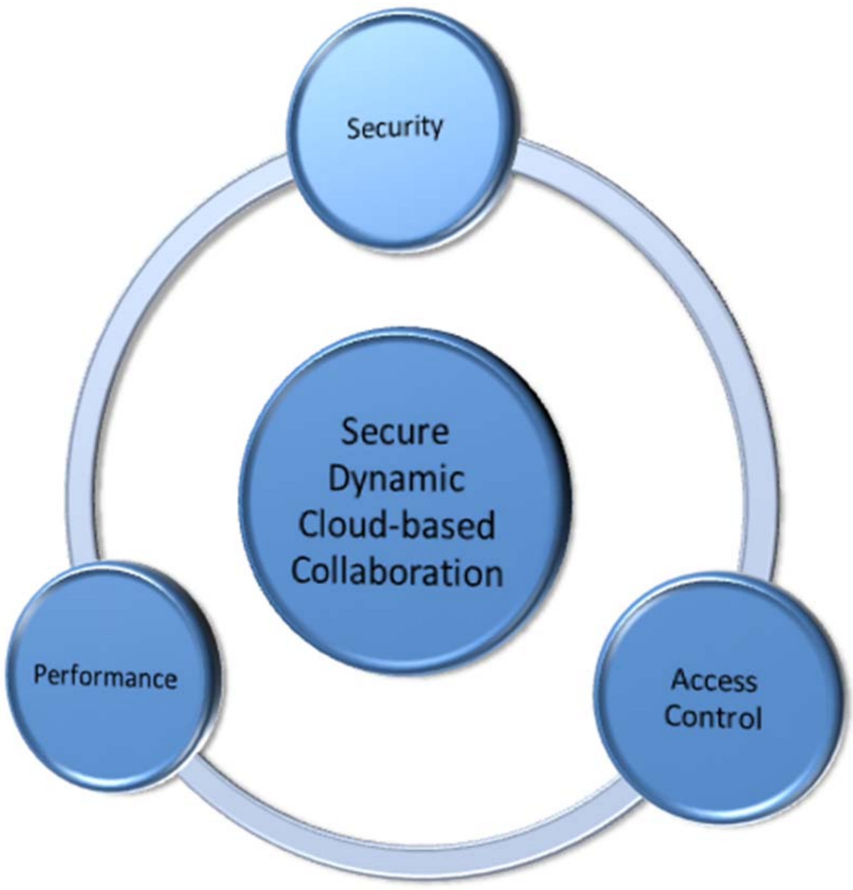
SECURE DYNAMIC CLOUD-BASED COLLABORATION WITH HIERARCHICAL ACCESS

Electrical and Computer Engineering
Technical Report ECE-TR-8



AARHUS UNIVERSITY
DEPARTMENT OF ENGINEERING

DATA SHEET

Title: Secure Dynamic Cloud-based Collaboration with Hierarchical Access
Subtitle: Electrical and Computer Engineering
Series title and no.: Technical report ECE-TR-8

Authors: Chris Piechotta, Adam Enø Jensen, and Martin Grooss Olsen
Department of Engineering – Electrical and Computer Engineering, Aarhus University

Internet version: The report is available in electronic format (pdf) at the Department of Engineering website http://www.eng.au.dk

Publisher: Aarhus University©
URL: http://www.eng.au.dk

Year of publication: 2012 Pages: 140
Editing completed: May 2012

Abstract: In recent years, the *Cloud* has emerged as an attractive way of hosting and delivering services over the Internet. This has resulted in a renewed focus on information security in the case where data is stored in the virtual space of the cloud and is not physically accessible to the customer. Through this thesis the boundaries of securing data in a cloud context, while retaining the benefits of the cloud, are explored. The thesis addresses the increasing security concerns of migrating to the cloud and utilising it for data storage.

The research of this thesis is divided into three separate areas: securing data in an untrusted cloud environment, ensuring data access control in the cloud, and securing data outside the cloud in the user's environment. Each area is addressed by separate conceptual designs. Together these comprise a secure dynamic cloud-based collaboration environment with hierarchical access. To further validate the conceptual designs, proof of concept prototypes have been constructed.

The conceptual designs have been devised by exploring and extending the boundaries of existing secure data-storage schemes, and then combining these with well-known security principles and cutting-edge research within the field of cryptography. The results of this thesis are feasible conceptual designs for a cloud-based dynamic collaboration environment. The conceptual designs address the challenges of secure cloud-based storage and allow the benefits of cloud-based storage to be utilised. Furthermore, this thesis provides a solid foundation for further work within this field.

Keywords: cloud, access control, digital rights management, security, cryptography, cloud storage, collaboration, software engineering and systems.

Supervisors: Peter Gorm Larsen and Joey W. Coleman.

Please cite as: Piechotta, C., Jensen, A.E., Olsen, M.G., 2012. Secure dynamic cloud-based collaboration with hierarchical access. Department of Engineering, Aarhus University. Denmark. 140 pp. Technical report ECE-TR-8.

Cover image:
Chris Piechotta, Adam Enø Jensen, and Martin Grooss Olsen.

ISSN: 2245-2087

SECURE DYNAMIC CLOUD-BASED COLLABORATION WITH HIERARCHICAL ACCESS

Chris Piechotta, Adam Enø Jensen, Martin Grooss Olsen

Aarhus University, Department of Engineering

Abstract

In recent years, the *Cloud* has emerged as an attractive way of hosting and delivering services over the Internet. This has resulted in a renewed focus on information security in the case where data is stored in the virtual space of the cloud and is not physically accessible to the customer.

Through this thesis the boundaries of securing data in a cloud context, while retaining the benefits of the cloud, are explored. The thesis addresses the increasing security concerns of migrating to the cloud and utilising it for data storage.

The research of this thesis is divided into three separate areas: securing data in an untrusted cloud environment, ensuring data access control in the cloud, and securing data outside the cloud in the user's environment. Each area is addressed by separate conceptual designs. Together these comprise a secure dynamic cloud-based collaboration environment with hierarchical access. To further validate the conceptual designs, proof of concept prototypes have been constructed.

The conceptual designs have been devised by exploring and extending the boundaries of existing secure data-storage schemes, and then combining these with well-known security principles and cutting-edge research within the field of cryptography. The results of this thesis are feasible conceptual designs for a cloud-based dynamic collaboration environment. The conceptual designs address the challenges of secure cloud-based storage and allow the benefits of cloud-based storage to be utilised. Furthermore, this thesis provides a solid foundation for further work within this field.

Acknowledgements

We would like to thank our academic supervisor Peter Gorm Larsen for his attention and advice during this thesis and Master's degree. We would also like to thank Joey W. Coleman for his invaluable input regarding the content and structure of this work.

Additional thanks goes to Giuseppe Ateniese and Matthew Green from the Johns Hopkins University, Kevin Fu from the University of Massachusetts Amherst, and Susan Hohenberger from the Massachusetts Institute of Technology, for their implementation of a cryptographic software library used in this thesis.

Finally, we would like to thank our family and friends for their patience and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the context of this thesis by describing its background and goals. The background material of this chapter directly informs the choice of concepts and technologies presented in in Chapter 2, and sets the context for all subsequent chapters.

1.1. Emergence of cloud services

The term *cloud computing* has been used in various marketing contexts and has been the subject of much attention. As a result, it does not have a clear definition. These factors, in combination, have been a cause of confusion. Although cloud computing is a term that has emerged within the last decade, the concept itself has roots in the forty year old idea of *computing as a utility* [Parkhill66]. In recent years cloud computing has emerged as an attractive paradigm for hosting and delivering services over the Internet. Cloud computing is attractive for businesses seeking scalability and availability at relatively low cost [Mather&09, Zhang&10]. The hardware and software provided by cloud computing is provided on an on-demand basis that, depending on usage, keeps the overall costs low. Compared to the traditional approach where resource provisioning is based on peak requirements, resources in the cloud can be acquired and released dynamically [Armbrust&09]. Furthermore, by outsourcing computing infrastructure, the challenges of its operation such as hardware failures, data distribution, and backups are transparent to its users [Wu&10].

1.1.1 Cloud services

The services provided by cloud computing are usually partitioned into three primary services, each with different levels of abstraction [Mather&09]. These services are:

Infrastructure as a Service (IaaS): At the lowest abstraction level, infrastructure providers have access to and manage large data centres. They are responsible for managing hardware to create pools of storage and computing resources by partitioning the hardware using virtualisation.

Platform as a Service (PaaS): Platform providers build upon an infrastructure service and provide operating systems and application frameworks which ease the burden of deploying applications.

Software as a Service (SaaS): Software as a service is at the highest level of abstraction and is where cloud applications are executed. These applications can be accessed on-demand over the Internet. Cloud applications are built to leverage the automatic scaling of the underlying services.

Storage as a Service (StaaS) is another service that is common in cloud computing [Wu&10]. Cloud storage allows users to store and access data from any location. Cloud storage systems are, in general, expected to have high availability, scalability, and performance [Vaquero&08].
Numerous other terms for services exist, in addition to the presented services, although the justification for some of them may be questionable. The phrase *Anything as a Service* (XaaS) spans all of them [Schaffer09]. Figure 1.1 shows the three primary services together with storage as a service in relation to the resource layers they govern. The figure also shows examples of services that, at the time of writing, are offered for each layer.

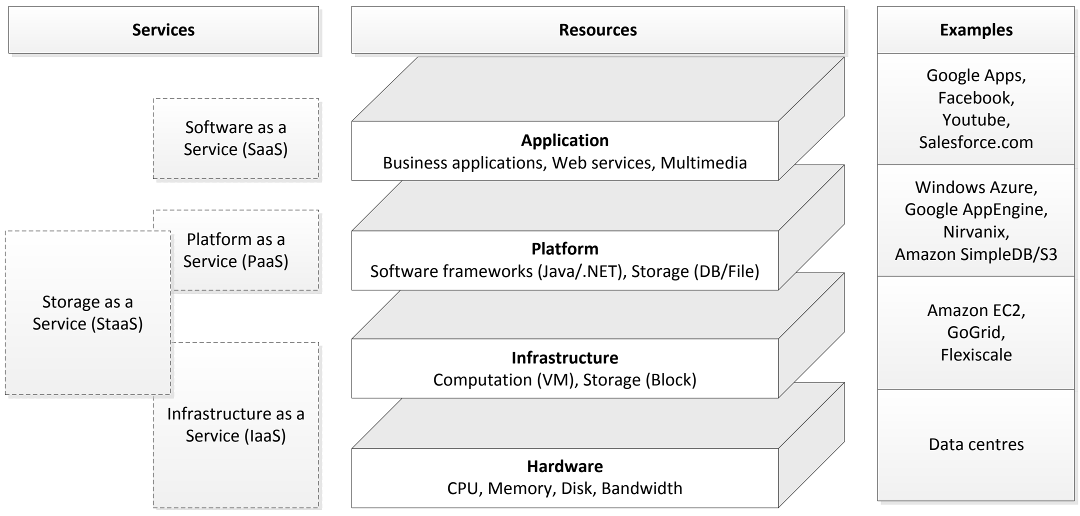

Services
Resources
Examples
Software as a Service (SaaS)
Platform as a Service (PaaS)
Storage as a Service (StaaS)
Infrastructure as a Service (IaaS)
Application
Business applications, Web services, Multimedia
Platform
Software frameworks (Java/.NET), Storage (DB/File)
Infrastructure
Computation (VM), Storage (Block)
Hardware
CPU, Memory, Disk, Bandwidth
Google Apps, Facebook, Youtube, Salesforce.com
Windows Azure, Google AppEngine, Nirvanix, Amazon SimpleDB/S3
Amazon EC2, GoGrid, Flexiscale
Data centres

Figure 1.1: Cloud resources, services, and examples thereof.

A *Cloud Service Provider (CSP)* is an entity which supplies any of the mentioned services. Some CSPs supply the services from the upper end of the service stack, others from the lower end. A few CSPs supply services that span the entire stack. CSPs supplying services from the upper layers who do not own an infrastructure rent one. Besides the services offered, a CSP can offer better physical security than a typical organisation because they maintain and guard large server farms. Some CSPs can also offer improved software security in the form of dedicated security teams [Potoczny-Jones11]. Because cloud computing is still relatively new, standards defining the requirements for cloud services are still nonexistent or not very widespread in the industry [Kern12]. However, some of the major CSPs have such a large influence on the cloud computing domain, that their service levels are becoming de-facto standards [Mohagheghi&11, Fratto12].
Aside from the services they offer, clouds can also be categorised, with the two main categories being public and private clouds [Armbrust&09, Mather&09, Zhang&10]:

Public cloud: In a public cloud services are provided to the general public by a CSP. As such, many consumers share the cloud. This helps keep down the cost for the individual consumer. However, public clouds have disadvantages. At the physical level, cloud storage is shared with others. Should a failure to separate the shared storage occur, the storage sharing could become a liability as it would allow consumers to access the data of others. Another disadvantage is the fact that consumers are subject to the CSP's security measures and have no immediate way of influencing these.

Private cloud: A private cloud is used exclusively by a single organisation. The organisation, or a third-party on their behalf, creates, runs, and maintains the cloud. Private clouds often have a higher quality of service than a public cloud, but require massive investments and are often compared to traditional server farms.

Combining a public and a private cloud results in a *hybrid cloud* where resources are split between the public and private part. Another type is a *virtual private cloud* where the underlying communication is virtualised. A virtual private cloud runs in a public cloud, but has some of the benefits of a private cloud.

1.1.2 Issues preventing cloud migration

There are many challenges for businesses considering migrating to the cloud. Security is one of the core challenges. In a ranked list, the most important challenge is "Availability of Service". The second and third most important ranked challenges are "Data Lock-In" and "Data Confidentiality and Auditability" [Armbrust&09].
A survey from 2008, made by the International Data Corporation (IDC), indicates security as a primary challenge [Zhou&10a]. Among the businesses asked, 74.6% indicated security as a challenge with regards to moving to the cloud. IDC has since, in December 2009, published an update to this survey [IDC09]. Figure 1.2 shows an overview of the the challenges and their importance, as indicated in the updated survey. Security is portrayed as a major challenge, but the figure also shows that security is not the only one.

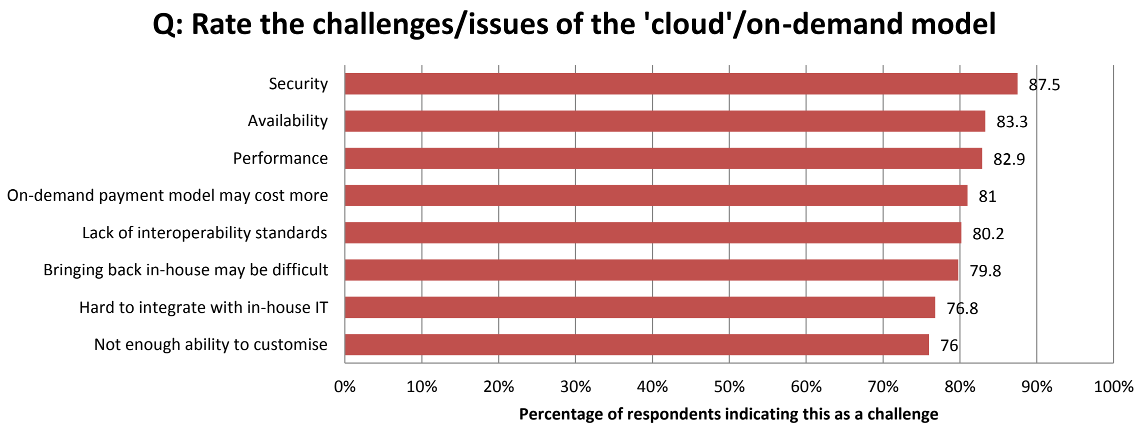


Figure 1.2: Survey of the challenges of cloud migration.

The lack of widespread standards for cloud services may cause an organisation, which is migrating to the cloud, to experience *vendor lock-in*. This results in the loss of the option of switching from one CSP to another, unless the organisation is willing to put in a major effort to accommodate already functioning cloud-based systems to the new CSP.
Another obstacle is the task of migrating to a cloud for the first time. This may be non-trivial if an organisation has a substantial amount of legacy applications. Each application needs to be analysed to determine its suitability, with regards to migration, as it must be scalable to properly benefit from the cloud [Mohagheghi&11].

The remainder of this chapter is organised as follows: the motivation for the work of this thesis is presented in Section 1.2, and the thesis goals are presented in Section 1.3. The relevance of the work is supported by a case which is presented in Section 1.4. A reading guide for the thesis is presented in Section 1.5, and the structure of the thesis is described in Section 1.6.

1.2. Motivation

The primary motivation for this thesis is the increasing usage of cloud-based systems. Several large companies use highly scalable cloud storage and computation solutions internally, and some also offer their cloud services for commercial use and as a product to individuals. This results in the cloud being of significant importance in modern society with regards to how information is stored and communicated. Therefore, we believe that the use and importance of cloud-based solutions will be increasing significantly in years to come.
A few months prior to the writing of this thesis, we took the course: *Advanced Pervasive Computing* and gained preliminary insight into the domain of cloud computing, its benefits, and liabilities. This insight has illustrated the need for a transparent security model which clearly states how data is protected inside the cloud. This need is strengthened with the increasing popularity of cloud based storage-applications, such as Dropbox [Dropbox12], Windows SkyDrive [SkyDrive12], and Google Drive [GoogleDrive12].
Research papers have presented several schemes on how to store and protect data inside and outside of a cloud context. Some have been implemented as prototypes and others remain purely theoretical schemes claiming to solve various issues and inconveniences. The main theme of most schemes is the use of cryptographic measures to achieve data security. A tendency among these schemes is the focus on specific technical functionality that offers little practical value by itself. The reason for this is likely the high level of complexity of cryptography which limits the scope of many research efforts to ensure adequate technical depth. We believe that these individually researched schemes can be improved and/or combined into a larger and more generic solution supporting the needs of a cloud-based collaboration environment.

1.3. Thesis goal, approach, and scope

1.3.1 Defining the goals

The goals of this thesis are:

1. To design a dynamic collaboration environment utilising the benefits of cloud storage while ensuring strong data security and fine-grained data access.
2. To improve our skills and knowledge with regards to designing systems that leverage the benefits of the cloud, improve our ability to research a scientific subject from different perspectives, and to contribute to the scientific community.

Any system can be described by its properties. To enable the comparison of designs for a dynamic collaboration environment, certain properties have been selected and will be the focus throughout this thesis. The chosen properties are those which we consider the most important in terms of addressing the issues preventing cloud migration. Furthermore, the dynamic collaboration environment should be as generic as possible to allow for its application in as many scenarios as possible. The properties are divided into three categories: security, access control, and performance. The dynamic collaboration environment covers both the cloud and the local environment at its users. However, the sets of important properties are not identical for both of these environments. Table 1.1 shows the properties selected for evaluating designs for a dynamic collaboration environment. The table also shows the category of the properties and the environments in which they are considered important.

Table 1.1: Properties for evaluating a dynamic collaboration environment.

Environment		**Category**	**Property**
Cloud	User	Security	Data confidentiality
	▨		Data integrity
	User		Data availability
	▨		Non-repudiation Auditability User-to-user anonymisation
		Access control	User addition and revocation
	User		Fine-grained access control
	▨		Access control delegation Many-to-many file sharing
		Performance	Scalability Durability

1.3.1.1 Property description and justification

The properties selected have been chosen for various reasons, which are briefly introduced and justified below:

Data confidentiality, integrity, and availability: As the surveys presented in Section 1.1 indicate, security is of great concern. Therefore, the dynamic collaboration environment should enforce security. Traditionally, security is understood as confidentiality, integrity, and availability. For the rest of this thesis the term *security* denotes these three properties, which are given a full definition in Section 2.2.

Non-repudiation and auditability: Non-repudiation and auditability are important for domains where errors or fraud might have large consequences, such as in the pharmaceutical industry. To support such domains, these properties should be supported.

User-to-user anonymisation: To support usage scenarios where entities involved in the system might attempt to collude or cartelise, user-to-user anonymisation should be supported.

User addition and revocation: The parties involved in a collaboration are often dynamic in nature and change over time. Thus, the dynamic collaboration environment should support addition and revocation of users.

Fine-grained access control: In most real-world scenarios users are trusted at different levels. Furthermore, to perform their tasks, users often only need access to a subset of the resources available. For these scenarios, fine-grained access control is essential.

Access control delegation: Access control delegation allows the dynamic collaboration environment to have multiple users manage access control at different levels. This avoids the potential bottleneck of having a single user manage access for all users.

Many-to-many file sharing: In general, a collaboration environment might be usable for some scenarios even if it supports only one-to-many or many-to-one file sharing. However, to avoid this constraint and remain generic, the dynamic collaboration environment should support many-to-many file sharing.

Scalability: A collaboration environment is a dynamic entity, which varies in size with regards the resources it stores and its amount of users. Therefore, it should support scalability.

Durability: Durability is important when storing large amounts of data for an extended period of time. The storage is inevitably going to experience problems, such as power interruptions or hardware failures. This could potentially cause data loss. Thus, the dynamic collaboration environment should possess high durabilty.

1.3.1.2 Cloud and user environments

All the properties described above are important in a cloud environment, but only some are relevant in a user's local environment. Of the six properties categorised under security only confidentiality and availability are considered relevant in the user's environment. Of the remaining properties only fine-grained access control is seen as relevant.

1.3.1.3 Additional properties considered

Several additional properties were considered for inclusion in the set of properties described above. The properties and the reasons for omitting them are the following:

Reliability: Reliability is a measure of a system's ability to perform its intended functions under a set of conditions over a period of time. While this is important to a dynamic collaboration environment, it is partly covered by availability and durability.

Maintainability: Maintainability is important for any system that is used over a longer period of time as defects are bound to be identified and must be corrected. However, for the purpose of devising a conceptual design for a dynamic collaboration environment, maintainability is less of a concern.

Usability: Usability is important for a dynamic collaboration environment to be accepted by its users. However, to a large extent this is something which can be addressed later and does not influence the functionality of the collaboration environment.

Interoperability: While integrating a dynamic collaboration environment with other systems is interesting, the dynamic collaboration environment, by itself, provides value to those concerned with migrating to the cloud.

1.3.1.4 Property rating scale

Solutions from related work and conceptual designs presented as part of this thesis are compared by examining their characteristics to determine how well the individual properties presented in Table 1.1 are supported.
Support for the described properties is rated at four levels. These levels are denoted by a single minus or up to three plusses, where three plusses is considered best. The scale is designed to make distinctions between designs presented in later chapters clear and to make the differences considered important in this thesis as clear as possible. By rating the characteristics of designs using the scale, and thereby comparing individual properties of the designs, an overall comparison of the designs is possible.
Each level of the scale is strictly better than the previous by accumulating positive characteristics of lower levels. This holds for all properties except user addition/revocation and scalability: these are reversed in the sense that higher levels of support do not accumulate positive characteristics, but instead eliminate negative characteristics. The characteristics required to attain a certain level are specific to each property. All properties have four levels defined, except non-repudiation and user-to-user anonymisation which have only two. Table 1.2 shows the granularity of the scale and the characteristics required for the specific levels for each property. The Data Owner (DO) is the entity administrating the collaboration environment and is described in further detail in Chapter 3.

1.3.2 Approach

The dynamic collaboration environment has been designed through several steps:

Inception: In this step, state of the art research, within the field of cloud-based collaboration systems and their supporting technologies, was gathered and analysed. This enabled us to evaluate the feasibility of the thesis goals and to improve on the rationale of the dynamic collaboration environment by defining a supporting case.

Elaboration: In this step, the dynamic collaboration environment was designed based on related works, existing technologies, and research.

Validation: In this step, the dynamic collaboration environment was validated by creating a proof of concept prototype to demonstrate the feasibility and integration of critical parts.

Three iterations of the steps described above have been performed, and three designs, each with its own focus, have been created. Although the designs have their own focus, they build upon previous designs and thereby support for the thesis goals is strengthened.

1.3.3 Scope

The scope of this thesis is limited to the use of cloud storage systems where on-line storage is supplied by a CSP and used to store data. This thesis does not deal with the cost of utilising a cloud, only the properties of one solution compared to another is considered.
Many solutions described in this thesis apply cryptography to obtain properties such as confidentiality. The properties and principles of these cryptographic schemes will be described and relied upon, but the internal mathematical workings of these cryptographic schemes are considered out of scope. Data is considered to be in either encrypted (ciphertext) or unencrypted (plaintext) form.

Table 1.2: Scale for rating the properties of a dynamic collaboration environment.

Property	Rating			
	-	+	++	+++
Data confidentiality	Data stored as plaintext	Confidentiality for data stored on a server or in a cloud	Confidentiality for data stored at a user	Data access patterns do not affect data confidentiality
Data integrity	Data integrity is not verifiable	A user can verify his own data stored on a server or in a cloud	The DO can verify any user's data stored on a server or in a cloud	Any user can verify any other user's data stored on a server or in a cloud
Data availability	Data is not available	Data is stored on a single secure server	Data is stored in a secure cloud environment	Secure offline data access in a user's environment
Non-repudiation	Actions can be repudiated	N/A	N/A	The DO and all user actions cannot be repudiated
Auditability	No auditability	The DO can inspect last access or change of existing data	The DO can inspect deletion of data	The DO can inspect access and modification history
User-to-user anonymisation	Users are not anonymous	N/A	N/A	Users are anonymous to each other
User addition and revocation*	No users except the DO exists	User addition and removal affects other users	User addition does not affect other users. Removal does	User addition and removal does not affect other users
Fine-grained access control	Users have read access to data on a server or in a cloud	Users can have read or write access to data on a server or in a cloud	Users can have access to only subsets of data on a server or in a cloud	Access control is enforced in the users' environments
Access control delegation	Delegation is not possible	Single level of access delegation	Multi-level access delegation	Delegation of all DO capabilities
Many-to-many file sharing	Single-user storage	One-to-one data sharing	One-to-many or many-to-one data sharing	Many-to-many data sharing
Scalability*	Data is stored at the DO	Data is stored on a server or in a cloud. The DO is required for authorisation and transfer	The DO is required for authorisation, but not transfer	The DO is not required for authorisation nor transfer
Durability	Data is not stored persistently	Data is stored persistently, but is lost if server or cloud is interrupted	Data is stored persistently and survives interruptions	Data storage is resistant to hardware failures

* For this property, the scale is reversed in the sense that higher levels of support do not accumulate positive characteristics, but instead eliminate negative characteristics.

1.4. Case study

1.4.1 A generic case

The generic case describes the need for a solution that allows data sharing between organisations of various sizes as well as individuals. The solution should be based on a cloud service as the benefits offered by such a service would, among other things, enhance availability and scalability. CSPs are regarded as untrusted third-parties. Therefore, data should never appear as plaintext inside the cloud environment. A selected group of users should be designated to control which users are granted access to specific data. For example, users should only have access to the data they need to fulfil their role, within their specific organisation. Adding new users and revoking access from existing users is necessary to support the dynamic structure of a collaboration environment. Fine-grained data access is considered an essential part of the desired solution. It should be possible to grant some users read-only access to the data, while others are allowed full read-write access. It should also be possible to audit which user has accessed data to track usage and operations on the data. This makes the solution more attractive to industries that require a higher level of auditability and traceability of their data.
Unauthorised backups and distribution of data should be prevented. As a consequence, security measures must be extended into the users' local environments to protect data once it is retrieved from the cloud storage. Non-repudiation and auditing of data should be supported in case data is leaked and later found in unauthorised distribution channels. This should enable the identification of the origin of the data and help identify the cause, or the user, responsible for the leak.

1.4.2 A concrete case example

As a concrete example of the generic case, consider an organisation, *BigCorp*, that needs custom mechanical parts built as a part of a larger project. To this end, they employ two separate contractors each of which delivers different mechanical constructs. The first contractor delivers a part that is dependent on a part that the other contractor delivers. Therefore they need to share details about how to interface with their respective parts. The first contractor needs to have read-write access, while the other only needs read access. BigCorp, however, does not want to manage access for all the workers working for the contractors. It wants to be able to give the contractors access based on their roles in the project and then let the contractors delegate granular access to their workers themselves.
BigCorp does not want the two contractors to have any detailed knowledge of each other. One reason is to prevent cartelisation, which could potentially drive up the price of the mechanical parts. BigCorp also wants to ensure that none of the project-plans end up in the wrong hands. The contractors want to keep their proposals private until it is ready for publication. This means strong security and protection of data in the contractors' local environment. Once a contractor has delivered his product, BigCorp wants to be able to revoke the access of that contractor without affecting the access of the other contractor.

1.5. Reading guide

Emphasis: Concepts, words, or sentences with special relevance are emphasised in the text by the use of an *italic typeface*.

User naming convention: Throughout this thesis the names *Alice, Bob, Carol, and Dave* are used when depicting users used to describe a conceptual design through diagrams and scenarios. A DO is depicted slightly different than the other users and can be recognised by a red user icon. The naming convention is used to keep consistency of diagrams throughout the thesis. The only exception to this is Chapter 5 where the notations *Consumer*, *Producer*, and *Publisher* are used due to the specific domain of this chapter.

Design comparison: Throughout this thesis, designs for a dynamic collaboration environment are compared by evaluating their support for the thesis goals. The comparisons are presented by tables and diagrams. For each compared design, the tables contain indications of how well the properties are supported. The diagrams used to present the comparisons are spiderweb or radar diagrams. In the diagrams, each property has a dedicated axis originating from the center of the diagram and a design is illustrated by lines crossing the axes. The area enclosed by the lines represents a design's overall support for the thesis goals. An example of a comparison between two designs can be seen in Figure 1.3.

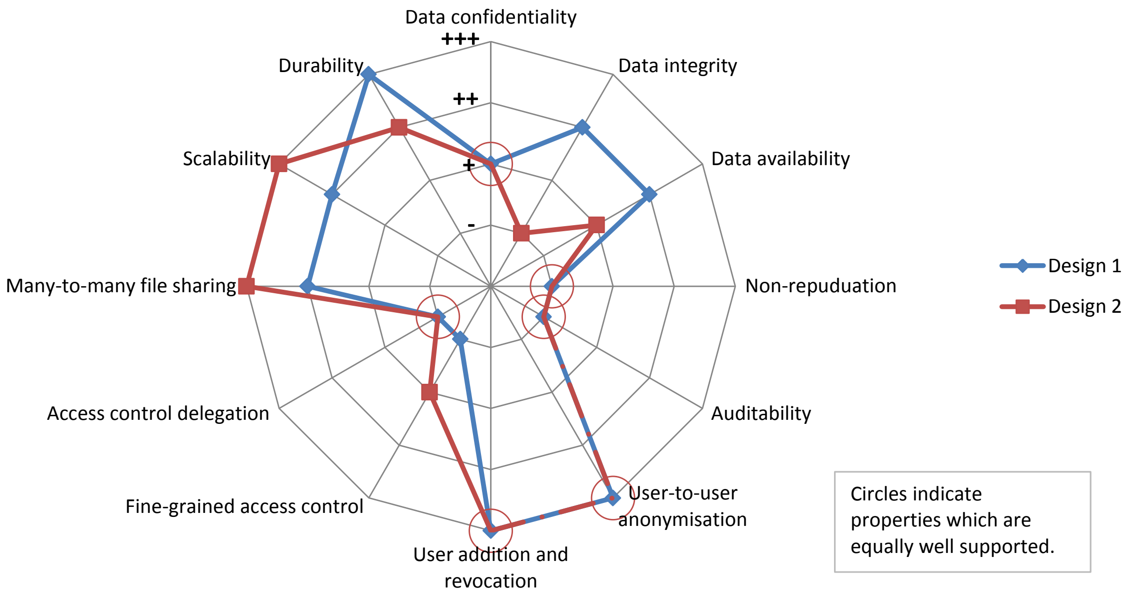


Figure 1.3: Example of a graphical comparison of two designs.

1.6. Structure

This thesis is divided into seven chapters and two appendices. Figure 1.4 shows an overview of the chapters and appendices along with lines indicating the suggested order in which chapters and appendices should be read. Chapter 2 contains the general theory for the thesis, explaining relevant security terms, concepts, and technologies.

Chapters 3, 4, and 5 each focus on subsets of the properties relevant to the dynamic collaboration environment. The chapters begin with a detailed presentation of related work, continue with a description of our contribution, in the form of a design for achieving the above mentioned properties, and a discussion thereof. These chapters are mostly independent, but should be read in lexical order to fully understand the thesis.

Chapter 6 describes proof of concept prototypes constructed to verify the designs presented in Chapters 3 and 4. Finally, Chapter 7 concludes the work of this thesis.

Appendix A presents the terminology used in this thesis and Appendix B contains details on the proof of concept prototypes, presented in Chapter 6.

Chapter 2: This chapter focuses on examining general theory, terms, concepts, and technologies relevant to the goal of the thesis.

Chapter 3: This chapter proposes a design for achieving confidentiality, integrity, and availability for data stored in an untrusted cloud by examining and improving upon related work.

Chapter 4: This chapter proposes a design for achieving fine-grained access control and auditability for data shared through a cloud.

Chapter 5: This chapter proposes a design for maintaining confidentiality, integrity, and availability when data is in a user's local environment.

Chapter 6: This chapter validates the designs proposed in earlier chapters by proof of concept prototypes.

Chapter 7: This chapter concludes the thesis, summarising, analysing, and discussing the results. Areas for further work are also established.

Appendix A: This appendix presents the terminology used in the thesis.

Appendix B: This appendix contains details on the proof of concept prototypes presented in Chapter 6.

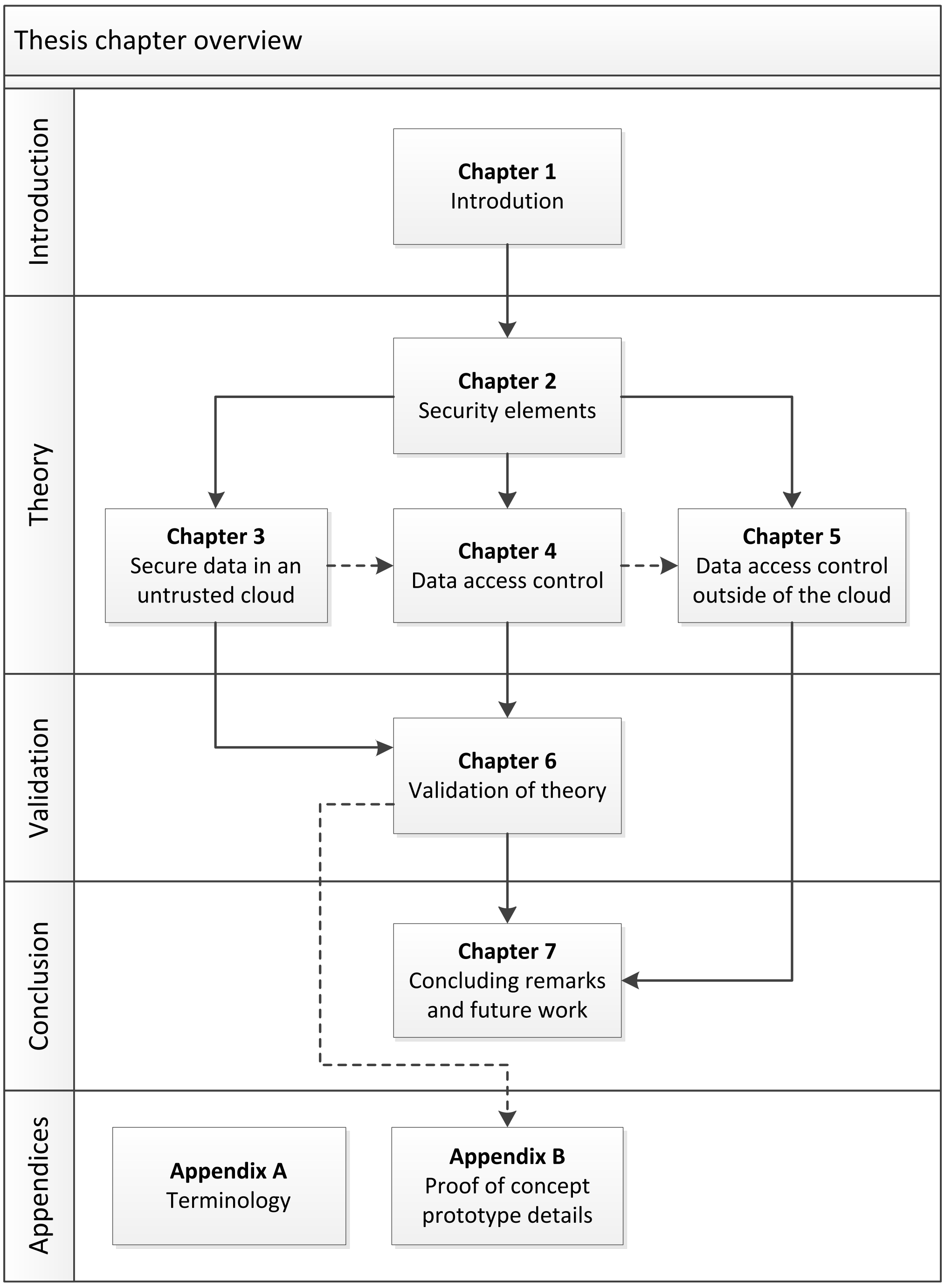


Figure 1.4: Overview of the thesis structure.

Chapter 2

Security elements

This chapter describes concepts and technologies used for achieving the thesis goals presented in Chapter 1. The material of this chapter ranges from generally known and widely used to very specific concepts and technologies. The content of this chapter forms the basis, and describes the security terminology, of the work presented in Chapters 3, 4, and 5.

2.1. Overview

This chapter describes concepts and technologies necessary to understand the results of this thesis. Section 2.2 gives a short introduction to the core principles of information security. Section 2.3 presents relevant cryptographic concepts, including cutting-edge research in cryptography. Finally, Section 2.4 describes concepts and schemes used to manage access control.

2.2. Information security principles

Information is an important asset and needs proper protection. The difference between information and data is that data is a specific representation of information, enabling the storage, communication, interpretation and processing of information. Traditional definitions of information security are based on three information security goals constituted by:

Confidentiality: Information that should only be disclosed to specific entities is considered confidential. Thus, to enforce confidentiality, disclosing information should be avoided except to the specific entities.

Integrity: Information integrity focuses on ensuring the fidelity of information with regards to its creation. Unauthorised modification or destruction of information, either accidental or deliberate, should be prevented. When examining data as the container for information, integrity breaches can be divided into two categories: Primary corruption, which is the modification of data when it is being transmitted, and secondary corruption, which is when data is already loaded onto a computer system [Harn&92].

Availability: Information availability refers to information being accessible to an authorised user when it is needed. As an example, when trying to access data from a server that is offline, there should be sufficient mechanisms in place to make this problem transparent to the user.

These three information security goals, Confidentiality, Integrity, and Availability, are by abbreviation referred to as CIA. When referring to security throughout this thesis the definition above is used.

2.2.1 Authentication

There are two kinds of authentication relevant to this thesis:

- Entity authentication, which is the process of verifying the claimed identity of an entity.
- Data origin authentication, which is the process of verifying the source and integrity of a message.

In general, authentication is based on any of the items below, or a combination thereof. Combining multiple items decreases the probability that an entity is presenting false evidence of its identity [Brainard&06]:

- Something you know: Password, PIN-code, etc.
- Something you have: Token, Certificate, Dongle, etc.
- Something you are: Fingerprint, Retinal pattern, DNA etc.

An example of authentication is challenge-response authentication where a user makes an access request and the host receiving the request sends a challenge to the user, who in turn responds [Diffie&92]. Access is granted if the response matches the expectation of the server. Using different challenges for each request helps prevent replay-attacks.

2.2.2 Authorisation

Authorisation is the process of determining access rights to a resource for an entity. Authorisation relies on authentication to verify the identity of the entity requesting access to the resource.

2.2.3 Non-repudiation

In information security, non-repudiation focuses on creating evidence that an action has occurred so that a user cannot falsely deny the action later. When working with data, this means that the integrity and origin of data can be proved. An example of non-repudiation would be a user who has modified a file, after which he can no longer deny that he was the one who made the modification.

2.2.4 Auditability

Auditability, in an information security context, refers to the ability to attribute actions on data to the performing entity. Thus, auditability enables tracing of actions performed to the entity who performed the action [Gerber&01].

2.3. Relevant cryptography

Many different approaches currently exist to ensure data confidentiality, and many of these approaches are cryptographic in nature. This section examines some of these approaches and goes into further detail about those of significant relevance to the thesis.

2.3.1 Enforcing data secrecy

Encryption is a widely used technique for enforcing data secrecy, but other techniques exist which may serve as a viable solution [Storer&09].

2.3.1.1 Secrecy by splitting

One approach towards secrecy is splitting of data into n parts in a way such that no combination of k parts reveals any information about the data, while any combination of m parts restores the data $(k < m \leq n)$ [Storer&09]. However, splitting data into n parts is not without problems as it introduces extra overhead and the need for n distinct storage locations. Data integrity can be ensured by using Error-Correcting Code (ECC) mechanisms at the cost of extra distinct storage locations. Figure 2.1 shows how data can be stored in this approach by splitting and storing data at several different locations while adding parity as an ECC mechanism.

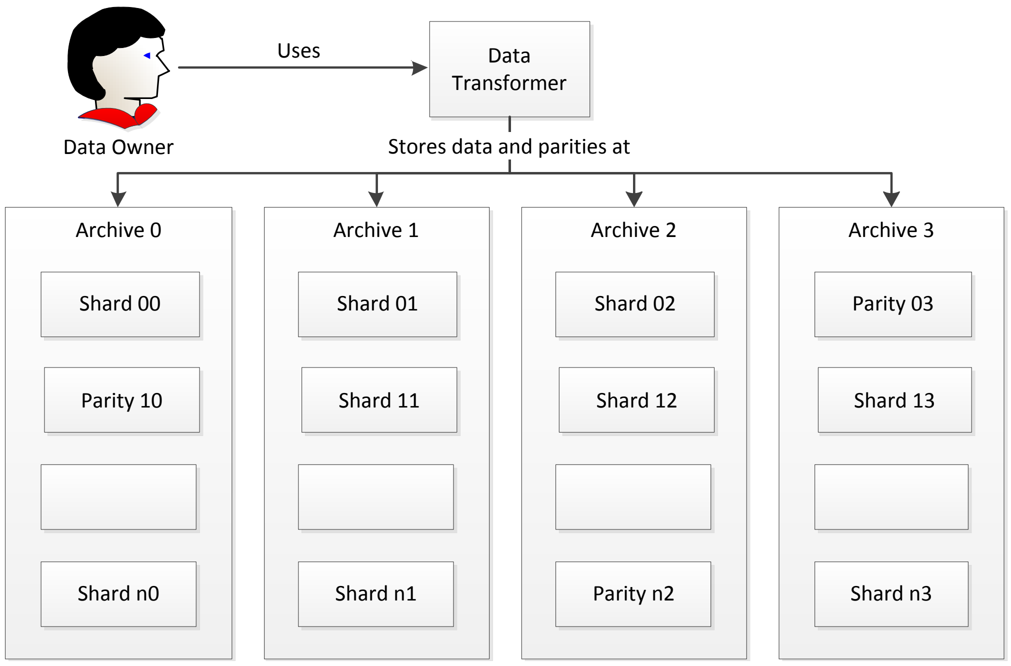


Figure 2.1: Data secrecy by splitting.

