UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

A Model for Compound Purposes and Reasons as a Privacy Enhancing Technology in a Relational Database

by

Wynand Johannes Christiaan van Staden

Submitted in fulfilment of the requirements for the degree

Philosophiae Doctor

in the Faculty of

Engineering, Built Environment, and Information Technology, University of Pretoria

July 2011

Abstract

The protection of privacy related information of the individual is receiving increasing attention. Particular focus is on the protection of user interaction with other users or service providers. Protection of this interaction centres on anonymising the user's actions, or protecting "what we do". An equally important aspect is protecting the information related to a user that is stored in some electronic way (or protecting "who we are"). This may be profile information on a social networking site, or personal information in a bank's database.

A typical approach to protecting the user (data owner) in this case is to tag their data with the "purpose" the collecting entity (data controller) has for the data. These purposes are in most cases singular in nature (there is "one" purpose – no combinations of purposes – of the data), and provide little in the way of flexibility when specifying a privacy policy. Moreover, in all cases the user accessing the data (data user) does little to state their intent with the data.

New types of purposes called *compound purposes*, which are combinations of singular or other compound purposes, are proposed and examined in this text. In addition to presenting the notion of compound purposes, compound reasons are also presented. Compound reasons represent the intent of the entity using the data (the data user) with the data.

Also considered are the benefits of having the data user specifying their intent with data explicitly, the verification of compound reasons (the data user's statement of intent) against compound purposes, the integration of compound statements in existing technologies such as SQL by providing a model for using compound purposes and reasons in a relational database management system for protecting privacy, and the use of compounds (purposes and reasons) as a method for managing privacy agreements.

KEY WORDS
Privacy enhancing technology, purposes, compound purposes, access control, verification

Contents

List of Figures

List of Acronyms

ACS Access Control Subsystem

AIP Allowed Intended Purpose

AP Allowed Purpose

BLAC Blacklistable Anonymous Communications

CSPS Complement to Suitable Purpose Set

CVV2 Card Verification Value Two

DAC Discretionary Access Control

DAG Directed Acyclic Graph

DBA Database Administrator

DNS Dynamic Name Service

DPS Domain Purpose Set

E-P3P Enterprise Platform for Privacy Preferences

EPAL Enterprise Privacy Authorisation Language

FICA Financial Intelligence Centre Act

RICA Regulation of Interception of Communications and Provision of Communication-related Information Act

FI Financial Institution

FTP File Transfer Protocol

GLB Greatest Lower Bound

GUI Graphical User Interface

HDAC Hybrid Discretionary Access Control

HDB Hippocratic Database

HP Hewlett-Packard

IP Intended Purpose

IT Information Technology

LUB Least Upper Bound

MAC Mandatory Access Control

MaxAL Maximal Acceptance Limit

MinAL Minimal Acceptance Limit

NM Nymble manager

OECD Organisation for Economic Cooperation and Development

OO Object Orientation

P3P Platform for Privacy Preferences

PCI-DSS Payment Card Industry Data Security Standards

PC Personal Computer

PDACS Purpose Driven Access Control Subsystem

PDP Policy Decision Point

PEREA Privacy-Enhanced Revocation with Efficient Authentication

PET Privacy Enhancing Technology

PII Personal Identifiable Information

PIP Prohibited Intended Purpose

PKI Public Key Infrastructure

PL Purpose Lattice

PM Pseudonym Manager

PR Public Relations

RBAC Role-Based Access Control

RDBMS Relational Database Management System

RFID Radio Frequency Identification

RS Reason Set

SFTP Secure File Transfer Protocol

SPS Suitable Purpose Set

SSO System Security Officer

TOR The Onion Router

UST Unlinkable Serial Transactions

XACML Extensible Access Control Markup Language

XML Extensible Markup Language

Soli Deo Gloria

Chapter 1

Introduction

> A city is a large community where people are lonesome together.
> – Herbert Prochnow

Entities providing services (enterprises, businesses, and so on) that require clients (customers) to divulge information about themselves have an unquestionable responsibility regarding the clients' information. In general, many businesses may operate without such information; however, it is becoming increasingly difficult to obtain the necessary goods and services without divulging some private piece of information regarding oneself to an entity providing some service.

Many useful services are offered in the online spectrum, and this is exactly where protection of collected information becomes an interesting topic. This is for two reasons: the ease with which large quantities of information are collected, stored, and collated, and secondly, businesses need information on the individuals that they serve to survive in a competing market.

In order to benefit from the services offered in this online spectrum, an individual will have to *relinquish complete control of a portion of their personal information*. This statement sounds harsh, but when one considers that in sharing information with another person, one cannot (in the context of online information sharing) control with whom else that information will be shared, then the statement is credible. It is especially relevant if the entity with whom one shares the information seeks information in order to remain a competitive business.

An example of relinquishing complete control of one's personal information is retail internet banking. In order to benefit from internet banking, a person will typically be required to provide the bank with personal information. In South Africa, for example, current legislation under the Financial Intelligence Centre Act (FICA) [36] requires that personal information not only be collected, but that proof of the physical living address of the person about whom the data is being collected be provided and recorded. The information collected by the bank (under

FICA) could be as small as a subset of the information that the bank already has on record, but the example illustrates that personal information is part of the agreement between the service provider and the individual wishing to subscribe to the service. Internet banking is a very good example of a service offered that should be tightly regulated in terms of actions involving privacy related information.

A more relaxed example of an internet service with specific privacy requirements would be a company offering the opportunity to download a trial version of their software. Many such companies require individuals to register (read: provide some personal information) before downloading the software. The information is typically used to profile potential buyers of software, and to be able to market future software products to individuals who have already intimated an interest in a current product.

Another, perhaps quintessential, example of online services in which privacy information is an absolute requirement is *social networking websites*. A social networking website provides benefits in the social realm; one is offered the opportunity to expand one's social network, to reconnect with old friends, and so on. However, in order to sign up for these types of services, one has to (quite understandably) provide personal information.

From the examples above it is clear that personal information is traded for some service; and a result the Internet user should have an inflexible attitude towards privacy (they should demand that their personal information be handled *responsibly* by everyone that collects it), and that the discussion of privacy is much wider than simply stating that *private information should be protected*. There are, for example, ethical questions about the individual's right to privacy, in particular, the question of whether one can have too much privacy – can the protection of privacy of the individual negatively impact the general well-being of the society which offers them this protection, and if this is the case, is there some line where privacy protection should stop in favour of protecting the interest of the community? The aim of this text is not to delve into the philosophy of the right to privacy and other epistemological questions of that nature. Instead, it is assumed that individuals are offered some level of privacy (no delineation is provided), and the considerations around protecting that privacy, no matter the level, are explored in detail.

The consideration of privacy protection (examined in this text) can thus be placed in three focus areas: firstly, an understanding of what privacy is (especially in the context of the internet), secondly, identifying what information would be considered privacy related information, and finally, understanding existing solutions to protecting privacy related information, in order to provide future solutions to privacy protection.

Understanding what privacy is, relates directly to the way in which it will be protected – the functionality of any computerised system is driven by what is ex-

pected of it, and as such the individual's expectation of a system that protects privacy is closely related to what individuals understand privacy to be. What exactly is considered to be privacy related information will be different according to each individual. Some individuals are naturally more protective of any information relating to their person, and will share very little information with others. Other individuals are more lenient in what they share. However, there are some common areas that are understood to be sensitive, such as one's residential address, or credit card information – really only the credit card number and the three digit Card Verification Value Two (CVV2) number on the back of the card are needed in order to make purchases with a credit card. Other pieces of information are not clearly sensitive or "not sensitive". An example is a postal address – for many individuals, receiving unsolicited mail is a cause for concern, particularly when one's postal address is the same as one's physical address. For others, sharing a postal address is less of a concern.

As a result of the inherent complexities in correctly categorising sensitive data and data that is not sensitive, it is difficult to provide a taxonomy of information which will clearly stipulate what types of information should be considered more sensitive than others. Such an endeavour would have to make demographic distinctions (to account for cultural differences) and create profiles on individuals. Apart from the very obvious *sensitive* data, because of personal choice many individuals may have, the safest choice is to allow the individual to determine what information is most precious to them. And therefore, mechanisms are needed to allow individuals (hereafter referred to as the *data owners*) to determine *for themselves* how the entity collecting information (hereafter referred to as the *data controller*) on them is to behave with the information.

The mechanisms that are used to protected the plethora of privacy related information can be divided into three categories: mechanisms in governance, law, and technology. In chapter 2 these methods are considered in more detail, but a quick mention of each is necessary to move the current discussion forward.

- Governance relates to best practice that a data controller may employ – promises made through published policies, checks and balances.
- Privacy protection through legislation – laws stipulate how privacy related information should be handled. Laws will also provide a clear view of what actions would constitute a violation of privacy.
- Technological means of protecting privacy – this is the use of technology to prohibit violation of privacy by ensuring that private information is treated correctly.

The focus of the work in this text is the technological means of protecting

privacy. The other two methods, being governance and legislation, are not considered in much detail (the text considers privacy protection from a computer science perspective rather than a business administration or legislative perspective). To understand the technological means of protecting privacy, it is necessary to introduce the categories of privacy related information. Privacy related information can be divided into two main categories (see chapter 2 for more detail), namely *what we do*, relating to communication actions (and other interactions), and *who we are*, information describing an individual (where they live, their banking details, and so forth).

One may argue that *who we are* is very close to *what we do* since a person's communication habits may very well be enough to describe them uniquely. However, *what we do* information is considered to be all information about a user's current actions. Once communication habits are stored for future reference, it is understood that the information transitions from *what we do* to *who we are*.

It is thus not surprising that the technological means of protecting privacy is divided into two types: technologies that protect communications privacy, and technologies that protect stored information that can be used to uniquely identify an individual (or Personal Identifiable Information (PII)). In addition to focusing on the technological aspects of protecting privacy, the attention in this text is also on the protection of PII. However, chapter 2 provides a detailed introduction to both technologies in order to provide an understanding of the position of the work presented here.

As mentioned, many services offered require some form of PII to be shared. It may also be the case (as with retail internet banking) that personal information is necessary in order to conduct business. A socially conscientious data controller should thus be concerned with protecting this private information. From a governance perspective there are many texts which dictate (or suggest) what mechanisms are necessary in order to be responsible regarding PII (see section 2.1.1).

In the privacy research field, (also see section 2.1.2), it is generally accepted that there are two phases in protecting PII. The first phase is concerned with making a promise to the data owner[1] regarding the use of their information. The second phase is enforcing that promise when data users request access to that data[2].

There are several problems with the way in which privacy protection is treated in current technologies. These are listed in the following paragraphs. This leads to the formulation of the research problem addressed in this text.

In the privacy binding phase, current technologies that enforce the protection

[1]Throughout the text no distinction is made between the data owner and someone acting on their behalf.

[2]Throughout the text no distinction is made between a data user or the principal acting on behalf of the data user

of privacy of the data owner make use of *purposes* that stipulate the *data user's* intent with data. When a data user wishes to access protected information, information in the data user's profile is matched against the stipulated reasons for storing information, and access is granted only if the information in the profile matches the stipulated reasons (the *privacy limitation phase*). The current method for expressing the purposes for storing information is somewhat limited in that the data user is typically only required to specify a list of purposes for which data is stored. Only in limited cases is a form of relationship between the purposes specified (and then only implicitly).

Another limitation is that in most cases agreements between the data controller and the data owner are constructed with very little choice offered to the data owner. A draconian approach is followed: "Either give us all your data (or at the very least all the data we want), or go away." This can lead to a fractious attitude from individuals who are privacy conscious – which should be everybody who subscribes to any service on the internet.

The problems mentioned here lead to the construction of the question that this text answers, and the following section will enunciate the components of that question.