2.3.2 Secrecy by encryption

Encryption is the process of transforming unencrypted data into encrypted data using an algorithm and one or more cryptographic keys. Decryption is the reverse operation. Encryption is suitable and often applied for ensuring data confidentiality. However, applying it effectively is not straightforward. A problematic aspect of cryptography is that once data is encrypted, the liability simply shifts from protecting the confidentiality of the data to protecting the confidentiality of the cryptographic keys [Diffie&76, Storer&09, Agudo&11]. Therefore, management of the cryptographic keys add to the complexity of employing encryption schemes. The advantage of using encryption is that, in general, keys are smaller and there is less to protect.

There is an on-going struggle between cryptographers and cryptoanalysists, where ciphers, once thought to be secure, now are cracked with ease. Given enough time, all cryptographic algorithms can be cracked and cryptoanalysis only assists in speeding up the process. The sole exception being one-time pads [Shamir83]. In the worst case, an inside attacker may gain knowledge of the key required to decrypt the ciphertext and thereby circumvent the encryption entirely. Accordingly, cryptography may be adequate for short term, but not long term storage [Storer&09].

There are two kinds of encryption: *symmetric* and *asymmetric*. Symmetric encryption has a single key used for both encryption and decryption. Asymmetric encryption uses a key pair consisting of a *public key* and a *private key*. The public key is used for encryption and the corresponding private key is used for decryption. Generally, asymmetric encryption does not perform as well as symmetric encryption because of the underlying mathematics used [Chen&10].

2.3.3 Searchable encryption

Searchable encryption is a scheme that allows search queries to be performed on a ciphertext. The first attempts to develop cryptographic techniques for searching inside encrypted data resulted in two schemes which form the basis of all subsequent work [Song&00]:

Sequential scan: In the sequential scan approach, the encrypted data is searched sequentially and thus, search time is linear, $O(n)$.

Index-based search: In the index-based search approach, a pre-computed index is used to improve the speed of searching. The index is a map of keywords to data and makes it possible to identify data associated with a particular keyword by performing a look-up in the index. A disadvantage of index-based search is that the index must be pre-computed and any change to the data necessitates updating the index. Another disadvantage is that the index takes up extra storage space. Even though index-based search has disadvantages, its ability to efficiently handle large amounts of data makes it the more widely used approach. Depending on the implementation, the search time can become logarithmic, $O(\log n)$.

To hide the meaning of keywords, encryption can be used to create tokens based on the keywords [Song&00]. Thus, the index becomes a map between tokens and data entities. The tokens are generated by the party wishing to search and are then sent to the storage provider which uses the token to search the index. Neither the tokens nor the ciphertext reveal information about the stored data or the keywords being searched for.

Even if data confidentiality is ensured during transmission and when data is stored, the access patterns may leak information since token generation is deterministic. The storage provider can over time, through statistical analysis, gain knowledge of the stored data. Obscuring these patterns is desirable, but may prove difficult [Dong&11].

Searchable encryption has, since its introduction, been improved with regards to efficiency, security, and flexibility [Chang&05]. Research has improved the efficiency and security, but also extended searchable encryption to a multi-user scenario [Curtmola&06]. *Symmetric Searchable Encryption* (SSE), *Asymmetric Searchable Encryption* (ASE), *Multi-user Searchable Symmetric Encryption* (MSSE), and *Efficient asymmetric Searchable Encryption* (ESE) have, subsequently, also been developed. For an overview of the properties of these searchable encryption schemes see Table 2.1.

Table 2.1: Searchable encryption schemes.

Type	**Underlying technique**	**Scenario**
SSE	Symmetric encryption	Single Reader & Single Writer (SR+SW)
ASE	Asymmetric encryption	Single Reader & Multiple Writers (SR+MW)
MSSE	Symmetric encryption & Broadcast encryption	Multiple Readers & Single Writer (MR+SW)
ESE	Asymmetric encryption	Single Reader & Multiple Writers (SR+MW)

SSE: This scheme is based on symmetric encryption and the key used for encryption and decryption is also used to generate search tokens. All known index-based SSE constructions can use any underlying symmetric encryption scheme [Kamara&11].

ASE: This scheme is based on asymmetric encryption. ASE is not as efficient or secure as SSE, but is more flexible with regards to expressiveness of search queries [Kamara&10]. ASE was first introduced by [Boneh&04].

MSSE: This scheme utilises SSE to enable a central entity to encrypt a message to a dynamically changing group of authorised users [Curtmola&06].

ESE: This scheme is based on ASE, but improves upon efficiency. The principle of ESE is to encrypt plaintext using asymmetric encryption while deterministically generating a token for each associated keyword [Bellare&07]. Because the tokens are generated deterministically they are suitable in a map of keywords and data entities. However, ESE is less secure than ASE because of the deterministic tokens [Kamara&10]. An ESE scheme can generate tokens by using hashing algorithms and this results in the ability to use any asymmetric encryption scheme [Bellare&07].

Despite a sizable research effort in searchable encryption, all current schemes have severe limitations. Several schemes do not support modifications to data once it is indexed [Song&00, Curtmola&06]. Another limitation is that no existing scheme supports the scenario of Multiple Readers & Multiple Writers (MR+MW). Also, all of the mentioned schemes support queries based on a single keyword only [Song&00, Chang&05, Curtmola&06, Kamara&10].
Research in the area of searchable encryption remains active and has progressed significantly. The authors of [Curtmola&06] have published a new version of their original paper which elaborates on the details of their original work [Curtmola&11]. Additional progress was also made on SSE to create a scheme that supports efficient addition and deletion of data [Kamara&11].
With regards to complex search queries, schemes supporting conjunction of search terms have been developed and later disjunction has been added [Katz&08]. Support for concepts such as multi-keyword ranked search have also been devised [Cao&11].

Searchable encryption is not a new concept and has been known for more than a decade. It has come a long way since its conception by [Song&00], but many of the current schemes still need to be improved with regards to performance, while support for some areas are still missing entirely.

2.3.4 Proxy re-encryption

The first description of Proxy Re-Encryption (PRE) is found in [Blaze&98]. PRE is based on asymmetric encryption and is a cryptographic technique which enables the transformation of a ciphertext encrypted by one key into a new ciphertext encrypted by another key. This is done without revealing anything about the original plaintext or the cryptographic keys, and allows PRE to be performed by a semi-trusted third party known as an *encryption proxy*.
The principle of PRE can be seen in Figure 2.2 and is as follows: two persons, Alice and Bob, each have a key pair. Alice creates a special PRE token based on her private key and Bob's public key and gives this token to the encryption proxy. By using the PRE token, the encryption proxy is now capable of transforming a message encrypted with Alice's public key into a message encrypted with Bob's public key. Bob can later decrypt the ciphertext using his private key.

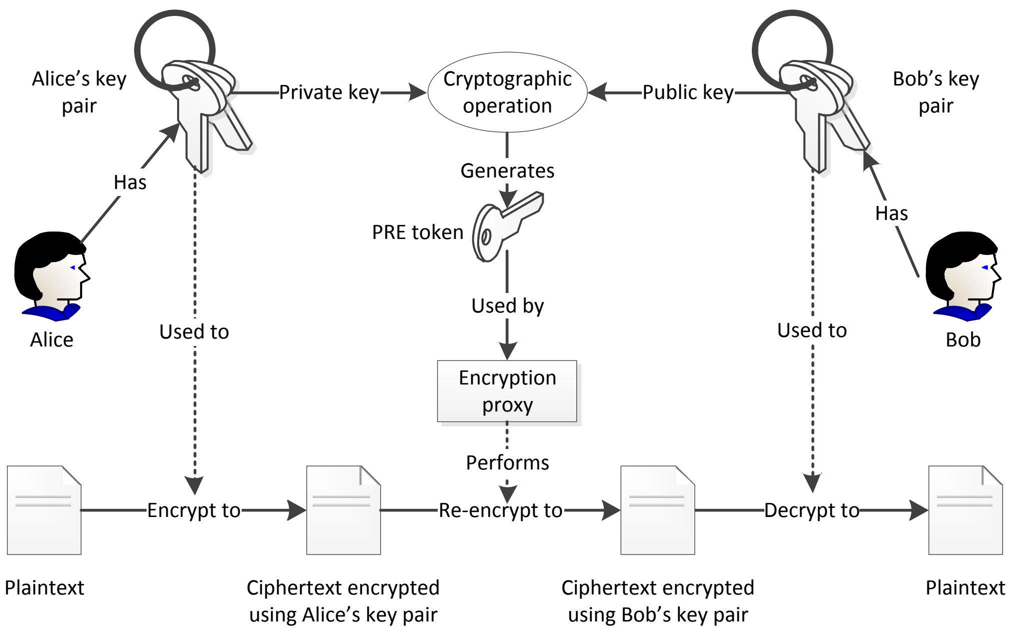


Figure 2.2: Overview of proxy re-encryption.

The original work by [Blaze&98] only supports bi-directional PRE. Consider two persons, Alice and Bob, the consequence is that it is not possible to transform a ciphertext from Alice to Bob without also enabling transformation from Bob to Alice. Should uni-directional transformation be required, this scheme is not usable [Ateniese&06].
Both transitive and non-transitive PRE schemes exist [Ateniese&06]. The principle of transitivity in a PRE context is the following: three persons, Alice, Bob, and Carol each have a key pair and the encryption proxy has a token to transform ciphertext from Alice to Bob and another token to transform from Bob to Carol. Transitivity allows an encryption proxy to transform from Alice to Carol by using both tokens and performing two transformations.

The properties of PRE schemes have improved in several ways since their introduction. One improvement is support for uni-directional PRE [Ivan&03]. Further improvements along with a comparison of PRE schemes are given in [Ateniese&06].
An inherent problem with PRE is the risk of any single user colluding with the proxy. In the scheme proposed by [Blaze&98], collusion between a user and a proxy can reveal another users' secret key. The collusion problem is partially solved in [Ateniese&06], but a general solution has not been found [Dong&11].
The performance of PRE may, for some purposes, be inadequate because it relies on relatively inefficient public key cryptography. This can be fixed by applying *hybrid encryption*.

2.3.5 Hybrid encryption

Hybrid encryption combines symmetric and asymmetric encryption and is meant to fix the inherent performance problems in asymmetric encryption. High-performance symmetric encryption is used to encrypt the bulk of the data using a randomly-generated key. This key is afterwards encrypted using asymmetric encryption. The resulting ciphertext is then the tuple of the symmetrically encrypted data and the asymmetrically encrypted key [Dong&11].

2.3.6 Digital signing

Digital signing is based on cryptographic key pairs. In digital signing, a user's signature for a message M is a value which depends on M and the user's secret key [Diffie&76]. Anyone is able to verify the validity of the signature using the user's public key. Digital signing is often used for verifying integrity and to achieve support for non-repudiation.

2.4. Data access control

Access control is essential to ensure fine-grained access to resources. To further ensure that access to resources can be managed and configured as necessary, a solid access control model, or a combination of multiple models, is needed. To prevent unauthorised access to resources located in multiple users' individual local environments access control is needed, not on the space where the resource is located, but on the resource itself. Insight into different access control approaches is provided in the following sections.

2.4.1 Data access control models

Data access control models are standard approaches for solving the issue of providing manageable access schemes to resources as described by the National Institute of Standards and Technology [NISTIR7657]. Some models use cryptography while others do not. There are three well-known security principles for access control models [Saltzer&75, Sandhu&96]:

Least privilege: A user is given only the permissions required for the user to perform his tasks.

Separation of duties: The objective of separation of duty is to prevent fraud and errors. This is achieved by distributing the permissions for a sensitive task among multiple users. An example from the financial world is when two persons must collaborate to sign a check.

Data abstraction: Abstract permissions, such as credit and debit for an account, are defined instead of the permissions like: read, write, execute, etc.

2.4.1.1 Access control by cryptography

Attribute-Based Encryption (ABE) is a type of fine-grained public key encryption where an encryption is associated with an access control policy [Yu&10, Zhao&11, Ion&11]. There are two sub-schemes of ABE [Yang&11]:

Ciphertext-Policy Attribute-Based Encryption (CP-ABE): In CP-ABE a user's private key is associated with attributes. An access structure over attributes is associated with data when it is encrypted and a user is only able to decrypt the data if the attributes of his private key satisfy the access structure of the ciphertext.

Key-Policy Attributed-Based Encryption (KP-ABE): KP-ABE uses the same principle of attributes and an access structure over attributes, but it is the reverse of CP-ABE. In KP-ABE, attributes are associated with data and an access structure is associated with a private key.

ABE is capable of fine-grained access control, but user revocation, for solutions supporting it, is inefficient as data must be re-encrypted [Singh&08, Yu&10].

2.4.1.2 Access control lists

Access Control Lists (ACLs) saw much use in the 1970s when multiple users using several terminals needed to be restricted from accessing data on shared computers systems [Saltzer&75]. Windows and UNIX systems alike make use of ACLs to manage resource access with varying degrees of complexity.
An ACL contains mappings between entities requesting access to a resource and the actions which the entities are allowed to perform. For example, a user trying to read from a resource. The resource has its own specific ACL specifying who has access to it and what actions are allowed. Thus, when an entity tries to access the resource, the operating system checks the ACL of the resource and determines whether actions requested by the entity are allowed.
In terms of performance and maintainability ACLs have limitations. The ACL for a particular resource must be accessed every time the resource is accessed. This hampers scalability and impedes performance. In a setting where many entities have different levels of access rights to resource objects, keeping ACLs up to date can be error prone and time-consuming. Least privilege is supported, while the other security principles for access control models are not inherent to ACL.

2.4.1.3 Role-based access control

Role-Based Access Control (RBAC) is an access control model that allows for grouping of different entities into categories based on similar access needs to fulfil a particular role. By mapping resource access to roles instead of individual entities, the need to maintain permissions for each entity is eliminated. Furthermore, entities are not limited to a single role, but may be assigned multiple roles granting them access to a broad array of resources. The downside to this, however, is the removal of granular access control of individual entities.
RBAC supports the three well-known security principles for access control models. However, the principle of least privilege is only supported if specific roles are created. Depending on the amount of specific roles created this may negate the benefits of RBAC.

2.4.1.4 Attribute-based access control

Attribute-Based Access Control (ABAC) is an access control model based on attributes that are associated either with the requested resource or the entity which is requesting access. The attributes

are used to determine access by comparing the attributes against a set of values [Wang&04a]. Attribute-based access control can be seen as a generalisation of role-based access control. When an entity requests access to a resource, the entity will either directly or indirectly provide a set of attributes which will be checked against attributes required for permission.
An advantage of ABAC is that the entity requesting access does not need to be known in advance to the system responsible for granting or denying access. As long as the supplied attributes meet the criteria of the rules governing access to the resources, access is granted. This is useful in large corporations where people may leave or join the organisation arbitrarily.
ABAC is rarely supported in operating systems, but commonly implemented at the application level using the open and widely accepted eXtensible Access Control Markup Language (XACML) standard [Priebe&06]. XACML is an eXtensible Markup Language (XML) dialect for defining ABAC policies.

2.4.1.5 Policy-based access control

Policy-Based Access Control (PBAC) is a model that standardises the ABAC model with regards to using ABAC at an enterprise level. Enterprise level organisations often need to be in compliance with legislation and this results in a need for tighter policies that control what resources should be available to whom and under what circumstances. Mechanisms that allow auditing of access to these resources should also be in place.
PBAC, like ABAC, uses attributes, but combines these with a set of circumstances under which access is requested. Rule-sets specify if access should be granted under organisational policy for those attributes under those circumstances. PBAC is a complex extension of ABAC. The added complexity is caused by extra maintenance of attributes and of enterprise level systems which needs to be deployed across the organisation to accommodate for PBAC.

2.4.2 Digital Rights Management

A Digital Rights Management (DRM) system is a specialised system for managing data access control. Most DRM systems offer a range of services used to publish, protect, and trace data. Data protection often consists of cryptographic schemes which are used to ensure confidentiality and integrity. DRM systems are often used as mechanisms for protecting digitally purchased goods such as music or movies. A typical DRM solution involves three distinct roles which each have different perspectives and goals [Michiels&05]:

Producer: The producer either produces or owns the right to data. He wishes to distribute data among selected groups of users and determines the access rights for the data such as read-only access.

Consumer: The consumer's main interest is the consumption of data. Consumption is often performed through a DRM client which is part of the DRM solution. By using a DRM client, control and protection of DRM protected data is made easier since decryption and details on decryption keys is hidden from the user.

Publisher: The role of the publisher is to manage the DRM system and perform distribution of DRM protected data. The task of monitoring data usage and revoking consumer access is also governed by the publisher. If unauthorised copies of data have been found in places where they are not supposed to be, the publisher is also responsible for investigating data leakage.

2.4.2.1 Interoperability

DRM solutions use a Rights Expression Language (REL) to formalise data control within the DRM system and unambiguously specify usage rules [Michiels&05]. As new methods for data distribution have emerged, the RELs have been extended accordingly. This has resulted in complex specialised extensions to some of the standard RELs. This increase in complexity and non-standard extensions has resulted in several non-interoperable RELs [Jamkhedkar&09]. This means, that although the common roles described in Section 2.4.2 can be identified in many DRM systems, most of the functionality and services in these DRM systems cannot be reused in other contexts. Furthermore, the implementation and semantics of the protocols used in most DRM solutions are deeply integrated with the REL. Changes made to the REL would in these systems ripple into the implementation of the specific DRM system [Jamkhedkar&09]. This problem, combined with the proprietary nature of most DRM systems, makes interoperability and re-usability very difficult.

2.4.2.2 Security

Licence Usage rights for DRM protected data are often expressed through licences assigned to authorised users. To ensure proper data protection, these licenses should only be issuable by a license service inside the DRM solution. If licenses can be forged, the protected data is no longer safe from unauthorised access. Another issue is how to maintain the integrity of licenses to prevent an authorised user, with a valid license, from editing the license to change their access rights. Furthermore, to prevent users from distributing valid licences, each individual licence must be uniquely associated with a specific user [Michiels&05].

DRM client When DRM protected data resides with a consumer, measures must be taken to ensure data confidentiality and integrity. The encryption and decryption details within the DRM client must be hidden from the user, as exposing these details could give away details which could lead to breach of confidentiality. DRM protected data with time-limited access needs a viable time keeping scheme to ensure access is granted only within the allotted time. This requires the time keeping scheme to be based upon a trusted clock which cannot be tampered with by the consumer. This can be achieved through the use of hardware clocks [Michiels&05].
Another concern is the way each individual DRM client is implemented. If all clients were to use the same cryptographic method and a common keying scheme, a single point of failure is introduced which would compromise the system if the cryptographic details were exposed on any of the DRM clients.

Tamper resistance DRM clients located in an uncontrolled environment are vulnerable to malicious users who might attempt to alter or tamper with the client to reveal the decryption process or key. Reverse engineering of the DRM client to reveal the cryptographic details can be hindered by obfuscating its implementation [Horne&02]. Individualisation of the DRM clients, so that one successful attack does not compromise other DRM clients is another method [Horne&02, Michiels&05].

2.4.2.3 Offline usage

If a consumer cannot establish a connection to the publisher, then none of the DRM protected data can be accessed if the DRM system does not support offline usage. An offline usage model allows users to temporarily access and read data already present at the DRM client without requesting the DRM publisher for permission. Offline usage requires the user and the environment, in which the DRM client is located, to be uniquely identifiable in order to establish a trust relationship between the DRM client and publisher.

2.4.2.4 Watermarking

The analog gap One of the biggest challenges in DRM is the analog gap or analog hole. Humans in general, are not able to manually decode digital messages. Therefore it needs to be presented in an analog environment, such as on a monitor as analog output. Because of this, it is vulnerable as an attacker is able to record the content in its unprotected form [Wolf&07]. A tool employed to tackle this problem is watermarking.
A watermark has two distinct properties: first, it is normally hidden from view and becomes visible only when using a special viewing process; second, the watermark itself contains information about the object it marks. A watermark usually has a record of its origin, when it was issued, who it was issued to and who has the overall responsibility for the data at the time of issue.
Digital watermarking is a process of embedding an imperceptible message about multimedia content into that content [Cox&01]. This practice can also be applied to text documents to aid in copyright protection [Jalil&09]. Watermarking is not able to prevent the original issuee of the watermarked data from sharing it with an unauthorised party, but may help in identifying the original issuee.
The ideal watermark leaves no perceptible difference between the watermarked and original data. A straightforward way to provide an imperceptible watermark is to embed it into the perceptually insignificant portion of the host data [Podilchuk&01]. This approach, however, leaves the watermark vulnerable to attacks, as it can be removed or altered without affecting the perceptible part of the host data.
As an example, the last couple of seconds of the credits in a video is not a significant portion as compared to the rest of the video content. A simple truncation of the video would remove such a watermark without affecting the main content of the video. To ensure robust watermarking, watermarks should be embedded in a important, but perceptually insignificant portion of the host data.

Shortcomings While it is true that data can be embedded with a watermark, it is important to notice that the watermark only provides traceability to the first issuee. It cannot identify users who redistribute data without permission when they receive it without permission themselves [Steinebach&07]. Another shortcoming is the inability to register when data is leaked. The only way to know if data is leaked is to find it somewhere it is not supposed to be. At this point, the data has already been compromised and confidentiality cannot be restored. However, identification of the issuee who leaked the data can help prevent future leakage of data from that source. A framework that deals with locating watermarks in arbitrary search spaces such as P2P networks, “Google spaces”, and eBay is proposed in [Wolf&07].

Chapter 3

Secure data in an untrusted cloud

This chapter describes a conceptual design used to ensure security of data stored in a cloud. The design supports efficient addition and revocation of users and ensures that only authorised users are able to access and modify data. These are core properties presented as thesis goals in Chapter 1. To achieve this, some of the concepts and technologies described in Chapter 2 are used. The conceptual design presented in this chapter provides the core for the extension proposed in Chapter 4. The design is validated in Chapter 6 through a proof of concept prototype.

3.1. Introduction

When cloud storage is utilised in a dynamic collaboration environment the benefits of the cloud are gained by exposing the stored data to security risks. Data security may be compromised when data is stored in the cloud or during transmission to and from the cloud. Thus, when using cloud storage, these security risks must be mitigated.

The assumptions made for this chapter are described in Section 3.2 and related work is described in Section 3.3. Section 3.4 describes our conceptual design for a dynamic collaboration environment. Finally, the conceptual design is discussed and compared to related work and the thesis goals in Section 3.5.

3.2. Assumptions

When using cloud storage, the Cloud Storage Provider (CSP) of the untrusted cloud is assumed to be *honest but curious*. This means that the CSP is expected to be curious to learn data content and has full access to everything stored in the cloud, but will honestly follow any protocol provided by the Data Owner (DO). Thus, the CSP will not actively manipulate data or communication. The same is assumed for any user accessing the untrusted cloud.

These assumptions can be loosened by adding extra memory checking to provide a robust solution at the cost of additional overhead [Curtmola&06]. Furthermore, it is assumed that the CSP may attempt to collude with an arbitrary user to attempt to gain knowledge about the stored data or queries of any other user.

3.3. Related work

This section describes and analyses how others have approached the challenge of a dynamic collaboration environment using cloud storage. Multiple solutions are examined, after which an overview is presented, comparing the properties of the solutions to the thesis goals.

3.3.1 Work by Kamara et al.

To ensure confidentiality of data stored in an untrusted cloud environment, a solution is presented in [Kamara&10]. The solution uses the cloud to store encrypted data and key and access management is handled by the Data Owner (DO).
The data stored in the cloud is encrypted by combining different encryption schemes. First, the data is encrypted using Symmetric Searchable Encryption (SSE) using a random encryption key. Then, the encryption key is encrypted using Attribute-Based Encryption (ABE). Finally, the encrypted data is stored in the cloud.
By using this encryption procedure, the data located in the cloud is made searchable to users with proper access privileges. The data can then be decrypted once downloaded by a user.
Users who should be able to access data in the cloud are issued credentials matching their access rights by the DO. These credentials are used to authenticate a user when he searches for data in the cloud. A token generator service is used to handle search requests and in response, the user is given a search token. The user then sends the search token to the cloud. Based on the search token, the cloud identifies which encrypted data should be returned to the user. See Figure 3.1 for an overview of the solution.

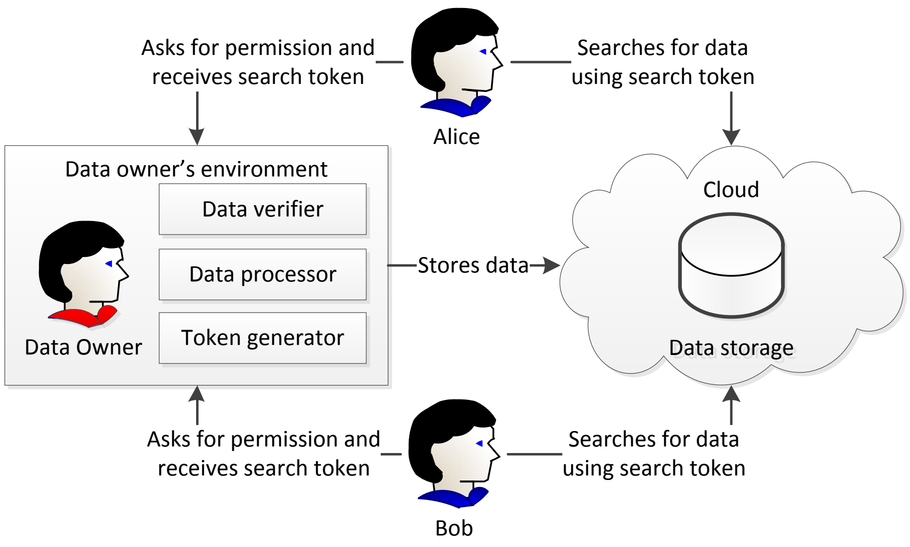

Asks for permission and receives search token
Alice
Searches for data using search token
Data owner's environment
Data verifier
Data processor
Token generator
Data Owner
Stores data
Cloud
Data storage
Asks for permission and receives search token
Bob
Searches for data using search token

Figure 3.1: Overview of solution presented by Kamara et al.

This approach offers several benefits, especially from a security perspective, since the cloud contains only encrypted data. Furthermore, the use of searchable encryption allows searching through encrypted data without downloading and decrypting it first, which improves the performance of the solution. Integrity is preserved by introducing a data verifier which the DO may use to verify the integrity of data stored in the cloud.

A rather large drawback of this solution is the fact that data cannot be altered. Data can be put into the cloud only when initialising the system and remains immutable afterwards. Although data is made searchable, the real benefits offered by a cloud are not utilised fully as it is used for storage only. To preserve confidentiality encryption and decryption of data is performed outside the cloud. Thus, it does not benefit from the computational power of the cloud.

The access management part of the solution is a potential bottleneck because it resides at the DO and is required for all searches. This impacts availability negatively. Another problem is that the encryption scheme used, does not offer support for a Multiple Reader + Multiple Writer (MR+MW) scenario. To support a dynamic collaboration environment, support for multiple readers and writers is necessary.

In [Kamara&11], an improved solution which builds on the work presented in [Kamara&10] is presented. The main improvement is the support for editing and deleting encrypted data in the cloud after its initialisation. Still, the suggested architecture lacks support for multiple writers. In addition to improvements in the scheme, the solution has been implemented and analysed with regards to performance.

The tokens used to request data from the cloud are generated deterministically and this could lead to breach of confidentiality since access patterns can be analysed through observation. This is a potential security flaw that the improved solution shares with the architecture presented in [Kamara&10].

3.3.2 Work by Dong et al.

An approach to form a solution supporting MR+MW is to combine searchable encryption and Proxy Re-Encryption (PRE). In particular, a uni-directional PRE scheme is combined with Efficient asymmetric Searchable Encryption (ESE) [Dong&08]. It should be noted that the solution presented here does not consider a cloud context.

Keys are handled by a separate and trusted Key Management Server (KMS). Introducing a trusted server is clearly a disadvantage, but the role of the server is limited to issuing and revoking users and can be put offline most of the time.

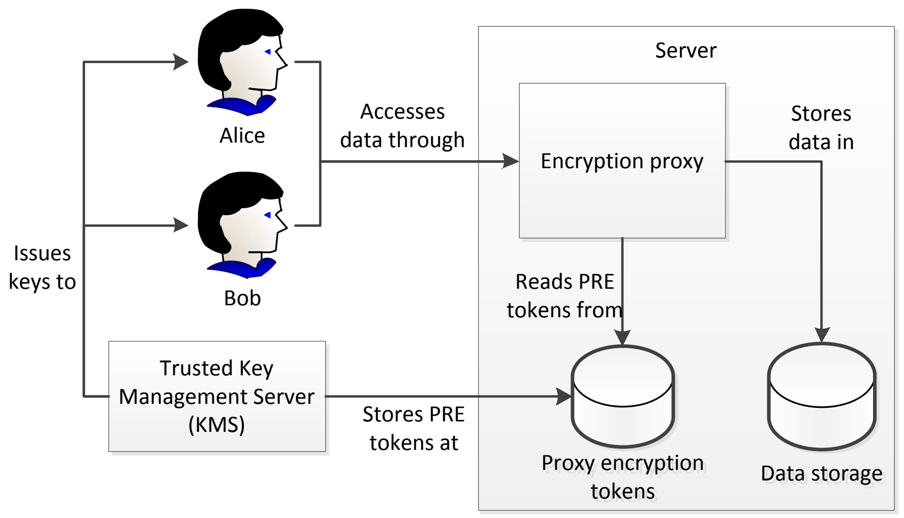


Figure 3.2: Overview of solution presented by Dong et al.

To begin with, the KMS generates a master key pair which is kept private by the KMS. When adding a user to the system, the KMS generates and issues a new key pair to the user. Additionally the KMS generates two PRE tokens.
When a user wishes to upload data, he associates the data with keywords and then encrypts the data and keywords using his public key. The user then transmits the resulting ciphertext to the server. The encryption proxy transforms the data and keywords to be encrypted by the master public key by using the token given by the KMS.
When a user wishes to download data, he uses his public key to encrypt the keyword to search for and transmits the search token to the server. The server transforms the search token to be encrypted by the master public key and performs the search. The identified data is transformed to the user-specific public key by using the second token generated by the KMS. Data is then transmitted to the user and decrypted using the user's private key. For an overview of the solution see Figure 3.2.
Because all users have distinct keys, a user can be efficiently revoked by removing the token used to transform ciphertext between the master public key and the user's public key. Because there are two tokens used for reading and writing data respectively, the solution also supports coarse-grained access control as removing one of them will prevent a user from either reading or writing data.
Overall the solution presented in [Dong&08] and in more detail in [Dong&11], which contains cryptographic proofs and implementation details, combines two unrelated cryptographic primitives to gain support for MR+MW while at the same time supporting efficient user revocation. This is achieved at the cost of introducing a trusted third party. There are, however, still unsolved issues such as support for integrity and fine-grained access control. Another significant issue with the solution is the inherent risk of collusion which is present in all known PRE schemes. To mitigate this the authors suggest splitting up the master key among several competing storage providers. However, this increases complexity and is not guaranteed to prevent collusion.

3.3.3 Comparison of related work

The two solutions approach the concept of secure data sharing in different ways. A traditional server is used in [Dong&08] while [Kamara&10] uses a cloud. By using a cloud [Kamara&10] improves availability and durability. Scalability is usually improved when using a cloud, but in the solution presented by [Kamara&10] all user authorisations are handled by the DO which limits scalability. The solution described by [Dong&08] does not suffer from this as the KMS in that solution can be offline while allowing users to access data.
The solution presented by [Kamara&10] has a significant limitation in that it does not support MR+MW. This is solved by [Dong&08] using PRE, but introduces the liability of potential collusion between the cloud and a user.
Both solutions provide confidentiality for data stored at a server or cloud, but neither support any kind of confidentiality for data stored locally at the user. Furthermore, neither solution supports non-repudiation, auditability, or access control delegation.
In the improved solution presented in [Kamara&11] data modification is supported, but this is possible only for the DO. Users are still unable to modify data without the help of the DO. In the solution by [Dong&08] only coarse-grained access control is possible. With regards to integrity, the solution presented by [Kamara&10] has built-in support for integrity checking, although only available to the DO. The solution by [Dong&08] does not support integrity checking. Both solutions support user addition and revocation and user-to-user anonymisation.

As Table 3.1 shows, the analysed proposals each have strengths and weaknesses. However, they are both lacking characteristics required to completely support many of the properties defined as goals in this thesis. A graphical presentation of the property rating can be seen in Figure 3.3. Overlapping points are circled for easy identification.

Table 3.1: Comparison of solutions proposed in related work.

Environment		Category	Property	Kamara	Dong
Cloud	User	Security	Data confidentiality	+	+
			Data integrity	++	-
	User		Data availability	++	+
			Non-repudiation	-	-
			Auditability	-	-
			User-to-user anonymisation	+++	+++
		Access control	User addition and revocation	+++	+++
	User		Fine-grained access control	-	+
			Access control delegation	-	-
			Many-to-many file sharing	++	+++
		Performance	Scalability	++	+++
			Durability	+++	++

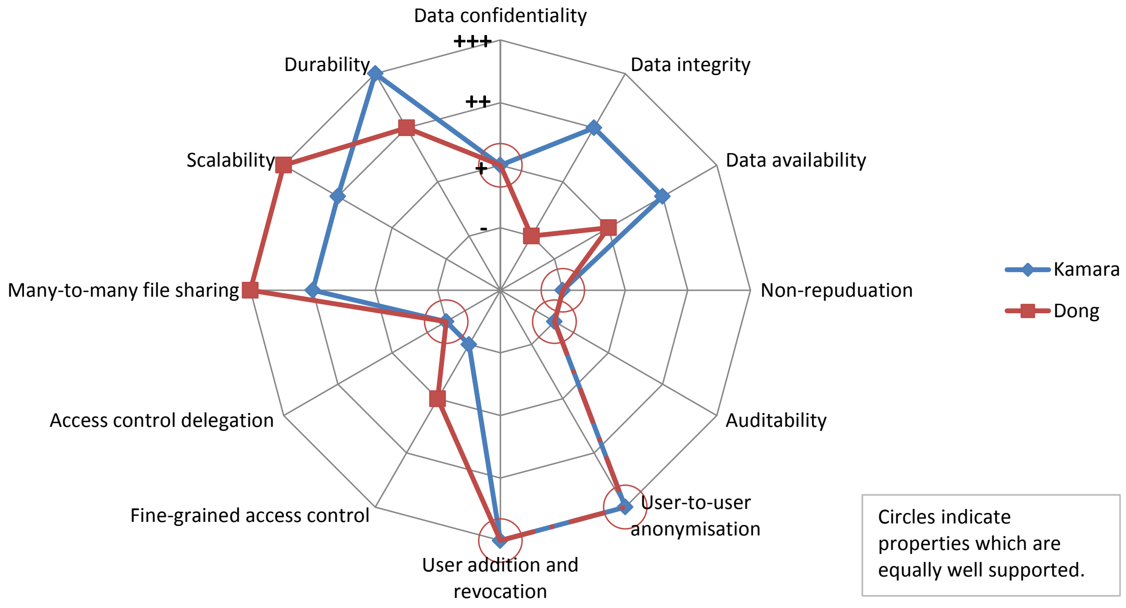


Figure 3.3: Graphical comparison of solutions proposed in related work.

3.4. Secure Dynamic Cloud-based Data-sharing

3.4.1 Overview

This section describes *Secure Dynamic Cloud-based Data-sharing (SDCD)* which is our conceptual design applying elements from the research and experiences of related work to form a dynamic collaboration environment as described in Section 1.3. An essential part of SDCD is based on cryptography and as with [Dong&08], PRE is combined with searchable encryption to attain support for MR+MW while being able to search for data based on keywords. Searchable encryption and PRE are explained in detail in Sections 2.3.3 and 2.3.4 respectively. This section starts by giving an overview of the conceptual design. Afterwards the terms important to SDCD are established and finally the dynamics of the conceptual design is described.

3.4.2 The conceptual design

SDCD uses ESE to enable searching on encrypted data as ESE is suitable to be combined with PRE. Combining these cryptographic primitives enables the conceptual design to have all shared data encrypted while at the same time supporting individual keys for users. Having individual keys for users makes it possible to efficiently revoke users.

The concept of a DO is central to SDCD. The DO is responsible for generating keys and PRE tokens for all users. This puts the DO in complete control of users having access to shared data. Although the DO is central to the system, users access data directly from the cloud and users depend only on the DO for being granted the initial access. Figure 3.4 gives a conceptual overview of SDCD.

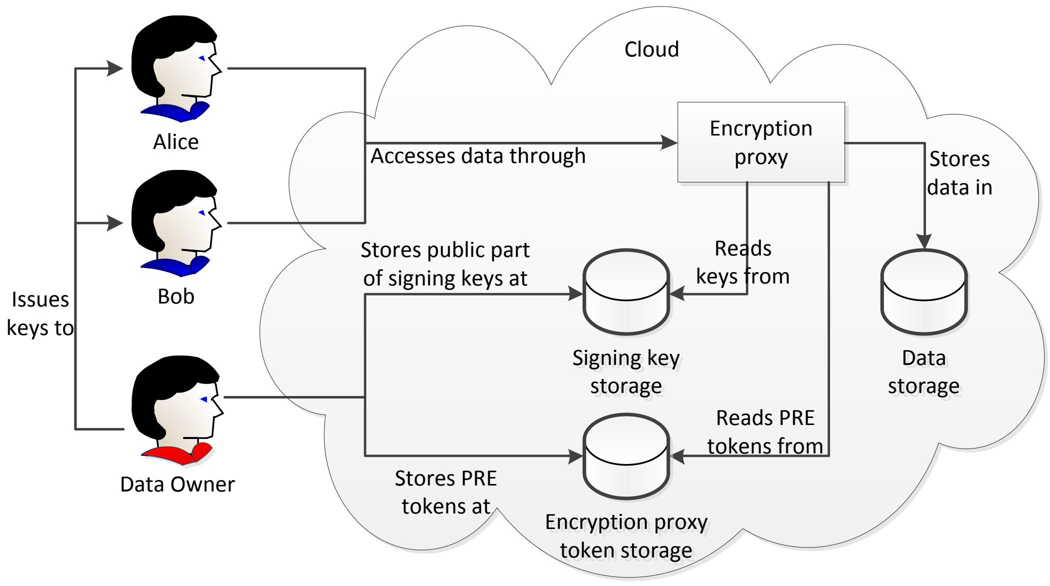


Figure 3.4: Conceptual overview of SDCD.

3.4.2.1 Data entity

In SDCD, a single unit of data is denoted a *data entity*. A data entity is essentially a wrapper for a resource that should be stored. It has an identifier, a non-empty set of attributes, and a payload. The identifier is randomly generated, but must be unique. The identifier is given to a data entity when the entity is created and remains unaltered throughout the lifetime of the entity. An attribute corresponds to a single keyword and contains two fields. The first is the encrypted keyword. The second is an identifier for the attribute which is deterministically generated from the unencrypted keyword. The payload is the resource that should be stored. To support hybrid encryption, the data entity also contains a field for the key used by a symmetric encryption scheme. Finally, each data entity contains the signature of the author. Table 3.2 gives an overview of a data entity.

Table 3.2: Overview of the fields in a data entity.

Field			Description
Identifier			Unique and immutable identifier.
Set of attributes	Attribute 1	Keyword 1	Set of attributes associated with the data entity. Each attribute is based on a single keyword and contains the encrypted keyword and an identifier which is deterministically generated from the unencrypted keyword.
		Identifier of attribute 1	
	$\vdots$	....	
		....	
	Attribute n	Keyword n	
		Identifier of attribute n	
Payload			The resource that should be stored.
Symmetric encryption key			The key used for hybrid encryption. The actual value depends on the applied scheme.
Signature			The author's signature of the whole data entity.

3.4.2.2 Keys and tokens

The DO owns a single *master key pair*. Future data entities stored in the cloud are encrypted using the master public key. Each user has two key pairs: a *PRE key pair* and a *signing key pair*. The PRE key pair allows a user to decrypt data entities received from the cloud. The corresponding *PRE token* is placed in the cloud.
The signing key pair is used by the user to sign data entities he creates or modifies. The signing public key is located in the cloud which uses it to verify integrity of data entities. All keys are generated by the DO and distributed to the user or cloud. In addition to the key pairs, each data entity has a unique symmetric encryption key. Finally, a *search token* is used when a user searches for data entities. Table 3.3 gives an overview of the keys and tokens used in SDCD.

Table 3.3: Overview of encryption keys and tokens used in SDCD.

Key or token	**Multiplicity**	**Availability**	**Purpose**
Master public key	One	The DO and all users	Encrypting data entities
Master private key	One	The DO	Generating PRE tokens
PRE public key	One per user	A single user	Generating PRE tokens
PRE private key	One per user	A single user	Decrypting data entities
Signing public key	One per user	The cloud and a user	Verifying integrity
Signing private key	One per user	A single user	Signing data entities
Symmetric encryption key	One per data entity	In a data entity	Efficient encryption of the payload of a data entity
PRE token	One per user	The cloud	Re-encrypting data entities
Search token	One for each search	The cloud and a user	Identifies data entities matching a keyword

3.4.2.3 Dynamics

Data entity communication When a user uploads a data entity to the dynamic collaboration environment, the data entity is encrypted in the user's environment. When the data entity subsequently is accessed, it is re-encrypted in the cloud and decrypted at the receiving user's environment. This means that the data entity is never available in the cloud as cleartext. This scenario is depicted in Figure 3.5. In the figure, the steps are:

1. The user, Alice, wishes to upload a data entity and prepares the data entity by encrypting it with the master public key.
2. Alice uploads the data entity to the cloud.
3. The cloud uses Bob's PRE token to re-encrypt the data entity.
4. The re-encrypted data entity is sent from the cloud to Bob.
5. Bob decrypts the data entity using his own private key.

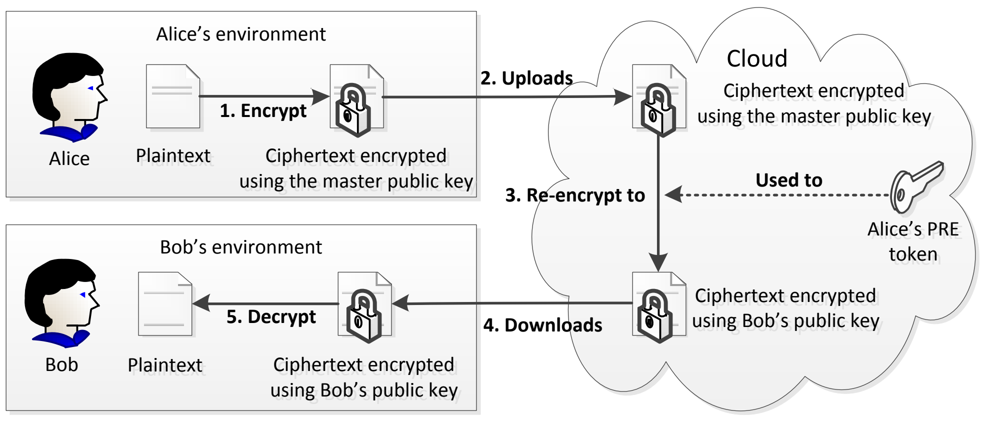

Alice's environment
Alice
Plaintext
1. Encrypt
Ciphertext encrypted using the master public key
2. Uploads
Cloud
Ciphertext encrypted using the master public key
3. Re-encrypt to
Used to
Alice's PRE token
Ciphertext encrypted using Bob's public key
4. Downloads
Bob's environment
Bob
Plaintext
5. Decrypt
Ciphertext encrypted using Bob's public key

Figure 3.5: Overview of how data flows in SDCD.

Operations The conceptual design offers support for a range of operations. These are described below and some of them are supported by sequence diagrams, showing the flow of the operation.

System initialisation: The DO generates a master key pair and permanently takes the role of DO by initialising the system.

User addition: When the DO decides to add a user to the system, the DO generates a new PRE key pair and a new signing key pair for the user. The DO also generates a single uni-directional PRE token. The DO uploads the PRE token and the public key of the signing key pair to the cloud. The DO gives the generated PRE key pair, signing key pair and the DO public key to the user. Figure 3.6 shows the process of adding a user.

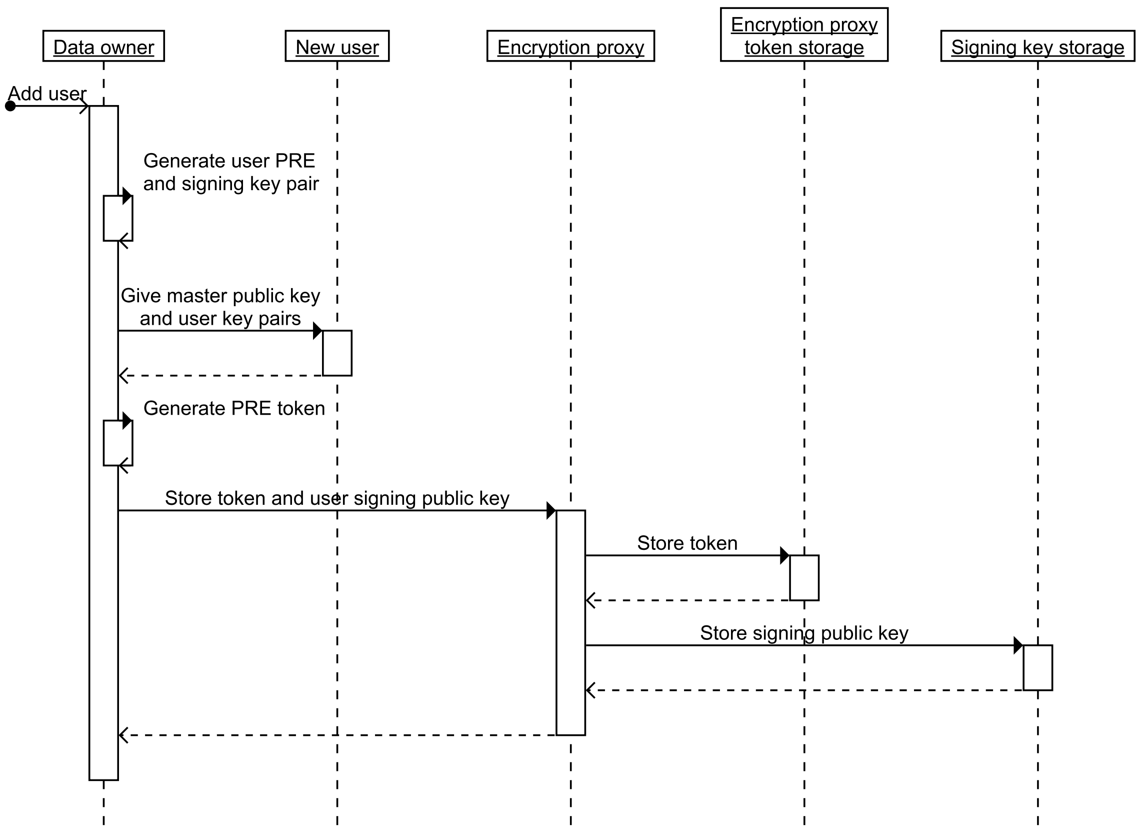

Data owner
New user
Encryption proxy
Encryption proxy token storage
Signing key storage
Add user
Generate user PRE and signing key pair
Give master public key and user key pairs
Generate PRE token
Store token and user signing public key
Store token
Store signing public key

Figure 3.6: Dynamics of how the data owner adds a user in SDCD.

User revocation: When the DO decides to revoke a user, the DO removes the user's PRE token from the cloud. The user's signing key is kept for later verification of integrity. Figure 3.7 shows how user revocation is performed.

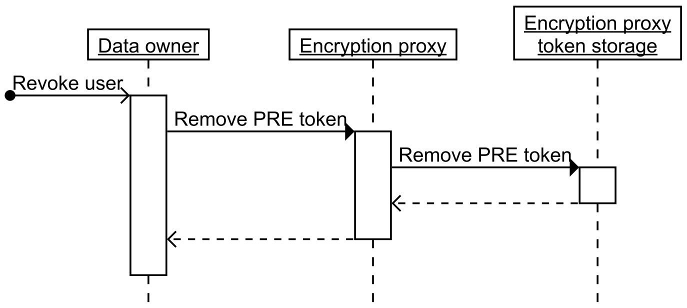


Figure 3.7: Dynamics of how the data owner revokes a user in SDCD.

Data entity creation: To share resources with others, a user selects a resource and associates it with at least one keyword. The resource is then wrapped in a data entity which is assigned a randomly generated identifier and a randomly generated symmetric key. Identifiers are generated for all attributes. The payload and keywords inside attributes are encrypted using the symmetric key which is then encrypted using the master public key. The user then signs the data entity using his private signing key.

The data entity is uploaded to the cloud which checks that the user is authorised to upload data by verifying the existence of the PRE token for the user. Finally, the cloud stores the data entity. For an overview of data entity creation, see Figure 3.8.

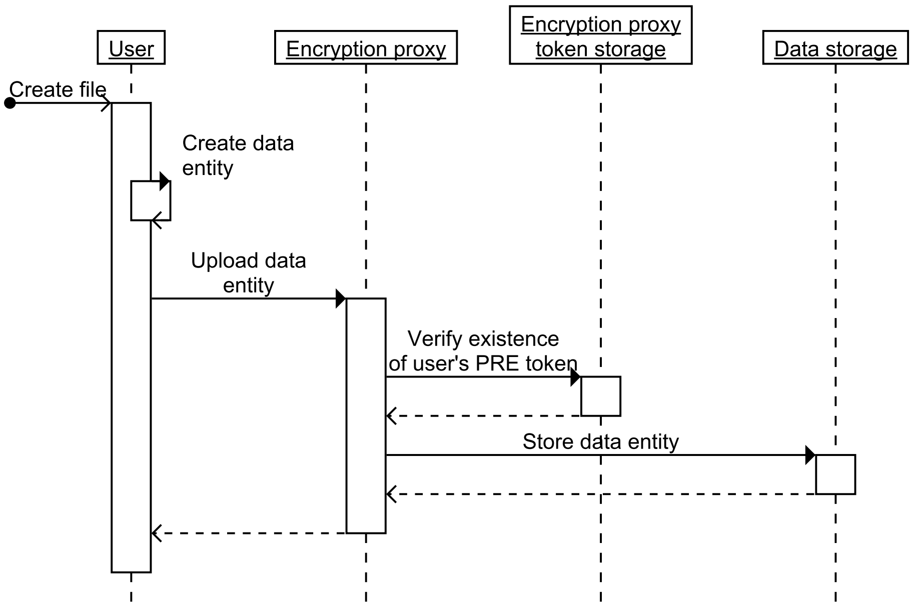


Figure 3.8: Dynamics of how a user creates a data entity in SDCD.

Data entity modification: Modification of a data entity is performed much like data entity creation. The only difference is that the identifier of an existing data entity indicates the entity to modify. When modifying a data entity, the modifying user becomes the author of the data entity.

Data entity deletion: To delete a data entity, a user uses the identifier of an existing data entity. The identifier is then sent to the cloud and indicates the data entity to delete.

Data entity searching: Data entity searching is based on a single keyword and performed by a user. An identifier is generated from the keyword to create a search token which is sent to the cloud. The cloud finds data entities by matching the search token to identifiers of attributes for all stored data entities. The cloud re-encrypts the found data entities by using the PRE token for the user performing the search. The found data entities are then returned to the user who decrypts the received data entities by using his private PRE key. The dynamics of data searching can be seen in Figure 3.9.

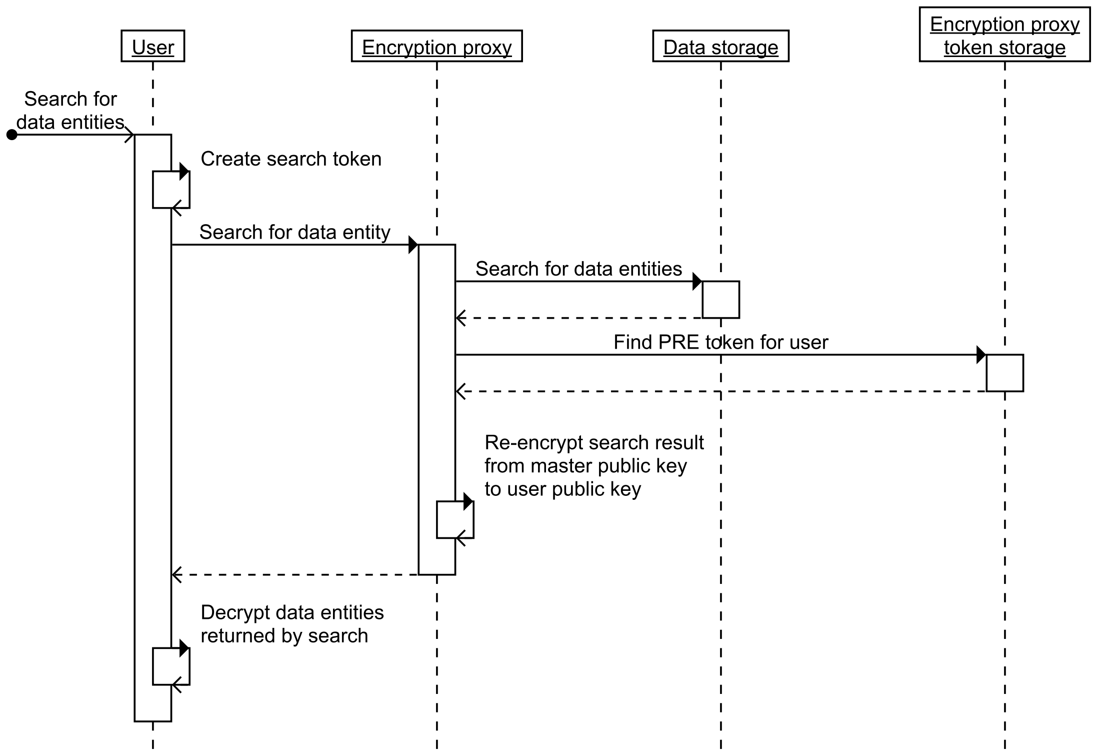


Figure 3.9: Dynamics of data entity search in SDCD.

Data entity integrity verification: Any authenticated user can at any point in time request the cloud to verify the integrity of any data entity. The cloud verifies the integrity by using the public signing key of the user who is the author of the data entity.

Data entity author authenticity verification: The DO can use the cloud to determine and verify the identity of the author of a data entity. The cloud knows the identity of the user creating a data entity as the user was authenticated, but to prove the identity, the cloud performs an integrity verification using the public signing key of the user. If the integrity verification succeeds, the authoring user cannot repudiate creating or modifying the data entity.

3.4.2.4 Verifying Integrity

To enable the cloud to verify the integrity of data entities, SDCD applies *digital signing*. A data entity is signed by its author so the integrity can be verified in the future. The integrity verification scheme makes it possible for any authenticated user to request the cloud to verify the integrity of any data entity, independent of who the author was. This is accomplished while keeping all users anonymous to each other. To verify integrity, digital signing is used resulting in added support for *non-repudiation*.

The cloud must retain all public keys used for verifying signatures, even after a user has been revoked. This remains the case as long as a user is the author of any data entity. Furthermore, a user may have multiple signing keys if he has been added, revoked, and added again. However, from the user's perspective, only a single signing key exists as he is unaware of the multiple signing keys kept by the cloud.

3.5. Discussion

SDCD combines searchable encryption and PRE in a way similar to [Dong&08] to ensure data confidentiality while supporting many-to-many resource sharing and being able to add and revoke users. SDCD further applies digital signing to obtain support for integrity verification and non-repudiation while hybrid encryption is applied to improve performance.
Users communicate directly with the cloud. This ensures availability and scalability as data sharing only depends on the cloud. A potential bottleneck is that only the DO is capable of granting and revoking access for users. User additions and revocations are less frequent events, compared to data entity creation and search. In large scale deployments, it is nonetheless a potential problem. SDCD provides only coarse-grained access control: either a user has no access or complete read- and write-access. Another limitation is the focus on securing data inside the cloud which means that data security is not ensured when data is located locally in the user's environment. SDCD does not require users to know each other and no information provided by SDCD helps to determine the identity of another user. As such, SDCD supports scenarios where all users are anonymous towards each other. Only the DO needs to know the identity of users. This prevents collusion between users.
When the DO decides to add a user to the system, the DO must authenticate the new user and the user's newly generated key pairs must be transferred in a secure manner. Furthermore, everyone communicating with the cloud must be authenticated. None of these problems are addressed by SDCD which requires other means of secure key distribution and authentication in general.

3.5.1 Comparison to related work

SDCD is designed to work in a cloud context which [Dong&08] is not. The cloud context increases availability, scalability, and durability. When users are added to the system, no further interaction with the DO is required. This enables users to share data entities even if the DO is not available.
Another important difference between SDCD and [Dong&08], is that SDCD does *not* re-encrypt data when it is uploaded to the cloud. Instead the master public key is used directly by users. The disadvantage to this approach is that read and write access cannot be distinguished. An authorised user has either full read and write access or no access. A benefit of this change is reduced complexity and improved performance in the cloud when data is uploaded. Another important benefit is that digital signing can be applied. Because re-encryption changes the binary layout of a data entity the approach presented by [Dong&08] would invalidate the signature created by a user.
By using PRE, SDCD introduces the risk of collusion between the cloud and a user similar to that of [Dong&08]. However, the benefit of using PRE is support for many-to-many resource sharing which outweighs this concern. The solution presented by [Kamara&10] does not share the concern for collusion, but on the other hand does not support many-to-many file sharing.

3.5.2 Comparison to thesis goals

SDCD supports many of the properties set as goals in this thesis. However, notably missing from SDCD is support for auditability, access control delegation, and security in general for resources in the users local environment. Support for this can be achieved and this is described in the following chapters.
Table 3.4 summarises and compares SDCD and the related work analysed earlier in this chapter to the thesis goals. Figure 3.10 is a graphical representation of the same comparison and a quick glance at the figure shows that we exceed the schemes of both [Kamara&10] and [Dong&08] as judged against the defined properties.

Table 3.4: Comparison of SDCD and solutions proposed in related work.

Environment		**Category**	**Property**	**Kamara**	**Dong**	**SDCD**
Cloud	User	Security	Data confidentiality	+	+	+
			Data integrity	++	-	+++
	User		Data availability	++	+	++
			Non-repudiation	-	-	+++
			Auditability	-	-	-
			User-to-user anonymisation	+++	+++	+++
		Access control	User addition and revocation	+++	+++	+++
	User		Fine-grained access control	-	+	+
			Access control delegation	-	-	-
			Many-to-many file sharing	++	+++	+++
		Performance	Scalability	++	+++	+++
			Durability	+++	++	+++

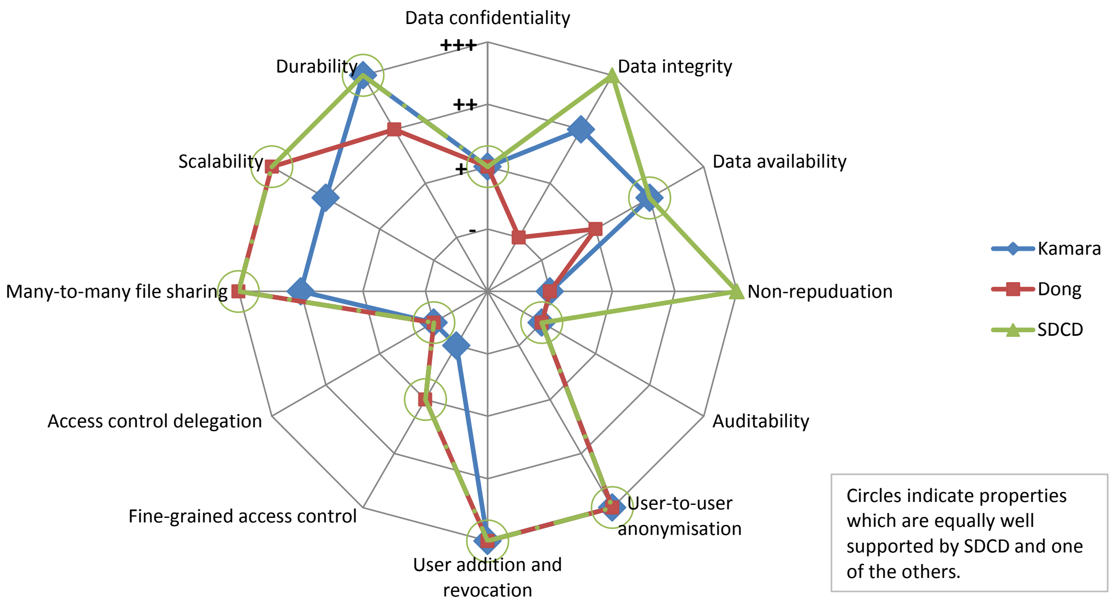


Figure 3.10: Graphical comparison of SDCD and solutions proposed in related work.

Chapter 4

Data access control

This chapter presents a conceptual design that achieves fine-grained data access and provides a manageable access control scheme, while allowing authorised users to delegate granular access control to other users. These are properties defined as thesis goals in Chapter 1. This chapter uses some of the concepts and technologies described in Chapter 2 and builds upon the design presented in Chapter 3. Chapter 5 further improves upon the design presented in this chapter and Chapter 6 validates the design.

4.1. Introduction

Fine-grained data access control is essential in any system where data is shared among multiple users with different levels of trust. To ensure data security, highly trusted users may be allowed full access while others are allowed only partial access to the shared data. In a system where users with different access rights exist, effective management of these rights becomes important.
The assumptions made for this chapter are described in Section 4.2. Related work is described in Section 4.3 and Section 4.4 describes a conceptual design for achieving fine-grained access. Finally, the design is discussed and compared to the related work and the thesis goals in Section 4.5.

4.2. Assumptions

This chapter assumes an existing data sharing design to build upon, and that this design has properties similar to the design presented in Chapter 3. The data sharing design is assumed to support many-to-many data sharing in a secure manner including support for confidentiality, integrity, availability, and non-repudiation. It is also assumed that an authentication mechanism is available.

4.3. Related work

As described in Section 2.4.1, several different access control models exist. This section describes two access control models which are relevant to this chapter.

4.3.1 Role-Based Access Control

Role-Based Access Control (RBAC) is a common scheme used to control data access. The main purpose of this model is to group permissions into entities that can be reused and assigned to users with similar needs. The general concept is described in Section 2.4.1.3. To facilitate use of this scheme, a National Institute of Standards and Technology (NIST) RBAC standard has been proposed. The purpose of the standard is to provide a mutual common ground for enabling and facilitating collaboration across organisations. From a consumer's viewpoint a benefit of the standard is the ability to compare different RBAC solutions, assuming they all follow the NIST standard [Ferraiolo&01]. The standard proposes a reference model used to define the terminology used in the context of RBAC. Interoperability between RBAC solutions that follow the standard is also increased, if not directly by the standard, then through potential Application Programming Interfaces (API) based on the standard.
The basic NIST RBAC functionality is called RBAC Core and describes the minimal requirements needed to conform to the standard. The RBAC Core requires implementation of roles, users, and permission. It also requires support for a many-to-many relation between roles and permissions and a many-to-many relation between roles and users. Another requirement is the ability to look-up the roles assigned to a user and vice versa. Similarly, the ability to look-up the permissions of a specific role and the roles having a specific permission is required. The requirements also state that users must be able to exercise multiple roles simultaneously [Ferraiolo&01]. An overview of the entities in the core part of the NIST RBAC standard can be seen in the Entity-Relationship diagram shown in Figure 4.1.

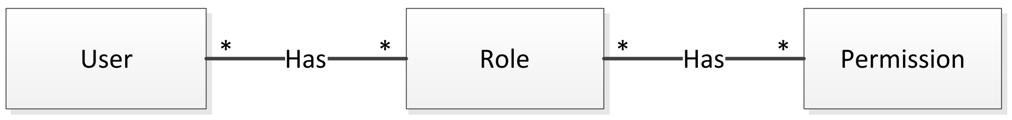


Figure 4.1: Core entities and their relations in the NIST RBAC standard.

In addition to the core aspects of RBAC, the NIST RBAC standard describes packages. Figure 4.2 depicts these packages, their purpose, and options for each package. Each package can be optionally selected, with exception of the mandatory Core RBAC package, and added to the final requirement package defining the access control model. The purpose of this package selection is to ease customisation of the RBAC to better support different usage scenarios and environments with different requirements [Ferraiolo&01].

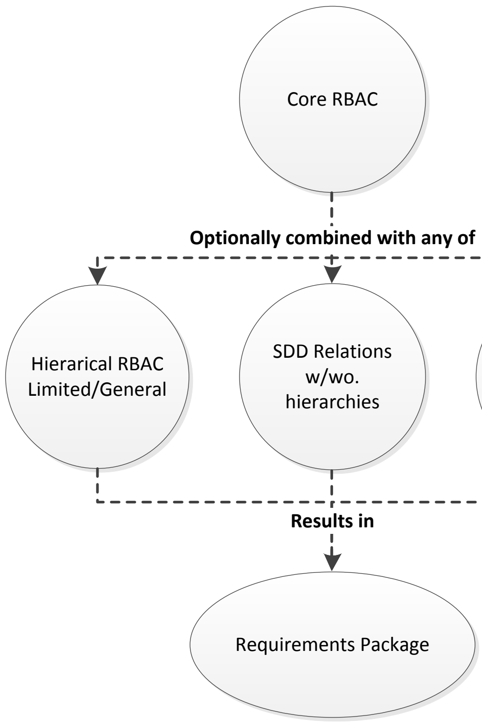

Core RBAC
Optionally combined with any of
Hierarical RBAC Limited/General
SDD Relations w/wo. hierarchies
DSD Relations
Results in
Requirements Package

Figure 4.2: Overview of NIST RBAC standard.

Hierarchical RBAC: The Hierarchical RBAC describes the requirements needed for supporting definition of role hierarchies. These can be defined either as *General Hierarchical RBAC*, supporting arbitrary hierarchies and allowing multiple inheritance, or *Limited Hierarchical RBAC*, limiting the hierarchy to tree-structures.

Static Separation of Duty: The purpose of Static Separation of Duty (SSD) relations is to enforce constraints, imposed on roles, in such a way that users cannot be assigned roles that conflict with the ones they are already assigned. For instance, if roles have conflicting permissions or negate one another. The SDD can be used in combination with a hierarchical RBAC structure, where constraint enforcement is performed on all inherited roles in the role hierarchy, when attempting to assign new roles to a user.

Dynamic Separation of Duty: Where SSD relations limit the roles at the time of assignment, Dynamic Separation of Duty (DSD) relations, which are variants of SSD relations, allow users to have potentially conflicting roles assigned, as long as the permissions that conflict are not activated at the same time.

The proposed standard enables interoperability and common understanding among solutions following this standard. The standard can be implemented at different levels which allows for customisation.
To enable automatic role assignment to individual users, a modified version of the original RBAC model has been proposed [Al-Kahtani&02]. This Rule-Based Role-Based Access Control (RB-RBAC) model automatically assigns permissions to users based on a set of attributes each individual user possesses. Rules are created to analyse these attributes and assign permissions to

the user accordingly. This RB-RBAC model uses *Seniority Levels* to enable comparison of rules which makes it easier to automatically assign permissions. It is also used to ensure rules follow a predefined role hierarchy and does not violate the constraints associated with this hierarchy.
A dedicated language is used, alongside the model, to express rules and related constraints. The primary benefit achieved through this automatic permission assignment is the dynamic role-permission assignment that allows large organisations to automatically assign roles to users. It is, however, important that the rules are defined properly to avoid accidentally assigning permissions to the wrong users.

4.3.2 Distributed Access Control in Clouds

An approach to access control in the cloud, different from the previous RBAC approaches, uses Attribute-Based Encryption (ABE) with Key Distribution Centres (KDC) [Ruj&11]. This scheme is called Distributed Access Control in Clouds (DACC) and allows data sharing and storage in a cloud environment, while the cloud is not able to access the data in unencrypted form. In this scheme, the cloud environment is assumed to be honest but curious. DACC claims to be collusion-proof in the sense that users cannot combine their decryption keys to access data that none of them are allowed to access. In addition, DACC also offers a user revocation scheme that allows revocation of users without having to redistribute cryptographic keys to its remaining users.

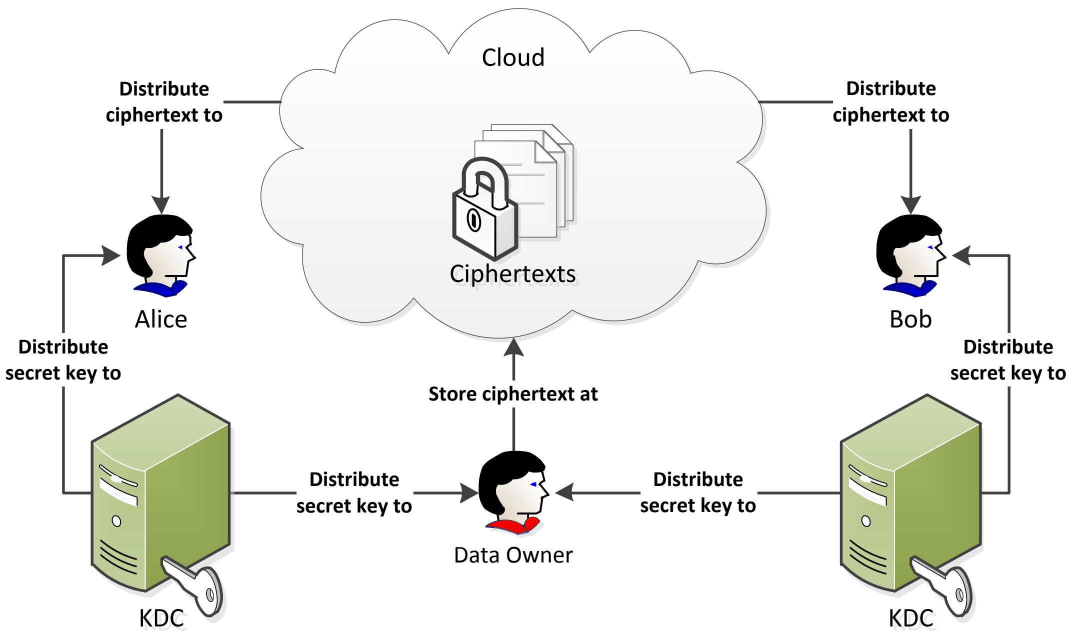


Figure 4.3: Overview of the DACC model.

The DACC model is depicted in Figure 4.3. The model allows data owners to place encrypted data in the cloud environment and define a set of attributes associated with the data. These attributes are assigned to users and determines what data they are able to access, i.e. a user gets the same attributes assigned as the data he has access to. The KDCs generate *secret keys* for users based on the individual user's attributes. These keys are used to decrypt the encrypted data users request from the cloud. To further increase data confidentiality, a secure protocol is used to transfer secret

keys from the KDC to users. The same protocol is used when the cloud transfers encrypted data to the users [Ruj&11].
The user revocation scheme used in the DACC model seems infeasible for general purpose usage scenarios. Whenever a user has his access revoked, the attributes he has been previously assigned are noted and data owners that own data containing one or more of these specific attributes are notified. These data owners are then responsible for assigning new attributes to their data and notifying remaining users with access to their data that an updated version is available [Ruj&11]. This scheme does not require reassignment of decryption keys, but a potential re-encryption of data is required along with other users being required to request the updated data from the cloud. Also, if user revocation is performed on a regular basis, the list of revoked attributes could potentially become very long. This requires data owners to use attributes different from the ones already revoked.

4.3.3 Comparison of related work

This section presents an overview of the related work by rating the properties of each analysed solution to the thesis goals according to the levels described in Table 1.2. The DACC scheme, presented by [Ruj&11], and the NIST RBAC scheme, presented by [Ferraiolo&01], approach access control in different ways.
As the NIST RBAC scheme is a standard and not an actual solution, the scheme is not applicable to most of the properties defined as goals in this thesis. Similarly, some of the properties are not considered by DACC.
DACC offers a different approach to access control than the ordinary role-based models. The attribute encryption scheme allows a more fine-grained access model than the role-user assignment used in RBAC. However, the user revocation approach in the DACC model is infeasible and cannot be used effectively in the context of generic usage scenarios. As user revocation impacts existing users in the DACC scheme, the rating is reduced accordingly. Another inconvenience concerning DACC is the use of off-cloud KDCs. The overall scalability is decreased as users need to interact with the KDCs to receive secret keys and attributes and the KDCs do not benefit from a cloud deployment. However, DACC benefits from the availability and durability of the cloud. A comparison between DACC and the NIST RBAC standard can be seen in Table 4.1.

Table 4.1: Comparison of models presented in related work.

Environment		Category	Property	NIST RBAC	DACC
Cloud	User	Security	Data confidentiality	N/A	+
			Data integrity	N/A	N/A
	User		Data availability	N/A	++
			Non-repudiation	N/A	N/A
			Auditability	N/A	N/A
			User-to-user anonymisation	N/A	N/A
		Access control	User addition and revocation	N/A	++
	User		Fine-grained access control	++	++
			Access control delegation	N/A	N/A
			Many-to-many file sharing	N/A	++
		Performance	Scalability	N/A	++*
			Durability	N/A	+++

* The DACC uses KDCs for user authorisation.

4.4. Cloud-based Hierarchical Access Control

4.4.1 Overview

This section describes *Cloud-based Hierarchical Access Control (CHAC)* which is our design for an access control model suitable for a dynamic collaboration environment. CHAC builds upon SDCD and adds support for fine-grained access control, access control delegation, and the ability to audit stored data. CHAC is partly inspired by the research presented in Section 4.3. CHAC will be presented in a top-down manner by first establishing key concepts and terminology and then defining the structure of the access control model. Finally, details regarding the management of entities in the access control model as well as auditability are described.

4.4.2 Access control

CHAC uses a Role-Based Access Control (RBAC) model to manage permissions that are used to access data stored in a cloud-based storage. The permissions are mapped onto roles that are assigned to users. The terms used in CHAC are as follows:

User: A user seen in the context of CHAC is a person or entity authorised to interact with the dynamic collaboration environment through one or more assigned roles. When a user is created, he is assigned a role granting him access to the dynamic collaboration environment. Users with no assigned roles have no access. A user may have multiple roles assigned.

Role: A role seen in the context of CHAC is an entity governing a user's allowable actions. Table 4.2 shows the definition of a role. A role has an identifier, a name, a number of permissions, and two flags indicating the type and status of the role. Furthermore a role grants access to a set of identifiers. The identifier uniquely identifies the role.

The permissions of a role grant users the right to access data entities and perform actions on these or to perform administrative actions. The flags indicate if the role is a normal role or a root role and if the role is active. A single role can be assigned to multiple users.

Root Role: A root role is a special kind of role that is invisible to other roles. This allows for creation of independent collaboration environments. A root role is the topmost role of a hierarchy. All of the actions for managing roles, explained later, do not apply to root roles.

Permission: A permission governs the ability to perform an action. Table 4.2 shows the permissions defined in CHAC. Users are granted permissions through role assignment.

Data Entity: A data entity as defined in Section 3.4.2 represent and encapsulates data that is to be shared with others using the dynamic collaboration environment. Access to a data entity is protected by permissions of roles and changes made to a data entity is visible to all users permitted to access the data entity.

Private Hierarchy: Private hierarchies can be created to have co-existing, independent role hierarchies. A Private hierarchy is managed by the root role located at the top of this hierarchy.

Table 4.2: Definition of a role and its fields.

Field	Description
Identifier	Unique and immutable identifier.
Name	The name of the role.
Manage sub-roles	Permission to create sub-role with subset of the current role's permissions and update and deleting existing sub-roles.
Create private hierarchy	Permission to create private hierarchies and adding a root role as the top-most role. The newly created root role is assigned to the user performing the action.
Assign and unassign roles	Permission to assign and unassign sub-roles to users.
Add users	Permission to create a new user and assign a role to him.
Set of data entity identifiers	Set of data entity identifiers which indicates the data entities that this role grants access to.
Data entity permission level	Indicates whether the role grants read-only or read and write access to data entities. Write access includes permission to create, update, and delete data entities.
Active flag	Indicates if the role is active or inactive. Inactive roles are suspended and cannot be used to access anything.
Root flag	Indicates if the role is a root role. If it is not a root role it is considered a normal role.

An overview of the conceptual structure of the CHAC design for accessing data can be seen in Figure 4.4.

1. First a user, Alice, requests a data entity from the cloud.
2. The access control manager validates the request against the roles that Alice is currently assigned. If Alice is authorised, the access control manager fetches the data entity from the data storage.
3. The data entity is then passed to the encryption proxy to be re-encrypted.
4. The encryption proxy fetches Alice's Proxy Re-Encryption (PRE) token from the encryption proxy token storage and re-encrypts the data entity to match her key pair.
5. The re-encrypted data entity is returned to the access control manager.
6. Finally, the access control manager returns the requested data entity to Alice.

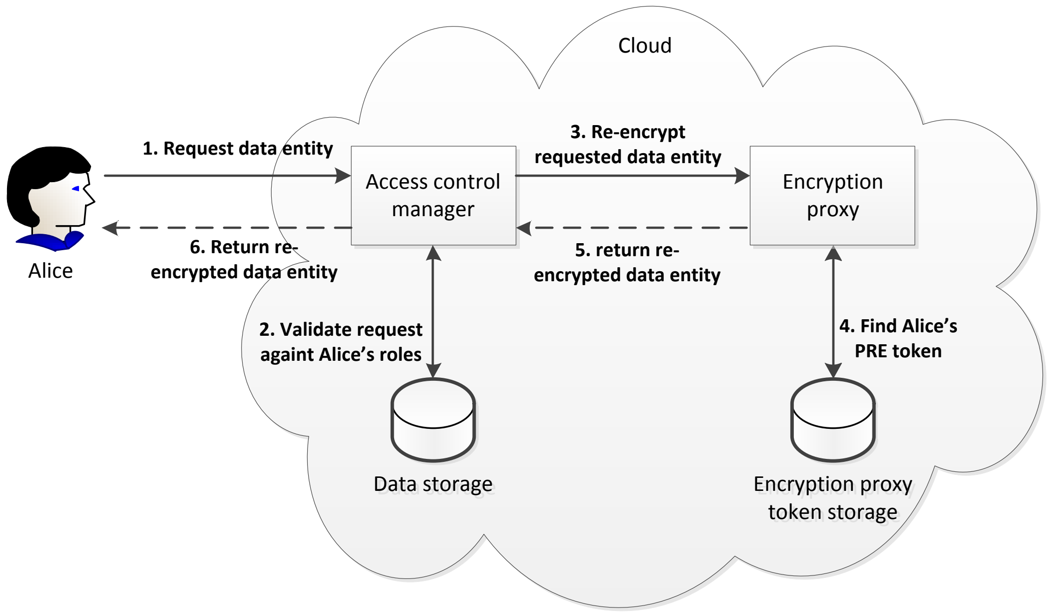
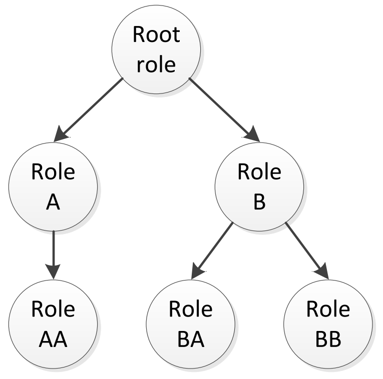

Cloud
1. Request data entity
Access control manager
3. Re-encrypt requested data entity
Encryption proxy
Alice
6. Return re-encrypted data entity
5. return re-encrypted data entity
2. Validate request againt Alice's roles
4. Find Alice's PRE token
Data storage
Encryption proxy token storage

Figure 4.4: Overview of the conceptual design of CHAC.

4.4.2.1 Access control structure

Role structure The roles in CHAC are structured as a hierarchy, more specifically a tree-structure. Figure 4.5 shows how roles are associated with each other. The permissions of each role is a subset of its parent's permissions. This structure allows inheritance of permissions while at the same time preventing child nodes from ever having permissions not held by the parent. The role at the root of this hierarchy is called a root role. When initialising the dynamic collaboration environment only a single root role exists, this role is assigned to the Data Owner (DO). The DO is responsible for creating the hierarchy by defining new roles. The creation of lower levels of the role hierarchy can then be delegated to the newly created roles. An example of this is shown in

Figure 4.5 where the root role creates Role A and Role B, then Role A can create Role AA and Role B can create Role BA and Role BB.



Role Permissions:
Role AA $\subseteq$ Role A $\subseteq$ DO
Role BA $\subseteq$ Role B $\subseteq$ DO
Role BB $\subseteq$ Role B $\subseteq$ DO

Figure 4.5: Overview of the role hierarchy.

To preserve user-to-user anonymisation, CHAC has rules determining visibility of roles and users. A role can, given the right permission, be used to see and manage all roles further down the tree on the same branch. That is, the role's sub-roles and their sub-roles, recursively. This also entails that a role can be managed by its parent roles, recursively. Thus, root roles can see and manage all roles and users assigned to those roles in the tree. For any sub-role that can be seen, the users assigned to the role can also be seen. Those rules apply to all normal roles. A root role cannot be managed by any role other than itself and is always invisible to all other roles.
A normal role cannot manage itself, only sub-roles can be managed. Thus, a user assigned to a role cannot assign that role to others. However, assuming a user has permission to manage sub-roles and assign sub-roles, he can create a sub-role with identical permissions to his own role and then assign this role to a user. An important consequence of this is that the user remains in control. He can see and manage the sub-role and unassign users again if desired. Root roles behave differently and can be used to manage themselves. This enables a user who is assigned to a root role to assign and unassign other users to the role and to delete the role.
The visibility rules can be seen in Figure 4.6 which illustrates role and user visibility from the perspective of Alice. Users and roles with dashed lines are invisible to her while users and roles with solid lines are visible. The users and roles invisible to Alice are also indicated by the dashed square surrounding them. In the figure, Alice can see her own role and its sub-role, role BB. She can also see Carol because Carol is assigned to role BB. However, she cannot see Bob, parent roles of her role, other branches of the hierarchy, or users assigned to those parts. In that same figure, the data owner can see everything.

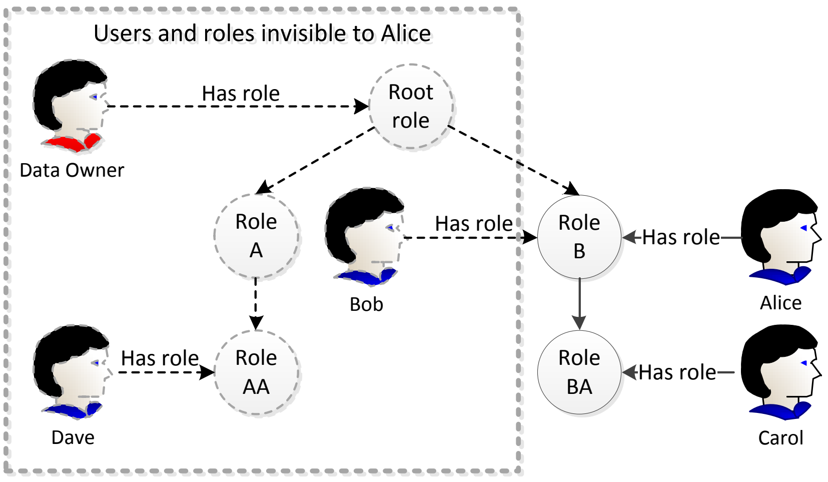


Figure 4.6: Example of the visibility rules in CHAC.

User structure The users in CHAC are kept in a *pool* of users. A user can be assigned roles, which govern the permissions the user has. Figure 4.7 shows a possible user-role relationship as a graph, where users are assigned to roles. Notice that users may have multiple roles assigned on multiple branches within the role hierarchy.