1.1 Problem Statement

The motivation for this research is the identification of shortcomings in current mechanisms used to protect PII: current mechanisms do not provide an expressive and flexible mechanism for constructing privacy promises, nor for forcing the users of data to state their intent with data.

The text provides a solution to this problem by asking and answering three questions:

1. How can the expressiveness of purposes bound to data be enhanced from the perspective of both the data controller and the data user?

2. Can these more expressive purposes be integrated with current technologies in a non-intrusive way?

3. Can this more expressive way in which purposes are stated help provide more flexibility in the agreements drawn up between the data controller and data owner?

1.2 Approach

In order to answer the questions posed in the previous section, it is first necessary to understand the nature of privacy protection systems. This includes the general concepts of privacy as well as the different technological approaches taken to protect it.

After this discourse a structure which supports a more expressive approach to purposes specification is given. The structure allows the efficient manipulation of purposes (and reasons). The manipulation of purposes (and reasons) is modelled mathematically, and algorithms are presented for access control based on purposes (and reasons).

The answer to the second question, that of integrating the proposed method with existing technologies, is then examined. It is shown how the proposed methods can be integrated with current data storage, access languages and systems in a non-intrusive manner. Data users who perform tasks that are not privacy related are not impacted, and those users who are concerned with privacy related work are required to explicitly state their intent with data, providing more verbose audit trails, and in general enhancing the privacy controls offered by the data controller.

The structure for the expressive forms used during purpose binding allows the concept of privacy agreements as discussed in another source [69] to be augmented, and even extended in places. It provides the opportunity for a relationship of trust to develop between the parties. This result is discussed in detail, and the work shows how such agreements may be managed.

The following section will provide a more detailed layout of the text.

1.3 Structure of the Text

The rest of the text is structured as follows: chapter 2 provides background information relevant to Privacy Enhancing Technologies (PETs), and the ideas discussed in this text. Chapter 3 provides a description of compound expressions, the structures that support them and the semantics behind them. An algorithm is given in chapter 4 for performing verification using compound expressions, as well as proofs that the verification is correct. Chapter 5 illustrates how compound expressions can be integrated into current technologies, specifically, Relational Database Management Systems (RDBMSs), which are currently the most widely used method for storing information on people. Finally, chapter 6 shows how the proposed structures and compound expressions augment and extend privacy agreements between the data controller and the data owner. Chapter 7 provides a conclusion and review of the work and ideas presented.

At times the masculine pronoun, e.g. he/him etc. is used in this text. However,

this is merely for convenience and should not be taken to refer exclusively to the male gender.

The following chapter provides background information on PETs in general as well as the goals of certain types of PETs, the theories behind privacy protection and the myriad of mechanisms that PETs employ in order to protect privacy.

Chapter 2

Privacy Enhancing Technologies

> It's a dangerous business, Frodo, going out of your door.
> – Bilbo Baggins, The Fellowship of the Ring

In order to understand the principles of privacy protection, it is necessary to define the concept of *privacy*. A rudimentary definition will provide valuable clues as to what aspects of privacy should be considered during a discussion on privacy protection. Generally speaking, privacy is the right to a state in which a person is not being watched[1]. However, a person may still enjoy privacy even though they are being watched, for example a person having consented to being observed as part of an experiment, or in a social context (think of the many "reality" programmes on television). The above definition is rather quixotic in nature, however, it does indicate that an important aspect of privacy is the individual's afforded ability to indicate if they would like to be observed or not.

As mentioned in the previous chapter, there are many opportunities for privacy violation in the online world, and many examples of such violations can easily be found through an online search using a search engine. At the time of writing, the most recent example was that of AT&T leaking e-mail addresses from users on their *iPads*, or that of all the data being removed from a user's personal cellular telephone *remotely*[2].

Many privacy concerns also surround the social web, and although the social web is an important way in which people interact [31], the internet is not the only way in which privacy can be violated. For example, an individual's movements

[1] Oxford English dictionary. sv 'privacy'. Oxford: Oxford University Press, 2001

[2] No stable URLs can be provided as a reference, since these reports are from on-line magazines which do not normally keep their content over an extended period of time, however the URLs are provided in TinyURL (http://www.tinyurl.com) form for the interested reader. Search words: AT&T leak of iPad e-mail addresses, URL: http://tinyurl.com/297vtf7 (accessed 2011-07-11). Search words: Remote wipe of personal phone, URL: http://tinyurl.com/29o9b48 (accessed 2011-07-11)

can be tracked without their consent using physical tags attached to individuals, such as Radio Frequency Identification (RFID), or simply through their cellular telephones [9, 22].

The plethora of ways in which people interact with each other on media such as the internet places certain responsibilities on all parties that own data (the people), those that control data (service providers, or enterprises collecting data) and those that use data (other people, or principals – ultimately, people make decisions on the use of data).

The responsibility placed on the incumbent can be concisely defined in terms of a complete, albeit terse, definition of privacy. Privacy is **the right of information self-determination** [4, 37]. This right affords the data owner the inalienable right to determine what information is gathered about them, who has access to this information and in what way the information may be used [4]. The individual can thus consent to being monitored, and also determine what information about them may be gathered and how this information may be used (included in this is the right to state with whom the information may be shared).

This right of information self-determination and its enforcement rests on the ideology that everybody will "play nicely" – and this is obviously not always the case. Thus, appropriate action is taken to ensure that an individual's privacy is protected. In the current technology climate, three ways in which the individual's privacy can be protected are available (considering that not everybody will play fair). All three may be considered to be founded on the basic principles of ethics. This text is in no way a consideration of morals or ethics, and the matter is not considered from a moral perspective[3].

It is of course all well and good to provide the individual with all sorts of rights; however, without any vehicle for enabling these rights, they are rather useless. Hence *the privacy problem*. *The privacy problem* is essentially a collection of scenarios that each constitute an action or inaction that would violate the privacy of a data owner [72].

The vehicles for enabling privacy can be summarised as follows: firstly, privacy may be protected by way of legislation. These are the laws that govern modern societies and seek to enforce a manner of acting upon the members of the society. Many countries have created privacy related laws in the past (that is, the early days of the computer revolution), but these had to change to fit the nature of the Internet [70] – the relative ease with which PII can be made available to large numbers of different users at geographically dispersed locations. This creates significant difficulties when a country's legislation only describes the controls

[3]The text considers *the privacy problem* amidst the continuous attempts by adversaries to violate an individual's privacy rights. In fact, answers to *the privacy problem* will hopefully one day successfully answer all the questions regarding the protection of an individual's privacy.

and mechanisms for privacy protection of PII stored on systems within it's own borders. New laws must consider the flow of PII across its borders.

Developing countries have only very recently started examining the problem of cross-border flow of information, and privacy in general. In South Africa, for example, two limited pieces of legislation govern electronic communication and access to information [34, 80]. In the developed world, there is much more involved legislation to protect individuals effectively[48, 1].

The second method of protecting privacy is through organisational safeguards. The motivation behind organisational safeguards may be legislation, or it may purely be the result of an enterprise wishing to behave responsibly – perhaps to strengthen its reputation. In these cases, the responsible enterprise will have a published privacy policy, and individuals within the enterprise are held accountable for their actions with the data they may come into contact with. Unfortunately, little else is done to enforce the privacy policy [5, 51].

The final method for protecting privacy is technical solutions and as was observed quite early in the development of the privacy related field, technical means are the expected way in which privacy is protected in a technological environment [38]. In this case, both the legislative and organisational safeguard approaches are supported by employing technological solutions to the privacy problem. For the individual whose privacy is at stake, an automated solution should be capable of taking a published privacy policy (that the individual agrees to) and enforcing the rules laid out in it.

The previous categories can be layered and may together become a comprehensive solution to the privacy problem [72]. In this stratified view of privacy protection, the technological safeguards consist of identity management, personal control and private communications. These safeguards may be supported by other technologies such as firewalls and encrypted communications channels – see section 2.1.1. Intrusion detection systems could also be seen as a direct safeguard for privacy protection. However, for the purposes of this text, it is viewed as a supporting technology that safeguards the security of private information.

Solving the privacy problem requires the analysis, modelling, deployment, enforcement and auditing of privacy policies. Once again it is, of course, possible for all these steps to be performed in of one the three areas just mentioned (legislation, organisational, technological). The discussion in this chapter (and indeed, the remainder of this document) is limited to the technological aspects of privacy protection.

In this chapter background information on technological solutions and research surrounding technological solutions are provided. The terminology, best practices, as well as some of the current systems which are concerned with solving the privacy problem are considered. These systems will be examined by placing them in specific categories, which will be introduced later.

The layout of this chapter is as follows: section 2.1 provides more detail on the technological approach that is used to protect privacy. In particular best practices and policy languages are discussed. Section 2.2 provides a discussion on PETs and how they are commonly classified. It also provides a classification of PETs as they fit in the context of this work. Sections 2.3, and 2.4 continues the classification of certain types of PETs, but now specifically based on the classification mechanism provided in section 2.2. Some of the current technologies implementing PETs is provided in section 2.5.

In the following sections, a detailed discussion of the technological approach to protecting privacy is provided.

2.1 The Technological Approach

The technological approach to solving the privacy problem relates to the application of current technologies and methods to provide a mechanism through which individuals can actively participate in the protection of their privacy.

In such a paradigm, privacy is dealt with in three phases [16, 76]:

- Information self-determination: The individual has the ability to specify what data about themselves can be used, who can use it and how it can be used.
- Parallel to this is the action by the organisations that wish to use data. Their *bona fide* actions will boil down to stating what they intend to do with information that they gather legitimately.
- Finally, these two actions are brought together by using automated tools that will determine if the organisation's statements (effectively their privacy policy) match the user's preferences.

All of these phases can (and should) in a technological framework be supported by providing software tools to service each phase. Many tools do exist to facilitate these phases and, as is the case with all tools, many come close to meeting expectations of a PET. In this chapter reference is made to some of the more interesting tools that have a bearing on the work presented later in this text.

Before these tools are considered further, the guidelines for privacy protection are outlined in the next section. This provides a set of goals that a privacy protection mechanism should attempt to reach. The first guideline, *best practices*, is discussed next.

2.1.1 Best Practices

As far back as 1980, the Organisation for Economic Cooperation and Development (OECD) published a set of principles which form the basis for most of today's theories regarding privacy [70]. These principles were intended as a way for cooperating countries to honour the use of private information regarding their citizens. As such, they form a good basis for enunciating the general principles of the protection of privacy.

The most important principles of privacy protection according to the OECD, are the acts of purpose limitation and purpose binding [37], which have been mentioned already.

Purpose limitation is the act of limiting access to information based on the purpose that the information will be used for. Thus, only information that is needed for a specific purpose is released. Moreover, information is only released if the stated intent matches the purposes bound to the information. The act of binding is simply to state what information will be used for – the way in which this statement is accomplished will be considered in chapter 5.

A second document which describes not only policy, but also mechanism is the joint study by the IPC/Registratiekamer [47]. The document outlines the different domains that form part of a computerised system, and which mechanisms should be embedded inside each in order to ensure the protection of privacy.

Security vs Privacy

A good question to ask is: *where does privacy fit in with security*? The simple answer is that security is the motivation and mechanism through which data is to be protected. Privacy protection encompasses security in that it may make use of security principles to enable the protection of the privacy of the individual. However, security alone cannot accomplish privacy protection.

Privacy protection, then, begins with good security: for example, having firewalls and good network security can go a long way towards protecting privacy. In fact, a good example of the use of security to help with the protection of privacy is the Payment Card Industry Data Security Standards (PCI-DSS) [75], which outline several key practices in order to protect information on individuals. However, on its own PCI-DSS offers no assurances as to the privacy of the individual other than ensuring confidentiality, and *confidentiality is not privacy*: confidentiality relates to ensuring that only authorised persons get access to data, it makes no provision for the data owner to state for what purposes his data may be used.