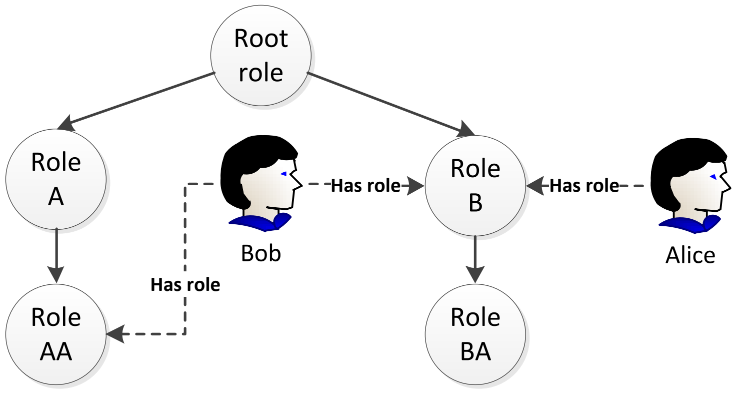


Figure 4.7: User-role relationship, with a user being assigned more than one role.

When data is submitted to the cloud it is encrypted with the DO's key pair and stored in the cloud. However, in order to retrieve and read data from the cloud, the data needs to be encrypted with the retrieving user's key pair for him to be able to decrypt it. Therefore, when retrieved, it must be re-encrypted. This results in users having an indirect hierarchical relationship in terms of their cryptographic key pairs.
In the scenario where more users are subsequently added, data is re-encrypted multiple times. Figure 4.8 shows how this affects the re-encryption process with n users. In the figure, Alice has added Bob who subsequently added another user ending with user n being added. When user n requests data, the data is first re-encrypted from being encrypted by the master public key to

being encrypted by Bob's public key, using Alice's PRE token. Then it is re-encrypted to being encrypted by the next user's public key, using the corresponding PRE token, until it is encrypted with user n's public key, enabling him to decrypt the data.

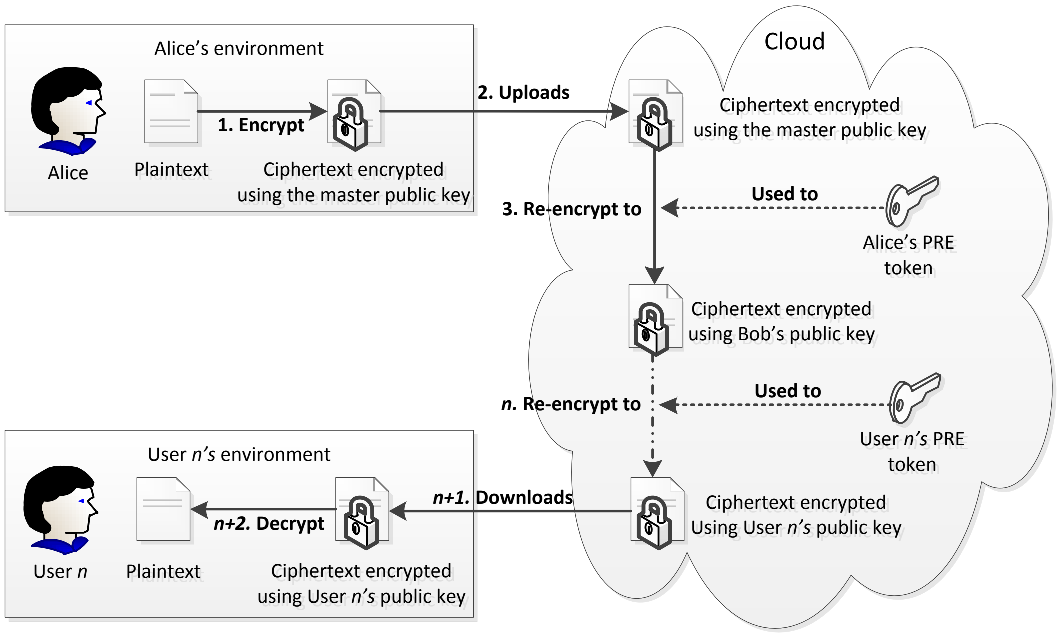

Alice's environment
Alice
Plaintext
1. Encrypt
Ciphertext encrypted using the master public key
2. Uploads
Cloud
Ciphertext encrypted using the master public key
3. Re-encrypt to
Used to
Alice's PRE token
Ciphertext encrypted using Bob's public key
n. Re-encrypt to
Used to
User n's PRE token
User n's environment
User n
Plaintext
n+2. Decrypt
Ciphertext encrypted using User n's public key
n+1. Downloads
Ciphertext encrypted Using User n's public key

Figure 4.8: Re-encryption flow with n users.

4.4.2.2 Role management

When managing normal roles, several actions are available to users with sufficient permissions. The actions described here do not apply to root roles as they are invisible. The actions for normal roles are the following:

Create: When creating a new role, the user performing the action must choose which of the roles assigned to him, or a sub-role of those roles, to base the new role on. The permissions of the new role must be a subset of the chosen role and the new role will become a child of the chosen role. The newly created role can later be assigned to users.

Update: A role can be updated by all its parents, recursively, provided it has the required permissions. If a role has its permissions altered and some permissions are removed, these changes are cascaded down the role hierarchy. This is done to ensure that the permissions of all child roles remain a subset of their parent's permissions. If a role had previously been permitted to create child roles and this permission is revoked, the existing created child roles are preserved. If a role is given wider permissions, this is not cascaded to sub-roles. Figure 4.9 shows how a role is updated and how the changes made to its permissions are cascaded to its child roles:

1. The figure shows the roles with associated permissions before an update is performed. Role A is subject to a role update.
2. The permissions to be removed are indicated using red text colour. The cascading effect of this update is shown with orange text colour in Role AB.
3. Finally, the result of the update operation is shown.

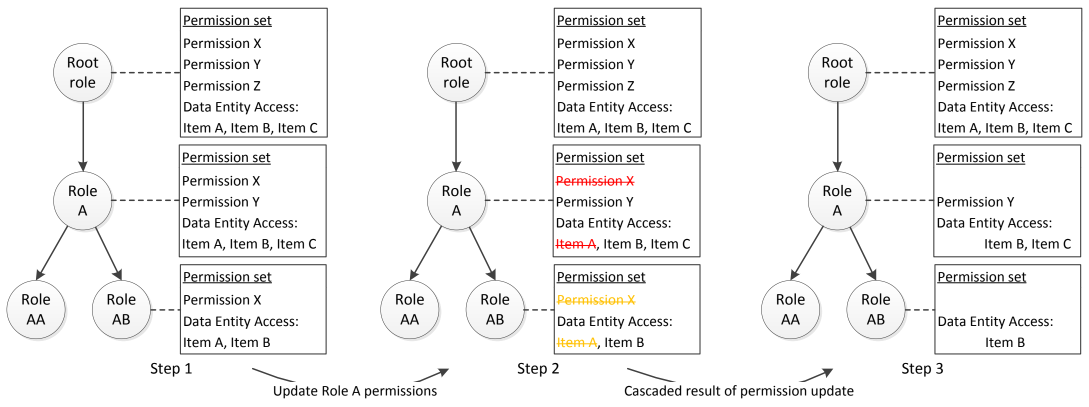


Figure 4.9: Flow of update role operation.

Delete: A role can be deleted only by the parent of the role or roles higher up in the role hierarchy. If the role being deleted has child roles, these roles are attached to the parent of the role being deleted. This scenario is shown in Figure 4.10:

1. Here the role hierarchy is shown before the delete operation is performed.
2. Here Role B is marked for deletion showing the affected relationships.
3. This last step shows the updated relationships where Role BA and Role BB are now direct children of the parent of Role B which in this case is the root role. Role B is completely removed from the hierarchy.

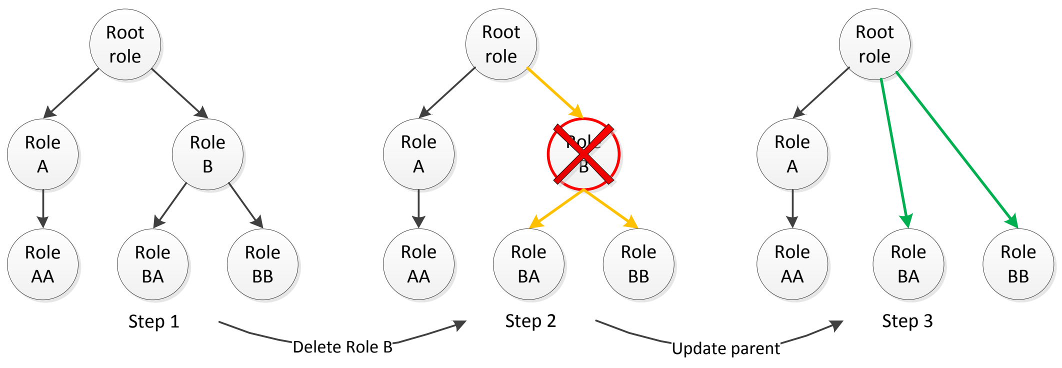


Figure 4.10: Flow of delete role operation.

Assign/Unassign: Roles can only be assigned to existing users. When a new user is added, that user is always assigned a role, but this is considered a different action which is described in Section 4.4.2.4 focusing on user management. To assign a role, a user chooses the desired role and the user to assign it to. Only roles and users visible to the user performing the action are eligible. If a user is unassigned from a role and has no more roles assigned to him, his access is effectively revoked and the user is removed.

Private hierarchy: In addition to ordinary role management CHAC offers the concept of multiple *private hierarchies*. The operations related to private hierarchies are explained in the next section.

4.4.2.3 Private hierarchy management

A *private hierarchy* is an independent role hierarchy that is detached from other role hierarchies. This private hierarchy is only associated with the original hierarchy through the user who created the private hierarchy and the role used for this purpose. Data entities cannot be shared between role hierarchies.

Create: Creation of private hierarchies requires a specific permission. When a private hierarchy is created, a root role is also created. The created root role is always granted full permissions and not a subset of its creator-role's permissions. This does not violate the original role hierarchy as data entities cannot be shared between separate role hierarchies. When creating a private hierarchy, the root role is always assigned to the user performing the creation. Figure 4.11 shows the scenario of a user capable of creating a new private hierarchy.

1. The initial role hierarchy is shown.
2. Alice creates a new private hierarchy with a root role as the topmost role of this new separate role hierarchy.
3. Alice now chooses to add an additional role to the new private role hierarchy.

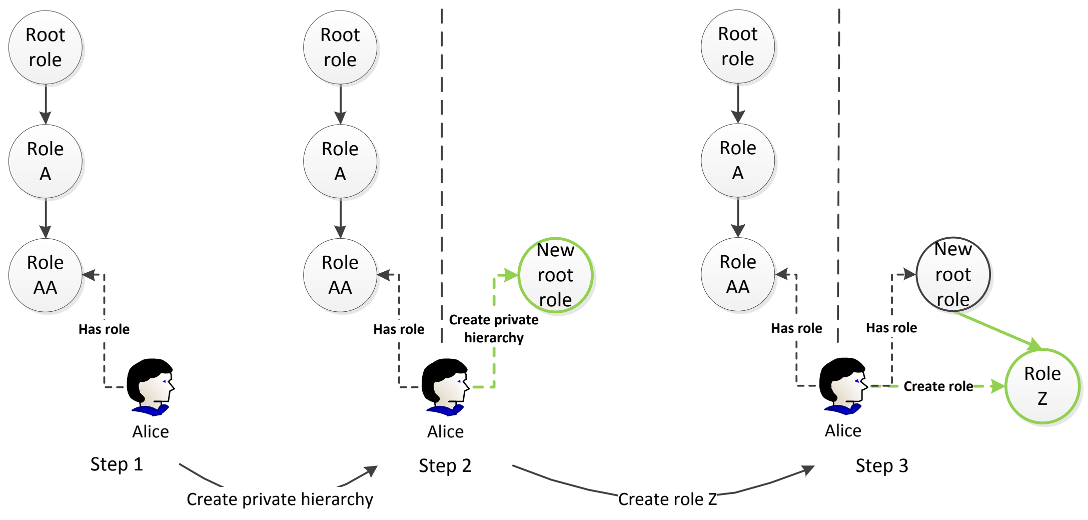


Figure 4.11: Flow of private hierarchy creation.

Delete: To delete a private hierarchy the root role of this hierarchy must be removed. This results in the entire role hierarchy, attached to the root role, being deleted. Note that this behavior is required by the fact that a root has no parent. If a private hierarchy is to be deleted two different options can be chosen to accomplish this operation as depicted in Figure 4.12:

1. The initial two role hierarchies are shown with a user assigned to Role A and the “Other Root Role”.

2a. If the “Other Root Role” is deleted, the entire associated private hierarchy is also deleted.

2b. If the last user is removed from the “Other Root Role”, the entire associated private hierarchy is also deleted.

3. Finally, the result of the delete operation is shown where only one role hierarchy remains.

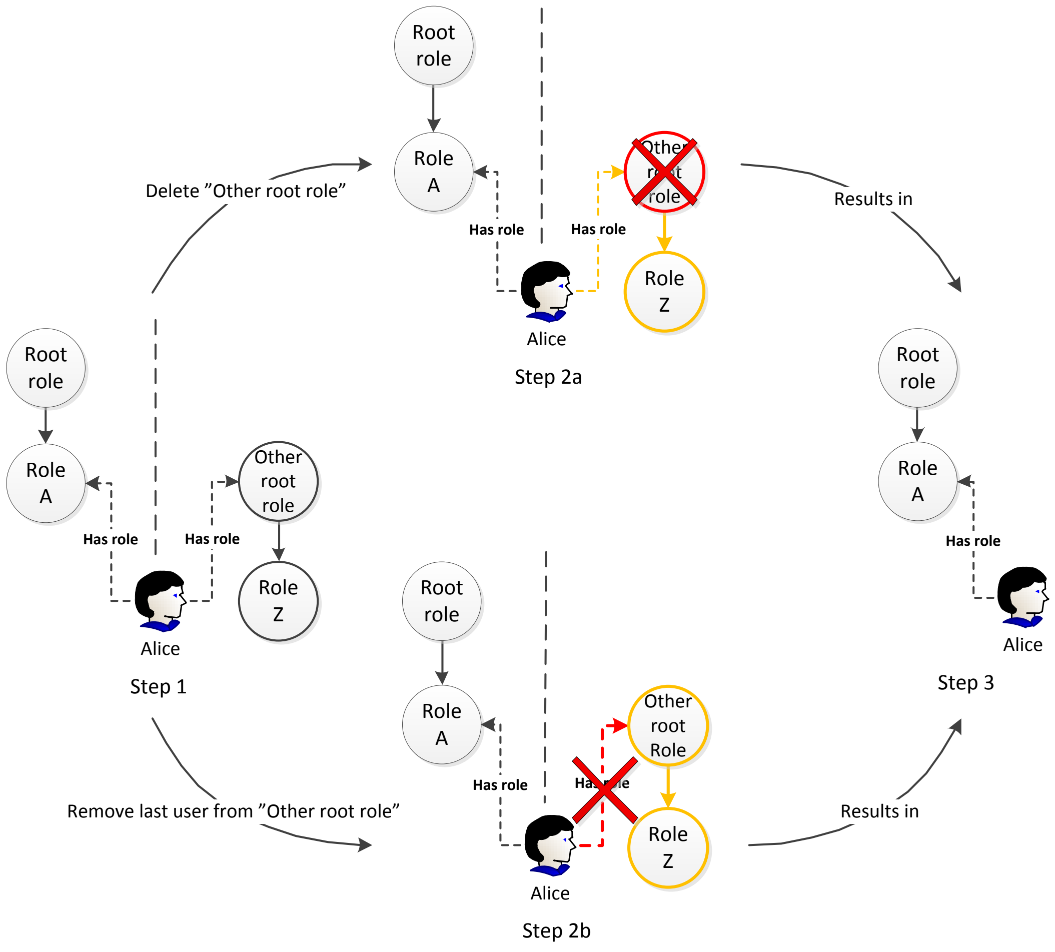


Figure 4.12: Flow of private hierarchy deletion.

An alternative scenario is shown in Figure 4.13:

1. First the original two role hierarchies with a user associated to Role A and the "Other Root Role" is shown.
2. Next, Role A, which was used to create the private hierarchy and is the child of the root role, is deleted. This results in a cascaded deletion of the private hierarchy associated with the "Other Root Role". If Alice has no other roles associated she is revoked as well. This is necessary as the remaining root role cannot see and manage Alice if she is only associated with the private hierarchy.
3. Finally, the remaining entities after the delete operation is shown.

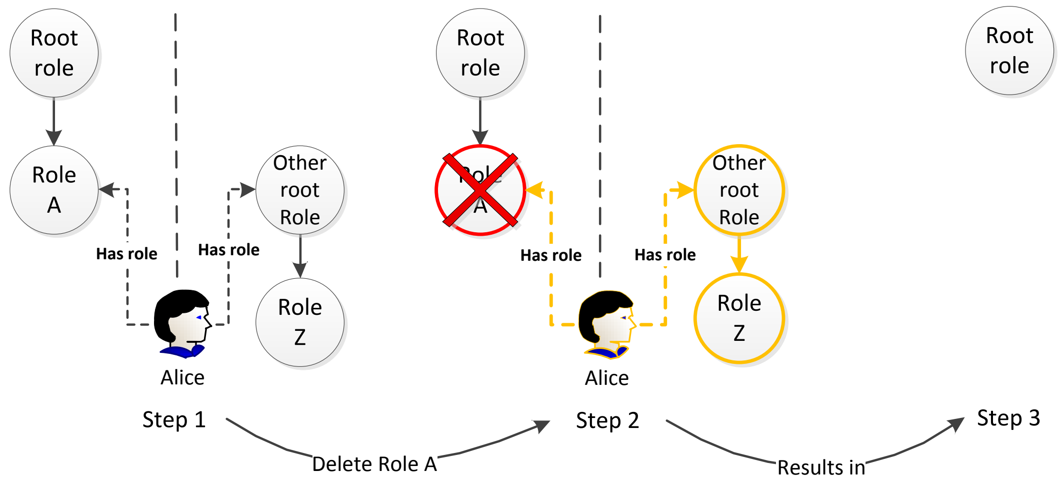


Figure 4.13: Alternative flow of private hierarchy deletion.

4.4.2.4 User management

When managing users, several operations are available to users with sufficient permissions. These operations are described below:

Create: Users are managed directly by other users only when being created, and are always assigned to a role when created. At the time of creation the user is issued a key pair for decrypting data, a signing key pair and a PRE token as defined in SDCD. In the event that a user already exists, but is created again, for example for the purpose of being assigned a role on a different branch in the role hierarchy, the system assigns him to the role and automatically links his existing user identity to the new role. When a user is created multiple times, he will be assigned multiple key pairs and PRE tokens. These are all valid and are kept to ensure that someone adding a user cannot tell that the user was already added by someone else. The user will then have multiple keys meant to be used for the different roles he is assigned.

Update: Users cannot be updated. However, the roles which they are assigned can be changed through the operations described under role management, see Section 4.4.2.2.

Delete: Users cannot be deleted directly, but the roles assigned to a user can be removed and when a user has no more roles assigned he is deleted. When a user is deleted his access is revoked, but a *shadow* of the user is retained. The shadow consists of the user's name, the PRE token, and the public part of his signing key. The name is required when performing audits to identify actions performed in the past. The PRE token is required to re-encrypt data entities for users the revoked user has added and the public part of his signing key is required to check the integrity of all data entities he is the author of. If the user is added to the system again later, he will be associated with his shadow.

Other considerations To handle situations where a leak occurs, such as a user sharing data with unauthorised entities, the *active* flags of roles are used. Users who can manage the roles suspected to be the origin of a leak can suspend those roles by making the roles inactive. This leaves the role hierarchy and user-role relationships intact, but prevents the users in the suspended roles from accessing the system through those roles.
In the case where a user has been assigned two roles with different permissions, for example a role granting read-only access to a data entity and another role granting read/write access to the same data entity, the role allowing read/write access takes precedence.
User-to-user anonymisation is retained through the user-role associations. Users are only able to see users in roles that are sub-roles, recursively, of their own roles. This effectively makes users assigned to a parent role and users assigned to roles in other branches of the role hierarchy anonymous to a user assigned to a child role.

4.4.2.5 Data entity management

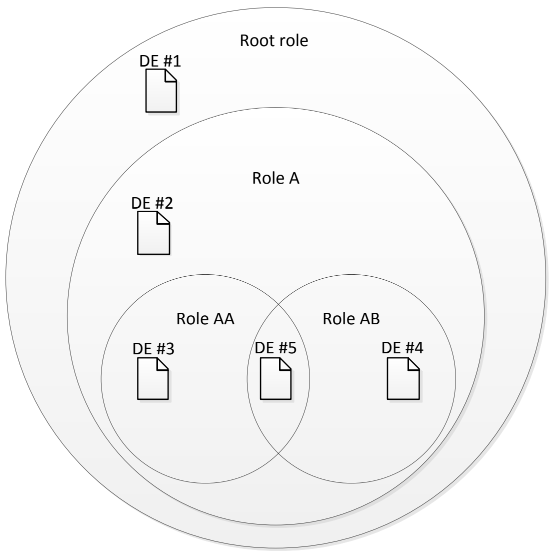
All data entities are stored separately from the information stored for roles and users. This means that for a role, which grants access to a set of data entities, a reference exist to each data entity. Multiple roles can reference the same data entity. When creating a sub-role which has access to data entities of its parent, an additional reference is created for each of the data entities they share. Thus, the relationship between references to a data entity and the roles giving permission to access that data entity is linear. It also means that any change made to a data entity through one role is visible when accessing the same data entity through another role. Figure 4.14 shows a Venn diagram of how data entities are shared between roles. In the scenario depicted, data entity #5 has four references.

4.4.3 Auditability

In the context of CHAC, auditability can be used to view actions performed by users. Auditing is an important security aspect, as it can be used to review events associated with a specific role, data entity, or user. It can determine what has occurred, when it occurred, and who was responsible.
Sufficient information must be stored in an audit log to ensure precise auditing. This means that whenever entities within CHAC are changed or new entities are created, these actions must be logged along with information about which user performed the action and through which role this action was performed. The logging is an automatic procedure performed by CHAC and it is transparent to users when they perform actions.
Auditing is performed through a root role as these are the only roles with the capability of accessing information on all entities in their respective hierarchies. If other roles were granted the ability to perform auditing, they would not have sufficient knowledge of the users in the hierarchy to see a complete audit log. Figure 4.15 illustrates a scenario where neither of the two users are able to construct a complete audit log for data entity #1. The roles that they each posses are incapable of

obtaining sufficient information of the role structure due to the anonymisation aspect of CHAC. Alice has no knowledge of Bob and vice versa and thus, they cannot see the other user's actions. If these users, through their roles, were granted access to view all other users associated to the role hierarchy then either user-to-user anonymisation would be lost or the viewed audit log would become incomplete.

In CHAC, audit logs are stored and maintained for users, roles, and data entities. Audit logs are kept even if an entity is deleted. This helps provide a complete picture of actions associated with even deleted entities and allows auditors to see when the entity was deleted and by whom. The audit logs are stored in the cloud storage in encrypted form using similar cryptographic techniques and keying schemes as the ones used to protect data entities.



Data entities referenced by roles:
Root role: {DE #1, #2, #3, #4, #5}
Role A: {DE #2, #3, #4, #5}
Role AA: {DE #3, #5}
Role AB: {DE #4, #5}

Figure 4.14: Role and data entity relationship.

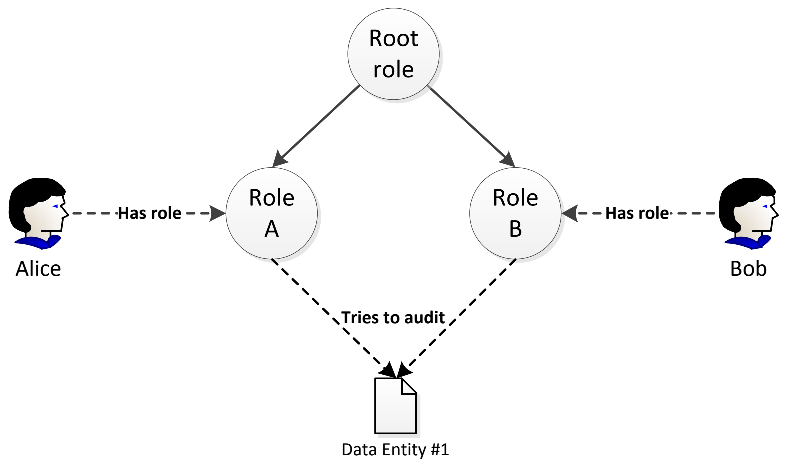


Figure 4.15: Users incapable of auditing a data entity.

4.5. Discussion

CHAC provides an access control model that allows fine-grained user access control and auditability. SDCD only offers coarse-grained access control that either allows complete access or no access to data entities. To improve this and fuse the two designs together, CHAC is designed as an extension of SDCD. Together they form a generic collaboration environment offering confidential cloud storage capable of enforcing fine-grained user access control and permission delegation through the use of a role hierarchy. This results in a transparent access control model with a clear set of rules defining how permissions are granted to users.

4.5.1 Private hierarchies

The purpose of private hierarchies is to support generic usage scenarios where users can create their own private areas within the dynamic collaboration environment. These can be shared with a selected group of users, which may remain a part of other role hierarchies. When created, a private hierarchy contains no data entities.
The root role, in a newly created private hierarchy, would be of little value if less than a full set of permissions were granted to it. Mainly because private hierarchies initially consist of only the root role itself.
If a root role of a private hierarchy is deleted, without deleting its sub-roles, the private hierarchy becomes unreachable with no administrating entity associated. Likewise, if the last user assigned to a root role is removed, the root role becomes unreachable as no other roles or users have knowledge of it. Therefore, in these situations, the hierarchy of a root role is always deleted in CHAC. The disadvantage of this approach is situations where a user or role is accidentally removed as this would cause data to be lost unintentionally.
Finally, if a role used to create a root role is deleted, then the root role and its private hierarchy is also deleted. If the root role was not deleted, then the DO who initialised the dynamic collaboration environment would have no way of ever deleting the root role, its hierarchy, and all the data entities accessible from roles of the hierarchy.

4.5.2 Anonymisation

The property of user-to-user anonymisation is preserved through the use of a role hierarchy. The hierarchies allow top-down knowledge of associated users and roles, which means that the different branches in the role hierarchy are invisible to each other. This helps preserve anonymisation.

4.5.3 Auditability

The introduction of auditability in CHAC could potentially negatively affect anonymisation as users performing audits require knowledge of all users within in the dynamic collaboration environment to get a complete audit. To deal with this issue only root roles are eligible to perform audits. However, this introduces the side-effect of auditors needing to be DO's and thereby be granted permissions which they should not have.

4.5.4 Life cycle management

A potential issue in CHAC is the life cycle management with regards to user keys and Proxy Re-Encryption (PRE) tokens. As can be derived from Figure 4.8 the user at the n^{th} level in the key pair hierarchy will require n re-encryptions to be able to decrypt requested data. This results

in the need to retain every PRE token for users having added him or his parents, which in turn results in the key pair hierarchy potentially becoming very deep. For instance, a user is added as a normal worker with his key pair being key pair n in the hierarchy. The user is then promoted and becomes project manager and needs to add more users to his team. The key pairs for these users would become key pair number $n + 1$. This may result in unchecked growth of the key pair hierarchy throughout the life cycle of CHAC and will degrade performance.
Another issue regarding key pair growth is the number of signing key pairs and PRE tokens associated with a single user. The PRE token and the public key of the user's signing key pair are kept as a shadow after he is revoked from the role hierarchy. The retained PRE token is used to allow other users, added by the revoked user, to retain access to data entities in the cloud. The retained signing key is used for verifying integrity of data entities created or edited by the revoked user. If this user is added again later, he will be assigned a new key pair, a new signing key, and a new PRE token. If this add-and-revoke process is repeated continuously the number of public signing keys and PRE tokens, stored in the cloud, will grow unckecked. This will render the cloud unable to determine what PRE token to use when the user requests data. Whenever an integrity check is requested for data, created or edited by this user, the cloud will, potentially, have to iterate through all public signing keys stored for the user. Currently this issue is not addressed by the design of CHAC. This is a subject for future work.
Whenever a user is revoked and then later added again the process of assigning new key pairs and PRE tokens should be transparent to the user. It must, nonetheless, be handled by the system.

4.5.5 Data versioning

When receiving a data entity, a user receives a copy of the requested data entity. The original copy of the data entity, which is stored in the cloud, can then be updated by someone else. If the user submits changes made to the data entity, the changes made by others are overwritten. This is not handled by CHAC, but could potentially be avoided by using a version control scheme.

4.5.6 Comparison to related work

The entity structure presented in the NIST RBAC standard [Ferraiolo&01] has been followed and implemented in CHAC. Furthermore, the use of a hierarchy to organise and relate roles to each other has also been used. The use of either Static Separation of Duty (SSD) or Dynamic Separation of Duty (DSD), as proposed by the NIST standard, has also been considered for CHAC. DSD is used for situations where a user is assigned two different roles; one allowing read-only access to a data entity, and another allowing read and write access to that same data entity. In situations like this, the role which permits read and write access takes precedence.
The user management of the DACC scheme, presented by [Ruj&11], is infeasible due to the fact that attributes assigned to revoked users can never be reused. This limits other users from using revoked attributes. In CHAC this issue is avoided and revoked users only leave a shadow. This is transparent to other users.
The DACC and CHAC concepts are compared in Table 4.3 and in Figure 4.16 along with the NIST RBAC standard. As DACC does not consider all of the aspects of CHAC, the two conceptual designs cannot be compared on all aspects. The Key Distribution Centres (KDCs), proposed in DACC, lowers the solution's overall scalability as they are not cloud-based. In CHAC access control is performed in the cloud environment and does not require off-cloud services. Availability and durability both benefit from the cloud deployments used in DACC and CHAC. Neither of the two designs consider data confidentiality in the clients' local environments.

Table 4.3: Comparison of CHAC and related work.

Environment		**Category**	**Property**	**Ruj**	**CHAC**
Cloud	User	Security	Data confidentiality	+	+
			Data integrity	N/A	+++
	User		Data availability	++	++
			Non-repudiation	N/A	+++
			Auditability	N/A	+++
			User-to-user anonymisation	N/A	+++
		Access control	User addition and revocation	++	+++
	User		Fine-grained access control	++	++
			Access control delegation	N/A	+++
			Many-to-many file sharing	++	+++
		Performance	Scalability	++*	+++
			Durability	+++	+++

* Ruj uses KDCs for user authorisation.

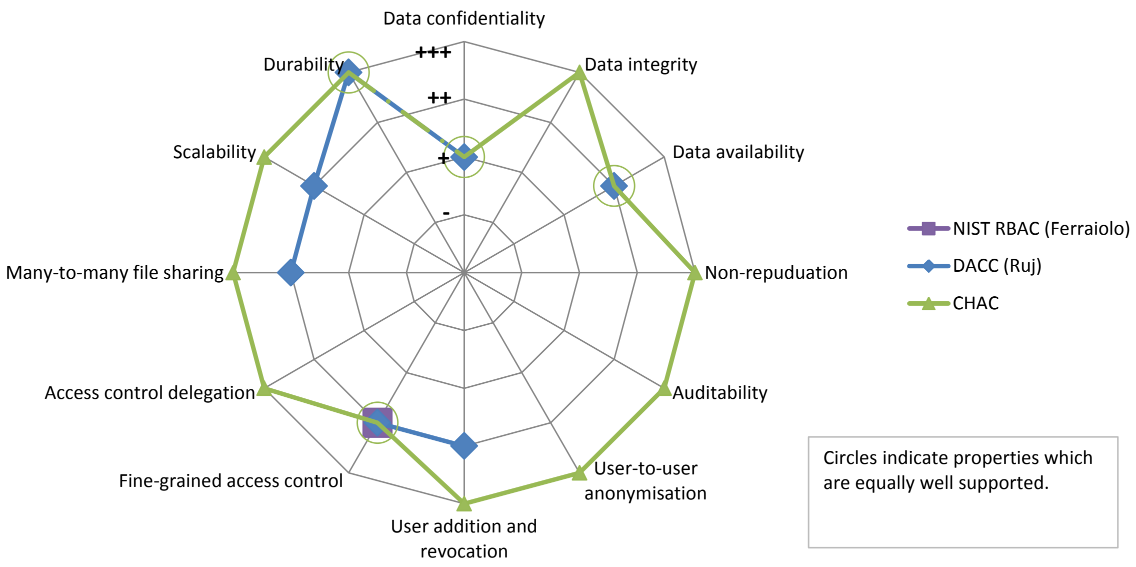


Figure 4.16: Graphical comparison of CHAC and related work.

4.5.7 Comparison to thesis goals

The presented conceptual design of CHAC adds support for fine-grained access control, access control delegation, and auditability. This is a significant improvement to the dynamic collaboration environment. Table 4.4 summarises and compares SDCD and CHAC to the thesis goals. Figure 4.17 is a graphical representation of the same comparison.

Table 4.4: Comparison of SDCD and CHAC.

Environment		Category	Property	SDCD	CHAC
Cloud	User	Security	Data confidentiality	+	+
			Data integrity	+++	+++
	User		Data availability	++	++
			Non-repudiation	+++	+++
			Auditability	-	+++
			User-to-user anonymisation	+++	+++
		Access control	User addition and revocation	+++	+++
	User		Fine-grained access control	+	++
			Access control delegation	-	+++
			Many-to-many file sharing	+++	+++
		Performance	Scalability	+++	+++
			Durability	+++	+++

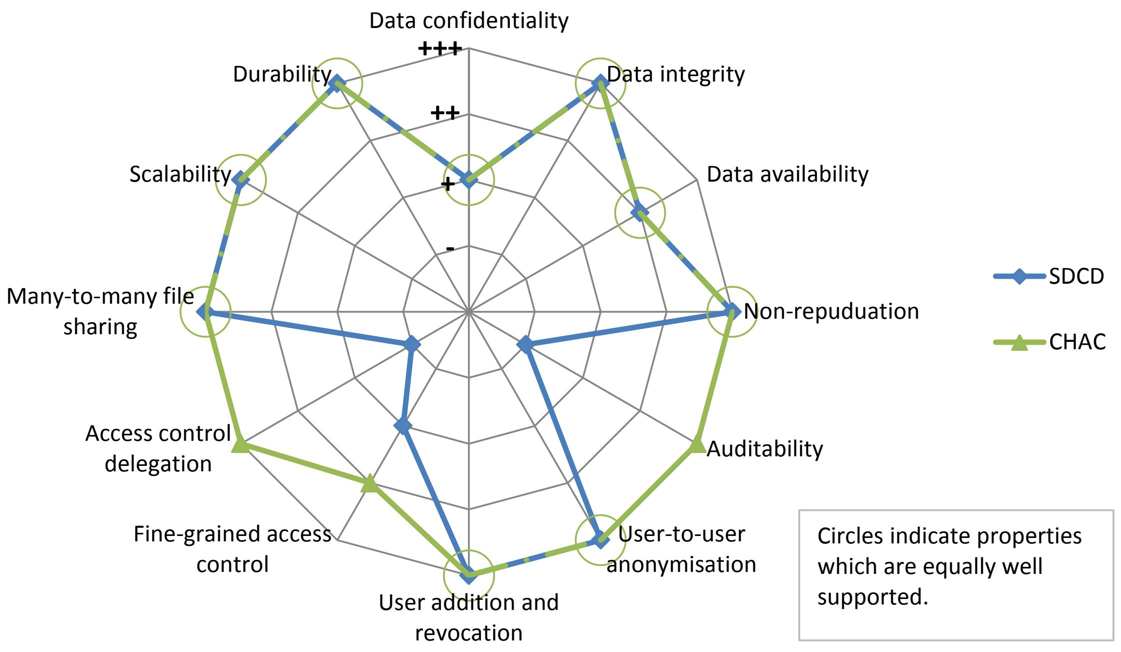


Figure 4.17: Graphical comparison of SDCD and CHAC.

Chapter 5

Data access control outside the cloud

This chapter presents a conceptual design focusing on ensuring the security of data once the data is in a user's local environment. This is accomplished by extending the support for the properties of confidentiality, availability, and fine-grained access control, defined in Chapter 1, to encompass the local environment of the user. This chapter relies on concepts and technologies presented in Chapter 2 and builds upon the design presented in Chapter 4 which in turn builds upon the design presented in Chapter 3.

5.1. Introduction

This chapter builds upon the designs presented in the two previous chapters and extends the security aspects of *Cloud-based Hierarchical Access Control (CHAC)*. To extend these aspects outside the cloud requires an off-cloud data protection scheme. Without such a scheme, users would be able to distribute data in its decrypted form to unauthorised entities. Therefore, to ensure the protection of data and allow for the detection of the origin of data leaks, a scheme capable of enforcing security locally in the user's environment and tracing copies of data is required.

The assumptions made for this chapter are described in Section 5.2. Existing schemes within in the field are presented in Section 5.3. Section 5.4 presents our conceptual design for ensuring protection of data in the user's local environment. Finally, Section 5.5 discusses and compares the conceptual design to the thesis goals.

5.2. Assumptions

The user is considered a potential liability to the confidentiality of the data he handles. The user might disregard the responsibility that comes with handling confidential data and distribute it to unauthorised entities. Thus, the user is a potential source of a data leak. As a consequence data stored in the user's local environment must be kept secure.

Data handled locally in the user's environment is vulnerable to the analog gap. It is assumed that the user does not exploit the gap, for example by taking pictures of a document shown on a monitor or by printing it and distributing it in its analog form. Thus, this chapter will only address security of the data in its digital form.

5.3. Related work

Over the last decade various schemes and technologies have been developed and suggested to prevent data from being distributed by entities authorised to access it, to entities not authorised to access it. One of the prominent methods is the utilisation of Digital Rights Management (DRM) systems. This approach is often used as a distribution mechanism to publish data from a data producer to a selected consumer or group of consumers. The producer specifies the terms by which data is distributed, often including cryptographic schemes to protect data once it is distributed to the consumer's environment. Various implementations and variations of DRM systems exist, offering different services, enforcing and supporting the terms of use such as payment services, limitations on usage, or time-based access to data [Guth03].

DRM solutions offer a large variety of services and ways of distribution. They have been used for many years and have been thoroughly tested in the industry. Because the properties of DRM systems are well-understood, DRM is a prime candidate for extending the cryptographic protection scheme of SDCD, described in Chapter 3, and integrate with CHAC described in Chapter 4.

The following sections describe some DRM solutions and their required properties. Section 5.3.1 describes the required properties of a solution which addresses the thesis goals of securing the data in a user's local environment. Existing concepts and solutions conforming to the presented properties is examined, starting with a typical DRM solution in Section 5.3.2 followed by a short commercial DRM solution comparison in Section 5.3.3. Finally, presentations of several schemes that are essential for ensuring support of the required properties are presented in Sections 5.3.4, 5.3.5, and 5.3.6.

5.3.1 Required properties

To add significant value to the envisioned dynamic collaboration environment, a potential DRM solution must have certain properties. The properties in Table 5.1, listed in arbitrary order, must be satisfied in order for the DRM solution to conform to the thesis goals:

Table 5.1: Properties which a DRM solution must have.

Property Name	**Property Requirement Description**
Access control	Protected data must be inaccessible to unauthorised users
Granular access	Read-only and read/write access rights must be supported
n resource types	A wide range of resource types must be supported
Tamper resistance	Robustness and resilience of the system must be ensured
Offline data usage	Protected data must be accessible while offline
Tracing of data	Data must be traceable to identify the origin of a leak
Utilise cloud	The benefits of a cloud deployment must be leveraged

Due to the proprietary nature of DRM systems, finding an existing DRM solution that supports all of the above mentioned properties is unlikely. Although there might be some systems that partially support the properties. However, to conform to all of them, an existing system is likely to require customisation to fit into the context of the dynamic collaboration environment.

5.3.2 A typical DRM solution

A typical DRM solution ensures access control of protected content in the user's local environment. Typically, this is achieved by using licenses and through the use of cryptographic schemes combined with an authentication entity, which is often a server [Liu&03]. Such a solution is depicted in Figure 5.1. It shows the flow of acquiring content and is comprised of four entities, each with distinct roles [Liu&03, Michiels&05]:

Consumer: The consumer is the entity desiring to make use of DRM protected content, as described in Section 2.4.2.

DRM client: The DRM client is responsible for negotiating with the content service and license service and presents the requested content to the consumer. Thus, the client must offer support for the content type to present it.

Content service: The content service handles incoming requests for content from the DRM client and returns protected content to the client to be presented to the consumer.

License service: The license service handles incoming requests from the DRM client for licenses to previously requested content and returns a license specifying the usage rights of the specific content, as described in Section 2.4.2.2. Content can only be consumed after acquiring a license from the license service.

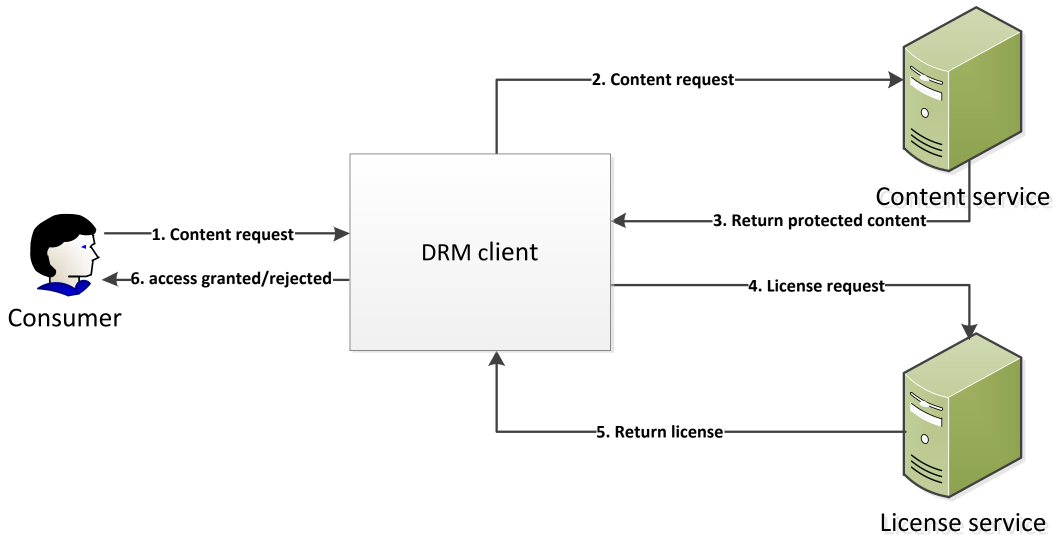


Figure 5.1: Typical DRM system structure.