PCI-DSS fails, simply because the first and second phases of any technology that aims to support the solution of the privacy problem involve the specification of privacy preferences by the individual, and a privacy policy by an organisation that

wishes to use information provided by individuals. PCI-DSS offers no purpose binding or limitation phase.

Since there are technologies that are not privacy capable, some tools that are privacy capable and that allow individuals and companies to specify their preferences and policies are considered below.

2.1.2 Policies and Policy Languages

A policy language exists either to allow the organisation to stipulate its reasons for using data or to allow the individual to specify their policy regarding the use of their personal data.

There are several policy languages that are to be considered: These are Platform for Privacy Preferences (P3P) [25], Enterprise Platform for Privacy Preferences (E-P3P) [88], Enterprise Privacy Authorisation Language (EPAL) [6], and the Extensible Access Control Markup Language (XACML) [68].

2.1.2.1 P3P

P3P [25] is widely regarded as the first practical policy language that allowed the enterprise, as well as the individual, to specify their stance on the collection of personal information. Its primary focus is web-browsing. In principle, the individual will construct a personal preference profile using some tool. The enterprise will construct its policy regarding the collection and use of personal information.

The browsing agent (the web browser, or other web client software) will contain functionality which will request the enterprise's policy, and compare the data owner's policy to that of the data controller. The result is that the policies are either compatible or not. If the policies are not compatible (the data controller does not share the data owner's view on privacy) then the data owner can still choose to go ahead with the recording of information, or they can opt out of giving the organisation any information at all. In many such cases the individual will no longer receive the benefit of a service from the particular organisation. In this text, this is referred to as a **do-or-die** approach to privacy, and chapter 6 will provide details on how this inflexible approach to solving the privacy problem can be mitigated.

Unfortunately, P3P has only a limited set of purposes that can be used in a privacy policy. Also, there is no relationship between purposes [25]. Enhancement of P3P has effectively stopped because of a low industry adoption rate [4].

[4]This inference is from the fact that the two authoritative websites that promote the use of P3P have admitted to halting work being done on P3P, firstly, http://www.w3.org/P3P, and secondly, http://www.p3ptoolbox.org which has indicated that the presence of the website is for archival purposes only.

Brandi and Olivier [13] use P3P to allow proxy servers to specify their privacy policies. This allows a user accessing some web server through a proxy to also examine the proxy's privacy policy. If the user does not agree to the terms, he is free to take appropriate action.

An enhancement of P3P, which would allow enterprises to publish enforceable privacy policies, is discussed next.

2.1.2.2 E-P3P

The problem with P3P is that at its core it is a mechanism that is built upon trust from the individual's perspective [88, 5]. The individual trusts the organisation to adhere to the privacy policy provided. However, in practice there is ample opportunity for the enterprise to act *mala fide*.

The E-P3P policy language [5] is an attempt at providing a standard way for the enterprise to publish its privacy policy (in much the same way as P3P), with more mechanisms that will allow the automation of transferring the policy to a practically implementable and enforceable policy [5].

The idea is that a policy is presented with a policy language that flows from E-P3P, and that the expression can be transferred to other technological devices (either logical or physical) that will enforce the policy.

There is an intended compatibility with P3P, and Ashley et al. provide sound reasoning for and examples of converting an E-P3P policy into a P3P policy [5]. The focus of this text excludes a discussion on such a conversion protocol, and the matter is not discussed further.

In the following section, an interoperability standard language EPAL is presented.

2.1.2.3 EPAL

EPAL [6] is an interoperability standard in which it is possible to define the privacy policies that govern the use of PII in an enterprise-wide Information Technology (IT) environment. The standardisation of a language such as this one allows many organisations to specify their enterprise-wide privacy policies, which will in turn allow interoperability between different organisations.

The core feature of EPAL is that it is data model independent. That is, it does not tie itself to a particular representation of data. It is foreseen that an enterprise PET will be EPAL enabled, and thus be able to input an EPAL policy and enforce the privacy policies encoded in the EPAL policy file.

The language itself caters for the specification of several elements within the privacy policies of the enterprise. It defines lists of hierarchies of data categories, user categories, and purposes, privacy actions, obligations and conditions. Data

categories, user categories and purposes are discussed in more detail below. Privacy actions and obligations are not discussed since they are beyond the scope of this text.

EPAL elements

Data categories The data categories represent high-level definitions of the types of data. Some examples might be postal address, credit card information, and so forth. The data categories are, however, placed in a hierarchy, allowing the possibility of providing more fine-grained descriptions of the data categories. This specificity will typically happen as one moves downward in the hierarchy of the data category.

User categories These define the users in the IT environment that will interact with the stored data. This data is once again placed in a hierarchy, allowing more general users at the top (groups, other organisations), and more specific users at the bottom (individuals).

Purposes These are placed in a hierarchy, and will typically indicate the intent for the data. More general purposes are placed at the top, and more specific purposes are placed closer to the leaves. The EPAL specification states that in order to use a specific purpose, the user either has to be granted access to it specifically, or the user should be granted the use of all the children purposes of that purpose.

2.1.2.4 XACML

XACML by the OASIS group [68] is the specification of an extensible policy specification language. Extensions to XACML abound as testimony to its application as an extensible policy language, for example: Kounga et al. [54] who use XACML to define preference based authorisation, and Rissanen et al. [85] who extend XACML to enforce distributed access control.

Policies in XACML are done in Extensible Markup Language (XML) to provide a standardised method for specifying policies. A standardisation in this way will allow an enterprise to provide a unified view of security and privacy policies across the entire enterprise.

Another benefit of using XML is that policies can be exchanged with other enterprises, allowing a more pervasive adoption of privacy practices. An important other benefit is then that a data subject's data can be tagged with a particular

policy, which can travel with it from enterprise to enterprise (using the privacy agreements presented in chapter 6, the same can be accomplished).

To provide a common policy language it is, of course, important to determine exactly what the requirements are for such a policy language. The OASIS group defines ten principles for such a policy language:

- A method for combining rules/policy into a policy set that applies to a decision request
- A flexible method to define how rules and policies are combined
- Dealing with multiple subjects acting in different capacities
- A method for basing an authorisation decision on attributes of the subject and resource
- A method for dealing with multivalued attributes
- A method for basing an authorisation decision on the values of a resource
- Provide a set of logical and mathematical operators on attributes of the subject resource and environment
- A method for handling a distributed set of policy components, while abstracting the method for locating, retrieving, and authorising the policy components
- Rapidly identifying the applicable policy
- An abstraction layer (hides application environment from the policy writer)
- Actions that must be performed in conjunction with policy enforcement

As well as specifying these ten principles, the XACML also defines standard language elements, discussed in the following section.

Language elements The XACML specification [68] provides a series of language elements which are used to specify a security and privacy policy. Aside from these base language elements, because the specification language is expressed in XML it is possible to extend the language itself, thereby allowing non-standard policies to be specified as well.

A lack of space and scope (the text is not concerned with specifying a *policy language*), precludes a detailed discussion of the XACML, and the interested

reader may read the XACML specification [68]. However, the core of the language is presented to clarify the principles of the language, and how it may apply to privacy protection in general.

XACML provides three top level elements which are used to construct any policy. These are `<rule>`, `<policy>` and `<policyset>`. Together these three elements form the basis for defining an access control policy in XACML.

`<policyset>` is the top level grouping for a set of rules. This element can also contain other policy sets.

`<policy>` contains rules that define the policy applicable to a certain decision request.

`<rule>` defines at the lowest level the rules that specify under which circumstances access is granted or denied for a particular decision request, defining the applicable rules that apply to a particular access context.

Method for specifying policy combinations The important aspect of XACML is that it allows a unified way of specifying/viewing a policy. This means that there will be some cases in which the same target (resource, data subject's information) will fall under the context of several policy sets. In order to solve the potential problem of having several conflicting policies, the XACML provides a simplified method for deciding on what the result would be of combining several policies.

This decision is in no way a complex one which results in the systematic combination of policies and rules. It is simply a way of stating what action to take whilst evaluating the rules of a policy. These rules are:

1. Deny-overrides: Whenever a rule evaluates to *deny*, stop evaluation, and the result of the decision is "deny".

2. Permit-overrides: Whenever a rule evaluates to *permit*, stop evaluation, and return the result of the decision as "permit".

3. First-applicable: A policy may include targets and resources that do not apply to this object itself. The policy is, however, still evaluated. A first-applicable states that the first rule/policy that applies to this object specifically should be evaluated.

4. Only-applicable is same in the sense that it will select the policy/rule that applies to this object, and only to this object. If no such policy exists, the result of the decision is "indeterminate".

Because of the extensibility, users of the XACML are permitted to specify their own combination algorithm.

Basing decisions on subject, resource attributes In many cases it is important that a decision be made based on the value of some attribute, for instance “access only allowed if user part of administration group”. In such a case, the user may have an attribute “admin” associated with it. Another example may be “access only allowed if user logged in via secure method”. It is important to make the distinction between the presence of attributes and the values of those attributes.

Decisions based on the values of attributes In this case it may be necessary to specify that a user will only be granted access if the time is between 15:00 and 16:00. In order to execute these types of access decisions based on the values of attributes, it may be necessary to perform calculations on the attribute values. For this reason, the XACML includes operators defined to work on a standard set of data types.

Rapidly identifying the correct policy This can be accomplished by indexing the policies on the target, or the resource.

Obligations In many instances it is necessary for the organisation to comply with some externally imposed regulation. This may be that the parent’s consent is needed before the data subject is contacted via e-mail and so on. These external obligations may be imposed by the data subject, best practice recommendations and legislation.

Data model The data model as considered by XACML is presented in figure 2.1.

The text now presents the classification of the foundational PETs.

2.2 Classification of Privacy Enhancing Technologies

In general, systems that attempt to solve the privacy problem can be categorised according to the functions that they perform. These categories allow the evaluation of the type of protection offered by the PET as well as the degree to which it can protect privacy.

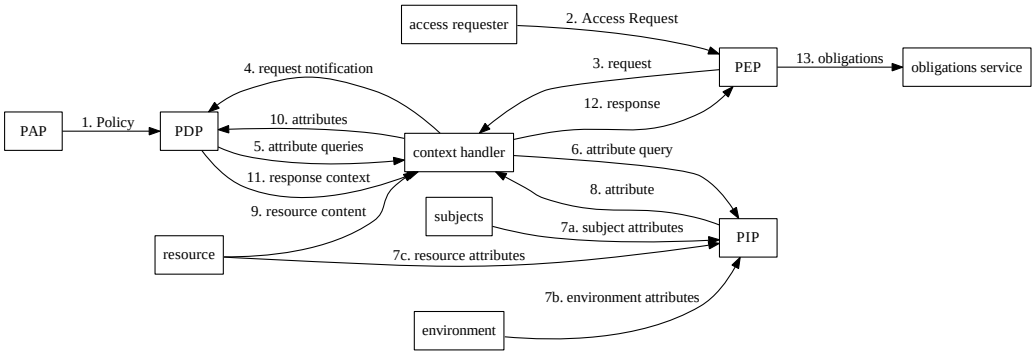

Figure 2.1: XACML data model

Some more generally accepted classifications are presented next, followed by a more general classification introduced in this text. The first point of interest, however, is the classification provided by the OECD.

2.2.1 Organisation for Economic Cooperation and Development Classification

The OECD provides four high level categories for PETs [71]: *personal PETs*, web-based technologies, information brokers, and network-based technologies.

2.2.1.1 Personal PETs

These are applications that are either installed locally on an individual's computer or deployed as part of a group policy on their network, or available as a service on a wider network. Users have the ability to configure these applications to act on their behalf and make critical privacy decisions.

- Cookie managers/blockers: These allow users to manage the cookies that are placed on their computers. This will include interrogation to determine which web-sites have placed cookies on their systems, and also viewing the content. Importantly enough, they will also allow users to decide whether or not a site can place a cookie on their computer.

  Most web browsers already provide some features of cookie management, allowing users to decide if they want to allow a particular site to place a cookie on their computer or deny it from doing so.

- Ad blockers: software which will block the delivery of advertisements. A more common example of this is *pop-up* blockers embedded in many web browsers today.

- Encryption software: Although encryption software originally falls into the realm of computer and information security, it does offer the ability to protect privacy if it is used to encrypt data streams and data stored on volatile and non-volatile storage media. There are many examples of encryption software, but perhaps the most relevant is the use of asynchronous encryption [78], which allows a user to encrypt data with the knowledge that the only person who can decrypt the message is the owner of the private key.

2.2.1.2 Web-based Technologies

Web-based technologies are technologies that allow users to communicate privately, and also to clearly specify their policies when interacting with a service provider.

- Anonymisers: These will allow the communication of a user to be private. The types of anonymisers are identified in section 2.2.4.2.
- The P3P: This allows the user to specify what information regarding their person may be divulged to a visited website. E-P3P, EPAL and XACML are not considered to be part of this classification, as they are employed by organisations to specify their privacy policies.

2.2.1.3 Information brokers

The information broker is a third party which will collect information on the user's actions, and allow the user to direct the usage of that information. Other parties can then purchase the information (restricted to that information as stipulated by the user) from the broker [2, 71].

2.2.1.4 Network-based technologies

- Proxies and firewalls: These can protect by hiding a user's internet protocol (IP), blocking certain content, hiding a user's Dynamic Name Service (DNS) lookups, or scanning for viruses. One example of such a tool is *privoxy*[5].

  An interesting application of a "privacy" proxy (arguably rooted more in paranoia than in a concrete interest in protecting privacy) is Scroogle[6]. The owner of the service funnels searches entered by visitors to the site to Google, effectively prohibiting profiling, since all searches seem to originate from a single IP address (Scroogle's).

[5] http://www.privoxy.org
[6] http://www.scroogle.org

- Privacy networks: These are typically implementations of Chaum's mixes (see section 2.3.1), for example *mixminion* [27], *mix-master* [24], and *The Onion Router (TOR)* [32].

2.2.2 A Privacy Architecture

Olivier [72] offers a categorisation of *PETs* which include non-technological solutions. These categories are *personal control, identity management, organisational safeguards,* and *private communications*. PETs that offer personal control are all those that allow an individual to control the use of their information. Identity management includes those PETs which safeguard an individual's identity from malicious use; several current technologies fall under this category, as well as the personal control category. Organisational safeguards are the same as those already mentioned in the introduction to this chapter; and private communications are those that provide the individual with the ability to communicate privately (which may include anonymous communications).

2.2.3 A Higher Level Classification

It is obviously important to recognise the fact that PETs can be classified in many ways, and this text appreciates this fact. However, in order to understand PETs from a user's perspective it is necessary to classify them according to their functionality. There are typically two types of actions that a user will take when interacting with another entity.

They will either send messages which will be consumed immediately, or they will send messages which will be stored for later consumption (or use). Messages that will be consumed immediately are typically commands that are sent to other computers, whereas messages that are stored for later consumption are typically information on the user themselves, such as name, age, gender, and so on. A typical example may be information that is used to conduct financial transactions.

E-mails are messages that are typically stored for later consumption, but will necessarily include information on the user such as their e-mail address.

In this text, messages that are consumed immediately are classified as *what we do* information, since it will describe browsing habits for example. Systems that protect what we do attempt to protect the individual by providing anonymity. Furthermore, messages that are stored for later consumption are classified in this text as part of *who we are* information, since they will typically describe the interacting entities as individuals. Systems that protect who we are act in scenarios where an individual cannot remain anonymous (even pseudonimity in these situations would create problems), and their information is imperative to provide them some service, or in order to comply with legislation.

2.2.4 Foundational Concepts

Before the discussion on PETs is continued in earnest, it is a good time to step to the side, and first introduce foundational concepts on aspects of privacy and PETs.

The progression of the discussion depends on the presentation and definition of two fundamental concepts in the privacy research field. These two concepts are briefly defined, with a more detailed discussion to follow.

The first fundamental concept is an anonymity set. Anonymity sets are used as a foundation on which to reason about privacy in a quantitative way. Using these sets, it is possible to start considering the probability with which a single element can be identified amongst other elements within the set, that is, determining who sent or received a message.

The second fundamental concept in a PET is the type of privacy that one may enjoy. These types all seem to stem from (and indeed are in many instances used interchangeably with) anonymity. The types of privacy are by general consensus accepted to be four: anonymity, unlinkability, unobservability, and pseudonimity [77]. In any one of these four, the data subject is supposed to be protected from some form of monitoring.

The concept of anonymity sets are now discussed in more detail.

2.2.4.1 Anonymity sets

Before a lengthy discussion of privacy terminology is given, one of the core concepts of many PETs is dealt with. An extremely important aspect of anonymity and privacy is the concept of measuring the degree to which an act of communication remains private [102], that is, the act of communication cannot be linked to a single individual. In order to formalise this measurement, the general notion of an anonymity set is put forward.

Consider a collection of people who are all communicating over some network. If all of these people communicate using some PET, then they all belong to what is commonly referred to as the *anonymity set*. This set is used to model an attack against anonymity. Loosely speaking, the ideal situation occurs when it is impossible for an attacker to uniquely identify any individual over another as a participant in some event.

It is also possible to define an unlinkability set, or an unobservability set. However, the same definitions can be applied to those sets as can be attributed to an anonymity set. Moreover, it is possible to think of anonymity in the context of an anonymity set as just that attribute, the individual's apparent likeness to other individuals in the set. So, it is possible to think of anonymity of observability and anonymity of linkability. For a more detailed discussion on the topics of unlinkability and unobservability the interested reader is encouraged to read a concise

summary by Fishcer-Hübner [37], as well as Pfitzmann and Hansen [77].

An attacker's ultimate goal is to attempt to distinguish between individuals in an anonymity set. He wishes to do this by associating a message with an individual. He can associate either a recipient with a message or a sender with a message, or both. Thus the anonymity set can be used as a baseline marker for determining the degree of anonymity that any particular PET offers.

Several methods have been proposed for determining the degree of anonymity that any one particular technology offers. These proposed methods assume either that there is some uniform distribution of probability over the anonymity set, or that the probability distribution is non-uniform.

Uniform distribution of the probability over the anonymity set is simply defined to be $1 - p$ [102, 30, 89], where p is the probability that a person in the set can be identified. Obviously as $p \rightarrow 0$, anonymity increases. However, as soon as p_i for subject i decreases, it must necessarily increase for $p_j, i \neq j$. The ideal is that $p_1 = p_2 = \ldots = p_n$, offering the highest degree of anonymity for the entire set of entities in the anonymity set.

A slightly more complex measure is presented: Statistically speaking, if A is the event that person a has sent a message, and B is the event that a message has been sent, then if $P(A|B) = P(A)$, then event A is not dependent on B, and perfect sender anonymity is achieved. If, however, $P(A|B)$ approaches 1, then A becomes more dependent on B, and no anonymity is provided.

Unfortunately, this quantification of the degree of protection offered by a PET represents the ideal case.

Non-uniform probability distribution over the anonymity set proposed by Diaz et al. [30] (and also independently by Serjantov and Danezis [89]), takes an information theoretic approach to determining the degree of anonymity provided by a system.

This calculation is based on the entropy inside a closed system representing a mode of attack that is focused completely on a single mode of communication. If the mode of communication changes, then the attack is no longer relevant and will have to be replayed.

The probability function $p_i = P(X = i)$ represents the probability that the event $X = i$, where $i \in S$. The entropy is then measured as

$$H(X) = -\sum_{i=1}^{N} p_i log_2(X) \tag{2.1}$$

The basic idea is that the drop in the level of entropy inside the anonymity set (and the assigned probabilities) is measured. If there is a large decline in

entropy then the attacker is starting to discover associations between individuals and messages. This approach is similar to that of an attacker attempting to guess the secret key used to encrypt a message. Once the number of guesses needed decreases, the probability of finding the correct key increases [33].

The following section deals with the concept of the type of privacy (introduced in the previous section) in more detail.

2.2.4.2 Anonymity, Pseudonymity, Unobservability, Unlinkability

The types of privacy are anonymity, pseudonymity, unobservability, and unlinkability.

Anonymity is that particular characteristic which states that a person remains anonymous when communicating with another party. It is also important to define what exactly what anonymity means. For the purposes of this text, anonymity is defined as follows:

Definition 1 (Anonymity) *Given a set of entities that could be responsible for an event (see section 2.2.4.1 (anonymity set)) that has taken place, anonymity is that aspect which renders the identity of the entity responsible for the event equally likely amongst all members of the set.*

Ideally the anonymity set is large, rendering an inference attack more difficult (an anonymity set with two members offers very little protection even if the probability that can be assigned to each member is equal).

In order to support anonymity, typically no information about the originator of messages or commands is stored. However, many PETs that attempt to provide anonymity in the realm of communications need to allow reverse communications (responses to messages). In these cases the detail of the originator would be stored in a reverse lookup table (see anonymous remailers: section 2.3.2).

In actual fact, there are several types of pure anonymity [7]:

- Sender anonymity
- Recipient anonymity
- Sender/recipient anonymity

The names of these types of pure anonymity are self-explanatory, and will not be discussed further.

[7] As opposed to anonymity of linkability and anonymity of traceability.

Pseudonymity Whenever one has a need to store information on a particular individual (in cases where someone may have to be held responsible for their actions) anonymity is protected by providing pseudonyms. The first remailers took this approach to privacy.

- Initial private pseudonyms
- Transactional pseudonyms

Transactional pseudonyms are considered to be the stronger of pseudonyms, as the association between the individual and the pseudonym changes per transaction. This makes it more difficult for the attacker to build a profile for an individual that can be used to track the individual.

Unobservability It should not be possible for an attacker to observe that a particular individual is communicating with another individual.

Unlinkability An attacker should not be able to link two individuals as communicating with each other.

The degree of privacy that each of these offers has also been studied outside of the realm of the anonymity set. In particular the approaches to presenting these concepts in a mathematical way all rely on probability theory, most of which deals with the entropy in communication systems as investigated by Shannon [90, 31, 26].

Statistically speaking, defining the degree of privacy requires an observation (B) by an adversary, and an event that takes place E. This event can be $E \in \{\text{A message is sent by person a, a message is received by person a}\}$. The adversary attempts to determine $P(E|B)$. In all cases, $0 < P(E|B) < 1$.

As a special case, unlinkability requires E to be some event that is common to two observed events that the attacker wishes to link together.

Whereas the attacker wants $P(E|B) \rightarrow 1$, the PET attempts to provide protection by ensuring that $P(E|B) = P(E)$, providing perfect anonymity, unlinkability, unobservability, or pseudonymity.

In the following section, PII is discussed.

2.2.4.3 Personal Identifiable Information

In the above cases, the typical concern is with hiding identity (or action) when an individual acts in some way. Generally, this type of protection is sufficient since information that was gathered during the action (by the trusted system) can easily be discarded after the transactions have taken place; thus: “do not store that which

is not needed". For example, if it is not necessary to store e-mail addresses, then don't (OECD guidelines [70]).

Consider, however, a situation in which discarding information is not as easy. There are a wide variety of online stores and other vendors which do store a plethora of information about individuals with whom they transact. However well they keep in line with the OECD guidelines, they will still have to store certain pieces of information (even if only temporarily).

Information that is stored in this way and that can be linked to an individual (telephone numbers, ID numbers, credit card numbers, etc.) is referred to as PII [88, 37].