As described above, and shown in Figure 5.1, a typical DRM solution adheres to the properties of access control in the form of the employed cryptographic schemes, the authentication entity, and granular access rights which are specified by the acquired license.

5.3.3 Commercial DRM solution comparison

Currently many different commercial DRM solutions exist. A common constraint on all of them is the fact that they only offer support for certain resource types, such as file formats, operating systems on the client side, and operating systems on the server side. To illustrate this, a short comparison of the Oracle Information Rights Management from Oracle Corporation, the Microsoft Windows Rights Management Services for Windows Server 2003 from Microsoft Corporation, and the Documentum IRM Services from Authentica can be seen in Table 5.2. For a full comparison of the above mentioned DRM systems as well as additional products, see [Zeng&10].

Table 5.2: Comparison of some Digital Rights Managements solutions.

Supported file formats	**Oracle**	**Microsoft**	**Authentica**
Adobe Acrobat	X		X
Microsoft Word	X	X	X
Microsoft Excel	X	X	X
Microsoft PowerPoint	X	X	X
Microsoft Outlook		X	X
Microsoft Access		X	
HTML	X		X
XML	X		X
eRoom			X
Lotus Notes			X
e-mail	X	X	X
OS support client side			
Windows Vista	X	X	X
Windows XP	X	X	X
Windows 2003		X	
Windows 2000	X	X	X
Windows NT		X	X
Windows 98		X	X
Linux			X
Macintosh OS			X
BlackBerry Phones	X		X
OS support server side			
Sun Solaris			X
Microsoft Windows			X
Microsoft Windows Server 2003	X	X	X
Apple Macintosh			X

As Table 5.2 clearly portrays, no single commercial DRM system conforms to the property of supporting n resource types. Thus, separate implementation to support each additional resource is required.

5.3.4 Tamper resistance

The main topic in [Horne&02] is the use of self-checking mechanisms to verify the system's integrity. Two types of self-checking are depicted: static self-checking which provides integrity checks once, at program start-up, and dynamic self-checking which offers continuous integrity checking during program execution. The dynamic self-checking being the main focus, is achieved using three components. The main component, which is the tester, is used to map large sections of executable code into smaller sets. The sets undergo a hashing process to scramble readability and pass the values on to a corrector acting as storage for the mapped and hashed values. The tester continuously recalculates and stores the hashed value of the code section. If changes should occur to the code section, the tester will detect the change in the hashed value compared to the stored value in the corrector. If the two hashed values differ, the tester contacts a tamper response component responsible for taking proper actions. The executable code is divided into sections and associated with several testers that store the hashes in individual correctors.
Several measures are used to protect this self-checking mechanism against attacks that either reveal or disable the mechanism. Obfuscating the testers to make them look like ordinary executable code and customising their individual implementations are some of the countermeasures used. Protection against the self-checking mechanism being detected by off-the-shelf debuggers is also added [Horne&02]. Custom made debuggers, however, cannot be prevented from stepping through the code and potentially discovering the self-checking mechanism. The components of the self-checking mechanism have been separated and designed in such a way that discovering one does not aid in locating others.

5.3.5 Offline usage scheme

An offline DRM scheme involving a Consumer role, a Publisher role and a DRM server is described in [Sankar&11]. Figure 5.2 shows the process of the offline DRM scheme:

1. In the registration phase, the consumer initially registers himself with the DRM server and requests a certificate granting him access to content. The DRM server generates two different certificates: one identifies the user and another identifies the machine from which the request originated. This is necessary to establish a trust relationship between consumer and publisher. Afterwards the certificates are sent back to the user and copies of the certificates are stored at the DRM server.

2. In the publication phase, the publisher encrypts and uploads content to the DRM server.

3. In the consumption phase, the consumer's DRM client uses the machine certificate to authenticate the platform on which protected data is to be accessed. The user certificate is then used to verify that the user has been granted permission to access the protected data. Along with the data, the consumer also requests a DRM user license to allow for offline access to protected content. This license can either be distributed directly from the publisher or through other sources, such as a physical media if the consumer is offline. The license associated with the protected data ensures that the machine identifier, which is described in the license, matches the one on the machine it is now located on.

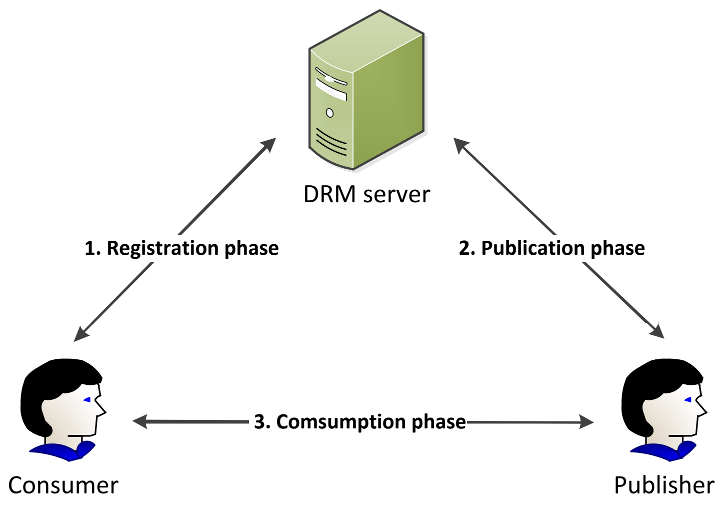


Figure 5.2: Offline DRM model.

5.3.6 Watermarking strategy

Currently there are two general usage strategies for the application of watermarking for assisting DRM systems [Wolf&07]:

Public watermarking: In public watermarking, the DRM system running on a consumer's computer embeds a watermark in the content when the content leaves the DRM system. That is, when the content is sent to an analog output channel. If the data re-enters the DRM domain, the watermark is retrieved and the DRM rules are re-assigned to the content. This is prone to attack, as an attacker may be able to perform operations on the content and see if the DRM rules are re-assigned to it. When this is no longer the case a successful attack has been identified.

Private watermarking: In private watermarking, the producer embeds the watermark and pays all of the computational costs. In this case, the watermarking is transparent to the consumer. The weakness described in the public approach is still present, however, only the producer is able to determine whether an attempt to remove the watermark has succeeded or failed. This effectively eliminates the quick and easy testing mechanism.

5.3.7 Comparison of related work

This section presents an overview and comparison of the related work by rating the properties of each presented scheme. The schemes are rated to the thesis goals, according to the levels described in Table 1.2, by their ability to support data confidentiality, data availability, or fine-grained access control in the user environment. The schemes presented above covers a typical DRM system, tamper resistance, offline usage, and watermarking. Each scheme covers a different aspect of ensuring confidentiality in the user's local environment and in combination they provide significant coverage. They are therefore considered important for the conceptual design which will be presented in Section 5.4. Table 5.4 shows a comparison of the presented schemes and shows that only the schemes presented by [Michiels&05] and [Sankar&11] cover fine-grained access control in the user's environment. This is achieved through the use of licenses and certificates to enforce access control at a granular level. Availability is only covered by [Sankar&11] through the use of an offline DRM scheme, which allows the user to access protected data without being forced to communicate with the DRM server.

Table 5.3: Comparison of schemes proposed in related work.

Environment		Category	Property	Horne	Sankar	Wolf	Michiels
Cloud	User	Security	Data confidentiality	++	++	++	++
	▨		Data integrity	N/A	N/A	N/A	N/A
	User		Data availability	N/A	+++	N/A	N/A
	▨		Non-repudiation	N/A	N/A	N/A	N/A
			Auditability	N/A	N/A	N/A	N/A
			User-to-user anonymisation	N/A	N/A	N/A	N/A
		Access control	User addition and revocation	N/A	N/A	N/A	N/A
	User		Fine-grained access control	N/A	+++	N/A	+++
	▨		Access control delegation	N/A	N/A	N/A	N/A
			Many-to-many file sharing	N/A	N/A	N/A	N/A
		Performance	Scalability	N/A	N/A	N/A	N/A
			Durability	N/A	N/A	N/A	N/A

5.4. User-Centered Data Security

5.4.1 Overview

This section describes our conceptual design for *User-Centered Data Security (UCDS)*. It uses the topics and technical methods described in Section 5.3. The topics and technical methods are used to achieve the required properties a DRM solution should conform to, as described in Table 5.1. The generic DRM role definitions presented in Section 2.4.2 match the entities contained in the conceptual design of CHAC which is presented in Chapter 4 and can thus easily be mapped to the context of this thesis. These mappings are depicted in Figure 5.3 and are described as follows:

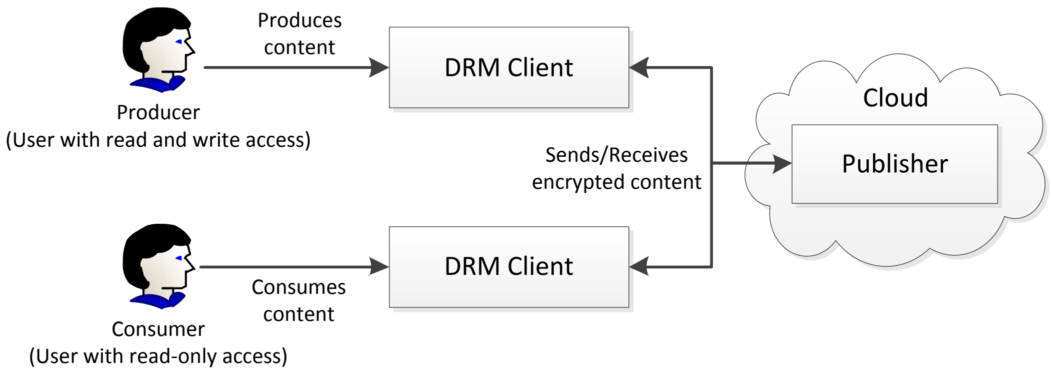

Produces content
DRM Client
Producer
(User with read and write access)
Cloud
Publisher
Sends/Receives encrypted content
DRM Client
Consumes content
Consumer
(User with read-only access)

Figure 5.3: A DRM solution in a cloud environment.

Producer: Seen in the context of this thesis, the users with read and write access rights are assigned the role of producers. They are the entities producing data to be shared.

Consumer: The consumer role is assigned to all users who need to access shared data stored in the cloud. Depending on which access rights they have been assigned, they can access a selected amount of data and perform their allowed operations on the data. A consumer can also be a producer, however, only if he has read and write access rights. Otherwise, the user only has a consumer role.

Publisher: The publisher role is assigned to an entity residing in the cloud environment. This role is the most active, with a need for high availability. Therefore, the availability offered by a cloud environment is very beneficial.

5.4.2 The conceptual design

Introducing a commercial DRM solution to the dynamic collaboration environment seems infeasible due to the lack of control of the cryptographic mechanics within these DRM solutions. This poses a problem as the data located in the cloud is already encrypted by the specific cryptographic scheme described in Section 3.4.2.
Furthermore, adding a commercial DRM solution would require data to be decrypted in the cloud and re-encrypted by the DRM solution. This would inevitably leave the data unencrypted in an untrusted environment. Putting together parts of existing DRM solutions is also considered infeasible due to the proprietary nature and lack of interoperability of most DRM solutions, as

described in Section 2.4.2.1. The preferred approach would be to utilise the cryptographic schemes and access control models introduced in Chapters 3 and 4 to implement similar functionality in a generic DRM solution.

5.4.2.1 Architecture

UCDS is implemented using an existing data protection scheme in conjunction with an access control model, such as the design presented in Chapter 4. UCDS reuses these aspects and extends them into the user's environment. This requires implementation of a DRM client located locally in the user's environment, controlling the decryption of content received from CHAC. The DRM client acts as a reader and editor where a user can perform his permitted actions on data. UCDS also supports access revocation and watermarking.
The drawback of such a design is that every resource type must be supported by the DRM client. This would necessitate specific implementations inside the DRM client for each individual resource type. Alternatively, a DO could chose not to use DRM protection on selected resource types. For instance, if the DRM client does not support a given type, the users would be presented with an unprotected copy once they have been authorised by the access control mechanism. By introducing the DRM solution, the control of cryptographic key management of CHAC would be handled within the DRM client and becomes transparent to the user.
The conceptual design is shown in Figure 5.4.

1. The flow begins at the DRM client where a request for content is made to the cloud.
2. The access control manager validates the request against the roles that Alice is currently assigned and, if validated, fetches the content from the data storage.
3. The content is then passed to the encryption proxy to be re-encrypted.
4. The encryption proxy fetches Alice's Proxy Re-Encryption (PRE) token from the encryption proxy token storage and re-encrypts the content to match her key pair.
5. The re-encrypted content is returned to the access control manager.
6. The access control manager returns the content to the DRM client and from this point two options are available.

7a. If the requested content is supported by the DRM client and the DO has specified that the content should be accessed through DRM protection, it is presented to Alice through the DRM client. Through this approach, content never leaves the DRM client in unencrypted digital form.

7b. If the content is not supported by the DRM client or the DO has specifically decided not to DRM protect it, the content is in its unencrypted form when in Alice's environment. This approach mitigates lacking support for resource types by sacrificing security locally in the user's environment.

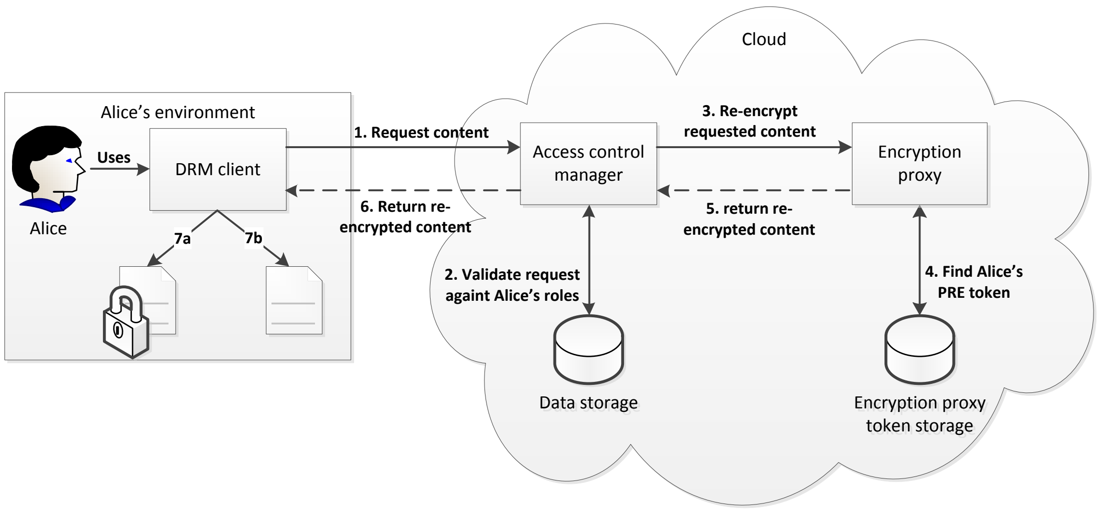

Cloud
Alice's environment
Alice
Uses
DRM client
7a
7b
1. Request content
6. Return re-encrypted content
Access control manager
3. Re-encrypt requested content
5. return re-encrypted content
Encryption proxy
2. Validate request againt Alice's roles
4. Find Alice's PRE token
Data storage
Encryption proxy token storage

Figure 5.4: Integrated DRM solution with DRM Client.

5.4.2.2 DRM client tamper resistance

Once data has been stored in the cloud, the DRM client is the only part of the dynamic collaboration environment that is responsible for decrypting data and operating on data in its unencrypted form. This, along with the DRM client being located locally in the users' environments, makes the DRM client a target for potential malicious users wanting to circumvent the data protection. For a DRM client to effectively enforce data access control and prevent unauthorised data access, tamper resistance must be implemented within the client. The approach, mentioned in [Horne&02] would provide the DRM client with a dynamic self-checking mechanism capable of detecting tampering attempts and triggering proper response actions. The triggered response can be integrated with other software protection solutions which would make the tamper resistance approach highly customisable. This is a desirable quality in the context of the dynamic collaboration environment, due to the support for specifying what response measures to take if the DRM client should detect a tampering attempt.

5.4.2.3 DRM client offline usage

The offline DRM model described in [Sankar&11] relies primarily on certificates. CHAC does not rely on certificates in its access control model, and would thus require additional support for certificate management. Data access restriction is already supported by CHAC. As described in Chapter 3, once the encrypted data is received in the user's local environment, the data decryption only requires the user's private key. By storing encrypted data locally in the user's environment, the DRM client could access this data instead of retrieving it from the cloud storage, decrypt it with the user's private key and present it to the user. This would grant the user offline access to the local encrypted data. However, it would not prevent him from accessing the data on other devices, though he would still need a DRM client with the exact same private key for decryption.

Data access restriction to a single device Restricting access to a single device could be achieved by introducing certificates identifying the device on which the data is to be decrypted. Another approach is to re-encrypt data under a new key consisting partly of unique identifiers from the given device, such as unique hardware serial numbers. Both approaches would introduce additional computational overhead, but the re-encryption approach would primarily require changes to the DRM client and would not require the additional implementation of certificate storage and management in the cloud as the first approach does. Figure 5.5 shows the re-encryption approach. It shows how encrypted data is received at the DRM client from either the cloud, through e-mail or from a physical media. The decryption and re-encryption process where encrypted data is decrypted and then encrypted using another scheme is also shown. This scheme encrypts data using a combination of the user's public key and a unique identifier which identifies the local environment on which the DRM client is located. This prevents the DRM protected data from being decrypted and accessed in other users' environments. Once the re-encryption process is complete, data is stored locally to enable later offline access. When a user chooses to access this specific data it is decrypted and presented to him by the DRM client.

5.4.2.4 Watermarking

To ensure traceability of data issued to a consumer, a watermarking scheme needs to be incorporated into the UCDS design. However, just as with existing commercial DRM solutions, existing commercial watermarking solutions are constrained. They only offer support for certain resource types and operating systems [Zeng&10]. Thus, to effectively integrate a watermarking scheme with the presented design it would be advantageous to select a watermarking scheme that supports watermarking of resource types supported by the design. Watermarking would then serve as a "second line of defence" by making resources under DRM protection in the users' local environments traceable. It would help identify the user who is leaking data, should a leak occur.

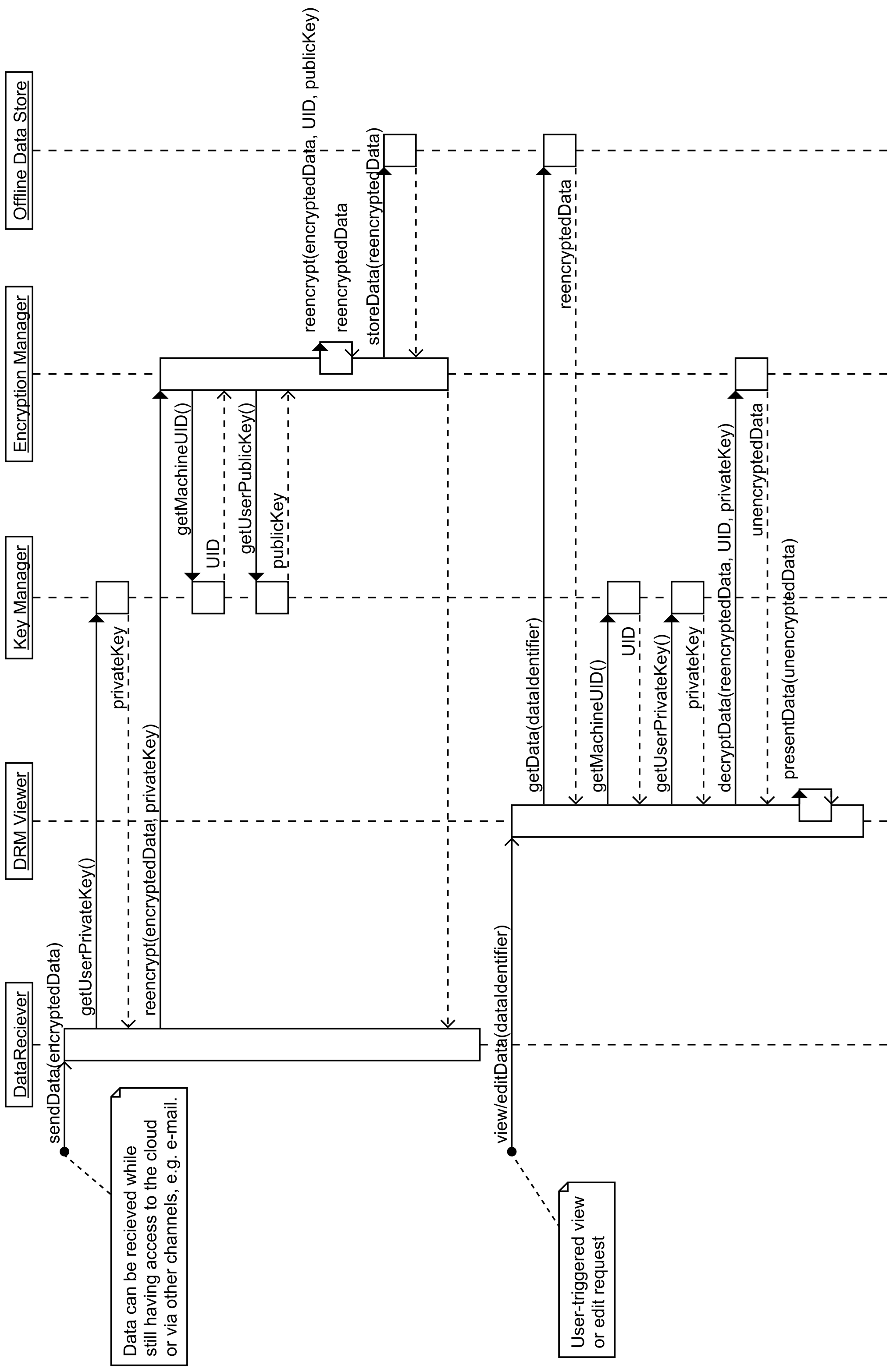

DataReciever
DRM Viewer
Key Manager
Encryption Manager
Offline Data Store
sendData(encryptedData)
getUserPrivateKey()
privateKey
reencrypt(encryptedData, privateKey)
getMachineUID()
UID
getUserPublicKey()
publicKey
reencrypt(encryptedData, UID, publicKey)
reencryptedData
storeData(reencryptedData)
view/editData(dataIdentifier)
getData(dataIdentifier)
reencryptedData
getMachineUID()
UID
getUserPrivateKey()
privateKey
decryptData(reencryptedData, UID, privateKey)
unencryptedData
presentData(unencryptedData)
Data can be recieved while still having access to the cloud or via other channels, e.g. e-mail.
User-triggered view or edit request

Figure 5.5: Dynamics of binding data to a single device.

5.5. Discussion

For most solutions which are based on cryptographic schemes, DRM is a good candidate for extending the data protection. A drawback of using DRM is the negative impact on usability as the user experiences no direct gain from it. As a consequence, the DRM element of the solution must take natural user interaction into account and remain transparent to legitimate users and permitted behaviours.

5.5.1 Interoperability

Looking at existing DRM solutions and their interoperability, the need for a custom DRM solution is evident. Interoperability issues are certain to occur when dealing with existing third-party components. This combined with the potential need for extending resource type support makes use of an existing commercial DRM solution difficult.

5.5.2 Offline DRM approach

The offline DRM approach, presented in Section 5.3.5, is the one considered feasible to integrate with CHAC. Time-limited usage of offline data must be introduced in the users' environments to support user revocation. Once users have received data from the cloud storage and stored it offline, they no longer need to be authorised by the DRM server. This means that users are not hindered in accessing data stored offline even though their access has been revoked. To overcome this issue periodic re-authorisation is necessary. This must be enforced by the DRM client and impacts the offline data availability to some extent, depending on the duration between the required re-authentication, but is a necessity seen from a security perspective.

5.5.3 Watermarking

While the introduction of watermarking is to aid as a second line of defence, it should be noted that currently no method exists to effectively track leaked watermarked data. Thus, the only way of discovering a leak is finding watermarked data where it is not supposed to be. At this point irreversible damage caused by the leak could already have been done.

5.5.4 Comparison to related work

While some of the presented schemes cover the same properties, they each contribute and compliment each other. Table 5.4 shows a comparison between UCDS and the presented schemes. While it is evident that the presented schemes only consider the user's local environment, the integration of these schemes into UCDS adds significant value to, and improve upon, the overall support of the thesis goals. Figure 5.6 shows a graphical comparison of UCDS and the presented schemes. Note that the data confidentiality property is achieved at the same level by all compared schemes. Thus, the individual scheme icons overlap here.

Table 5.4: Comparison of UCDS and schemes proposed in related work.

Environment		**Category**	**Property**	**Horne**	**Sankar**	**Wolf**	**Michiels**	**UCDS**
Cloud	User	Security	Data confidentiality	++	++	++	++	++
			Data integrity	N/A	N/A	N/A	N/A	+++
	User		Data availability	N/A	+++	N/A	N/A	+++
			Non-repudiation	N/A	N/A	N/A	N/A	+++
			Auditability	N/A	N/A	N/A	N/A	+++
			User-to-user anonymisation	N/A	N/A	N/A	N/A	+++
		Access control	User addition and revocation	N/A	N/A	N/A	N/A	+++
	User		Fine-grained access control	N/A	+++	N/A	+++	+++
			Access control delegation	N/A	N/A	N/A	N/A	+++
			Many-to-many file sharing	N/A	N/A	N/A	N/A	+++
		Performance	Scalability	N/A	N/A	N/A	N/A	+++
			Durability	N/A	N/A	N/A	N/A	+++

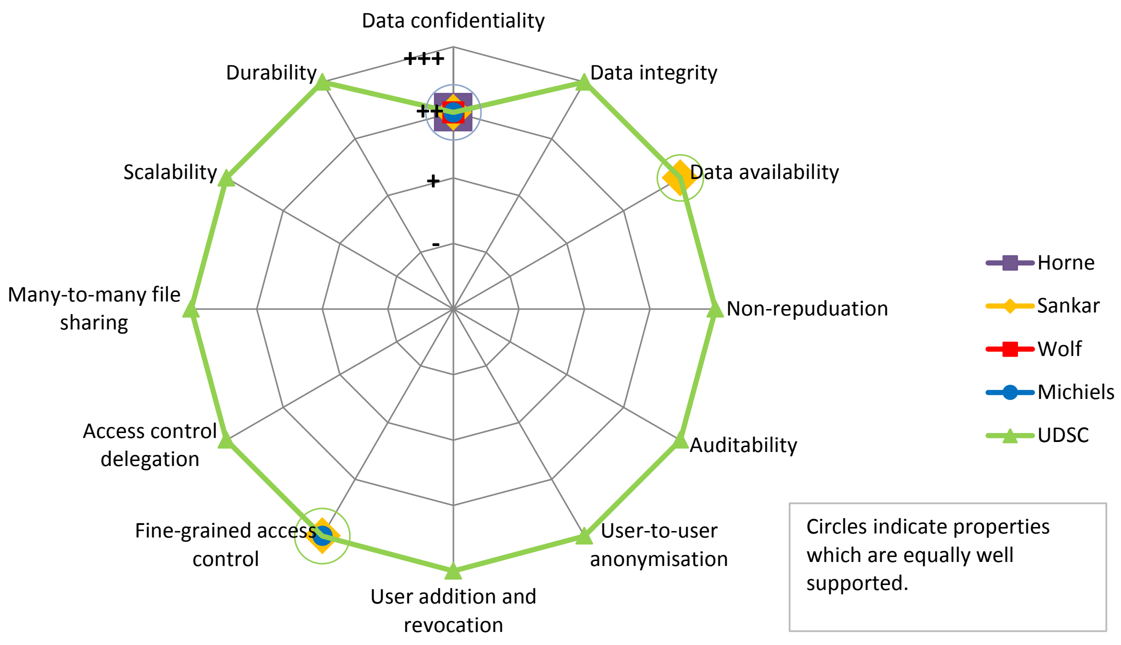


Figure 5.6: Graphical comparison of UCDS and schemes proposed in related work.

5.5.5 Comparison to thesis goals

Through the incorporation of the presented schemes into the conceptual design that is UCDS a higher coverage of the thesis goals is achieved. The presented conceptual design of UCDS conforms to the properties required for a DRM solution to add value to the dynamic collaboration environment. Table 5.5 shows the support the design offers for these properties. Table 5.6 summarises and compares CHAC to UCDS with regards to the thesis goals. Figure 5.7 is a graphical representation of the same comparison.

Table 5.5: Comparison of required properties and properties of UCDS.

Property	**Property Requirement Description**	**Support**
Access control	Protected data must be inaccessible to unauthorised users	Yes
Granular access	Read-only and read/write access rights must be supported	Yes
n resource types	A wide range of resource types must be supported	Yes*
Tamper resistance	Robustness and resilience of the system must be ensured	Yes
Offline data usage	Protected data must be accessible while offline	Yes
Tracing of data	Data must be traceable to identify the origin of a leak	Yes*
Cloud utilisation	The benefits of a cloud deployment must be leveraged	Yes

* Support for resource types must be implemented individually.

Table 5.6: Comparison of CHAC and UCDS.

Environment		**Category**	**Property**	**CHAC**	**UCDS**
Cloud	User	Security	Data confidentiality	+	++
	▨		Data integrity	+++	+++
	User		Data availability	++	+++
	▨		Non-repudiation	+++	+++
			Auditability	+++	+++
			User-to-user anonymisation	+++	+++
		Access control	User addition and revocation	+++	+++
	User		Fine-grained access control	++	+++
	▨		Access control delegation	+++	+++
			Many-to-many file sharing	+++	+++
		Performance	Scalability	+++	+++
			Durability	+++	+++

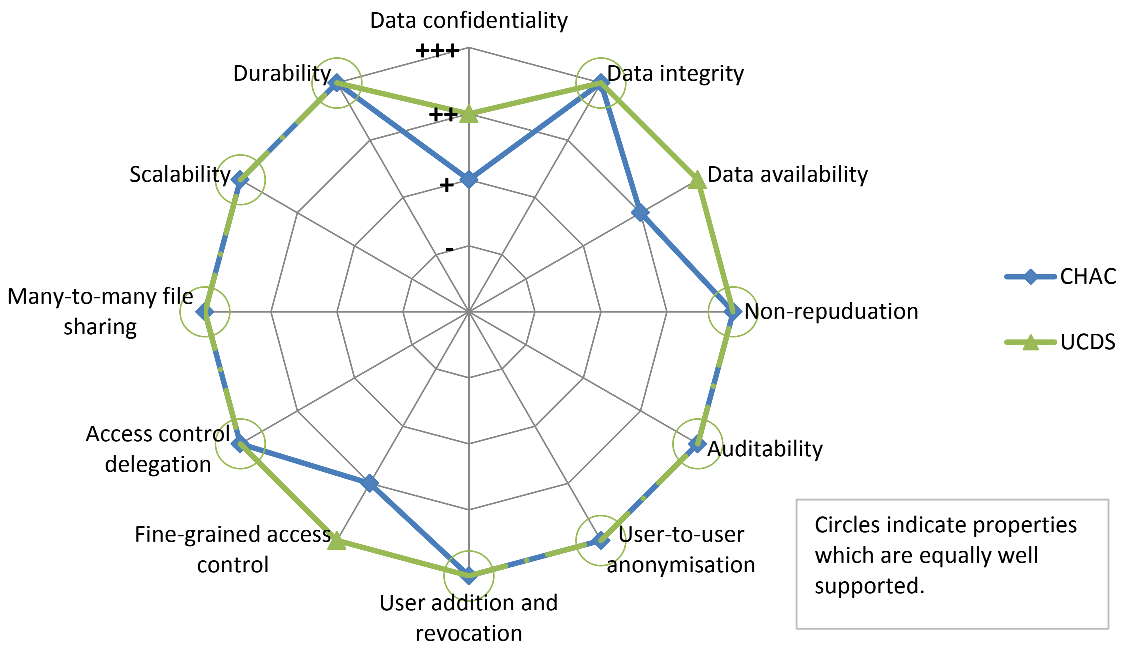


Figure 5.7: Graphical comparison of CHAC and UCDS.

Chapter 6

Validation of theory

This chapter focuses on the validation of the conceptual designs, described earlier in this thesis, by proof of concept prototypes. Chapter 1 defines the thesis goals which form the validation criteria. Chapter 3 and Chapter 4 each present a conceptual design that is validated here.

6.1. Introduction

To validate the conceptual designs presented in earlier chapters, proof of concept prototypes have been constructed. In this chapter, "proof of concept prototype" is in general shortened to "prototype". The prototypes demonstrate the feasibility for entire designs as they are described in earlier chapters and not only for a single yet complex component such as Proxy Re-Encryption (PRE). The prototypes focus on the high-risk and/or complex parts that are relevant to the conceptual designs. As a consequence, other parts which are of lesser importance to the conceptual designs are skipped.

Prototypes have been made for *Secure Dynamic Cloud-based Data-sharing (SDCD)* and *Cloud-based Hierarchical Access Control (CHAC)*. These conceptual designs are described in Chapter 3 and Chapter 4 respectively. As the conceptual design for CHAC builds upon the design for SDCD, the prototype for validating CHAC also builds upon the prototype for validating SDCD. No prototype has been built for *User-Centered Data Security (UCDS)* as the effort required to create a prototype of a Digital Rights Management (DRM) system outweighs its benefits. Aside from the non-trivial issues of supporting multiple resource types and integrating with the access control model of CHAC, validation of the conceptual design of UCDS does not significantly benefit from a prototype.

The following sections present the two prototypes created by giving an overview of each prototype. The descriptions of each prototype are based on the properties of the dynamic collaboration environment. For each property relevant for SDCD and CHAC, the sections present the aspects of the prototypes which influences the prototypes' abilities to validate SDCD and CHAC respectively.

Support for most properties is graded into four levels as seen in Table 1.2. Ideally, the prototypes validate each property up to the level which the conceptual designs support, but for reasons explained later, this is not the case for all properties.

Section 6.2 describes the prototype created for validating the conceptual design of SDCD which is described in Chapter 3. Section 6.3 considers the conceptual design of CHAC from Chapter 4. Finally, Section 6.4 discusses the validations.

6.2. Validating Secure Dynamic Cloud-based Data-sharing

6.2.1 Overview

This section describes how the conceptual design of SDCD is validated by a prototype. To support the thesis goals, the conceptual design of SDCD uses several cryptographic components and its prototype combines all of them into a single system which validates most of the conceptual design of SDCD.
As in the conceptual design of SDCD, its prototype uses a cloud for storing data entities. To enable the Data Owner (DO) and the users to interact with the cloud, the SDCD prototype has a *user client* and an *administration client*. The SDCD prototype contains a single cloud-based service and with respect to the cloud service categories discussed in Section 1.1, the service belongs in the Software as a Service category. The cloud-based service exposes a *storage service*, a *PRE service*, and has an internal storage for data entities. The storage service makes it possible for users to store and share data entities. The PRE service is used for all PRE operations.
The cloud services, the user client, and the administration client are implemented using the Microsoft .NET framework [Microsoft12]. The .NET framework contains many of the required features for building distributed systems speeding up the prototype creation. An overview of the SDCD prototype is depicted in Figure 6.1. The figure shows how a single cloud-based service contains the two services mentioned above and how two users: Alice and Bob through their user clients accesses data entities stored in the cloud.

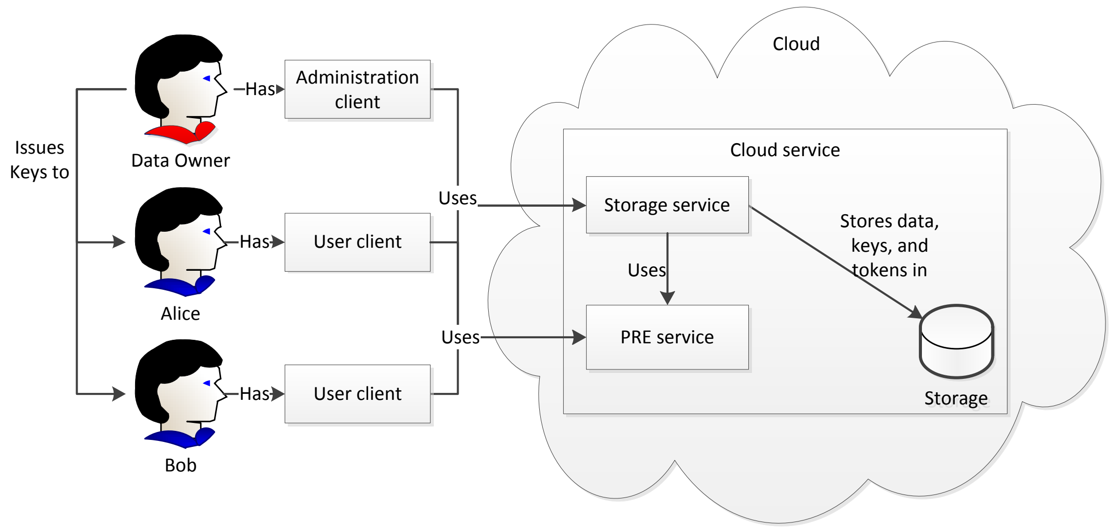


Figure 6.1: Overview of the SDCD proof of concept prototype.

6.2.2 Property validation

As described in Chapter 3, the conceptual design of SDCD supports many of the properties presented as the thesis goals. This can be seen in Figure 6.2 which reiterates the findings of Chapter 3 and shows a comparison of the conceptual design of SDCD to the thesis goals. The following sections each focus on a particular property which the conceptual design of SDCD partly or completely supports. For each property, the aspects of the SDCD prototype which influences the prototype's ability to validate the conceptual design of SDCD is presented.

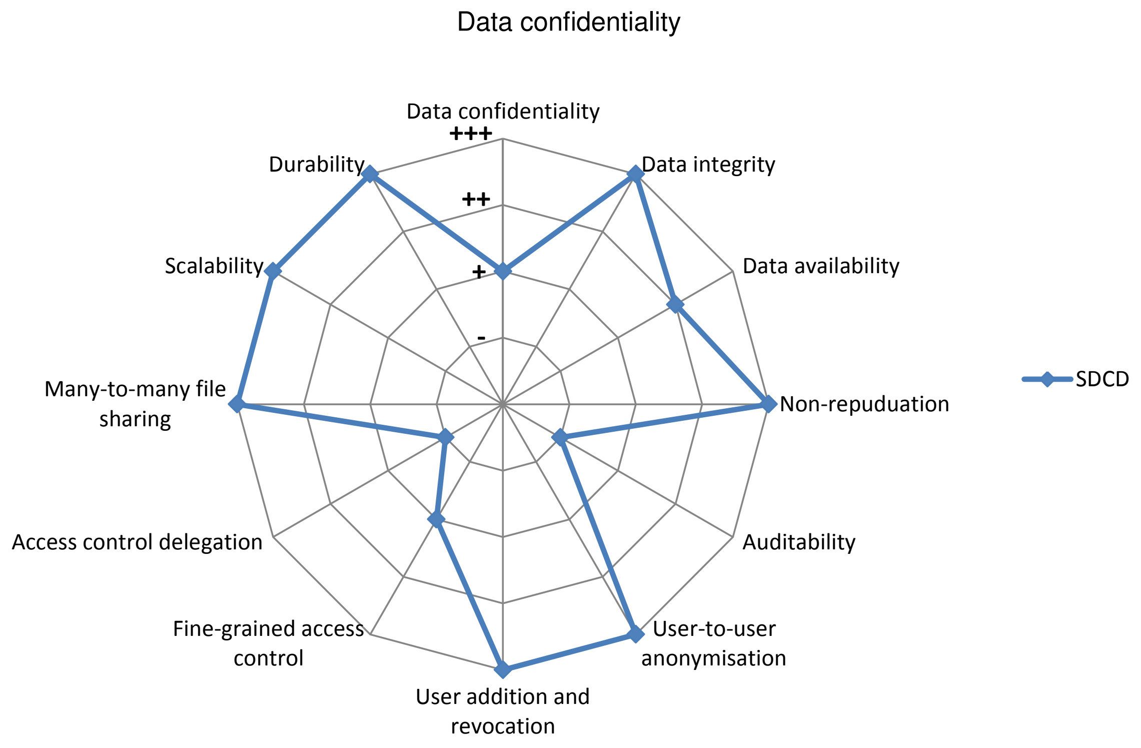


Figure 6.2: Graphical overview of how well SDCD supports the thesis goals.

6.2.2.1 Data confidentiality

The conceptual design of SDCD relies on multiple cryptographic building blocks and their combined ability to achieve confidentiality. In particular, the conceptual design of SDCD applies uni-directional PRE, Efficient Searchable Encryption (ESE), and hybrid encryption. The cryptographic building blocks and their properties are vital, but their internal workings are out of scope of this thesis. As a consequence, the SDCD prototype primarily uses third-party implementations and only resorts to custom implementations if no other options are available.

Proxy re-encryption: For the purpose of PRE the SDCD prototype uses a third party implementation of a uni-directional PRE scheme developed by Ateniese et al. [Ateniese&06]. This library has limitations which affect the SDCD prototype's ability to validate some of the properties supported by the conceptual design of SDCD. The implementation made by Ateniese et al. is available for academic use under the name: "The JHU-MIT Proxy Re-cryptography Library" (PRL) [Green&07]. PRL is at a very early stage of development and it is no longer actively maintained [Ateniese&12]. This resulted in a considerable effort to integrate PRL into the SDCD prototype. PRL requires that all instances of the library used in a system has an identical set of public cryptographic parameters. Keys and PRE tokens generated by one instance of the library cannot be used by another instance with different public cryptographic parameters. For this purpose, PRL supports serialising/deserialising of the parameters, but a flaw in the library's code prevents usage of the deserialised parameters. In the conceptual design of SDCD, the DO, all users, and the cloud itself performs PRE operations when performing actions, such as generating keys, decrypting, or re-encrypting data entities. Normally, the operations would be performed locally by generating a single set of public PRL parameters when initialising the system and distributing the parameters. The PRL limitations prevent this and as a consequence, a single PRL instance must be used in the SDCD prototype. This necessitates that the instance is located in the cloud.

Searchable encryption: The SDCD prototype uses the "Encrypt-and-Hash" scheme for the purpose of ESE [Bellare&07]. The SDCD prototype uses a custom implementation written by the authors of this thesis. The implementation combines the asymmetric encryption already provided by PRE with keyword hashing for generating search tokens. The custom implementation seems secure as it implements a proven secure scheme, but it has not been reviewed by cryptographic experts.

Hybrid encryption: The symmetric encryption scheme used for hybrid encryption is the "Advanced Encryption Standard" (AES) [FIPS-197]. The SDCD prototype uses the .NET framework's built-in support for AES which is considered secure. A data entity is encrypted by AES using a random key. This random key is in turn encrypted using the master public key.