The protection of PII receives a significant amount of attention not only in the privacy field, but also in the security field [75], and also (as has been mentioned) forms the pivot of the research in this work. Protecting PII protects the privacy not only of those individuals who make use of technologies such as the internet, but also those individuals about whom some data is stored in a database somewhere by virtue of them having filled in a competition form, or having applied for credit at some financial institution.

In the following section, the idea of placing the purposes for which information is stored in hierarchies is considered more closely.

2.2.4.4 Hierarchies

A natural way to think about the many purposes that one may have regarding data is to consider these purposes in hierarchies. One may express a desire to have an e-mail address in order to contact an individual for the purposes of market research. However, one may also be more specific and state that one would like to store the e-mail address of an individual in order to do market research relating to after-sales service. Market research relating to after-sales research is much more specific than a general notion of market research, but it is still market research.

Using hierarchies allows the organisation to provide a comprehensive list of ever-increasing specific reasons why data is stored, making it easier for individuals to determine what the root of a particular purpose regarding data could be.

The following section provides a brief note on the auditing aspects of PETs.

2.2.4.5 Auditing

In order to determine if a PET has honoured the promises it published in a privacy policy, it should be possible to ask questions regarding actions taken with information to determine if there are any discrepancies.

An auditing mechanism can be limited to storing the collection of access requests, and associated results which can be manually investigated to determine

if an access request violated some published privacy policy. On the other hand, it can perform complex analysis of the access requests, and automatically determine if the access request resulted in the violation of a published privacy policy. Auditing falls outside of the scope of this document.

In the following section, the first classification of PETs presented in this text is described in more detail.

2.3 Protecting What We Do

A technology that protects what we do is an attempt at protecting an individual's actions from prying eyes by prohibiting other individuals from linking a certain communication action to an individual. That is, it attempts to allow the individual to interact with other entities, and will not allow another entity to observe the exact nature of the interactions – effectively anonymising actions.

From the definition of anonymity, it is painfully clear that it may be possible to determine that a person is communicating. However, it may be impossible to determine the nature of the communication, such as the content of the message, or with whom that person is communicating.

The technologies discussed in the following sections all attempt to protect privacy in this way. Certainly none of them provide complete privacy, but most of them provide at least some form of anonymity, unlinkability, untraceability, and unobservability.

The following paragraphs purposefully avoid discussions on the means that have been proposed to attack the presented PETs, as this would be beyond the focus area of the work in this document. Moreover, such a discussion would easily fill as many texts such as this one.

The following section puts forward the first concrete proposal that was made for a PET. This PET is considered to be the starting point of privacy protection research [26].

2.3.1 Chaum's Mixes

In 1981, David Chaum [18] proposed a model for protecting the content of e-mail, as well as the anonymity of the person communicating by obfuscating the content of the mail, and sending the communication through a special service provider that would obfuscate the message (or parts of the message) in such a way that the origin or destination of the message could not readily be determined by an attacker.

Protecting the content of the e-mail is very simple, since a relatively short time before Chaum's proposal, the concept of public key cryptography was made

known. Chaum's idea was also to make use of public keys to encrypt e-mail messages, prohibiting attackers from reading the content, and thereby determining the identity of the individual sending the message (or the identity of the recipient, for that matter).

The simplest type of MIX is a single machine that acts as the MIX service provider [18, 38, 37, 28]. A more complex scheme consisting of more than one MIX (called a cascade), attempts to provide more anonymity by forming a chain of MIXes which would make it more difficult for an adversary to determine if the incoming message came from the "true" sender, or if the outgoing message is going to the "true" receiver.

A single mix scenario entails that the sender adds a random string of bits to the original message, and encrypts the result with the public key of the intended recipient. Another sequence of random bits is generated and added to the previously calculated result. This result is then encrypted using the public key of the MIX, and the message is sent to the MIX.

The MIX decrypts the message, and forwards it to the intended recipient, which decrypts the message and can respond to it.

Allowing responses to originating messages requires that the return address should also be provided. The originator of the message will typically encrypt his address and pass that on to the recipient as part of the message. The recipient then responds to the MIX, by passing the encrypted address back. The MIX decrypts the address and encrypts the message using the randomly prepended sequence of bits as a key. The originator then decrypts the message using the private portion of the random sequence of bits.

This sequence of actions inherently means that the MIX must not be a compromised system. An extended scenario for the MIX notion is to have several of these MIXes. This provides a platform for the creation of a system that is even better suited to protecting privacy.

Cascades A cascade is a sequence of MIXes [102, 26, 37] which all exchange messages with one another in order to protect the anonymity of the message originator. The concept works the same as the single mix scenario, except that messages are now encrypted multiple times starting at the inner layer (encrypting for the recipient), and adding encryption layers for every MIX the message has to pass through.

This type of system enhances privacy in two ways. Firstly, since the message is encrypted multiple times, an adversary now needs to determine the keys of multiple recipients, if he manages to intercept the message. Secondly, even if the adversary manages to take control of one of the mixes, all he will be able to

determine is which computer sent him a message and to which other computer he is supposed to send a message. It is difficult to determine if the recipient is the intended recipient and if the sender is the originating sender.

Avoiding attacks on MIX systems is done by batching messages in order to avoid latency attacks [35], padding/splitting messages to ensure that no distinction can be made between messages by examining message length [28, 35, 64], and adding "cover-traffic" to the messages sent [59]. These messages often lack security features and are often vulnerable to other attacks [28]. A full discussion on the attacks and methods for countering them is beyond the focus of this text, however.

The following section presents first, second and third generation implementations of the MIX idea. These implementations are known as remailers since they will *remail* an e-mail, and provide privacy in the same fashion as MIX.

2.3.2 Remailers

The beginnings of PET implementations were all rooted in the most common form of communication then available on the internet - the element that made it popular: e-mail. The ideal of hiding one's identity when communicating with someone else is akin to leaving an anonymous note addressed to someone, and arguably this is in many cases a very desirable action.

There are four generations of anonymous mailers, which will be discussed very briefly in the following sections.

Type-0 The type-0 remailer is a simple implementation of the MIX, but it only supports a subset of the features of a MIX as proposed by Chaum [37]. Individuals that wish to send e-mail anonymously use a single mail server provided by the type-0 remailer. The remailer strips PII from the incoming message, rewrites the message envelope, and sends the mail to the recipient.

In order to facilitate reverse emails, that is, allow recipients to respond to received e-mails, an associative array, stored on the remailer itself, is utilised. Any response to an anonymous e-mail would be routed by examining the associative array to determine the original sender of the mail (perhaps by looking at the original "from" data in the e-mail). This from data will most likely be a pseudonym. Unfortunately, this reverse lookup functionality means that the originator of a message can still be traced in the event of intervention by an arm of the law. This, of course, is not always a bad situation, since allowing law enforcement agencies the ability to determine the origin of certain e-mails may save lives, and thus it has the benefit of preserving accountability.

The first implementation of this type of remailer was the well-known *penet.fi*

hosted by Julf Helsingius [26, 37, 35]. Unfortunately, at the time when Helsingius provided the service, the *establishment* frowned on privacy of this fashion, and an accusation of wrongdoing by one of the users of Helsingius's service resulted in the service being closed down [26, 40].

Cypherpunk Type-I remailers or Cypherpunk [102, 40, 56] [8] remailers solved several shortcomings of the Type-0 remailer.

It allowed remailing, supported encrypted e-mails from the source, and avoided storing any form of mapping from sender to recipient.

MixMaster Type-II remailers (of which MixMaster [37, 24, 27] is the archetypal example) improve even further upon the Type-I remailers (and are in essence closer to the Chaumian design of a mix [67]). They always support chains of mixes, they make use of fixed length messages (padding messages to achieve a constant length for all messages), support reordering of messages, and also employ techniques to protect against replay attacks.

MixMaster also provides background noise (randomly generated messages to further obfuscate traffic).

MixMinion MixMinion [27] is effectively a Type-III remailer. The major contribution from MixMinion (over a Type-II remailer) is that it allows for bidirectional anonymous communication – recipients can reply to messages received anonymously in an anonymous fashion.

The following section contains an enhancement of the remailer and MIX implementations, although it is generalised to any type of internet communication.

2.3.3 Onion Routing and Crowds

Onion Routing Onion Routing [82, 15, 41] relies on the methods introduced by the concept of MIXes to encrypt a message in layered format. As the message passes through the chain of participating servers, layers are peeled off, until the recipient finally peels off the final layer.

This implies that routes are calculated beforehand (in order to allow each layer of the onion to be created). Each onion layer adds a layer of encryption to the message that is sent to the destination.

A standard Public Key Infrastructure (PKI) scheme can be used to accomplish the encryption of layers.

[8] The term Cypherpunk is rooted in social bedrock: the remailer was conceived and implemented by the *cypherpunk* discussion group.

The Onion Router (TOR) TOR [32, 61] is designed specifically with low level latency networks such as web browsing in mind. It offers the individual the opportunity to browse these networks anonymously. The use of TOR is designed to be extremely easy – it is used in conjunction with a back-end proxy server, which in essence allows one to point any service at the proxy server, thus having the potential to render any service anonymous.

The TOR project provides an easy-to-install application which any user should be able to install and use on their Personal Computer (PC). It has a near zero-button interface (save for starting TOR and changing the web browser's setup to use TOR – which can be accomplished with the click of a button).

Recent enhancements to TOR [62] reduces the bandwidth cost of relay selection in the TOR network which allows low-bandwith clients to join the TOR network.

Crowds In Crowds [83] and their derivatives the central concept is that of a *Jondo*. A Jondo[9] is basically any entity that wants to be part of the anonymising network. Whenever an entity wants to send a message over the anonymising network, the message is passed to either one of two candidates. It can be passed to the intended recipient, or it can be passed to another Jondo in the crowd. The selection of the direct recipient *versus* the intended recipient is determined by generating a random number and using this number to determine whether or not to forward the message to the intended recipient or to forward it to another Jondo.

Each Jondo goes through this process. The advantage of crowd-based routing is that no pre-set path is specified. Thus there is even less chance of the adversary discovering the identity of the sender since the route of each message from the sender is likely to change from the previous message – the attacker cannot use previously discovered information to aid in future investigation.

2.3.4 Flocks

Flocks is a technological solution similar to TOR [73, 74]. It allows anonymous web browsing, but it can be utilised for more than this. It uses existing proxy servers to act as Jondos. This means that very little development is necessary, and depending on the load of the flock, one can reap the benefit of caching employed by the proxies. Another important benefit is that very little effort is needed to provide anonymity for other protocols (File Transfer Protocol (FTP), Secure File Transfer Protocol (SFTP), etc.) since these protocols are typically supported by proxies without any customisation of the software required.

[9] From "John Doe"

2.3.5 Blender

Anonymity is not only a desire in the traditional client/server model of networking. Peer-to-peer networks such as BitTorrent [23], which provides scalable delivery of content, may also benefit. Indeed, some protocol extensions already exist for protecting the privacy of those making use of the peer-to-peer network in the form of BitBlender [7]. BitBlender is in essence an onion router, routing data between peers in the anonymity network.

2.3.6 Dining Cryptographers

For sake of completeness, the dining cryptographers [19] are also mentioned briefly. The proposal by Chaum provides a proved way of unlinking actions from individuals by offering disjunctive views of the actions, and loss of information when comparing results.

Ultimately though, this solution appears to be impractical for large-scale systems because of the large number of keys that have be shared, as well as the requirement for reliable broadcast media [102].

The Herbivore system [39] allows anonymous browsing and internet services – it successfully makes use of the dining cryptographers' principles [52, 35], with some modifications: users are placed in (dynamically) managed hierarchies called *cliques*, which are star topologies, and are in turn connected to larger star topologies, allowing different *cliques* to communicate with each other. The topologies and hierarchies alleviate the high bandwidth requirement imposed by the standard dining cryptographers network [52].

In the following section, accountability in the face of anonymity is considered in some detail. It is only considered from a "what we do" perspective, and is important since users may not always exhibit the behaviour that their anonymous status privilege entitles them to.

2.3.7 Accountability in the Face of Anonymity

Ensuring that the privacy rights of an individual are protected has to be juxtaposed against the rights of individuals who would seek to abuse their anonymous status [46]. To counteract abuse it is necessary to be able to *ban*, or *blacklist* a user, ideally without a violation of their right to privacy. There are several proposed solutions to the problem of ensuring anonymity and still providing accountability, but a complete discussion of each will not be provided. The highlights of the well-known ones are given.

Unlinkable Serial Transactions (UST) by Syverson et al. [92] was a first attempt at banning a misbehaving user after having discovered that they have been

misbehaving. The problem with this approach is that miscreants can only be banned after having been pointed out as behaving badly (once their interaction with a service provider ends) and not during their interaction [46].

To counter this behaviour, and also to provide continued privacy enjoyment, *pseudonymous servers* were proposed. These are in fact a progression of proposed techniques to allow banning of anonymous users from an online community such as WikiPedia[10] [96, 46].

Pseudonymous servers [50, 20] allow users to register on the server, and obtain a pseudonym. This pseudonym is then used to log into other websites and services that have partnered with the pseudonym server. Users that misbehave can be blacklisted by means of their pseudonym, avoiding revealing the identity of the misbehaving individual (protecting their right to privacy). Unfortunately, this assignment of pseudonyms does not provide a significant degree of anonymity, since it is always possible to link the pseudonym to the registered user (see section 2.3.2 on the Penet remailer).

To solve (amongst others) the problem of *back-linking* (mentioned above), Tsang et al. [96] proposed a system called *Nymble*, which implements *transactional pseudonyms* discussed in section 2.2.4.2 – Nymble is discussed in the following section, however, it is worth noting that the same authors of the *Nymble* system also proposed two other, seemingly similar, systems for blacklisting misbehaving users called *Blacklistable Anonymous Communications (BLAC)*, and *Privacy-Enhanced Revocation with Efficient Authentication (PEREA)*.

Both of these suffer a lack of linking to a unique resource, and both suffer from scalability problems [46]. Since they are variations of the Nymble theme, they are not considered further, and the interested reader is referred to Tsang et al. [94, 95] for more information.

Nymble Nymble consists of a Pseudonym Manager (PM) and a Nymble manager (NM). The user requests a pseudonym through the PM (this pseudonym is generated based on the user demonstrating that he controls some resource that will ostensibly be difficult to falsify). The pseudonym is provided for a certain *linkability window* – once a day, or once an hour, for example.

The user then contacts the NM through an anonymising network, which will assign an *ordered* collection of *nyms* (or tickets) to the user based on the server he wishes to contact. At this point the PM knows only the resource/pseudonym pair, and the NM only knows the pseudonym/website pair.

A service provider may block a misbehaving user by presenting the nymble associated with the misbehaving user, and receiving a *seed*. The seed allows the

[10] http://www.wikipedia.com

server to notify the NM that all nymbles associated with the provided seed are blacklisted.

Users can check their blacklist status by connecting to the NM server before connecting to the server from which they were blacklisted. In this way, the users do not reveal a new nymble to the server from which they are blacklisted, preventing the server from linking them to their previous misbehaviour. Moreover, past connections and future connections from the same user will not be linkable to the current blacklist, providing better anonymity than pseudonym servers providing a pseudonym.

Unfortunately, the Nymble system does allow back-linking if the PM and NM collude, de-anonymising the user's actions completely [46].

Henry et al. [46] proposed enhancements of the Nymble system to make back-linking difficult through the use of a credential manager which replaces the PM, and issues credentials that are unable to be recognised at a later stage. This ensures that back-linking becomes difficult.

The following section deals with PETs from the second broad category as introduced by this document, that of protecting who we are.

2.4 Protecting Who we Are

In many cases it is imperative that organisations store information regarding their clients [21, 44]. This may be required purely as a way to conduct business in a legitimate fashion, or as required by law. In South Africa, two pieces of legislation (FICA [36] and the Regulation of Interception of Communications and Provision of Communication-related Information Act (RICA) [84]) force institutions to verify the identity of customers/clients. An audit should be able to reveal that the verification took place. Thus the organisation is forced to store information on the customer/client.

Protecting who we are thus aims at limiting access to information for legitimate purposes only. A gross oversimplification is made that the reason specified is a legitimate reason – the fact that the organisation may lie as to what constitutes a legitimate reason is ignored.

The following sections offer some principle examples of systems that protect who we are.

2.4.1 Task-based Privacy Protection

Fischer-Hübner [37, 38] proposed a task-based privacy model which aims at ensuring that a data user may only get access to data if they are authorised to perform

a certain task, and then only if the task to be performed requires the data in question to be completed. Additionally, the data may only be accessed if the purpose of the task corresponds to the purpose of the task for which the data was collected.

2.4.2 Hippocratic Database

The tenet of the Hippocratic Database (HDB) [4] (after the oath of Hippocrates taken by medical doctors) is to limit access to information based on certain "promises" made by the database. These promises are stored as part of the database schema and are used during information retrieval.

These promises made by the HDB make use of purposes in order to accomplish privacy protection. A promise consists of a purpose for which the data is used (thus purpose binding), the retention period for the data, external recipients, and data users that may request access to the information. Access to information is granted only if the data user is allowed access (based on the access control for the object), and if their profile contains purposes that match the purposes bound to the data.

Based on its design, the reasons for accessing data that are to be provided are not hierarchical, they are strings that are stored in the database schema. Moreover, they are passively supplied. That is, a set of reasons are to be supplied as part of a data user's profile, and those reasons are used when determining whether access to information can be granted.

In chapter 5 it is argued that requesting access to information should entail active participation by the data user, and an explanation is given of exactly how users can be enabled to state their intent with data.

Research around the HDB includes not only the implementation of a database schema that allows the binding of purposes to data, and so on, but also the enforcement of these rules (purpose limitation), as well as auditing compliance within the HDB.

Purpose limitation is readily accomplished by rewriting queries based on the purposes bound to the data. The RDBMS is charged with normal access control to the data itself (using the standard, GRANT/DENY model [42]).

Auditing compliance allows the verification that the HDB did in fact honour the statements it made about privacy [3]. It uses the RDBMS log to analyse the queries that were executed, and determines the data that was released by the HDB. This analysis not only seeks out intentional violations of the privacy safeguards, but also situations in which data was unintentionally released (a mistake from the controlling mechanism).

Karjoth et al. [51] extend this idea of an active privacy system proposed in the HDB through the proposal of the E-P3P, which would allow the privacy policy to

be applied by the HDB without having to manually decorate every data element in the schema with a purpose.

2.4.3 Privacy Enforcement with HP Select Access

Casassa Mont et al. [65, 66] provide a proof of concept of a privacy aware system for privacy governance. This proof of concept is constructed using a product series delivered by Hewlett-Packard (HP). Their model consists of:

1. Explicitly modelling personal data which allows the implementor of the privacy policy to describe the data (location, type, etc.).
2. Writing privacy policies, which will allow policy implementors to create a policy, and so describe policy constraints and conditions – including data transformation and filtering.
3. An authorisation framework, which will make access control decisions based on the privacy policy.
4. A mechanism that will intercept access requests and enforce the privacy policy.

Data users can either decorate their access request to data, or rely on their default profile settings when requesting access to data. Depending on the policy that is bound to data, the data is transformed in an "appropriate way" as is described by the constraints. The notion that a data owner can provide consent to allow or deny the data user access to the information is close to the idea of a privacy contract between the enterprise and the data owner. A model for privacy contracts is considered in more detail in the following section, and one relating specifically to compound purposes and reasons is discussed in chapter 6.

Typically, if a data user requests access to information, only that information which meets their intent is returned. These privacy policies are flexible since data subjects can use opt-in or opt-out privileges to indicate which statements of intent will result in access to data.

The access to information in this scenario is limited to examples where the data user presents a single notion of intent. That is, the data user will typically state the data they need access to, and provide a single statement of intent (if at all). This statement of intent is then examined by the Policy Decision Point (PDP), which may modify the original request statement in order to limit access to data that matches the statement of intent.

An important note is that the statement of intent is once again (based on the examples provided) limited to a single purpose. In other words, it governs the whole request for access through the use of a single purpose.

2.4.4 Privacy Contracts

Oberholzer and Olivier [69] suggest the idea of a privacy contract that is a more formal agreement between the data owner and the data controller. They maintain that a privacy contract may consist of one of four categories of agreements between the organisation and the data subject.

A very important issue to understand is the idea that an organisation may not be able to conduct business if it does not collect certain pieces of information about a data subject. In the Oberholzer model a privacy contract consists of mandatory agreements. These agreements must be acceded to by the data subject – the organisation needs this information in order to conduct business. Additionally, the contract also has a component of optional agreements. Agreements allow the actualisation of transactions between the organisation and the individual. Transactions are defined in terms of actions (logically) and specific purposes that indicate the reasons for storing or accessing the individual's data.

They further expand upon the idea of mandatory and optional privacy agreements by providing four categories that cater for the different needs that individuals may have regarding their PII. These categories (levels) describe the types of transactions that may occur between data subjects and data users.

Level 0 defines mandatory transactions. Level 1 transactions allow the data subject to opt in to some optional transactions. These transactions are part of a "set menu" and the data subject either consents to them or does not. Level 2 allows the privacy pragmatist the opportunity to specify some additional purposes for which their data can be used in optional transactions defined at level 2. Level 3 allows the privacy fundamentalist the opportunity to determine which data elements can be used in the optional transactions consented to at level 2.

Privacy contracts are a powerful way to allow data subjects to control how their information is used.

2.4.5 Privacy Protection for Advanced Data Management Systems Using RBAC

Byun et al. argue [14] that there is a need for purpose-based access control in more complex data models such as XML and Object Orientation (OO) – an extension to work they did previously. This text does not consider their previous work as the later model provides some more interesting results. The first of these is that they identified a way in which to determine how a data user's intent with information is specified during an access request. The second is that they provide a notion of prohibited purposes – purposes that may not be used in a statement of intent during an access request. And finally, they show how to model purpose-based access control for data stored in a hierarchy.

2.4.5.1 Purposes in the Statement of Intent

As has been stated before, an important aspect of controlling access to data using purposes is that an access request should be accompanied by a statement of intent. This statement of intent contains the list of purposes for which the data will be used. The work by Casassa Mont et al. has shown that it is useful to allow an access request to have an explicit way to express the statement of intent.

In the Byun model, the statement of intent is inextricably intertwined with the role that the data user assumes when accessing the information. This is expressed in the model by associating the statement of intent with roles in the Role-Based Access Control (RBAC) model. When a data user is assigned to a role, they will also receive a statement of intent. This statement is passed implicitly when an access request is presented by the data user.

2.4.5.2 Purposes and Prohibited Purposes

In the Byun model, data binding entails associating what is called Intended Purpose (IP) with the data. An IP consists of the purposes that may be used when requesting access to information (the Allowed Intended Purpose (AIP)), and the purposes that may not be used (the Prohibited Intended Purpose (PIP)), or more formally as:

Definition 2 (Intended Purpose) *IP = <AIP,PIP>*

2.4.5.3 Controlling Access to Data

A data user will request access to information, at which point the purposes associated with that role will be used to determine if access can be granted. The purposes associated with a role and consequently presented during an access request are referred to as the Allowed Purpose (AP). Access is only ever granted if the purposes in the AP are implied or included by the AIP, and not implied or included by the PIP. A purpose y is implied by a purpose x if in a hierarchy of purposes y is taken to be a descendant of x. A purpose x is included if, simply stated, it is present in the AIP or PIP. More formally stated,

Definition 3 (Byun Purpose-based Access Control) *Access in the Byun model is granted if and only if* $\forall x \in AP, x \notin_b PIP^x \in_b AIP$. $\notin_b$ *and* $\in_b$ *are defined to mean "not included or implied by an element in" and "included or implied by an element in".*

Take note that the principle of denials take precedence is followed: access is denied if the presented purpose is included or implied by an element in PIP.