A critical consequence of the PRL limitations is that all key pairs including the private keys at some point in time are available as plaintext inside the memory in the cloud and thus, the SDCD prototype suffers a breach of confidentiality.

6.2.2.2 Data integrity

The conceptual design of SDCD includes the ability to verify data integrity by using digital signing. For this purpose the SDCD prototype applies the "Digital Signature Algorithm" (DSA) which is an asymmetric digital signature scheme [FIPS-186-3]. The SDCD prototype uses the .NET framework's built-in support for DSA which in turn relies on the underlying operating system's implementation. This implementation is considered secure. When a data entity is created, its authoring user signs the data entity using his personal signing key. DSA enables the cloud to verify integrity of the data entity by using the signature associated with a data entity and the public part of the authoring user's signing key.

6.2.2.3 Data availability

Microsoft Windows Azure (MWA) has been chosen as the cloud environment for the SDCD prototype. MWA has been chosen because it delivers suitable cloud services and because it offers close integration with the .NET framework. This helped reduce the time needed to create the SDCD prototype. In MWA, a service may have multiple instances running inside distinct virtual machines. All service instances handle incoming service requests in cooperation and the individual instances can be reset, migrated to another data centre, or otherwise be temporarily unavailable.

As described earlier, the SDCD prototype has two cloud-based services: a storage service and a PRE service. In the SDCD prototype, they are both hosted by a single MWA service instance and the SDCD prototype does not attempt to handle migration or resets. As such, the SDCD prototype does not gain improved availability from using a cloud. Instead the availability of the SDCD prototype is comparable to that of a traditional server.

6.2.2.4 Non-repudiation

The SDCD prototype achieves non-repudiation by using the DSA algorithm to sign data entities as described above in the section considering integrity. As only a single user has access to the private key required to sign a data entity, only he can be its author.

6.2.2.5 User-to-user anonymisation

When a user receives a data entity from the cloud, the data entity contains a number of identifiers. One for itself and one for each of its attributes. In the SDCD prototype, identifiers for data entities, attributes, and users are based on Universally Unique IDentifiers (UUID) [Leach&05]. The identifier of a data entity is randomly generated. The identifier for each data entity attribute is generated by hashing the unencrypted keyword of the attribute.

For hashing, the SDCD prototype uses the hashing algorithm Message-Digest 5 (MD5) [Rivest92]. This hashing algorithm is no longer considered adequate for security sensitive purposes, but because of their size, its hash values are suitable for conversion to UUIDs. As with any hashing algorithm, hash values are not guaranteed to be unique. This means that two users with distinct user names could end up with identical identifiers. Despite this theoretical risk, the likelihood of collision is very low and is not considered by the SDCD prototype. Furthermore, the risk of identifier collision is not limited to identifiers generated by hashing, but is also present for identifiers which are randomly generated.

As with identifiers for data entity attributes, generating user identifiers by hashing user names ensures that identifiers do not leak any information regarding the user except equality. Because of this, support for user-to-user anonymisation is achieved. However, as the identifiers are deterministic, they may be used to analyse access patterns. The signature of a data entity and the public keys of other users' signing key pairs are kept in the cloud. This enables the cloud to verify the integrity of data entities while preserving user-to-user anonymisation.

6.2.2.6 User addition and revocation

The SDCD prototype supports addition and revocation of users. Furthermore, the PRE scheme used in the SDCD prototype allows the additions and revocations to be performed without affecting other users.

To verify integrity for data entities, the conceptual design of SDCD requires the public part of the signing key to be kept even after a user has been revoked. A limitation of the SDCD prototype is the lack of support for multiple keys for each user. This is an issue in the scenario where a user is added, then revoked, and then added again using the same identifier. All other aspects of user addition and revocation are supported.

6.2.2.7 Fine-grained access control

The conceptual design of SDCD does not have full support for fine grained access control. Either a user has full access or no access at all. In the SDCD prototype this is also the case. When a user is created, a PRE token is placed in the cloud and associated with the new user. In the SDCD prototype, the existence of the PRE token associated with a particular user determines whether he should have access or not. As such, coarse-grained access control is supported by the SDCD prototype.

6.2.2.8 Many-to-many file sharing

In the conceptual design of SDCD, usage of a PRE scheme is central for achieving support for many-to-many file sharing. Through the re-encryption, made possible by the PRL library, the SDCD prototype achieves support for many-to-many file sharing.

6.2.2.9 Scalability

In the conceptual design of SDCD, the DO is necessary only for user management. Once a user has been given his keys, he can access data entities independently of the DO. The SDCD prototype achieves scalability through the usage of a cloud which makes the dynamic collaboration environment and all the data entities stored within, available to other users without relying on the data owner.

6.2.2.10 Durability

MWA provides several ways to store data [Franks&12]. Some storage types are temporary and some are permanent. Each virtual machine that hosts a service instance has a local file-system and the SDCD prototype uses this file-system. Using the local file-system allows faster development as normal file system operations can be used. The other data storage types offered by MWA requires the use of specialised Application Programming Interfaces (API) [Richter12]. As the local file-system is temporary storage, data entities stored there are lost if the service instance is reset. As a consequence durability of the SDCD prototype is negatively affected and not as good as in the conceptual design of SDCD.

6.2.3 Summary

The SDCD prototype implements many aspects of the conceptual design of SDCD and in particular attention is paid to the complex cryptographic parts and their interactions as using a particular cryptographic element could prevent usage of another. The SDCD prototype consists of source code written in C, C++, and C# and is a distributed system running on multiple platforms. It contains complex components and the effort put into constructing the SDCD prototype was approximate two person-weeks. For a technical overview of the SDCD prototype, the interested reader is directed to Appendix B.

The SDCD prototype validates all the properties to the level supported by the conceptual design of SDCD except for three: confidentiality, availability, and durability. Furthermore the SDCD prototype omits several features which limit its real world usability. An example of this is that communication between the administration client, user client, and the cloud is not secured. Another example is that authentication is simulated i.e. authentication is based on user names without using passwords.

The limitations of PRL are the root cause of the most significant deviations from the conceptual design of SDCD. For production usage these limitations need to be fixed, but as of Spring 2012, we were unable to locate any other PRE implementation.

The limitations of PRE cause differences in communication flow and directly limits durability. If the PRL instance is lost, the stored data entities cannot be decrypted and they become inaccessible. As data entities become inaccessible by losing the PRL instance, no effort has been made to ensure that data entities are written to a persistent storage type. This affects durability and as a consequence, the SDCD prototype is not able to validate durability to the level supported by the conceptual design of SDCD.

As key and PRE token generation operations are performed in the cloud, confidentiality of all stored data is breached. However, the breach of confidentiality is caused by implementation details that given time, can be fixed. A workaround would be to perform all PRE operations on a trusted server outside the cloud, but this would not add significant value for the purpose of validating the conceptual design of SDCD.

Finally, availability normally benefits from utilising a cloud, but the SDCD prototype uses a single

MWA service instance and thus, does not gain improved availability. Instead the single instance is comparable to a single traditional server.

In spite of the deviations and omissions, the SDCD prototype still validates most of the conceptual design of SDCD. Table 6.1 is a comparison of how well each property of the conceptual design of SDCD of validated by the SDCD prototype and Figure 6.3 is a graphical presentation of the same comparison.

Table 6.1: Comparison of the SDCD prototype and the conceptual design of SDCD.

Environment		Category	Property	SDCD	SDCD prototype
Cloud	User	Security	Data confidentiality	+	-
	▨		Data integrity	+++	+++
	User		Data availability	++	+
	▨		Non-repudiation	+++	+++
			Auditability	-	-
			User-to-user anonymisation	+++	+++
		Access control	User addition and revocation	+++	+++
	User		Fine-grained access control	+	+
	▨		Access control delegation	-	-
			Many-to-many file sharing	+++	+++
		Performance	Scalability	+++	+++
			Durability	+++	+

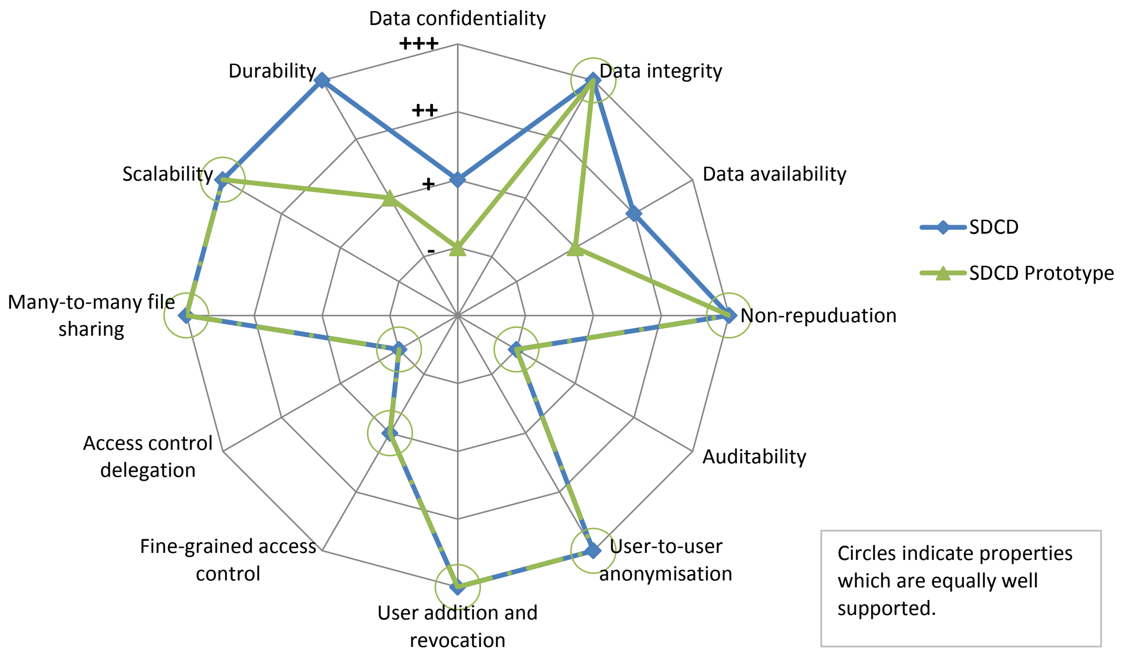


Figure 6.3: Graphical comparison of the SDCD prototype and the conceptual design of SDCD.

6.3. Validating Cloud-based Hierarchical Access Control

6.3.1 Overview

This section describes how the conceptual design of CHAC is validated by a prototype. The CHAC prototype builds upon the SDCD prototype. The CHAC prototype attempts to make as few modifications to the features of the original SDCD prototype as possible. The CHAC prototype focuses on adding features required to validate the properties which the conceptual design of CHAC improves compared to the conceptual design of SDCD. As the CHAC prototype builds upon the SDCD prototype, the limitations of the prototype are inherited. As will be described later, this influences the prototype's ability to validate the conceptual design of CHAC.
The existing clients from the SDCD prototype have been extended to support the data access control model described in the conceptual design of CHAC. The administration client still handles system initialisation and users and has been extended to manage roles. The user client is used when accessing data entities, but has been extended to support the concept of roles.
The cloud service contains the same two services as in the SDCD prototype: a storage service and a PRE service. However, the storage service has been extended to enforce data access rules. The PRE service is identical to the one in the SDCD prototype.
An overview of the CHAC prototype is depicted in Figure 6.4. In the figure, a DO has added two users: Alice and Bob. Alice have been given permission to manage roles and manages roles for Bob using the administration client. Bob does not have this permission and thus, does not use the administration client. In the figure, the DO uses only the administration client, but if he so desired, could use the user client to access data entities.

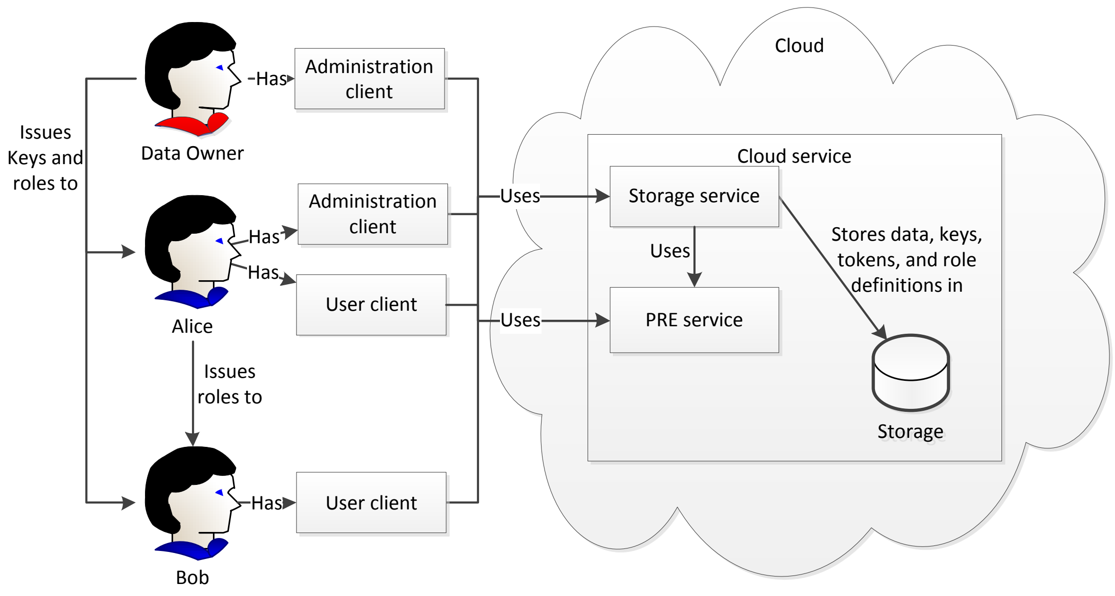


Figure 6.4: Overview of the CHAC proof of concept prototype.

6.3.2 Property validation

The CHAC prototype focuses on validating the properties which the conceptual design of CHAC improves upon. An overview of the properties and a comparison of the conceptual designs of

SDCD and CHAC can be seen in Figure 6.5. As can be seen from the comparison, the conceptual design of CHAC improves upon the conceptual design of SDCD for the properties: auditability, fine grained access control, and access control delegation.

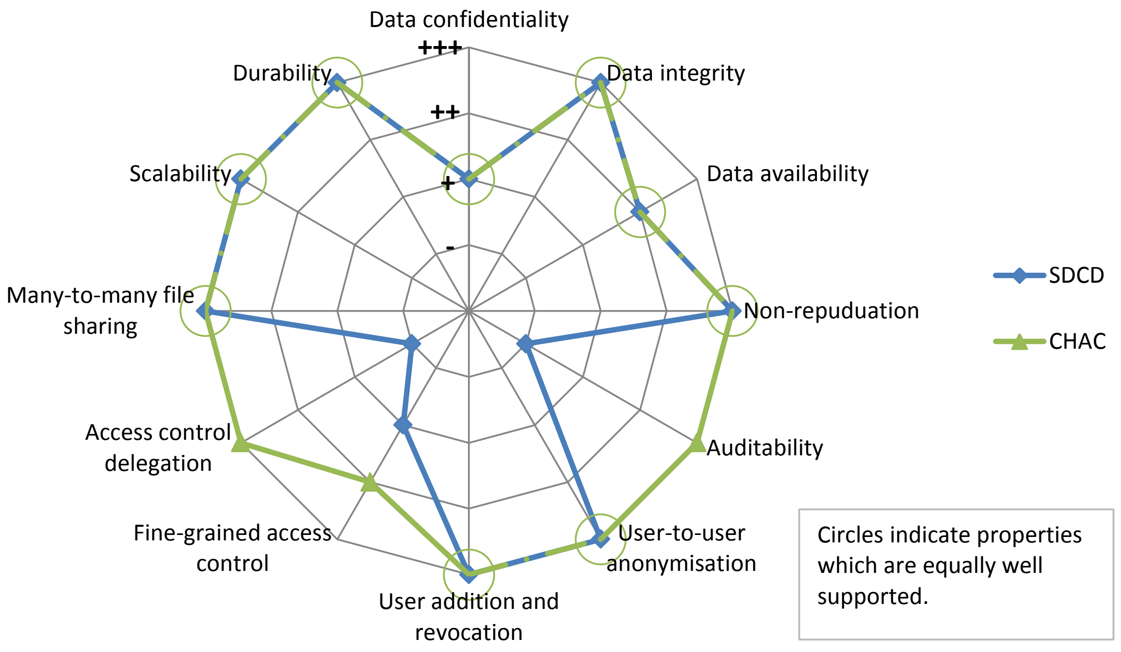


Figure 6.5: Graphical comparison of SDCD and CHAC.

The following sections focus on aspects of the CHAC prototype which influences the prototype's ability to validate the three properties where the conceptual design of CHAC improves upon the conceptual design of SDCD.

6.3.2.1 Fine-grained access control

The CHAC prototype has, with one exception, full support for the concept of a role as described in the conceptual design of CHAC. The flag which indicates whether a role is active has not been implemented. This flag is used for example to set roles as inactive for scenarios where leaks occur and those with access to the leaked data should be locked out of the system. Although this feature has not been implemented the remaining functionality based on roles is not influenced.

As in the conceptual design of CHAC, the CHAC prototype organises roles in a tree-based hierarchy. The rule stating that the permissions of sub-roles must always, except for root roles, be a subset of their parent's permissions has been implemented. This rule is enforced by the CHAC prototype when creating and updating roles. The prototype supports users, assignment and unassignment of roles and allows users to be assigned multiple roles at the same time. Likewise, the rules defining visibility and management of other roles and users, from the perspective of a particular user, is supported by the prototype.

Private hierarchies introduce extra complexity in the conceptual design of CHAC and the CHAC prototypes supports all the concepts and operations related to private hierarchies. This includes their creation and the rules defining situations in which they are deleted. The rules defining visibility and management of roles and users, from the perspective of a particular user, also takes private hierarchies into consideration. By supporting virtually all the features and operations related to roles and their hierarchies, the prototype supports fine-grained access control.

6.3.2.2 Access control delegation

The CHAC prototype makes it possible for the DO to add users. For each new user a new PRE key pair, signing key pair, and PRE token is generated. The generated PRE token is stored in the cloud. Users who have a role granting them permission to add new users are not able to generate new keys as described in the conceptual design of CHAC. Instead, their own keys are shared with the new user. This is due to a limitation in the PRL implementation of PRE inherited from the SDCD prototype. In general, PRE schemes can be transitive, but PRL does not support this. As a consequence, even if a key pair and a matching PRE token were to be generated, the cloud would not be able to re-encrypt multiple times as required by the conceptual design of CHAC. With respect to the thesis goals, this means that the DO is not able to delegate all his capabilities and thus, the CHAC prototype's support for access control delegation is limited.

6.3.2.3 Auditability

The CHAC prototype does not support auditability. Implementing support for auditability would require logging all accesses and modifications of data entities, roles, and users. The logs would have to be encrypted using the master key pair and stored in the cloud. Finally, client support would have to be built to enable users who are assigned to root roles to inspect the logs. Implementing these steps would require a significant effort which outweighs the benefits in terms of validating that part of the conceptual design of CHAC.

6.3.3 Summary

The CHAC prototype focuses on access control and in particular management and delegation of access control. The features which are built on top of the SDCD prototype, forming the CHAC prototype, consist of definitions and associated logic for managing roles, users, and data entities. In total, the effort put into constructing the CHAC prototype is about one person-week. For a technical overview of the CHAC prototype, the interested reader is directed to Appendix B.
The concept of roles, defined in the conceptual design of CHAC, makes multiple scenarios possible and the CHAC prototype supports most, but not all of them. Auditability is not supported and thus, this aspect is not validated. The prototype also inherits the limitations of the SDCD prototype and its inability to fully validate confidentiality, availability, and durability. The extent to which the CHAC prototype validates each of the properties of CHAC can be seen in Table 6.2. A graphical presentation of the same comparison is shown in Figure 6.6.

Table 6.2: Comparison of the CHAC prototype and the conceptual design of CHAC.

Environment		Category	Property	CHAC	CHAC prototype
Cloud	User	Security	Data confidentiality	+	-
	▨		Data integrity	+++	+++
	User		Data availability	++	+
	▨		Non-repudiation	+++	+++
			Auditability	+++	-
			User-to-user anonymisation	+++	+++
		Access control	User addition and revocation	+++	+++
	User		Fine-grained access control	++	++
	▨		Access control delegation	+++	++
			Many-to-many file sharing	+++	+++
		Performance	Scalability	+++	+++
			Durability	+++	+

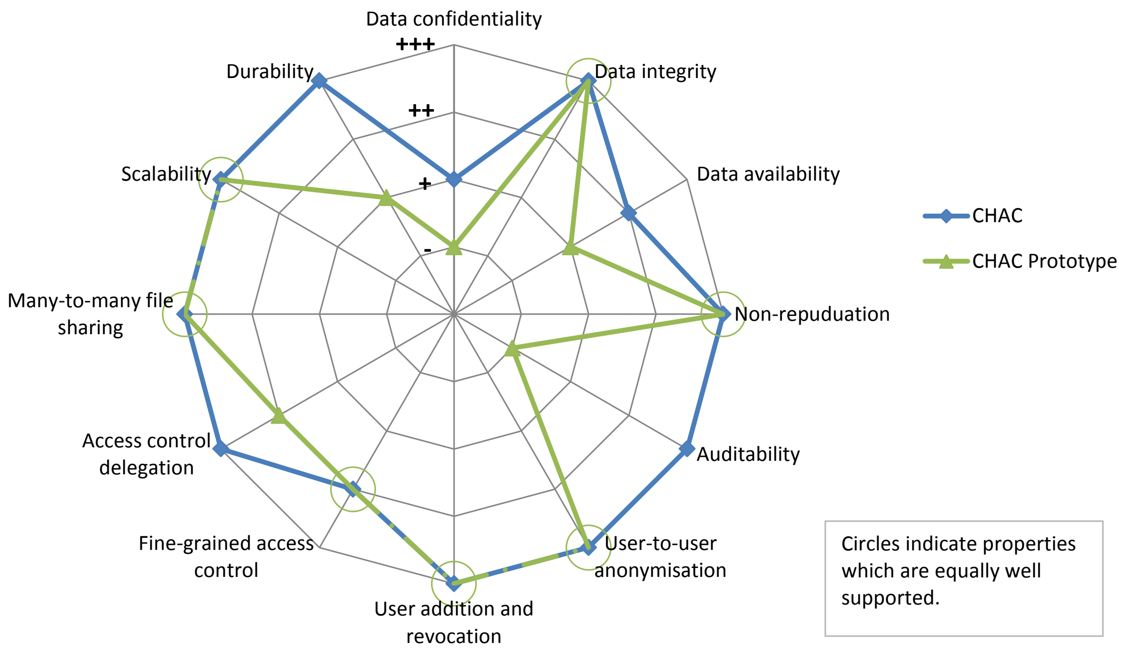


Figure 6.6: Graphical comparison of the CHAC prototype and the conceptual design of CHAC.

6.4. Discussion

As part of this thesis, a prototype has been constructed to validate the conceptual design of SDCD and another prototype has been constructed to validate the conceptual design of CHAC. The focus of the prototypes has been to validate as much of the conceptual designs of SDCD and CHAC as possible. This means that the prototypes attempt to validate all properties to the level supported by their respective conceptual designs. The prototypes succeeded in doing so, with a few exceptions.

Chapter 7

Concluding remarks and future work

This chapter concludes on the results achieved in this thesis. The thesis goals defined in Chapter 1 are related to the conceptual designs presented in Chapters 3, 4, and 5. Together the conceptual designs comprise the results of this thesis.

7.1. Introduction

This chapter concludes on the conceptual designs presented in earlier chapters and on the thesis in general. This includes elaboration on areas where the conceptual designs can be improved. This chapter also contains reflections on our personal learning outcomes.
In Section 7.2, the achieved results are presented and discussed with regards to the thesis goals and the theory validation presented in Chapter 6. Future work, expanding on each of the three conceptual designs, is outlined in Section 7.3. Our personal learning outcomes are described in Section 7.4, detailing the process of authoring this thesis. Finally Section 7.5 concludes the chapter and thesis.

7.2. Achieved results

The initial goals of this thesis, as presented in Chapter 1, were:

1. To design a dynamic collaboration environment utilising the benefits of cloud storage while ensuring strong data security and fine-grained data access.

2. To improve our skills and knowledge with regards to designing systems that leverage the benefits of the cloud, and improve our ability to research a scientific subject from different perspectives and contribute to the scientific community.

To be able to design a secure dynamic cloud-based collaboration environment with support for fine-grained data access, it was essential to study existing solutions and approaches. Chapter 2 presents information about security related concepts and technologies relevant for developing a design satisfying the thesis goals.
The design for the envisioned *secure dynamic cloud-based collaboration environment with hierarchical data access* is comprised of the contribution of three designs, each being an extension of

the former, thus resulting in a simple dependency graph as depicted in Figure 7.1. The designs focus on different aspects of the goals as they are described in Chapter 1, two of which have been further validated by proof of concept prototypes. Each design is assessed in its own subsection below and finally a section is dedicated to a comparison and evaluation of the designs.

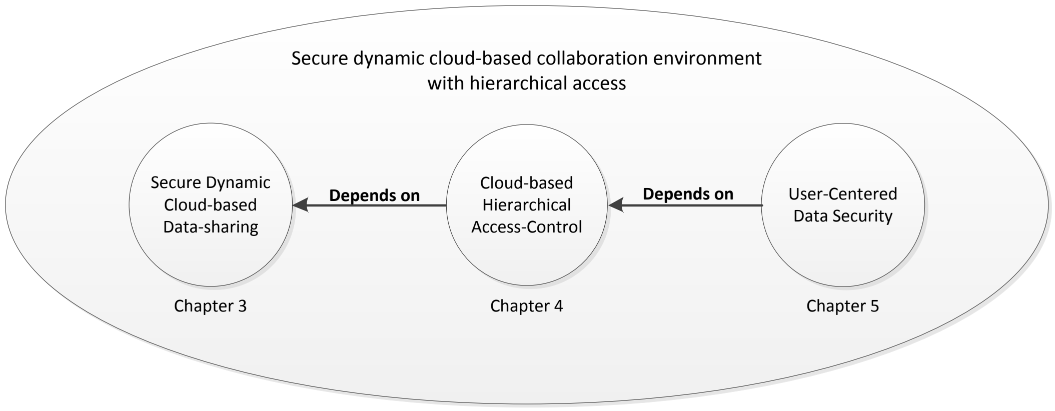

Secure dynamic cloud-based collaboration environment with hierarchical access
Secure Dynamic Cloud-based Data-sharing
Depends on
Cloud-based Hierarchical Access-Control
Depends on
User-Centered Data Security
Chapter 3
Chapter 4
Chapter 5

Figure 7.1: Dependencies of the designs.

7.2.1 Secure Dynamic Cloud-based Data-sharing

In Chapter 3 our conceptual design for *Secure Dynamic Cloud-based Data-sharing (SDCD)* used for storing data securely in the cloud was presented. The design employs a combination of symmetric and asymmetric cryptography to ensure confidentiality. When data is submitted and stored in the cloud, it is encrypted by the same master key regardless of who submits it. This master key is generated by a Data Owner (DO) when the system is initialised. The DO is responsible for the system and is the only entity who can manage users and verify integrity.

Integrity and non-repudiation is ensured by using digital signatures. All users have individual signing keys which are used when submitting data to the cloud. The digital signature allows the DO to verify integrity and verify the identity of the author.

The concept of Proxy Re-Encryption (PRE) is applied to support many-to-many file sharing while retaining the ability to add and revoke users. This allows all users to submit data encrypted by the same key while receiving and decrypting data using a user specific key. Furthermore, a user specific key can be revoked without adversely affecting the other users, by removing a PRE token from the cloud. User-to-user anonymisation is achieved by restricting the capabilities of users to only encompass data access and data integrity checking. None of these operations reveal the identity of other users. All other operations such as adding new users are available to the DO only.

When a user has been granted access and has received his keys, no further communication between the DO and the user is required. This provides scalability and availability as no bottleneck is introduced and the user communicates directly with the cloud. Availability is also improved by using a cloud as it has numerous service instances located at geographically distributed locations.

Although SDCD supports many of the properties defined as goals in this thesis, some properties are notably missing. These are fine-grained access control, access control delegation, and auditability. There is no concept of access control in SDCD and all authorised users are always able to access all data. Likewise there is no concept of auditability.

In SDCD the focus is on ensuring data security while utilising the benefits a cloud envivonment provides. SDCD effectively ensures data security and prevents the Cloud Storage Provider (CSP) from gaining access to stored data in its unencrypted form. However, once data is in the users' local environments security can no longer be maintained.

The concept of PRE, which is applied by SDCD, is vulnerable to collusion where a user and the CSP can collude to obtain information about data that they could not access on their own. This is an unsolved problem with PRE, however, this liability is made harder to exploit because nothing in the cloud can be used to identify users. Although communication between users and the cloud is assumed to be adequately protected a CSP can still, in theory, identify users by other means such as monitoring the destinations of outgoing data.

The SDCD conceptual design is validated by a proof of concept prototype. The prototype validates SDCD by implementing many of its features, and focuses on the complex cryptographic parts. Although the prototype has limitations and deviates from SDCD in a number of places, these are mostly caused by limitations in a third party PRE library. However, these limitations do not invalidate the SDCD design. In summary, the proof of concept prototype strengthens the argument that SDCD is able to fulfill many of the thesis goals.

7.2.2 Cloud-based Hierarchical Access Control

Chapter 4 presented a design named *Cloud-based Hierarchical Access Control (CHAC)*. CHAC is a customised Role-Based Access Control (RBAC) model with SDCD at its core and enables support for fine-grained data access. An auditability scheme has been introduced, in addition to the access control model, adding the security aspect of auditing to the design.

CHAC is structured as a hierarchy, achieving a manageable structure that can easily be mapped onto job or usage functions in an organisational structure. Our initial designs had a user hierarchy, existing in parallel with the role hierarchy, to enable a structural dependency between users. However, this approach was deemed infeasible due to the complexity of maintaining and continuously validating two different hierarchies with intertwining dependencies. To simplify the access control model, the user hierarchy was abstracted away and only the role hierarchy remains, with the users now in a flat structure, or a *pool* of users.

Private role hierarchies were introduced to the access control model to enable addition of independent hierarchies. However, caution must be taken when managing roles and users in the hierarchy from which the new hierarchies originate. This is due to the cascading effect of deleting the creator role of an additional hierarchy or the user that created it. This could potentially result in the deletion of the entire hierarchy.

An auditability scheme has been included as part of the CHAC design along with the access control model. Auditability is considered a valuable feature as it allows designated users to survey the behaviour of other users and monitor how entities evolve over time in the collaboration environment. In its current form CHAC only allows root roles to audit. This is due to the fact that being able to audit contradicts the general concept of anonymisation. If users in the lower tiers of the hierarchy had permission to audit, anonymisation would be violated.

The current life-cycle management of the PRE key pair hierarchy is based on the relationship between key pairs in the form of PRE tokens. To allow a user in a lower tier of the hierarchy to decrypt data, the encrypted data must first go through a chain of re-encryptions to pass down through the hierarchy ending at the user. As the collaboration environment evolves over time, the PRE key pair hierarchy potentially becomes very deep. This is undesirable as the additional re-encryptions result in an additional computational overhead and expose structural information about the hierarchy in the form of re-encryption duration. This information could be used for *timing attacks*

which could potentially reveal cryptographic details on how data is protected [Kocher96]. This is a subject for future work.
The CHAC theory is validated through a proof of concept prototype which is based on the prototype validating SDCD. The auditability aspect is the only part not included in the prototype as this adds little value to the argument that CHAC is a feasible approach for a design conforming to the thesis goals. As the CHAC prototype is based on the SDCD prototype, the CHAC prototype is subject to the same third party PRE library limitations as SDCD. However, this remains an implementation specific detail and does not invalidate the prototype.

7.2.3 User-Centered Data Security

Chapter 5 presented a design that extends CHAC to encompass the local environment of the users and is named *User-Centered Data Security (UCDS)*. To achieve security in a user's local environment, UCDS applies Digital Rights Management (DRM). Several commercial DRM schemes exist. However, due to their proprietary nature, none were considered suitable as they lack support for arbitrary resource types and extending an existing solution is infeasible, again due to the proprietary nature of DRM solutions. Furthermore, interoperability in DRM systems is problematic, resulting in integration with the existing access control model of CHAC becoming very challenging.
Instead of using an existing DRM system, UCDS proposes the development of a custom system allowing for integration with the access control model of CHAC. The definitions of licenses and rights could even be extended to encompass the definitions used in CHAC. Furthermore, the encryption schemes of SDCD can be reused in place of conventional DRM encryption. Data persisted in a user's local environment is stored in its encrypted form when it is received from the cloud and accessed by a trusted DRM client. To enable this functionality a custom DRM client is essential.
When a user is revoked from the system his status in the cloud changes. To ensure the revocation is cascaded to the user's local environment, the DRM client re-authorises the user. However, this means that a user must have access to the cloud to access local data using the DRM client, resulting in usability being severely affected. This is solved by UCDS with the support for offline data access, where access to data is granted without having to re-authorise the user. Offline access and the ability to perform a user revocation in a user's local environment are two conflicting properties. A particular usage scenario must weigh the needs for the properties and adjust accordingly. UCDS proposes a solution to this through a scheme imposing time constraints on data, where the user is forced to periodically re-authenticate with the cloud to ensure continued access to data when offline.
For the scenario where data is leaked the DRM scheme supports watermarking. However, watermarking cannot help in tracking leaked data, it can only help with identifying the source of a leak. This assumes, however, that the data is first discovered where it is not supposed to be.
No proof of concept prototype has been built to validate UCDS. This decision was made because a proof of concept prototype for this design was deemed to add little value compared to the effort required for its creation.

7.2.4 Comparison and evaluation

The three conceptual designs focus on different aspects of the thesis goals, where each subsequent design improves the former incrementally supporting more of the thesis goals than the former design provided by itself. Table 7.1 shows a comparison between SDCD, CHAC, and UCDS. A graphical overview of the same comparison can be seen on Figure 7.2. Notice that confidentiality is not supported to the highest level defined in the rating scale in Chapter 1. This is only achievable by ensuring that the CSP is unable to deduce any information about the requested data, for example by performing statistical analysis. To hide access patterns a technique for Private Information Retrieval (PIR) exists [Ishai&06]. However, this technique has not been used in our work for efficiency concerns. Any PIR-based technique needs to access all the data which is available to a user in the cloud. This is very inefficient in any large scale system [Cao&11]. Had the technique been included, it would have prevented full utilisation of the performance offered by the cloud. Thus, the reason for utilising a cloud for the dynamic collaboration environment would have been invalidated.

Table 7.1: Comparison of SDCD, CHAC, and UCDS.

Environment		**Category**	**Property**	**SDCD**	**CHAC**	**UCDS**
Cloud	User	Security	Data confidentiality	+	+	++
			Data integrity	+++	+++	+++
	User		Data availability	++	++	+++
			Non-repudiation	+++	+++	+++
			Auditability	-	+++	+++
			User-to-user anonymisation	+++	+++	+++
		Access control	User addition and revocation	+++	+++	+++
	User		Fine-grained access control	+	++	+++
			Access control delegation	-	+++	+++
			Many-to-many file sharing	+++	+++	+++
		Performance	Scalability	+++	+++	+++
			Durability	+++	+++	+++

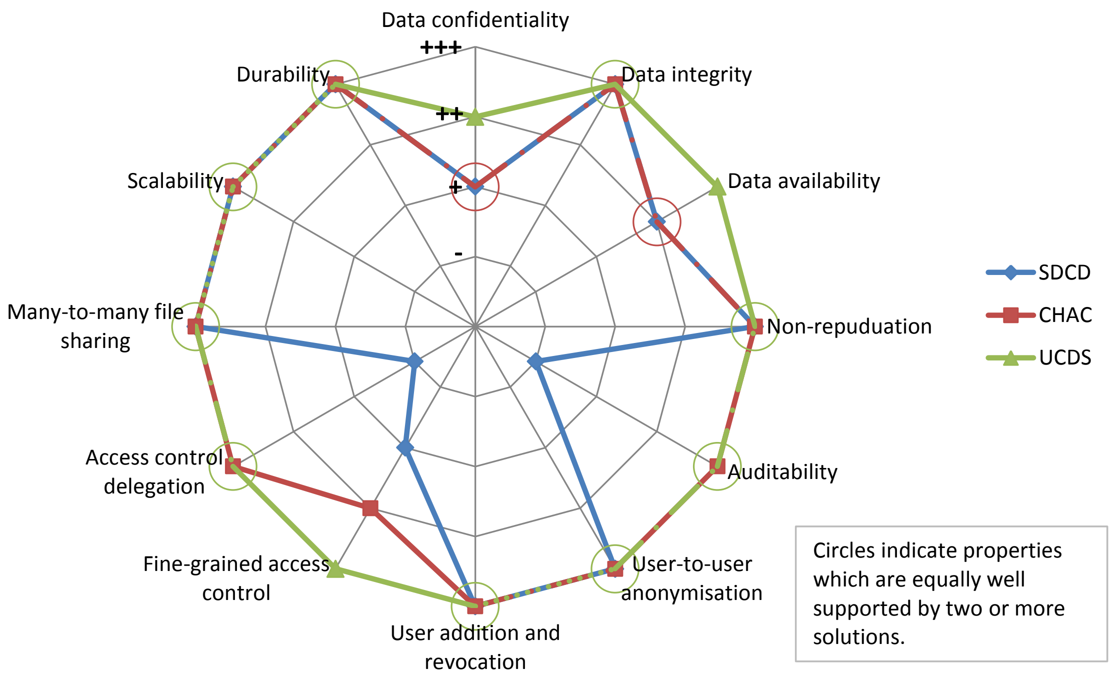


Figure 7.2: Graphical comparison of SDCD, CHAC, and UCDS.

7.3. Future work

During the progress of this thesis several interesting areas for further work have been identified. The areas are divided into sections for SDCD, CHAC, UCDS, and one for areas which do not directly fit into any of the three. For some of them, suggestions on how they can be approached are given.

7.3.1 Secure Dynamic Cloud-based Data-sharing

Currently users have two destinct key pairs, one for signing data and one for enabling proxy encryption. The key pair used for proxy encryption has been generated to have certain cryptographic properties, but this key pair might also be suitable for digital signing. Using a single key pair for both operations would simplify key management and distribution. A user's public key is required to be located in the cloud to enable the cloud to verify integrity. However, it is unknown whether the CSP is able to deduce the master private key from the user's PRE token and the master public key. Additional research is necessary to resolve this.

Another area for further study is improving the capability of the dynamic collaboration environment to handle multiple simultaneous DOs. This should include mechanics to revoke a DO's master key pair once a DO leaves the collaboration environment, without adversely affecting other authorised entities which access the environment. The potential benefit is a truly dynamic collaboration environment where all parties are made equal by removal of the constraint of having a single entity in complete control. Each DO will be independent from each other with the same properties and abilities as the DO in the original conceptual design thus, no liabilities or violations of anonymisation will be introduced from this approach in future works.

7.3.2 Cloud-based Hierarchical Access Control

7.3.2.1 Improved confidentiality

Introducing the access control model with its role hierarchies, users, and data entities requires that information on these entities are stored in the cloud. Information such as user and role names are encrypted while residing in the cloud, but to function correctly the cloud must be able to identify all entity types and thus, the identifiers of roles, users, and data entities cannot be encrypted. Although identifiers reveal nothing by themselves the structure of users, roles, and data entities contain many identifiers and by examining the structure itself, information might be deduced. To improve confidentiality, the CSP should be prevented from deducing information from the structure.

When users access data stored in the cloud, data confidentiality is ensured by having data encrypted until it reaches the users' environments. To further improve confidentiality of data stored in the cloud, the CSP should be prevented from analysing the access patterns of users. Access patterns can be statistically analysed to find tendencies which in turn may disclose information.

A potential solution to resolve this problem could be the introduction of an additional cloud acting as a proxy, redirecting user requests to the actual cloud storage. This could potentially hide the identity of users requesting data though it is still uncertain if this will suffice to hide all details of the origin of the user request as the storage cloud would still need to validate the request through its access control model.

The aspect of Rule Based Role-Based Access Control (RB-RBAC) [Al-Kahtani&02] which allows automated role delegation is an interesting aspect that would be worth considering as a future addition to CHAC. It could potentially reduce the time spent managing and delegating roles and allow users to focus on other activities.

7.3.2.2 Dedicated audit roles

By adding auditability to CHAC, selected users are able to audit actions and events associated with users, roles, and data entities. The current concept only allows auditing by users with root roles assigned. As a consequence, users who need to perform audits are also assigned all the permissions of a root role. As a future improvement, it is desirable to separate the ability to perform audits from the root roles.
From a usage perspective, an ideal solution would be to have the concept of a dedicated audit role. The audit role would exist at the same level in the role hierarchy as root roles, but with read-only access to all data. This way users assigned to the auditing role would have the required knowledge of the entire role hierarchy and the entities within to perform effective audits while at the same time being prevented from modifying anything.

7.3.2.3 Improved life cycle management

The life cycle of CHAC involves creation of multiple PRE tokens and key pairs. When a user adds another user, the number of re-encryptions required for the new user to access data entities grows. This influences performance. A potential way to mitigate this is by performing “structural cleanups”. If the root of the hierarchy is no longer needed its children are examined recursively. If only one branch connected to the root of the hierarchy is needed, then the root and its other children can be removed. This should be transparent to the user. However, the challenge of this approach is to determine when a key pair is no longer needed. A different approach for solving this issue could be a many to many relationship between key pairs. However, this would result in an exponential amount of PRE tokens for each user in the system, which in turn would severely impact performance and scalability in terms of numbers of users. Therefore this approach is not recommended.
Another issue in CHAC is that a single user will have many key pairs if the user has been added multiple times. Each of the key pairs have an associated PRE token stored in the cloud. When re-encrypting a data entity for the user, the cloud needs to select one of these PRE token. A potential solution is to create an explicit association between a key pair and a PRE token in the cloud. The association is established when the key pair and PRE token is created, for instance by a single randomly generated identifier. The identifier is saved in the cloud along the PRE token and given to the user alongside his new key pair. When the user accesses data entities in the cloud, the identifier of the key pair he wishes to use is sent to the cloud and indicates the PRE token to use.
Finally, multiple signing keys exist for a user which has been added multiple times. When verifying integrity, the cloud needs to determine which signing key pair to use. When verifying integrity the issue of selecting the correct signing key can be solved by associating each data entity directly to its corresponding signing key instead of the authoring user.
The issues described regarding life cycle management are not addressed in the design of CHAC nor the CHAC prototype. Improving upon the design of CHAC to solve the issues, for instance by the approaches suggested above is a subject for future work.

7.3.3 User-Centered Data Security

7.3.3.1 Supporting resource types

Directions of future work for UCDS includes researching a new approach towards achieving better and wider resource type support in the DRM client. This is currently the main issue in UCDS. Each resource type requires individual implementation to support viewing and editing through the client, which drastically lowers usability. We suggest an implementation approach using a controlled virtual machine (VM) to act as a DRM client. The VM would run an entire operating system thereby being able to support and run commercial applications originally associated with the corresponding resource. As an example, Microsoft Word could handle its associated textural resource types within the VM. This would require the VM to be sealed off so information cannot be extracted from the VM except when sending data to the cloud. This approach would improve usability due to the different resource types being associated with their corresponding applications.

7.3.3.2 Prototype Implementation

Although the required effort is substantial, it would be valuable to validate the theory and concepts of UCDS by a proof of concept prototype. If the existing prototypes of SDCD and CHAC are used as the foundation, UCDS could be integrated with the existing cryptographic schemes and access control model.

7.3.4 General improvements

A rather large, yet potentially significant improvement of the dynamic collaboration environment would be the ability to control access to individual parts of a given data resource. This increase in access control granularity allows collaboration at an even more detailed level. Specifically, this would allow different access rights to different sections in documents.

7.4. Personal learning outcomes

Working on this thesis has been a first-time experience in conducting research in academia for all the authors of this thesis. Through the thesis, the authors have gained the ability to get a quick overview and extract important details of scientific research papers. This ability is essential as some research papers are of great value while others provide little to no useful information. One of the primary personal learning outcomes of the thesis is that the authors now have a much deeper knowledge of what a cloud is, what its properties are, and how they can be utilised.
The acquired knowledge of cryptography and in particular the cryptographic schemes used in the thesis will undoubtedly prove invaluable in the future. This knowledge is relevant in the current IT landscape where security is an increasing concern.

7.4.1 Consequences of being a group

Being a three-person group required focus on aligning the individual understandings and perceptions of the thesis goals to make sure it was properly reflected throughout the thesis. This required additional rework and rewriting of chapters. The extent of this thesis also required hard work and coordination while adapting to English as the written language.

Group discussions have been held several times during the thesis to make sure all agreed on the common goal and that the best possible solution or approach was chosen. Aligning the expectations of three people increased the workload, but having the time and resources of three people available for proofreading, reviewing, and general quality improvement has been invaluable. Especially, as reviews performed by others than the main author often tend to be more thorough.

To better support three persons simultaneously working on the thesis, the work was parallelised and each author was assigned primary areas of responsibility. This helped optimise the work effort and has helped maintain good progress throughout the thesis.

In addition to our internal thesis group discussions, weekly meetings have been conducted with attendance of the thesis supervisor and co-supervisor when possible. This has provided valuable feedback on the written work and suggested approaches for the further progress on the thesis. Through these meetings questions and uncertainties were addressed and clarified. The meetings also provided a good foundation for keeping the supervisor and co-supervisor up-to-date regarding the progress of the thesis. Additionally, these meetings provided new perspectives on topics in the thesis because of the supervisors' experience and different perspectives. This taught us to direct our attention to matters and approaches, that had not previously received sufficient attention, and encouraged us to explain concepts in a simpler and more clear manner.

7.4.2 Shifting focus

The main focus of the thesis has changed a lot since the initial work was conducted. Some of the challenges originally expected had already been completely or partially solved in research works by others. Thus, the focus shifted towards solving the shortcomings of these research works by combining different approaches into a single and more refined solution. This has been necessary to distance our work from what has already been researched and published. The original goal of achieving data security in a cloud environment through architectural approaches, while preserving other software qualities, has not changed. However, techniques for achieving the main goal have become more refined as knowledge has been gained by analysing related research and by validating the presented designs by creating proof of concept prototypes.

Choosing cryptography to achieve data confidentiality and integrity required the acquisition of extensive knowledge on this subject. Although extensive insight on the mathematical algorithms operating behind the different cryptographic schemes was not necessary, it was necessary to fully understand the key management and general terms and properties of the schemes. This included understanding the abilities and liabilities associated with each cryptographic scheme to find the optimal combination. For example, the choice regarding the use of symmetric or asymmetric encryption and how the decision of using either would impact the result.

7.4.3 Unforeseen synergies

During development of the proof of concept prototype for validating SDCD, an issue emerged. A third party library implementing the vital PRE concept was originally intended to be used to perform re-encryption of data when it is first put into the cloud and again when it is transferred from the cloud to the user requesting the data. The third party library implemented non-transitive proxy encryption and was only capable of re-encrypting data once. This rendered the proof of concept prototype unable to validate the original design of SDCD. This resulted in brainstorming on how to overcome the issue. The solution to the re-encryption issue consisted of changing the way the proxy encryption scheme was being used in SDCD. This, in turn, opened up for the possibility of allowing data integrity checks to be performed by the cloud. The lesson learned from this is that it is not always obvious how and which processes end up influencing a solution. To get the best possible solution multiple processes need to be performed and multiple perspectives need to be considered.

7.4.4 An engineering perspective

Through the education of becoming masters in software engineering, we have been trained to divide a seemingly impossible problem into smaller solvable parts. This methodology has been applied throughout the work on this thesis.
As an example, to define and describe CHAC, initially the thesis goals were considered to determine the purpose and contribution of CHAC to the thesis. Several concepts such as role-based access were deemed suitable and they were combined into a single solution. This solution was complex and perhaps "over-engineered", but after a few iterations it was simplified to what CHAC currently is. The current design is simpler and also easier to communicate while still fulfilling the intended thesis goals. This process has added to the skill of analysing requirements and context to come up with an appropriate solution.

7.5. Final remarks

The goals of this thesis were to design a dynamic collaboration environment leveraging the benefits of a cloud environment, and to improve our skills and knowledge on this subject. We are pleased with this thesis successfully meeting the specified goals. The only shortcoming being the lack of support for confidentiality at the highest defined level. Achieving the highest level of support has been noted for future work.
The dynamic collaboration environment has been built in increments to constitute a complete solution. By splitting up the collaboration environment into multiple sub-solutions, in essence, three distinct designs have emerged. Furthermore, the properties of the dynamic collaboration environment are supported by a generic case and as such, the designs are generic in nature and bring value to many different scenarios. Having multiple distinct and generic designs enable a potential organisation in need of a dynamic collaboration environment to select the solution which best matches its needs.
The conceptual design of the dynamic collaboration environment has been constructed through the use of many different techniques and approaches. By working with the subject for an extensive period, we have become confident in our skills and our acquired knowledge of performing scientific research, presenting solutions, and communicating designs and concepts.
In conclusion, using a cloud holds great potential as an on-line and practically always available storage platform, but before widespread storage of sensitive data can begin, security concerns need to be addressed. This thesis presents a solid design addressing these concerns.
Through the points raised in the future works section we sincerely hope that others will incorporate and improve upon our designs, giving this field of research the attention it deserves.

References

[Agudo&11] Agudo, Isaac and Nuẽz, David and Giammatteo, Gabriele and Rizomiliotis, Panagiotis and Lambrinoudakis, Costas. Cryptography Goes to the Cloud. In Lee, Changhoon and Seigneur, Jean-Marc and Park, James J. and Wagner, Roland R., editors, *S*ecure and Trust Computing, Data Management, and Applications, pages 190–197, Springer Berlin Heidelberg, 2011. [cited at p. 16]

[Al-Kahtani&02] Al-Kahtani, M.A. and Sandhu, R. A model for attribute-based user-role assignment. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 353 – 362, 2002. [cited at p. 41, 94]

[Armbrust&09] Armbrust, M. and Fox, A. and Griffith, R. and Joseph, A.D. and Katz, R.H. and Konwinski, A. and Lee, G. and Patterson, D.A. and Rabkin, A. and Stoica, I. and others. *A*bove the clouds: A berkeley view of cloud computing. Technical Report, Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, 2009. [cited at p. 1, 2, 3]

[Ateniese&06] Ateniese, Giuseppe and Fu, Kevin and Green, Matthew and Hohenberger, Susan. Improved proxy re-encryption schemes with applications to secure distributed storage. *A*CM Trans. Inf. Syst. Secur., 9:1–30, February 2006. [cited at p. 18, 19, 79, 121]

[Ateniese&12] Ateniese, Giuseppe and Fu, Kevin and Green, Matthew and Hohenberger, Susan and Olsen, Martin and Jensen, Adam and Piechotta, Chris. Proxy Re-Cryptography Library. An email conversation betweeen the authors of PRL and the authors of this thesis. March 2012. Correspondence occured on March 17 2012. [cited at p. 79]

[Bellare&07] Bellare, Mihir and Boldyreva, Alexandra and O'Neill, Adam. Deterministic and efficiently searchable encryption. In *P*roceedings of the 27th annual international cryptology conference on Advances in cryptology, pages 535–552, Springer-Verlag, Berlin, Heidelberg, 2007. [cited at p. 17, 80]

[Blaze&98] Blaze, Matt and Bleumer, Gerrit and Strauss, Martin. Divertible protocols and atomic proxy cryptography. In Nyberg, Kaisa, editor, *A*dvances in Cryptology - EUROCRYPT'98, pages 127–144, Springer Berlin / Heidelberg, 1998. [cited at p. 18, 19]

[Boneh&04] Boneh, Dan and Di Crescenzo, Giovanni and Ostrovsky, Rafail and Persiano, Giuseppe. Public Key Encryption with Keyword Search. In Cachin, Christian and Camenisch, Jan, editors, *A*dvances in Cryptology - EUROCRYPT 2004, pages 506–522, Springer Berlin / Heidelberg, 2004. [cited at p. 17]

[Brainard&06] Brainard, John and Juels, Ari and Rivest, Ronald L. and Szydlo, Michael and Yung, Moti. Fourth-factor authentication: somebody you know. 168–178, 2006. [cited at p. 14]

[Cao&11] Ning Cao and Cong Wang and Ming Li and Kui Ren and Wenjing Lou. Privacy-preserving multi-keyword ranked search over encrypted cloud data. In *I*NFOCOM, 2011 Proceedings IEEE, pages 829–837, april 2011. [cited at p. 17, 93]

[CertiVox&12] CertiVox. MIRACL Crypto SDK. `http://certivox.com/index.php/solutions/miracl-crypto-sdk/`, 2012. Accessed Apr 19 2012. [cited at p. 121]

[Chang&05] Chang, Yan-Cheng and Mitzenmacher, Michael. Privacy Preserving Keyword Searches on Remote Encrypted Data. In Ioannidis, John and Keromytis, Angelos and Yung, Moti, editors, *A*pplied Cryptography and Network Security, pages 391–421, Springer Berlin / Heidelberg, 2005. [cited at p. 17]

[Chen&10] Lanxiang Chen and Shuming Zhou. The comparisons between public key and symmetric key cryptography in protecting storage systems. In *C*omputer Application and System Modeling (ICCASM), 2010 International Conference on, pages V4–494 –V4–502, oct. 2010. [cited at p. 16]

[Cox&01] Ingemar J. Cox and Matthew L. Miller and Jeffrey A. Bloom. *D*igital Watermarking: Principles & Practice, 2001. Morgan Kauffman Publishers, San Francisco, October 2001. [cited at p. 23]

[Curtmola&06] Curtmola, Reza and Garay, Juan and Kamara, Seny and Ostrovsky, Rafail. Searchable symmetric encryption: improved definitions and efficient constructions. In *P*roceedings of the 13th ACM conference on Computer and communications security, pages 79–88, ACM, New York, NY, USA, 2006. [cited at p. 17, 25]

[Curtmola&11] Curtmola, Reza and Garay, Juan and Kamara, Seny and Ostrovsky, Rafail. Searchable symmetric encryption: Improved definitions and efficient constructions. *J*ournal of Computer Security, 19(5):895–934, 2011. [cited at p. 17]

[Diffie&76] Diffie, W. and Hellman, M. New directions in cryptography. *I*nformation Theory, IEEE Transactions on, 22(6):644 – 654, nov 1976. [cited at p. 16, 19]

[Diffie&92] Diffie, Whitfield and Oorschot, Paul C. and Wiener, Michael J. Authentication and authenticated key exchanges. *D*esigns, Codes and Cryptography, 2:107–125, 1992. [cited at p. 14]

[Dong&08] Dong, Changyu and Russello, Giovanni and Dulay, Naranker. Shared and Searchable Encrypted Data for Untrusted Servers. In Atluri, Vijay, editor, *D*ata and Applications Security XXII, pages 127–143, Springer Berlin / Heidelberg, 2008. [cited at p. 27, 28, 30, 36, 37]

[Dong&11] Dong, Changyu and Russello, Giovanni and Dulay, Naranker. Shared and searchable encrypted data for untrusted servers. *J*. Comput. Secur., 19:367–397, August 2011. [cited at p. 16, 19, 28]

[Dropbox12] Dropbox. About Dropbox. `https://www.dropbox.com/about`, May 2012. Accessed May 14 2012. [cited at p. 4]

[Ferraiolo&01] Ferraiolo, David F. and Sandhu, Ravi and Gavrila, Serban and Kuhn, D. Richard and Chandramouli, Ramaswamy. Proposed NIST standard for role-based access control. *A*CM Trans. Inf. Syst. Secur., 4(3):224–274, August 2001. [cited at p. 40, 43, 57]

[FIPS-186-3] Locke, G. and Gallagher, P. FIPS PUB 186-3: Digital Signature Standard (DSS). *N*ational Institute of Standards and Technology, 2009. [cited at p. 80]

[FIPS-197] FIPS, N. FIPS PUB 197: Advanced encryption standard (AES). *N*ational Institute of Standards and Technology, 2001. [cited at p. 80]

[Franks&12] Franks Larry and Windows Azure Product Team. Data Storage Offerings on the Windows Azure Platform. `http://social.technet.microsoft.com/wiki/contents/articles/1674.data-storage-offerings-on-the-windows-azure-platform.aspx`, 2012. Accessed Apr 19 2012. [cited at p. 82]

[Fratto12] Fratto Mike. Should Amazon Define Cloud Standards? `http://www.networkcomputing.com/cloud-computing/232800457`, April 2012. Accessed May 8 2012. [cited at p. 2]

[Gerber&01] Gerber, M. and von Solms, R. and Overbeek, P. Formalizing information security requirements. *I*nformation Management & Computer Security, 9:32–37, 2001. [cited at p. 14]

[GoogleDrive12] Google. Google Drive Overview. `https://developers.google.com/drive/overview`, May 2012. Accessed May 14 2012. [cited at p. 4]

[Green&07] Green, Matthew and Ateniese, Giuseppe and Fu, Kevin and Hohenberger, Susan. The JHU-MIT Proxy Re-cryptography Library. `http://spar.isi.jhu.edu/~mgreen/prl/`, 2007. Accessed Apr 19 2012. [cited at p. 79, 121]

[Guth03] Guth, Susanne. A Sample DRM System. In Becker, Eberhard and Buhse, Willms and Gunnewig, Dirk and Rump, Niels, editors, *D*igital Rights Management, pages 150–161, Springer Berlin / Heidelberg, 2003. [cited at p. 62]

[Harn&92] Lein Harn and Hung-Yu Lin and Shoubao Yang. A software authentication system for information integrity. *C*omputers & Security, 11(8):747 – 752, 1992. [cited at p. 13]

[Horne&02] Horne, Bill and Matheson, Lesley and Sheehan, Casey and Tarjan, Robert. Dynamic Self-Checking Techniques for Improved Tamper Resistance. In Sander, Tomas, editor, *S*ecurity and Privacy in Digital Rights Management, pages 77–124, Springer Berlin / Heidelberg, 2002. [cited at p. 22, 65, 70]

[IDC09] IDC. New IDC IT Cloud Services Survey: Top Benefits and Challenges. `http://blogs.idc.com/ie/?p=730`, 2009. Accessed Apr 19 2012. [cited at p. 3]

[Ion&11] Ion, M. and Russello, G. and Crispo, B. Enforcing Multi-user Access Policies to Encrypted Cloud Databases. In *P*olicies for Distributed Systems and Networks (POLICY), 2011 IEEE International Symposium on, pages 175 –177, june 2011. [cited at p. 20]

[Ishai&06] Yuval Ishai and Eyal Kushilevitz and Rafail Ostrovsky and Amit Sahai. Cryptography from Anonymity. In *F*oundations of Computer Science, 2006. FOCS '06. 47th Annual IEEE Symposium on, pages 239 –248, oct. 2006. [cited at p. 93]

[Ivan&03] Anca Ivan and Yevgeniy Dodis. Proxy Cryptography Revisited. In *i*n Proceedings of the Network and Distributed System Security Symposium (NDSS), 2003. [cited at p. 19]

[Jalil&09] Jalil, Z. and Mirza, A.M. A Review of Digital Watermarking Techniques for Text Documents. In *I*nformation and Multimedia Technology, 2009. ICIMT'09. International Conference on, pages 230–234, IEEE, 2009.

[cited at p. 23]

[Jamkhedkar&09] Pramod A. Jamkhedkar and Gregory L. Heileman. Digital rights management architectures. *C*omputers &amp; Electrical Engineering, 35(2):376 – 394, 2009. Circuits and Systems for Real-Time Security and Copyright Protection of Multimedia. [cited at p. 22]

[Kamara&10] Kamara, Seny and Lauter, Kristin. Cryptographic cloud storage. *F*inancial Cryptography and Data Security, 136–149, 2010. [cited at p. 17, 26, 27, 28, 36, 37]

[Kamara&11] Kamara, S. and Papamanthou, C. and Roeder, T. *C*S2: A semantic cryptographic cloud storage system. Technical Report, Technical Report MSR-TR-2011-58, Microsoft Research, 2011. http://research. microsoft. com/apps/pubs, 2011. Accessed Apr 19 2012. [cited at p. 17, 27, 28]

[Katz&08] Katz, Jonathan and Sahai, Amit and Waters, Brent. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In Smart, Nigel, editor, *A*dvances in Cryptology - EUROCRYPT 2008, pages 146–162, Springer Berlin / Heidelberg, 2008. [cited at p. 17]

[Kern12] Kern Justin. The State of Cloud Standards. `http://www.information-management.com/news/cloud-computing-standards-security-SLA-OMG-DMTF-10021934-1.html`, February 2012. Accessed May 8 2012. [cited at p. 2]

[Kocher96] Kocher, Paul. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Koblitz, Neal, editor, *A*dvances in Cryptology — CRYPTO '96, pages 104–113, Springer Berlin / Heidelberg, 1996. [cited at p. 92]

[Leach&05] Leach, P. and Mealling, M. and Salz, R. RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace. `http://tools.ietf.org/html/rfc4122`, 2005. Accessed Apr 19 2012. [cited at p. 81]

[Liu&03] Liu, Qiong and Safavi-Naini, Reihaneh and Sheppard, Nicholas Paul. Digital rights management for content distribution. In *P*roceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21, pages 49–58, Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 2003. [cited at p. 63]

[Mather&09] Mather, T. and Kumaraswamy, S. and Latif, S. *C*loud security and privacy: an enterprise perspective on risks and compliance. O'Reilly Media, Inc., 2009. [cited at p. 1, 2]

[Michiels&05] Michiels, Sam and Verslype, Kristof and Joosen, Wouter and De Decker, Bart. Towards a software architecture for DRM. In Proceedings of the 5th ACM workshop on Digital rights management, pages 65–74, ACM, New York, NY, USA, 2005. [cited at p. 21, 22, 63, 67]

[Microsoft12] Microsoft. Microsoft .NET Framework. `http://www.microsoft.com/net`, May 2012. Accessed May 16 2012. [cited at p. 78]

[Mohagheghi&11] Mohagheghi, P. and Sændther, T. Software Engineering Challenges for Migration to the Service Cloud Paradigm: Ongoing Work in the REMICS Project. In Services (SERVICES), 2011 IEEE World Congress on, pages 507 –514, july 2011. [cited at p. 2, 3]

[NISTIR7657] Privilege Management Conference Collaboration Team. A Report on the Privilege (Access) Management Workshop. Technical Report IR 7657, National Institute of Standardards and Technology, March 2010. [cited at p. 19]

[Parkhill66] Parkhill, D.F. The challenge of the computer utility. The Challenge of the Computer Utility, Addison-Wesley Pub. Co., 1966. [cited at p. 1]

[Podilchuk&01] Podilchuk, C.I. and Delp, E.J. Digital watermarking: algorithms and applications. Signal Processing Magazine, IEEE, 18(4):33 –46, jul 2001. [cited at p. 23]

[Potoczny-Jones11] Potoczny-Jones, I. Cloud Security Risk Agreements for Small Businesses. 2011. [cited at p. 2]

[Priebe&06] Priebe, T. and Dobmeier, W. and Kamprath, N. Supporting attribute-based access control with ontologies. In Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on, page 8 pp., april 2006. [cited at p. 21]

[Richter12] Richter Jeffrey. Data Storage Offerings in Windows Azure. `https://www.windowsazure.com/en-us/develop/net/fundamentals/cloud-storage/`, 2012. Accessed Apr 19 2012. [cited at p. 82]

[Rivest92] Rivest, R. RFC 1321: The MD5 Message-Digest Algorithm. `http://tools.ietf.org/html/rfc1321`, 1992. Accessed Apr 19 2012. [cited at p. 81]

[Ruj&11] Ruj, S. and Nayak, A. and Stojmenovic, I. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, title=DACC: Distributed Access Control in Clouds, pages 91 –98, nov. 2011. [cited at p. 42, 43, 57]

[Saltzer&75] Saltzer, J.H. and Schroeder, M.D. The protection of information in computer systems. Proceedings of the IEEE, 63(9):1278 – 1308, sept. 1975.

[cited at p. 19, 20]

[Sandhu&96] Sandhu, R.S. and Coyne, E.J. and Feinstein, H.L. and Youman, C.E. Role-based access control models. *C*omputer, 29(2):38 –47, feb 1996. [cited at p. 19]

[Sankar&11] Veerubhotla, R.S. and Saxena, A. A DRM framework towards preventing digital piracy. In *I*nformation Assurance and Security (IAS), 2011 7th International Conference on, pages 1–6, dec. 2011. [cited at p. 65, 67, 70]

[Schaffer09] Schaffer, H.E. X as a Service, Cloud Computing, and the Need for Good Judgment. *I*T Professional, 11(5):4–5, sept.-oct. 2009. [cited at p. 2]

[Shamir83] Shamir, Adi. On the generation of cryptographically strong pseudorandom sequences. *A*CM Trans. Comput. Syst., 1(1):38–44, February 1983. [cited at p. 16]

[Singh&08] Singh, Aameek and Liu, Ling. Sharoes: A Data Sharing Platform for Outsourced Enterprise Storage Environments. In *P*roceedings of the 2008 IEEE 24th International Conference on Data Engineering, pages 993–1002, IEEE Computer Society, Washington, DC, USA, 2008. [cited at p. 20]

[SkyDrive12] Microsoft. SkyDrive Overview. `http://windows.microsoft.com/en-GB/skydrive/any-file-anywhere`, May 2012. Accessed May 14 2012. [cited at p. 4]

[Song&00] Song, Dawn Xiaodong and Wagner, David and Perrig, Adrian. Practical Techniques for Searches on Encrypted Data. In *P*roceedings of the 2000 IEEE Symposium on Security and Privacy, pages 44–55, IEEE Computer Society, Washington, DC, USA, 2000. [cited at p. 16, 17, 18]

[Steinebach&07] Steinebach, M. and Hauer, E. and Wolf, P. Efficient Watermarking Strategies. In *A*utomated Production of Cross Media Content for Multi-Channel Distribution, 2007. AXMEDIS '07. Third International Conference on, pages 65 –71, nov. 2007. [cited at p. 23]

[Storer&09] Storer, Mark W. and Greenan, Kevin M. and Miller, Ethan L. and Voruganti, Kaladhar. POTSHARDS - a secure, recoverable, long-term archival storage system. *T*rans. Storage, 5:5:1–5:35, June 2009. [cited at p. 15, 16]

[Vaquero&08] Vaquero, L.M. and Rodero-Merino, L. and Caceres, J. and Lindner, M. A break in the clouds: towards a cloud definition. *A*CM SIGCOMM Computer Communication Review, 39(1):50–55, 2008. [cited at p. 2]

[Wang&04a] Wang, Lingyu and Wijesekera, Duminda and Jajodia, Sushil. A logic-based framework for attribute based access control. In *P*roceedings of the 2004 ACM workshop on Formal methods in security engineering, pages 45–55, ACM, New York, NY, USA, 2004. [cited at p. 21]

[Wolf&07] Wolf, P. and Steinebach, M. and Diener, K. Complementing DRM with digital watermarking: mark, search, retrieve. *O*nline Information Review, 31(1):10–21, 2007. [cited at p. 23, 66]

[Wu&10] Wu, J. and Ping, L. and Ge, X. and Wang, Y. and Fu, J. Cloud Storage as the Infrastructure of Cloud Computing. In *I*ntelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on, pages 380–383, IEEE, 2010. [cited at p. 1, 2]

[Yang&11] Yanjiang Yang and Youcheng Zhang. A Generic Scheme for Secure Data Sharing in Cloud. In *P*arallel Processing Workshops (ICPPW), 2011 40th International Conference on, pages 145–153, sept. 2011. [cited at p. 20]

[Yu&10] Yu, S. and Wang, C. and Ren, K. and Lou, W. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *I*NFOCOM, 2010 Proceedings IEEE, pages 1–9, Ieee, 2010. [cited at p. 20]

[Zeng&10] ZENG, W., PARKIN, S.E., VAN MOORSEL, A. Digital Rights Management. 2010. [cited at p. 64, 71]

[Zhang&10] Zhang, Qi and Cheng, Lu and Boutaba, Raouf. Cloud computing: state-of-the-art and research challenges. *J*ournal of Internet Services and Applications, 1:7–18, 2010. [cited at p. 1, 2]

[Zhao&11] Zhao, Fangming and Nishide, Takashi and Sakurai, Kouichi. Realizing Fine-Grained and Flexible Access Control to Outsourced Data with Attribute-Based Cryptosystems. In Bao, Feng and Weng, Jian, editors, *I*nformation Security Practice and Experience, pages 83–97, Springer Berlin / Heidelberg, 2011. [cited at p. 20]

[Zhou&10a] Minqi Zhou and Rong Zhang and Wei Xie and Weining Qian and Aoying Zhou. Security and Privacy in Cloud Computing: A Survey. In *S*emantics Knowledge and Grid (SKG), 2010 Sixth International Conference on, pages 105–112, nov. 2010. [cited at p. 3]

Appendices

Appendix A

Terminology

***ABAC*:** Attribute-Based Access Control. A control model based on attributes that are associated either with the resource requested or the entity which is requesting access.

***ABE*:** Attribute-Based Encryption. Fine-grained access control model based on public key encryption associated with an access control policy.

***ACL*:** Access Control Lists. Access control model where authorisation is based on mappings between entities requesting access to a resource and the actions the entities are allowed to perform on a particular resource.

***AES*:** Advanced Encryption Standard. NIST standardised symmetric encryption scheme used globally. It is classified for top secret by the National Security Agency of the United States.

***API*:** Application Programming Interface. Specification of a set of software interfaces and classes which enables software components to communicate.

***ASE*:** Asymmetric Searchable Encryption. Asymmetric encryption scheme which gives confidentiality while allowing searching the ciphertext for keywords.

***CIA*:** Confidentiality, Integrity and Availability. Abbreviation covering all three terms.

***CP-ABE*:** Ciphertext-Policy Attribute-Based Encryption. Special kind of ABE where a user's private key is associated with attributes and an access structure over attributes is associated with data.

***CSP*:** Cloud Service Provider. Refers to any company or organisation from which you can buy cloud services. Examples of actual cloud service providers are Amazon, Goggle and Microsoft.

***Data Entity*:** Single unit of data used in S1 where it is essentially a wrapper of a file adding various meta data.

***DO*:** Data Owner. Central entity in S1 which is responsible for initialising the system as well as granting and revoking access for users.

***DRM*:** Digital Rights Management. System specialised in managing data access control. Is often used to protect audio and video.

***DSA*:** Digital Signature Algorithm. NIST standardised digital signature scheme.

***ECC*:** Error-Correcting Code. Any algorithm which allows transmitting data over unreliable or noisy communication channels while preserving integrity. This is usually achieved by adding redundant information and restoring the original message on the receiving side.

***ESE*:** Efficient asymmetric Searchable Encryption. ASE schemes which support more efficient searching than normal ASE schemes do.

***IaaS*:** Infrastructure as a Service. Cloud service layer at the bottom of the service layer stack. Consists of pools of storage and computing resources partitioned using virtualisation.

***KP-ABE*:** Key-Policy Attributed-Based Encryption. Special kind of ABE where data is associated with attributes and a user's private key is associated with an access structure over attributes.

***MD5*:** Message Digest 5. Widely used cryptographic hash scheme producing a 128bit hash value. Designed in 1991 and is no longer considered suitable for security purposes as it is not collision resistant.

***MR+MW*:** Multiple Readers+Multiple Writers. Term used to describe a data sharing solution supporting both multiple readers and multiple writers at the same time.

***MR+SW*:** Multiple Readers+Single Writer. Term used to describe a data sharing solution supporting multiple readers at the same time, but not only a single writer.

***MSSE*:** Multi-user Searchable Symmetric Encryption. Searchable encryption scheme based on SSE, but combined with broadcast encryption to obtain support for MR+SW.

***MWA*:** Microsoft Windows Azure. Commercial cloud platform provided by Microsoft.

***NIST*:** National Institute of Standards and Technology. United states federal technology agency working with the industry to develop and apply technology, measurements, and standards.

***PaaS*:** Platform as a Service. Cloud service layer building on the IaaS layer. It provides operating systems and application frameworks.

***PBAC*:** Policy-Based Access Control. Model that standardises of the ABAC model in an enterprise environment which may require compliance with legislation.

***PRE*:** Proxy Re-Encryption. Encryption scheme which allows a message from being encrypted with one key pair to being encrypted with another key pair.

***PIR*:** Personal Information Retrieval. A technique allowing a user to retrieve an item from a server without revealing which item is being retrieved.

***PRL*:** The JHU-MIT Proxy Re-cryptography Library. PRE prototype implementation created by researchers at Johns Hopkins University and Massachusetts Institute of Technology.

***RBAC*:** Role-Based Access Control. Access control model that allows grouping of different entities into categories based on similar access needs to fulfil particular roles.

***REL*:** Rights Expression Language. Machine-processable language used to express rights in a DRM system.

***SaaS*:** Software as a Service. Cloud service layer at the top of the cloud service stack. Consisting of cloud applications with individual purposes built to leverage the automatic scaling of the cloud.

***SSE*:** Symmetric Searchable Encryption. Searchable encryption based on symmetric encryption while allowing searching of the ciphertext for keywords.

***SR+MW*:** Single Reader+Multiple Writers. Term used to describe a data sharing solution supporting multiple writers at the same time, but only a single reader.

***SR+SW*:** Single Reader+Single Writer. Term used to describe a data sharing solution supporting only a single reader and a single writer.

***UUID*:** Universally Unique IDentifiers. UUID is an identifier standard used in software. The size of a UUID is 128bit and the standard defines how it should be generated to allow generation of unique identifiers without central coordination.

***WCF*:** Windows Communication Foundation. API part of the Microsoft .NET framework 3.0 for building distributed and service-oriented systems.

***XaaS*:** Anything as a Service. Term used to describe how virtually anything in some situations are considered services which may be offered by a cloud.

***XACML*:** eXtensible Access Control Markup Language. Standard which defines a declarative access control policy language implemented in XML and a processing model describing how to evaluate authorisation requests according to the rules defined in policies.

***XML*:** eXtensible Markup Language. Markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

Appendix B

Proof of concept prototype details

B.1. Introduction

This appendix constitutes a brief technical overview of the proof of concept prototypes created to validate the conceptual designs of *Secure Dynamic Cloud-based Data-sharing (SDCD)* and *Cloud-based Hierarchical Access Control (CHAC)*. The purpose of this appendix is to present the interested reader with extra information regarding the overall design of the SDCD and CHAC prototypes.

B.2. Secure Dynamic Cloud-based Data-sharing prototype

In this section a brief technical overview of the prototype created to validate SDCD is presented. An overview of the prototype is shown in Figure B.1. The figure shows the Data Owner (DO) and two users. They use an administration client and a user client to access data stored in a cloud. The cloud exposes a Proxy Re-Encryption (PRE) service and a storage service. The PRE service handles generation of key pairs, encryption, re-encryption, and decryption. The storage service is responsible for storing data entities and ensuring their availability.
The SDCD prototype has been developed using the Microsoft .NET framework and has been written in C#.
The content and relations of entities used in the prototype is presented in Section B.2.1 after which the operations of the prototype is described in Section B.2.2. Finally, persistence is described in Section B.2.3 and the implementation effort, put into constructing the SDCD prototype, is summarised in Section B.2.4.

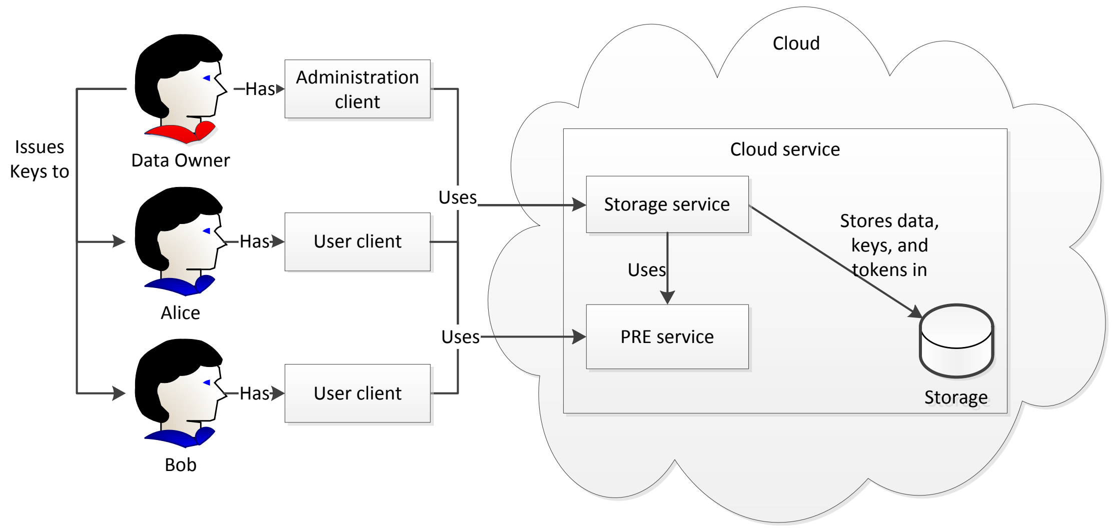


Figure B.1: Overview of the SDCD proof of concept prototype.

B.2.1 Structure

The SDCD prototype has a small model for describing a data entity and its related entities. Each entity is implemented by a C# class. A user is not implemented as a class. Users exist only as a particular instance of the `System.Guid` structure of the Microsoft .NET framework which represent their identifier.

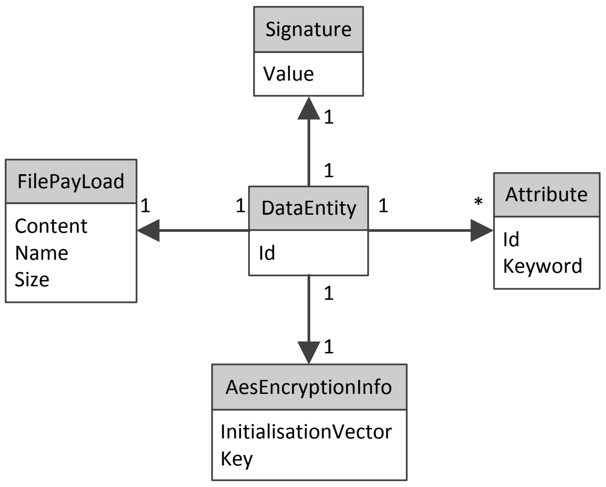


Figure B.2: Entity Relationship diagram of a data entity.

B.2.2 Operations

SDCD defines a number of operations, but not all of them are implemented as part of the SDCD prototype. This section describes the implemented operations. Sequence diagrams have been created for some of them.

System initialisation: Figure B.3 shows an overview of the system initialisation. The DO initialises the system by using the administration client. His identifier is generated from his name and sent to the cloud which stores the identifier for future authentication of the DO. Finally the DO's master key pair is generated.

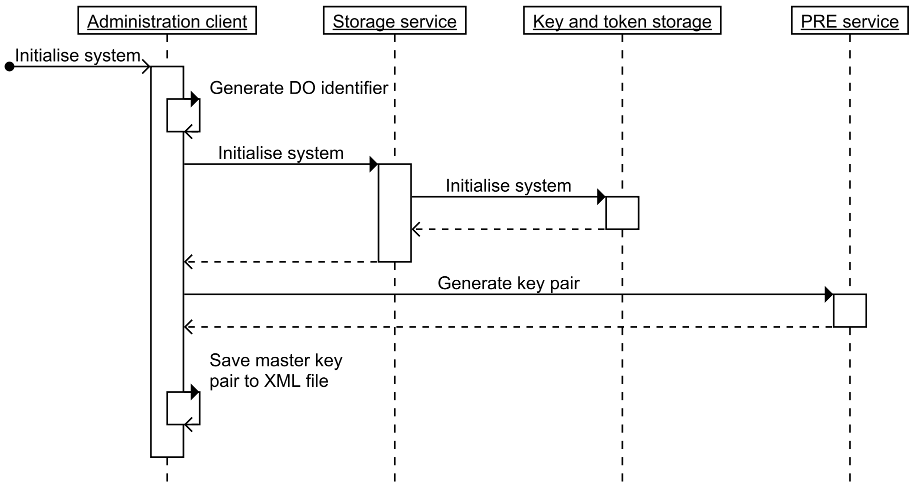
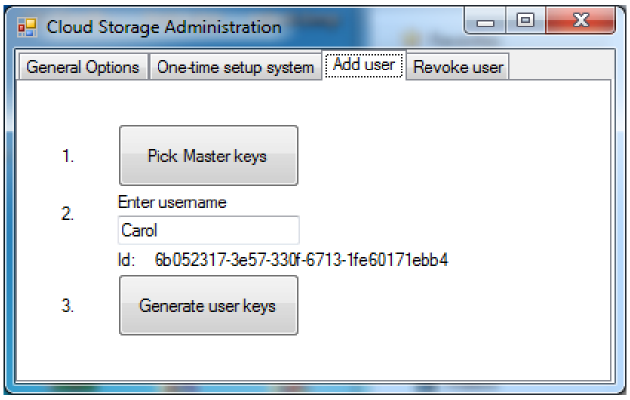


Figure B.3: Dynamics of how the system is initialised in the prototype.

User addition: The DO adds a new user by selecting the master key pair and generates an identifier for the user based on the user's name. Then a signing key pair, a PRE key pair, and a PRE token is generated for the user. The user's identifier and PRE token key pairs are then sent to the cloud and given to the user as defined by the conceptual design of SDCD. In this process, the cloud authenticates the DO and adds the user. Figure B.4 shows a screenshot from the administration client of the SDCD prototype where a user is being added.



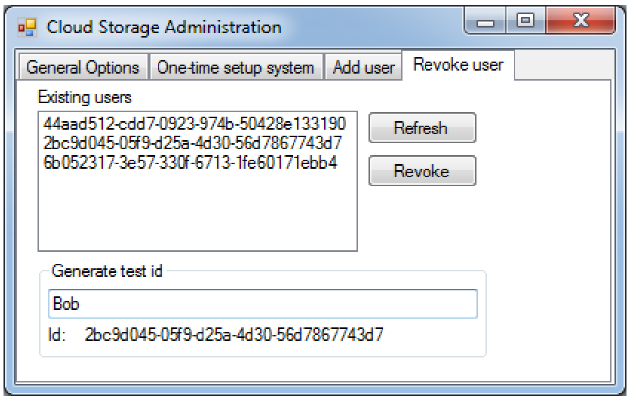
Enter username

Carol

Id: 6b052317-3e57-330f-6713-1fe60171ebb4

Figure B.4: Screenshot of user addition in the administration client of the SDCD prototype.

User revocation: The DO revokes a user by using the administration client to obtain a list of identifiers for all current users. The DO correlates the received identifiers to actual users himself as the SDCD prototype does not implement this step. He then revokes a specific user by using the user's identifier. When doing this, the cloud authenticates the DO and removes the PRE token of the user. The public part of the user's signing key pair is retained. Figure B.5 shows a screenshot from the administration client of the SDCD prototype where a user is being revoked.



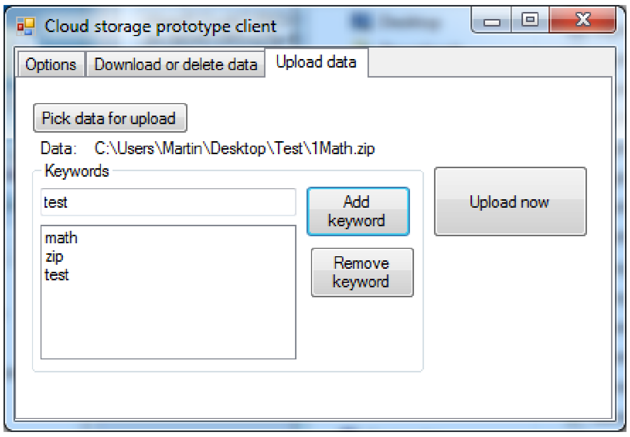
44aad512-cdd7-0923-974b-50428e133190
2bc9d045-05f9-d25a-4d30-56d7867743d7
6b052317-3e57-330f-6713-1fe60171ebb4

Bob

Id: 2bc9d045-05f9-d25a-4d30-56d7867743d7

Figure B.5: Screenshot of user revocation in the administration client of the SDCD prototype.

Data entity creation: To create a data entity, a user selects his keys and then the file he wishes to upload to the cloud. One or more keywords are then associated with the file by the user. A random AES key is generated and the file contents, file name, and keywords are encrypted using symmetric encryption after which the keywords are hashed. The AES key is then encrypted using the master public key and an identifier is generated for the data entity. Finally the data entity is signed using the user's signing key pair and transmitted to the cloud. The cloud authenticates the user and verifies the integrity of the data entity. The data entity is then stored in the cloud. Figure B.6 shows a screenshot of the user client of the SDCD prototype where a data entity is being created by selecting and uploading a file.