2.4.5.4 Hierarchical Data Models

Data models that contain hierarchies of objects, such as topic maps, XML and OO data models, will typically allow meta-data of the parents to be inherited by the child objects. This also means the opportunity to allow child objects to inherit the access control rules of the parent objects. That is, implicitly, the access control policy that applies to a parent object is expected to be applied to the child object.

The Byun model handles this situation by allowing the definition of strong intended purposes and weak intended purposes. A data object is labelled with a set of intended purposes (both strong and weak), and so objects that inherit the properties of parents or of objects that describe their type are labelled with strong and weak intended purposes. A strong intended purpose means that the purpose cannot be in conflict with another purpose. The strong purpose will always be chosen if a conflict occurs. A weak intended purpose can be chosen either way.

2.4.6 Minimal Disclosure, and Delegation of Authorisation

Massacci et al. [60] argue that the tenet of the HDB is not enough to accomplish proper privacy-based access control. They propose that an additional three concepts are necessary in order to fulfil the requirement of protecting the privacy of the individual. These are hierarchies of purposes, limiting disclosure and delegation of authorisation. The following sections cover each of these concepts in more detail.

2.4.6.1 Hierarchies of Purposes

Hierarchies of purposes are Directed Acyclic Graphs (DAGs). The decomposition is an indication of how purposes relate to parent purposes, and if purposes are required or optional when attempting to fulfil the parent purpose. Ultimately, the parent purpose is the purpose that needs to be fulfilled.

Purposes are decomposed into AND and OR subpurposes. An AND subpurpose pair indicates that both sub-purposes are required in order to fulfil the parent purpose, and an OR subpurpose pair indicates that either of the purposes are enough to fulfil the parent purpose. In such a scheme a parent purpose might be “Send parcel” and the subpurposes may be “Send to postal address OR send to physical address”.

2.4.6.2 Limiting Disclosure

The idea of organising purposes into hierarchies also allows the PET to determine which information is required as a minimum in order to complete a particular task.

For example, it may be necessary to send a customer an invoice. For this action, the data user does not need any information other than, for example, the title and name of the customer, and the postal address to which to send the information.

Given the service that the data subject wishes to receive, and the purposes as decomposed into their AND and OR decompositions, it is possible to determine the minimum amount of information needed from the data subject in order to render the service. The benefit is that the data user gets the minimum authorisation (exactly what is necessary) in order to complete the task.

2.4.6.3 Delegation of Authorisation

Because purposes have been placed in a hierarchy, it is now possible to delegate the subtasks and goals to different entities in order to meet the parent goal (purposes) as specified by the purpose DAG.

In the following section, auditing is discussed briefly.

2.5 Current Technologies Implementing PETs

Many of the topics discussed thus far focused largely on the typical "enterprise" selling some product, or service. This enterprise would require some subset of personal information about a user in order to conduct business. However, there are service providers whose business model differs from this classical view.

These enterprises follow a business model of introduction. People are invited to make use of their services (which will ostensibly allow them to keep in contact with friends, family, and colleagues); and at the same time enterprises that do have some service to sell will be able to advertise their products (services). Services such as these are commonly referred to as *social networking*, and popular examples are *Facebook*[11], *LinkedIn*[12], *Twitter*[13] and *MySpace*[14].

It is easy to understand that it is imperative that a person's profile on these sites be a true reflection of the person. In order to verify the authenticity of these persons these service providers require PII. This PII may be verified by interrogating the person, but this very seldom happens.

These service providers have a very important responsibility toward data owners. Since they will typically attract a variegated type of privacy conscious person, their privacy mechanisms, by default, must be aimed at absolute protection unless the person stipulates otherwise.

[11] `http://www.facebook.com`
[12] `http://www.linkedin.com`
[13] `http://www.twitter.com`
[14] `http://www.myspace.com`

There are also many examples of work which attempts to infer information about a person on one of these networks [58, 57, 103], and also attempts to provide anonymisation of the "links" between entities in social networks. The problem of preserving anonymity in social networking is not applicable to the work presented in this text, though; preserving anonymity in a social (public) database requires protection in the face of regular users of the social network exploring content. The focus of this work is on non-social (non-public) databases. That is not to say that the techniques presented here are not applicable.

The course of discussion for PETs has covered the relevant topics for the proposals set forth in the work, and concluding remarks are provided in the following section.

2.6 Summary

In this chapter all the research regarding privacy protection that is relevant to the work in the rest of this text was introduced. The terminology and some of the more fundamental research on the topic of protecting privacy were given. Several categorisations for PETs were discussed, as well as definitions for these categorisations. The discussion also introduced two new broad categorisations of PETs, namely what we do protection, and who we are protection.

What we do privacy protection primarily protects the individual's interactions with other people or services where the protection of identity and privacy is necessary. Examples of such technologies are rooted in the concept of MIX as presented by Chaum. Typically, the individual's interactions with other individuals or service providers are protected by obfuscating the individual packets that make up communication sessions. Some more examples of these types of PETs are anonymous remailers and onion routers.

Who we are protection deals exclusively with situations in which an organisation collects information on the individual in order to provide a service. In these cases it is important that the organisation act responsibly with the individual's data. Thus access to such information should be protected. The primary way in which PETs protect who we are is by binding purposes to the data that is stored, and then only granting access to that information once the correct purposes for accessing the information are provided. The primary focus of the rest of this text is based on protecting who we are information by providing a model for more complex usage of purposes and reasons.

In the following chapter, a detailed introduction to the principles behind compound purposes and compound reasons is provided.

Chapter 3

Compound Purposes and Reasons

May we together become more than the sum of both of us.
– Surak of Vulcan

In the previous chapter the tenets of PETs were introduced, and the goals of such technologies in attempting to solve the privacy problem were outlined.

In this chapter the concept of **compounds** in a privacy context is introduced. Compound purposes provide a strong and flexible way of representing privacy policy implementations by allowing ad hoc creation of complex purposes for storing information. This is done by using existing purposes that the data controller has already defined. Compound reasons are used by the data user as a *statement of intent* and are used during access verification to determine why the data user wishes to use the data they are requesting access to. They are also used to create audit logs, which allow for non-repudiation from the data user's perspective.

The concepts behind compounds are discussed by first considering the representation of purposes and reasons, and then providing a structure that forms the foundation for creating compound purposes and reasons. The existence of this structure is central to the concepts of compound purposes and reasons as it allows reasoning about the *relative suitability* of reasons that are used during the limitation phase and the purposes used during the binding phase. It also eases the amount of work necessary in order to manage purposes and reasons.

Throughout the chapter, definitions of the exact semantics of compound purposes and reasons are also given. These definitions provide a common understanding behind what the data owner, the data user and the data controller may agree upon, and consequently lay the foundation for the discussion of privacy agreements in chapter 6. These agreements are important for various reasons: they allow the data controller to engender trust, they provide a definitive understanding for the data owner regarding what their data will be used for, they provide a means for the data user to clearly state what they intend to do with the data, and

also allow the actions taken with data to be audited.

The layout of the chapter is as follows: Section 3.1 provides a brief introduction to the ideas behind compound purposes. A distinction is made between reasons and purposes in 3.2, and the representation of purposes during an access request is discussed in section 3.3. In section 3.4 the placement of purposes in a structure that will ultimately support compound purposes and reasons is explained. Section 3.5 provides a gentle introduction to the idea of compound purposes.

3.1 Preliminaries

Before embarking on the discourse which will define purposes, compound purposes and their notations, some background that will clarify the reader's understanding of the chapter is provided.

During the course of this chapter the term *purpose* is used to denote the intended use of a datum as specified by the *data controller*. The data controller may use a myriad of ways to bind purposes to data (a Graphical User Interface (GUI), or text-based configuration file, and so forth). However, for the moment the concern of this chapter is not with such a mechanism, which will be considered in again chapter 6.

The term *reason* is used to denote the intent of the *data user* (which may also be a principal, that is, something requesting access to information on behalf of the data user). The data user will (either implicitly or explicitly) present their intent with the data they are requesting access to during an access request. In chapter 5 it is argued that data users should explicitly state their intent with data, and not be allowed access to PII simply because they have the correct entries in their user profile. In the simplest case, one may think of a reason and a purpose to be the same thing. However, this need not be the case, and this is explored further in section 3.2.

Throughout this text a "shorthand notation" is used to avoid long textual representations. A purpose, as may be used by the data controller, is written as ϕ_n, and a reason, as may be presented by a data user, is written as ρ_m. A typical data controller will have a set of purposes that are specific to the domain that they operate within (such as the financial domain, engineering domain, etc.). These purposes may be defined by some governing body, or an association of interested partners. Defining an ontology [43] can also be a way to standardise the use of purposes and reasons. However, the creation of an ontology for purposes and reasons clearly falls outside the scope of this text, and is not discussed further. For ease of reference, throughout this text, this set of purposes that the data controller has defined is referred to as the Domain Purpose Set (DPS).

Whenever a reference is made to *purpose* it is taken to mean a *singular* or *atomic* purpose. Recall from chapter 1 that an atomic purpose or reason is a purpose or reason that represents a singular purpose for storing or accessing information. A simple example of such a purpose may be "to personalise browsing experience". When a reference has to be made to a *compound purpose*, it will be done explicitly, that is, as *compound purpose*.

In the following section the distinction between purposes and reasons (and implicitly also compound purposes and compound reasons) is discussed in greater detail.

3.2 Distinction between *Reasons* and *Purposes*

At this point it is necessary to first examine the distinction between reasons and purposes. A concrete understanding of the need for a distinction provides a good foundation when examining the mechanism that is employed during the verification stage of the limitation phase. The impact that such a distinction has on the overall employment of compounds can then be examined and understood.

From the literature in the previous chapter it is apparent that purposes are in general associated with a data user's profile, or the roles that are conferred on the user. Access to data is granted if the user's profile allows them access to the data as one would expect from a normal access control system (without privacy controls); additionally, access is only granted if the user's profile contains the right purposes.

The distinction that is made between purposes and reasons exists because it is important from an auditing and trust perspective that a data user express their intent with data (see chapter 5). An expression of intent makes it possible to provide a vast amount of flexibility during an access request. In systems that do not support RBAC, for example, one may specify a *role-based* reason for accessing information, such as "Because I am the Public Relations (PR) manager". Alternatively, one may wish to provide reasons that exist because of external influences, such as "Fraud investigation", or "Revenue audit".

In order to use reasons effectively and meaningfully, it is imperative to understand the relationship between a *reason* and a *purpose*. For verification during an access request to data to be accomplished correctly, the data user's statement of intent must be comparable to the (compound) purpose bound to the data. This requirement implies that the statement of intent (the reason) either consists of a (compound) purpose, or that it is possible to map the reason to some (compound) purpose. This text, as has been stated already, is not concerned with the creation of conversion protocols and that avenue of discussion is not considered further. To ease discussion, however, it is assumed that reasons and purposes are used

interchangeably, and that (compound) reasons are used during the access request phase.

Definition 4 (Reason) *A (compound) reason is expressed as a (compound) purpose, or as some other predefined (compound) reason.*

Considering a reason to be logically equivalent to a (compound) purpose also provides the opportunity to organise reasons into some structure which makes it possible to compare reasons to other reasons.