Data: C:\Users\Martin\Desktop\Test\1Math.zip

test

math
zip
test

Figure B.6: Screenshot of data entity creation in the user client of the SDCD prototype.

Data entity modification: This is not implemented in the prototype. Data entities are immutable due to this. Once uploaded to the cloud storage data entities cannot be changed.

Data entity deletion: To delete a data entity, it must first be retrieved by searching which is described below. Once data entities are found, the user requests the cloud to delete a data entity based on its identifier.

Data entity searching: A user is able to search for data entities through the user client by selecting his keys and entering a search keyword. A search token is created from the keyword and sent to the cloud. The cloud identifies data entities by comparing the search token to identifiers of their associated attributes. The cloud then re-encrypts the AES keys of the data entities and returns the data entities. The payloads of the data entities are omitted to preserve bandwidth. The search operation is shown in Figure B.7.

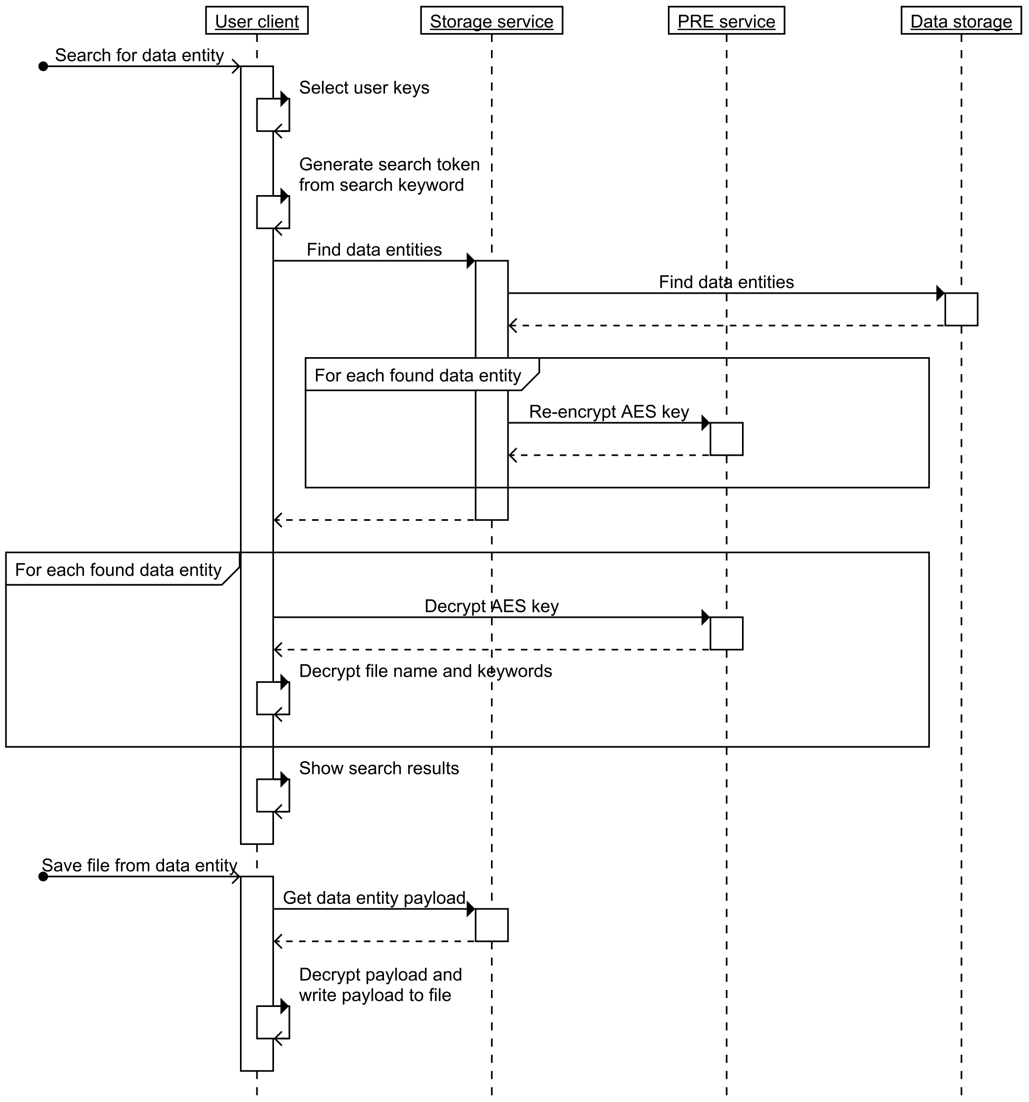
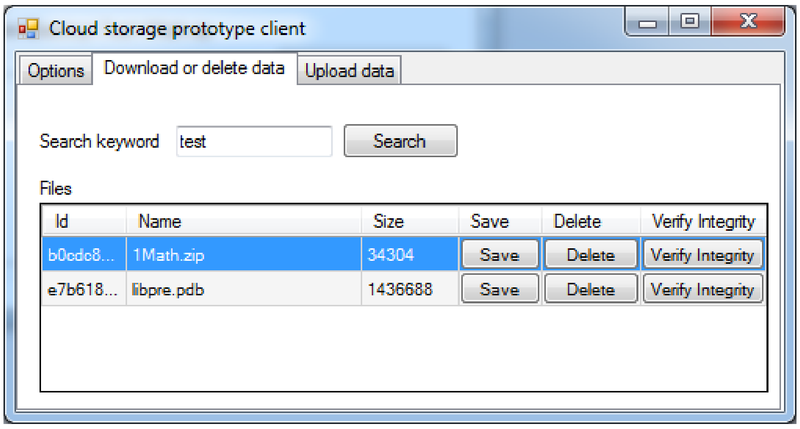


Figure B.7: Dynamics of how a data entity is searched for and how its payload is saved.

At the user client the AES keys are decrypted and then used to decrypt the file name for each data entity, which is then presented to the user. To save a file encapsulated in a data entity the user selects the corresponding file name. This results in the payload of the chosen data entity being downloaded, decrypted using the corresponding AES key, and finally written to a file in the user's environment. Figure B.8 shows a screenshot of a search operation performed using the user client.



Id	Name	Size	Save	Delete	Verify Integrity
b0cdc8...	1Math.zip	34304	Save	Delete	Verify Integrity
e7b618...	libpre.pdb	1436688	Save	Delete	Verify Integrity

Figure B.8: Screenshot showing a search result for the keyword "test" in the user client.

Data entity integrity verification: The process of data entity integrity verification is shown in Figure B.9. To verify the integrity of a data entity, data entities must be searched for according to the operation described above. Once data entities are found, the user requests the cloud to verify integrity based on the identifier of a data entity. The cloud authenticates the user and looks up the public sign key of the author of the data entity. The data entity is then found and its integrity is verified. Finally, the result of the verification is sent to the user.

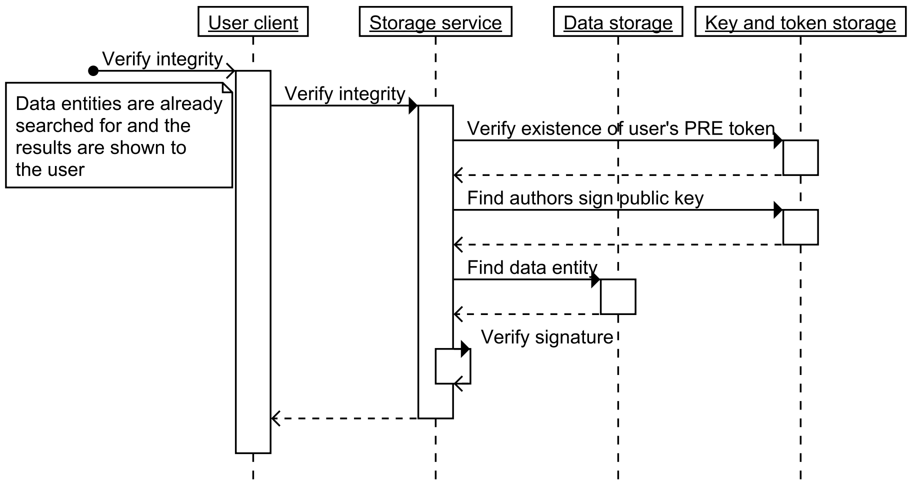

User client
Storage service
Data storage
Key and token storage
Verify integrity
Data entities are already searched for and the results are shown to the user
Verify integrity
Verify existence of user's PRE token
Find authors sign public key
Find data entity
Verify signature

Figure B.9: Dynamics of how the integrity of a data entity is verified in the prototype.

Data entity author authenticity verification: This is not implemented in the prototype. Therefore the DO is unable to verify the identities of data entity authors.

B.2.3 Persistence

The SDCD prototype stores data entities and their related information in the cloud. The data entity payloads are stored in separate files and the remaining meta-data regarding data entities are stored using eXtended Markup Language (XML) files. Several fields of the information associated with a data entity is binary. In the XML files this information is encoded. Examples of the XML files stored in the cloud by the SDCD prototype can be seen below.

Some of the binary values associated with a data entity are large and in the examples below, they have been truncated to fit the page. Code Listing B.1 is an example of the information stored about users and code Listing B.2 is an example of the information stored about a data entities. From the examples it can be seen that while the XML files reveals structural information, they do not reveal information such as the names, actual keywords, or payloads of the stored data entities.

```xml
<?xml version="1.0"?>
<UsersMetadata xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DataOwnerId>44aad512-cdd7-0923-974b-50428e133190</DataOwnerId>
  <UserMetadata>
    <UserMetadata UserId="44aad512-cdd7-0923-974b-50428e133190">
      <DelegationToken>QAAAAD43x+ ... CED26lQ==</DelegationToken>
      <SignPublicKey>BgIAAAAiAABE ... uyKN2EodqB8</SignPublicKey>
    </UserMetadata>
    <UserMetadata UserId="2bc9d045-05f9-d25a-4d30-56d7867743d7">
      <DelegationToken>QAAAAEyVNT ... 6HKt/Ww==</DelegationToken>
      <SignPublicKey>BgIAAAAiAABE ... 47N13RnuQxe</SignPublicKey>
    </UserMetadata>
    <UserMetadata UserId="6b052317-3e57-330f-6713-1fe60171ebb4">
      <DelegationToken>QAAAACxgVY ... tvvOG5g==</DelegationToken>
      <SignPublicKey>BgIAAAAiAABE ... NmZBan4eFVj</SignPublicKey>
    </UserMetadata>
  </UserMetadata>
</UsersMetadata>
```

Listing B.1: Example of an XML file storing information about users in the SDCD prototype.

```xml
<?xml version="1.0"?>
<StorageMetadata xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Entities>
    <EntityMetadata>
      <Id>b0cdc830-66a5-44f1-b645-a1a9644be9fc</Id>
      <Name>7WMWFm13N3YGdhzrI1cbAQ==</Name>
      <Size>34304</Size>
      <Attributes>
        <EntityAttribute Id="a383b420-90b4 ... ae02db930429">
          <Keyword>5/P996umfoCYL3JkpCsHWg==</Keyword>
        </EntityAttribute>
        <EntityAttribute Id="f87be32b-a2c1 ... c4b8700c145d">
          <Keyword>6RQAGdSfXxtkhJvUQnSirw==</Keyword>
        </EntityAttribute>
        <EntityAttribute Id="00bc84ba-ea42 ... 31992a8d66d5">
          <Keyword>iEQROL6cUfEeEJNNpol3jw==</Keyword>
        </EntityAttribute>
      </Attributes>
      <AesKey>AUAAAAAZRHtd ... d8Z8kVU8S8gfGKTAmbSGBTk=</AesKey>
      <AesIV>AUAAAACGEcK5p ... IBYHF6zE1iKspvW/KCwsI0MI=</AesIV>
      <Signature>acC4Fb08O ... g4I3CqQZUYvt4ce4gnQ==</Signature>
    </EntityMetadata>
    <EntityMetadata>
      <Id>e7b618ed-bd3b-4a50-b28f-96f9d2f434a0</Id>
      <Name>os/nPfpZkD0GPQ8pNPyydQ==</Name>
      <Size>1436688</Size>
      <Attributes>
        <EntityAttribute Id="a383b420-90b4 ... ae02db930429">
          <Keyword>qFD9EQHPEi48Ie5/vIP8CQ==</Keyword>
        </EntityAttribute>
        <EntityAttribute Id="c1d4b5cd-2c7d ... 48a39309ca14">
          <Keyword>iqxr4BeeMJSiB5NKuAcz8Q==</Keyword>
        </EntityAttribute>
        <EntityAttribute Id="88cc54db-2a7e ... 631e4d158e71">
          <Keyword>Nf/xFcmoePet9aVvTu7QRQ==</Keyword>
        </EntityAttribute>
      </Attributes>
      <AesKey>AUAAAAAYbMME ... nRdofsjttMtopRagMWjXnXk=</AesKey>
      <AesIV>AUAAAAAtoZRUw ... wfcqZlJoPWUJxJEyxVWt/fy0=</AesIV>
      <Signature>UHA+UsH47 ... m2UkfIIhLuA5+Ve4Z4g==</Signature>
    </EntityMetadata>
  </Entities>
  <Attributes>
    <AttributeMetadata>
      <AttributeId>a383b420-90b4 ... ae02db930429</AttributeId>
      <EntityIds>
        <EntityId Id="b0cdc830-66a5-44f1-b645-a1a9644be9fc" />
        <EntityId Id="e7b618ed-bd3b-4a50-b28f-96f9d2f434a0" />
      </EntityIds>
    </AttributeMetadata>
    <AttributeMetadata>
      <AttributeId>f87be32b-a2c1 ... c4b8700c145d</AttributeId>
      <EntityIds>
        <EntityId Id="b0cdc830-66a5-44f1-b645-a1a9644be9fc" />
      </EntityIds>
    </AttributeMetadata>
    <AttributeMetadata>
      <AttributeId>00bc84ba-ea42 ... 31992a8d66d5</AttributeId>
      <EntityIds>
        <EntityId Id="b0cdc830-66a5-44f1-b645-a1a9644be9fc" />
      </EntityIds>
    </AttributeMetadata>
    <AttributeMetadata>
      <AttributeId>c1d4b5cd-2c7d ... 48a39309ca14</AttributeId>
      <EntityIds>
        <EntityId Id="e7b618ed-bd3b-4a50-b28f-96f9d2f434a0" />
      </EntityIds>
    </AttributeMetadata>
    <AttributeMetadata>
      <AttributeId>88cc54db-2a7e ... 631e4d158e71</AttributeId>
      <EntityIds>
        <EntityId Id="e7b618ed-bd3b-4a50-b28f-96f9d2f434a0" />
      </EntityIds>
    </AttributeMetadata>
  </Attributes>
  <Authors>
    <AuthorMetadata>
      <AuthorId>2bc9d045-05f9-d25a-4d30-56d7867743d7</AuthorId>
      <EntityIds>
        <EntityId Id="b0cdc830-66a5-44f1-b645-a1a9644be9fc" />
        <EntityId Id="e7b618ed-bd3b-4a50-b28f-96f9d2f434a0" />
      </EntityIds>
    </AuthorMetadata>
  </Authors>
</StorageMetadata>
```

Listing B.2: Example of an XML file storing information about data entities in the SDCD prototype.

B.2.4 Implementation effort

The SDCD prototype successfully validates the conceptual design of SDCD. This is done by using different technologies, frameworks, and platforms and by writing and compiling many lines of code. The prototype uses a PRE library implementing a PRE scheme described by Ateniese et al. [Ateniese&06]. The implementation of the PRE scheme is done by Ateniese et al. and is available under the name: "The JHU-MIT Proxy Re-cryptography Library" (PRL) [Green&07]. This library in turn uses the "MIRACL library" which is a commercial third-party library [CertiVox&12]. The MIRACL library was compiled from its source code which has not been modified for the use in the SDCD prototype. The PRL library, as it was obtained from its authors, contained compilation errors which we fixed, however, most of the source code for the library has not been modified. Finally several lines of code has been written specifically for the SDCD prototype. Except for code which is part of the Microsoft .NET framework, Table B.1 presents an overview of the amount of code used to construct the SDCD prototype. The table does not show the code generated by tools used to design the user interface of the administration and user client. This would add approximately 1100 additional lines of code for the SDCD prototype. In total, the effort for developing the SDCD prototype was approximately two person-weeks.

Table B.1: Approximate number of lines of code in the SDCD prototype.

Area of prototype	Language	Lines of code	Description
MIRACL library	C	≈ 32000	Commercial third-party library used by the PRL library.
PRL library	C/C++	≈ 3800	Library implementing the PRE scheme used by the SDCD prototype.
The code we produced	C#	≈ 4250	Code written specifically for the SDCD prototype.
Total	Mixed	≈ 40050	Total number of lines of code compiled for constructing the SDCD prototype.

B.3. Cloud-based Hierarchical Access Control prototype

The CHAC prototype builds upon the SDCD prototype and extends its existing functionality to support fine-grained access control and access control delegation. An overview of the CHAC prototype can be seen in Figure B.10. In this figure, the data owner and two users use the administration client and user client to interact with the cloud. Like in the SDCD prototype, the cloud has two services and stores data entities, but in the CHAC prototype the cloud also stores information regarding roles.

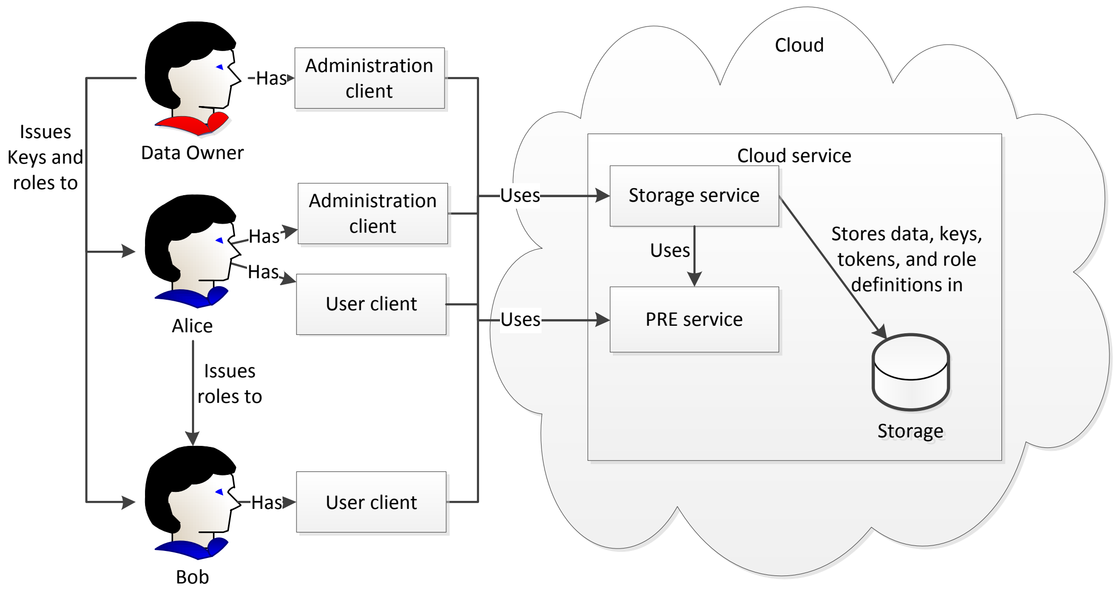


Figure B.10: Overview of the CHAC proof of concept prototype.

B.3.1 Structure

The CHAC prototype uses the same definition of a data entity as the SDCD prototype. Figure B.11 shows the part of the prototype's model which is related to a data entity. The CHAC prototype extends the SDCD prototype by defining roles which govern access to data entities. The CHAC prototype also has a user entity. Figure B.12 shows the relationships in the CHAC prototype. It shows a many-to-many relationship between roles and users and a one-to-many relationship between roles. Thereby, the concept of sub-roles is modeled.

B.3.2 Operations

CHAC defines an additional number of operations extending SDCD. Not all of the operation are implemented as part of the CHAC prototype. This section describes the operations that have been implemented. Sequence diagrams and screenshots have been created for some of these operations. Through the administration client a number of operations are available. The client is shown in Figure B.13.

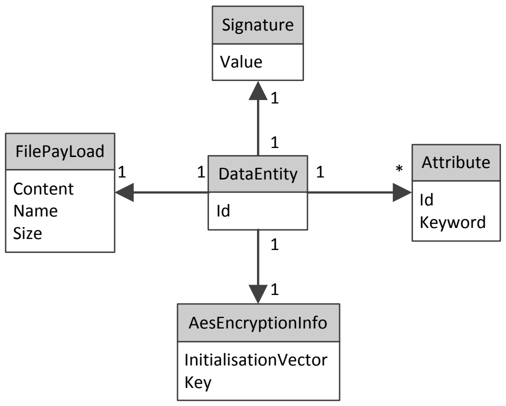


Figure B.11: Entity Relationship diagram of a data entity.

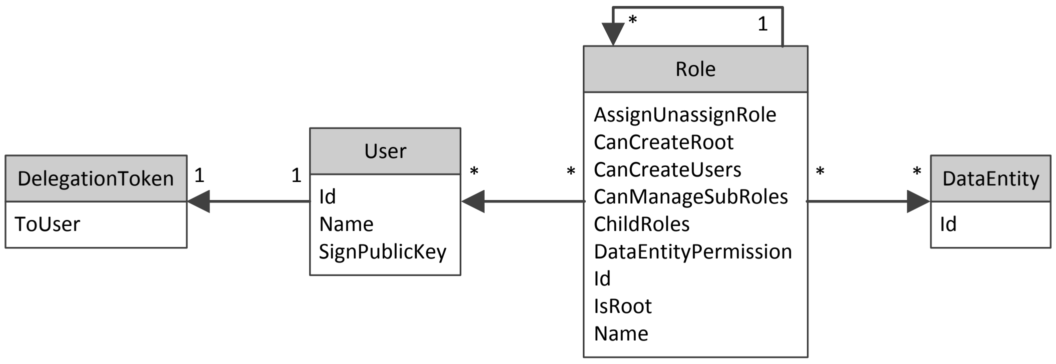
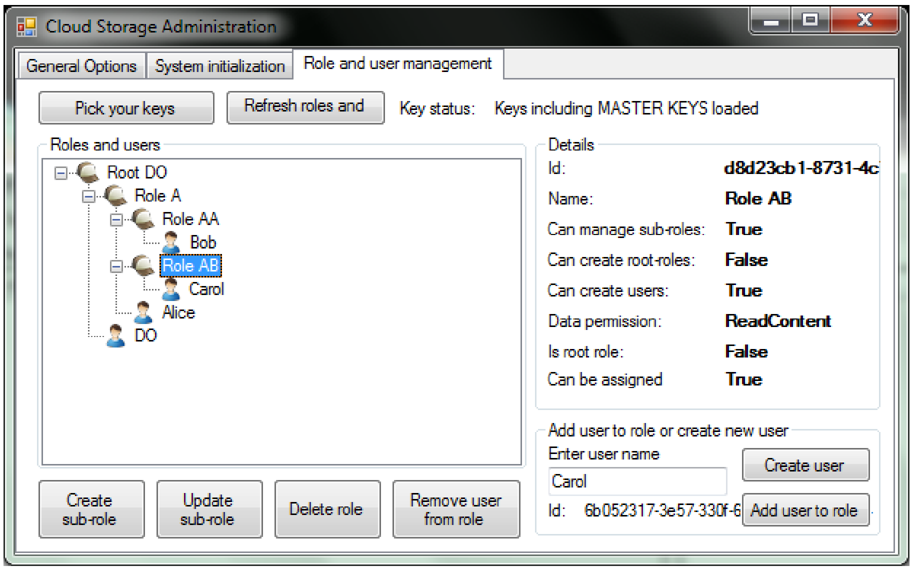


Figure B.12: Entity Relationship diagram of a data entity.

Create sub-role: Figure B.14 shows how a sub-role is created. This operation allows a user, with the appropriate permission, to create additional sub-roles by selecting the role to base the new sub-role on. Additionally, the user must select a subset of permissions which the sub-role is granted. The information is then sent to the cloud. When the sub-role is created it is shown in the role hierarchy. The permission details of a role can be see in Figure B.13 where the current permissions of the selected Role AB is shown.

Update sub-role: Existing roles can be updated by either adding or removing some of its permissions. Adding permissions is still limited to a subset of the permissions of the parent role of the role being edited. If permissions are revoked this change is cascaded down to sub-roles of the role being edited. This is done through use of the administration client.

Delete role: Deletion of roles is also possible through the administration client. A role is selected and the “Delete role” option is used. Sub-roles of a role being deleted are attached to the parent of the role being deleted. Any users assigned to the role being deleted are removed from the role. If a user is not assigned to any other roles this user is removed from the hierarchy.



Key status: Keys including MASTER KEYS loaded

Roles and users

- Root DO
  - Role A
    - Role AA
      - Bob
    - Role AB
      - Carol
    - Alice
  - DO

Details

Id:	d8d23cb1-8731-4c
Name:	Role AB
Can manage sub-roles:	True
Can create root-roles:	False
Can create users:	True
Data permission:	ReadContent
Is root role:	False
Can be assigned	True

Add user to role or create new user

Enter user name: Carol

Id: 6b052317-3e57-330f-6

Figure B.13: Screenshot showing the administration client used for role and user management.

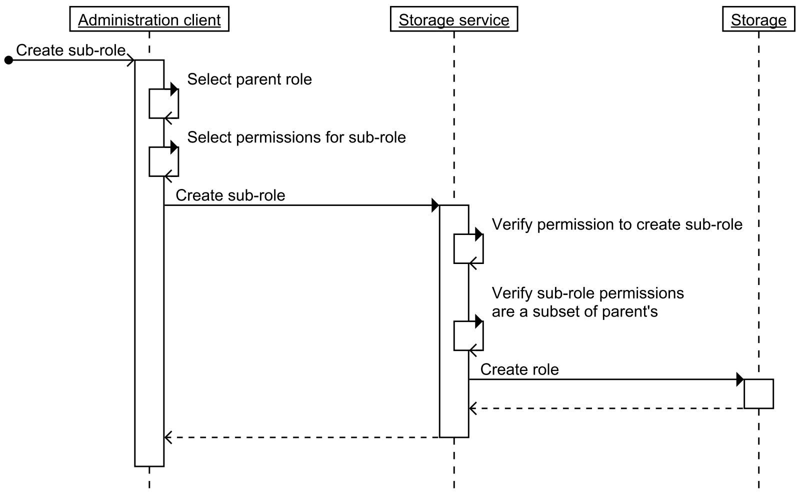

Administration client
Storage service
Storage
Create sub-role
Select parent role
Select permissions for sub-role
Create sub-role
Verify permission to create sub-role
Verify sub-role permissions are a subset of parent's
Create role

Figure B.14: Dynamics of how the roles are created in the prototype.

Add user to role: Figure B.15 shows a sequence diagram where the process of adding a user to a role is shown. After selecting a role in the hierarchy a user can be assigned to this role, provided the user trying to add him has the permission to perform this action.

Remove user from role: Users can also be removed from roles. To do so, a user must be selected in the role hierarchy and then the “Remove user from role” option must be selected, see Figure B.13. This removes the user from the role. If the user is not assigned to any other roles he is removed from the hierarchy. Again, only users with permission to perform this action can remove other users.

Create private hierarchy: The prototype also allows creation of private role hierarchies. This is done by creating a new role. If the user creating the new role is allowed to create private hierarchies, he can mark the new role as “root role” and through this root role the new private hierarchy is created. The user is then automatically assigned this root role.

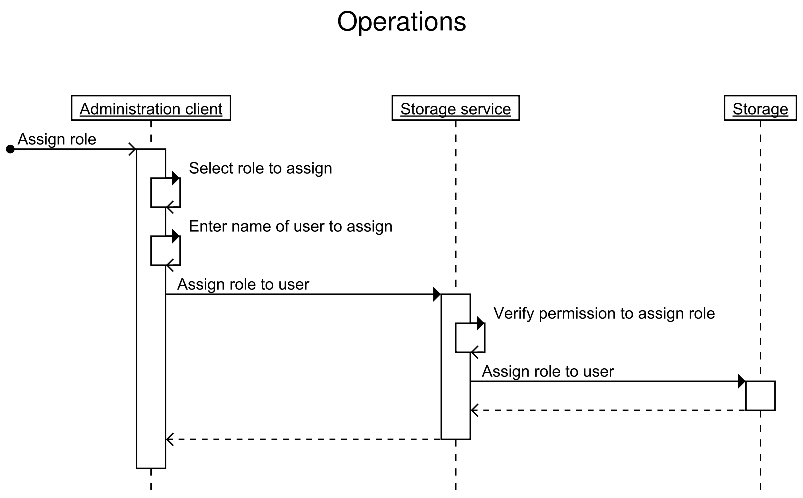
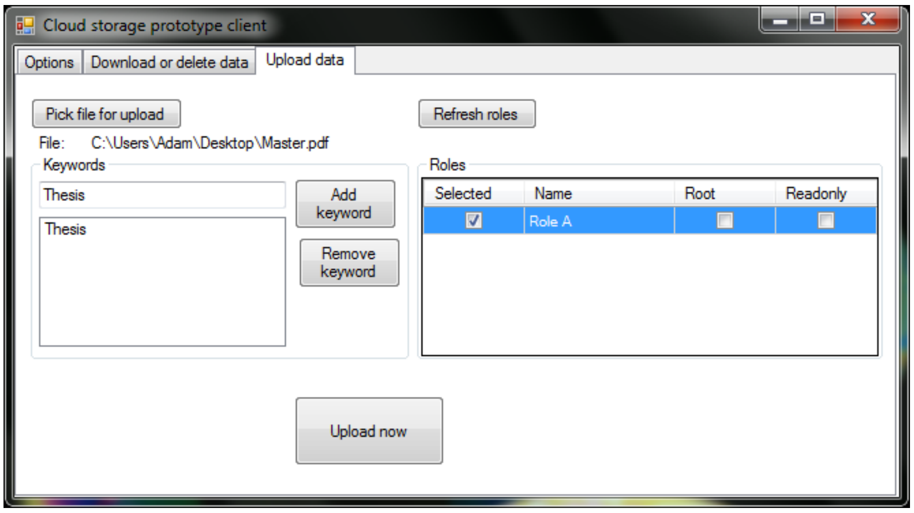


Figure B.15: Dynamics of how the roles are assigned to users in the prototype.

Auditability: This is not implemented in this prototype. Therefore, auditing of entities in the collaboration environment is not supported.

The user client created as a part of the SDCD prototype has been extended in the CHAC prototype. The following operations have had their features extended:

Create data entity: Fine-grained data access is now implemented as a part of the user client. Figure B.16 shows options made available. When a user is uploading a new data entity he can choose which roles should have access to the data entity. If access is granted to a specific role it can be configured to only allowing read-only access.



File: C:\Users\Adam\Desktop\Master.pdf

Keywords

Thesis

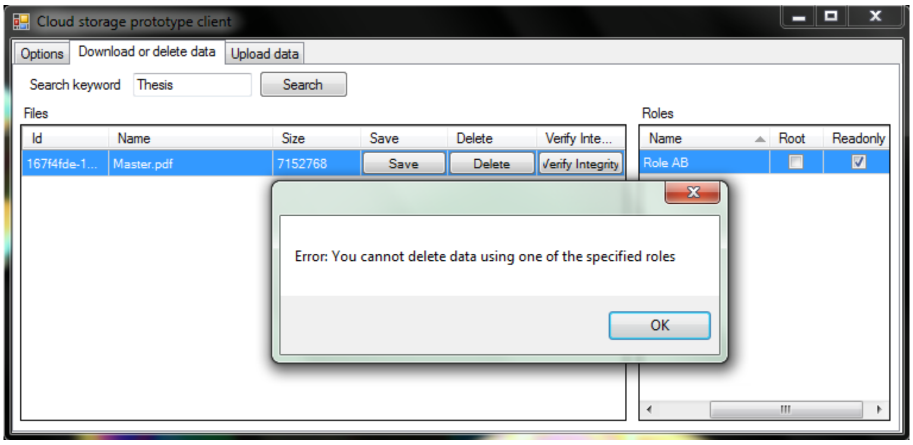
Thesis

Roles

Selected	Name	Root	Readonly
☑	Role A	☐	☐

Figure B.16: Screenshot showing a user uploading a data entity.

Data entity search: Fine-grained data access has been implemented when searching for data entities through the user client. The user will only be able to search for entities that are visible to the role or roles that the user is assigned to. Read-only access is enforced in such a way that users cannot delete or alter data entites that are marked as read-only. Figure B.17 shows an example of a user attempting to delete a data entity. The role, through which the user is attempting to delete the data entity, has read-only access to the data entity and thus it cannot delete the entity.



Error: You cannot delete data using one of the specified roles

Figure B.17: Screenshot showing a user attempting to delete a data entity.

B.3.3 Persistence

Like in the SDCD prototype, the CHAC prototype also stores information in XML files. As the information stored in CHAC has been extended to include roles, the information stored has likewise been extended. Examples of the XML files used in the CHAC prototype can be seen below.

Code Listing B.3 is an example of how user key pair information is stored. It is created by a user and then sent to the specific user it is associated with. Code Listing B.4 is an example of the data stored in the cloud containing information about roles, user, and data entities. From the examples it can be seen that while the XML files reveal structural information, they do not reveal information such as the names, actual keywords, or payloads of the stored data entities.

```xml
<?xml version="1.0"?>
<KeyCollection  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <PublicKey>QAAAAI1IfM8T9Im9JD ... /dkvIif4WMs6K1c=</PublicKey>
  <PrivateKey>FAAAABrkqaQnwsz1j ... EbRSjEeXoVB7b1U</PrivateKey>
  <SignKeys>BwIAAAAiAABEU1MyAAQ ... fPzYt70BiQ2/dfqH/</SignKeys>
</KeyCollection>
```

Listing B.3: Example of an XML file containing user key pair and signing key. Used in the CHAC prototype.

```xml
<?xml version="1.0"?>
<StorageMetadata xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <DataOwnerUserId>5f0b00b5- ... -d294fc8eb7af</DataOwnerUserId>
  <DataOwnerRoleId>5bd74c67- ... -7f0c6ff5126a</DataOwnerRoleId>
  <Users>
    <UserMetadata Id="5f0b00b5-6bd1-1203-8e6e-d294fc8eb7af">
      <Name>AUAAAACHyX2qqSw301Q2kp4jEtVd9Gq0gnbo=</Name>
      <SignPublicKey>BgIAAAAiAABEU ... 3F8StinRC</SignPublicKey>
      </UserMetadata>
      <UserMetadata Id="44aad512-cdd7-0923-974b-50428e133190">
      <Name>AUAAAABA65LJpdxycJmR/04BW5SFG9IcVdr8=</Name>
      <DelegationToken>QAAAAG/2i ... nDST7Cw==</DelegationToken>
      <SignPublicKey>BgIAAAAiAAB ... 5Sq3oR/x57C</SignPublicKey>
    </UserMetadata>
    <UserMetadata Id="2bc9d045-05f9-d25a-4d30-56d7867743d7">
```

```xml
      <Name>AUAAAACd0fsMwI1wGnoMIy2AfhV0Xq11qlJ5X=</Name>
      <DelegationToken>QAAAAARSH ... c0ZQFPw==</DelegationToken>
      <SignPublicKey>BgIAAAAiAAB ... tsOg85xs9sR</SignPublicKey>
    </UserMetadata>
    <UserMetadata Id="6b052317-3e57-330f-6713-1fe60171ebb4">
      <Name>AUAAAAAh7m7UN+RBBuKNEEk1AiPxkSxnWRd96=</Name>
      <DelegationToken>QAAAACHjX ... gMjncyA==</DelegationToken>
      <SignPublicKey>BgIAAAAiAAB ... 0BiQ2/dfqH/</SignPublicKey>
    </UserMetadata>
  </Users>
  <Roles>
    <RoleMetadata Id="5bd74c67-8bd0-41ea-b29c-7f0c6ff5126a">
      <Name>AUAAAABidsCKy0ooUXHl4AzuSXasjxlkvUn2=</Name>
      <AssignUnassignRole>true</AssignUnassignRole>
      <CanManageSubRoles>true</CanManageSubRoles>
      <CanCreateRoot>true</CanCreateRoot>
      <CanCreateUsers>true</CanCreateUsers>
      <IsRoot>true</IsRoot>
      <DataEntityPermission>ModifyContent</DataEntityPermission>
      <ChildRoles>
        <ChildId Id="3de5fb87-a13b-48a9-b6ec-219819969a0b" />
      </ChildRoles>
      <Users>
        <ChildId Id="5f0b00b5-6bd1-1203-8e6e-d294fc8eb7af" />
      </Users>
      <DataEntities>
        <EntityId Id="167f4fde-125f-4c29-aff2-b5364d254c4f" />
      </DataEntities>
    </RoleMetadata>
    <RoleMetadata Id="3de5fb87-a13b-48a9-b6ec-219819969a0b">
      <Name>AUAAAACa/kWSXmYHlxasOGhXBF22CSeWcaZ5=</Name>
      <AssignUnassignRole>true</AssignUnassignRole>
      <CanManageSubRoles>true</CanManageSubRoles>
      <CanCreateRoot>true</CanCreateRoot>
      <CanCreateUsers>true</CanCreateUsers>
      <IsRoot>false</IsRoot>
      <DataEntityPermission>ModifyContent</DataEntityPermission>
      <ChildRoles>
        <ChildId Id="ed3d3d57-9d8b-49a8-8835-dd86c9761ff6" />
        <ChildId Id="4c41962c-2d49-457c-93dc-b96b3970175f" />
      </ChildRoles>
      <Users>
        <ChildId Id="44aad512-cdd7-0923-974b-50428e133190" />
      </Users>
      <DataEntities>
        <EntityId Id="167f4fde-125f-4c29-aff2-b5364d254c4f" />
      </DataEntities>
    </RoleMetadata>
    <RoleMetadata Id="ed3d3d57-9d8b-49a8-8835-dd86c9761ff6">
      <Name>AUAAAABl+ibun+Y7Ts1WJmWslg09JQJVNn6V=</Name>
      <AssignUnassignRole>true</AssignUnassignRole>
      <CanManageSubRoles>true</CanManageSubRoles>
      <CanCreateRoot>false</CanCreateRoot>
      <CanCreateUsers>true</CanCreateUsers>
      <IsRoot>false</IsRoot>
      <DataEntityPermission>ModifyContent</DataEntityPermission>
      <ChildRoles />
      <Users>
        <ChildId Id="2bc9d045-05f9-d25a-4d30-56d7867743d7" />
      </Users>
      <DataEntities>
        <EntityId Id="167f4fde-125f-4c29-aff2-b5364d254c4f" />
      </DataEntities>
    </RoleMetadata>
    <RoleMetadata Id="4c41962c-2d49-457c-93dc-b96b3970175f">
      <Name>AUAAAAADB/VXg+jn+Eqqp9sHxz2H7VyY31wJ=</Name>
      <AssignUnassignRole>true</AssignUnassignRole>
      <CanManageSubRoles>true</CanManageSubRoles>
      <CanCreateRoot>false</CanCreateRoot>
      <CanCreateUsers>false</CanCreateUsers>
      <IsRoot>false</IsRoot>
      <DataEntityPermission>ReadContent</DataEntityPermission>
      <ChildRoles />
      <Users>
        <ChildId Id="6b052317-3e57-330f-6713-1fe60171ebb4" />
      </Users>
      <DataEntities>
        <EntityId Id="167f4fde-125f-4c29-aff2-b5364d254c4f" />
      </DataEntities>
    </RoleMetadata>
  </Roles>
  <Entities>
    <EntityMetadata>
      <Id>167f4fde-125f-4c29-aff2-b5364d254c4f</Id>
      <AuthorId>44aad512-cdd7-0923-974b-50428e133190</AuthorId>
      <Name>5iLOQt+n9c7Exd7qS9Bcdg==</Name>
      <Size>7152768</Size>
      <Attributes>
        <EntityAttribute Id="e20191de- ... -1afa-6367a6ff865f">
          <Keyword>6bmU1txD3Si0fFFsNo5Pzw==</Keyword>
        </EntityAttribute>
      </Attributes>
      <AesKey>AUAAAABtqzBYSB/21Yy05B0y ... wdHdNHL8XE=</AesKey>
      <AesIV>AUAAAAAHOddnMVOC7hXPWjc+K ... yeJzVt6iLA=</AesIV>
      <Signature>Mdi1yzEoXn7kCknx1TrB6 ... a6S7ag==</Signature>
    </EntityMetadata>
  </Entities>
</StorageMetadata>
```

Listing B.4: Example of an XML file storing information about roles, users, and data entities in the CHAC prototype.

B.3.4 Implementation effort

The CHAC prototype successfully validates most of the conceptual design of CHAC. Like for the SDCD prototype, the CHAC prototype is built by using different technologies, frameworks, and platforms. As the CHAC prototype builds upon the SDCD prototype, the code of the SDCD prototype is reused. The MIRACL library and the PRL libraries are reused without modifications. The code written specifically for the SDCD prototype has undergone only minor modifications.
Besides the code reused from the SDCD prototype, several lines of code has been written specifically for the CHAC prototype to support its concept of roles and the associated functionality. Except for code which is part of the Microsoft .NET framework, Table B.2 presents an overview of the amount of code used in the CHAC prototype. The number of lines of code shown in the table do not include C# code generated by using tools for designing the user interface of the administration and user client. This would add another $\approx$ 900 lines to the code produced for the CHAC prototype. In total, the effort for developing the SDCD prototype was approximately one person-week.

Table B.2: Approximate number of lines of code in the CHAC prototype.

Area of prototype	Language	Lines of code	Description
MIRACL library	C	$\approx$ 32000	Commercial third-party library used by the PRL library.
PRL library	C/C++	$\approx$ 3800	Library implementing the PRE scheme used by the SDCD prototype.
Inherited from the SDCD prototype	C#	$\approx$ 4250	Code written specifically for the SDCD prototype.
Code we produced for the CHAC prototype	C#	$\approx$ 1800	Code written specifically for the CHAC prototype.
Total	Mixed	$\approx$ 41850	Total number of lines of code compiled for constructing the CHAC prototype.

Piechotta, C., Jensen, A. E., Olsen, M. G., Secure Dynamic Cloud-based Collaboration with Hierarchical Access, 2012.

Department of Engineering
Aarhus University
Edison, Finlandsgade 22
8200 Aarhus N
Denmark

Tel.: +45 4189 3000