The existence of reasons also provides the ability to have truly ad hoc statements of intent such as “Because Alice said I could”, which (even though the example is quite simple) provide a means to allow access to data for explicit execution of duty. A reason may be, for example, “because I have been granted power of attorney”. In both of the above cases, the reason may be logically equivalent to some compound purpose. One may thus think of a compound reason being to a compound purpose what a view is to a table in an RDBMS.

The presence of reasons, which are defined in terms of purposes or other reasons, suggests that special care must be taken to ensure that the meaning behind the reason matches the purposes that are used to define it. So, it is important that “Because Alice said I could” be auditable, and carry the meaning clearly linked to the intent behind “Because Alice said I could”. For this reason, it is imperative that both the reason that is provided and the definition of that reason form part of the audit trail.

The following section explores the representation of purposes in more detail.

3.3 Purpose (Reason) Representation Properties

This section provides a brief introduction to the considerations behind the representation of purposes. The way in which purposes are represented is important because it allows people to communicate with systems about purposes, and it determines the way in which purposes will be used when performing limitation. It also allows people to talk to other people about purposes in the privacy context.

An enterprise’s privacy policy should ideally start off as a legal document from which purposes can be identified and extracted for use in a privacy context. Here it is important that purposes be uniquely identifiable in the document, and that no two purposes overlap. The extraction process will no doubt produce short phrases that represent the purposes, and these textual representations may be considered the baseline from which a machine enforceable privacy policy may be constructed. Methods have been proposed to allow the extraction of privacy requirements for role engineering [45], and this may be an efficient place to start the definition of the privacy policy.

As the privacy policy changes, some overlap may be introduced. The extraction of privacy requirements from the policy must be carefully managed to ensure that these overlaps are detected (a wholly manual process)– the model proposed in this text has no way of detecting these overlaps since it considers the purposes provided to it at a symbolic level (purposes are elements in sets). It may be possible to use ontologies to annotate the purposes, and allow the system to discover overlaps. The construction of ontologies falls outside of the scope of this text, and is mentioned as a future area of study in section 7.

Long textual phrases can become cumbersome, and no doubt, the System Security Officer (SSO) will rather employ mnemonic devices to keep the machine representation of purposes concise, yet clear. With some forethought, it is entirely possible that a standard set of mnemonics can be constructed, and shared between enterprises. Regular users may not want to make use of these mnemonic devices, and the system should allow them the ability to make use of the "long textual" phrases. The SSO can use the mnemonic representations on a low-level, and the regular users can use the textual representation which will be mapped to the mnemonic representations.

The most important aspect (put forward in the previous section) of using mnemonics is that there must be some mapping between the mnemonic and the textual representation, and that this mapping will not change during the lifetime of a particular purpose.

From the discussion above, as far as the representation of purposes goes, there are three important properties.

Definition 5 (Purpose Representation Properties)

1. *There must be a bijection between a privacy context purpose and a privacy policy document purpose.*

2. *There must be a bijection between the mnemonic representation of a purpose, and the textual phrase that is communicated between the parties agreeing on the privacy policy.*

3. *This mapping cannot change once it has been assigned.*

■

Throughout this text it is assumed that a finite set of purposes is used, that each one of these purposes is unique, and that they conform to definition 5. Simple mnemonics are used to represent purposes; with the understanding that even

though the link between a mnemonic and its meaning as understood by the enterprise using it may initially be of interest, as this text progresses to the verification of reasons against purposes, this link will no longer be as important – the focus shifts to the way in which a purpose, or reason is represented, and how verification is accomplished. For clarity in exposition the text represents purposes using the Greek symbols ϕ, and ρ (as well as their capital representation). For example, a purpose such as "Send e-mail to client" may be labelled as ϕ_1, while another purpose such as "Send billing information" may be labelled as ρ_2. Ultimately, the SSO may write a privacy policy using these labels, or a data user may state their intent in using them.

Even if an enterprise decides on a set of purposes that are different from another enterprise, it is possible that they may publish their purposes and the way in which to compare purposes, so that another enterprise may write a *protocol* converter between two different sets of DPSs. The protocol converter must be based on a mutual agreement between the two enterprises, and must also be audited to ensure that the purposes are mapped correctly (it will be relatively easy for enterprise A to enjoy access to sensitive information by simply creating one purpose and mapping that to the purpose used to protect the most sensitive information in enterprise B's database). The notion of such a protocol converter is not the subject of this text, and is not considered further.

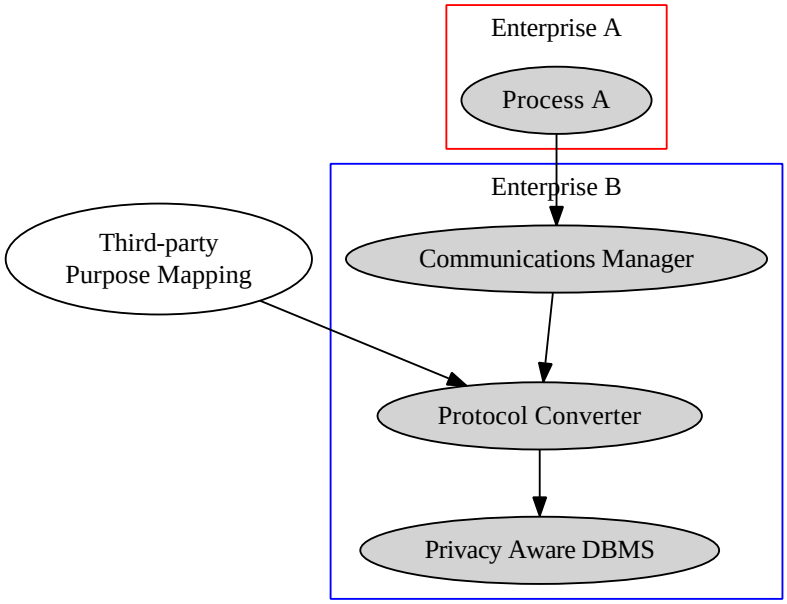

Figure 3.1: Implementing protocol converter between two enterprises

Having considered purposes and reasons, and how they may be represented by

an enterprise wishing to employ them, appropriate way in which to organise said purposes will also be considered in more detail in the following section.

3.4 A Structure for Purposes

In this section the most prominent and widely accepted structure that is used for purposes is examined. It is then argued that a *purpose lattice* is better suited to purposes than this structure. The reason for using a lattice also ties in strongly with the concept of compound purposes and reasons.

The benefits of organising purposes into a hierarchical structure have been expressed by many researchers [12, 88, 6]. The most common reason is that organising any type of token into a hierarchy reduces the number of tokens that have to be managed by, for example, the SSO, or Database Administrator (DBA) [81].

Inheritance is obviously a necessary design implementation of the system that manages these tokens. One example of tokens that inherit the properties of other tokens is that of RBAC [87]. Roles are conferred on users, and those roles inherit the properties of other roles in the hierarchy.

Notable similar proposals are EPAL [6], and Hyper-graphs [60]. The structure presented in this chapter will be juxtaposed to these structures in order to elucidate the differences (see following section).

It has already been indicated, earlier in this chapter and in work done elsewhere such as EPAL and XACML, for example, that the placement of purposes in a hierarchy provides a more effective use of purposes when constructing privacy policies and access requests.

In fact, placing purposes in the correct structure allows the definition of the relationships that may exist between purposes. These relationships are important, firstly, because they allow the data controller and data owner to come to a mutual understanding on the subject of the relative sensitivity of privacy related information, and secondly, because they provide a way to create complex new purposes during an access request or during the limitation phase, rather than having to expressly create these purposes and incorporate them into the structure used to store purposes.

The following section expands on the idea of the relationship between purposes.

3.4.1 The Relationship between Purposes

Suppose an enterprise stores (amongst other things) the e-mail addresses of clients. These addresses can, for example, be used to send out general news about the enterprise, billing information, catalogues about products on offer, and much more.

In a simple scenario, the enterprise will publish its privacy policy stating, hopefully in no uncertain terms, that it only uses e-mail addresses to send out catalogues. The HDB (section 2.4.2) principle states that a data user will thus only gain access to e-mail addresses if they have the right reasons as part of their user profile for doing so.

Suppose one customer might not be too concerned about receiving the odd commercial e-mail about special offers and other marketing material. This customer might even be fine with receiving their invoices per e-mail. If some of the purposes that were discussed previously are considered in more detail, then it becomes clear that the act of sending e-mail (and thus storing an e-mail address for the purposes of sending e-mail), and the act of sending a product catalogue by e-mail, or even sending billing information by e-mail are related. In fact, "sending e-mail" can be thought of as a much more general purpose for storing e-mail addresses, while the other purposes are a bit more specific.

Using a hierarchy, it now becomes easy to define a complete set of purposes in this manner. It is easy to see that such a hierarchy offers a rich mechanism for enabling privacy. Purposes in hierarchies elsewhere have been defined as a simple way to manage purposes.

Only two systems, at the time of writing, consider a special relationship between purposes that are placed in a hierarchy. EPAL uses the hierarchy to define the concept of completeness. Purposes that are in child nodes of a purpose should all be accumulated before the actual parent purpose can be used. Purposes that form part of the RBAC, by Byun et al. [14], use role inheritance to allow additional purposes to be inherited by a role.

In neither of these are purpose lattice used as a specific way of organising purposes, neither employs the idea of more specific and less specific purposes explicitly, and in neither are compound purposes used specifically.

Another comparable system is Hyper-graphs [60]. The Massacci proposal considers hierarchies of purposes as they relate to tasks. Children tasks can be considered conjunctions and disjunctions, but the concept of explicitly expressing compounds during information storage and access is never addressed.

3.4.1.1 Purpose Lattice

A small sample purpose lattice is provided in figure 3.2 to illustrate. A more general model of the intention with a purpose lattice is presented in figure 4.1.

Consider figure 3.2. Sending e-mail is a parent node in the hierarchy to the sending of billing information purpose and of catalogues. However, the sending of catalogues and of billing information are seemingly different purposes altogether (one may be considered a marketing action, whilst the other is part of the completion of a transaction). That is to say, no relationship exists between them, other

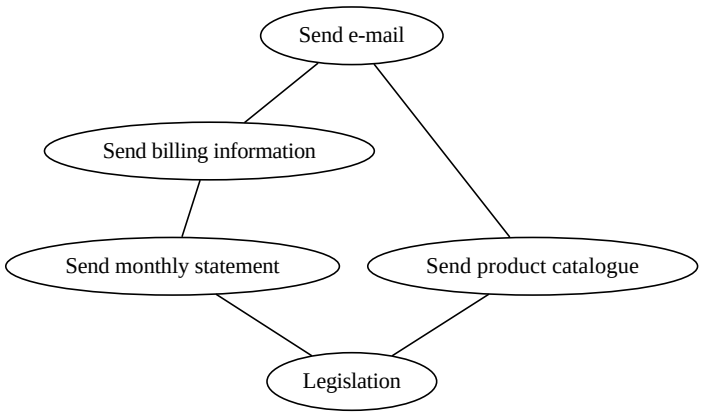

Figure 3.2: A simple purpose hierarchy

than sharing the same parent node.

Although adequate, the hierarchy of purposes presented here is not sufficient. Suppose, for instance, that a law enforcement agency requests access to data of all the clients stored in the database. Surely one purpose that is given, perhaps that of "compliance with a subpoena", should be enough to gain access to all stored information? Attempting to specify each *strongest* purpose for each branch of the hierarchy can be quite cumbersome for a hierarchy sporting hundreds of entries. Moreover, the actual purpose for using the data will not be presented to gain access to the data, since the data user has to artificially present every possible strongest reason in the hierarchy to get access to information. If one were to perform an audit on the access to this data, then clearly there would be a discrepancy.

Quite intuitively then, one would correctly guess that if every leaf purpose were the same strongest purpose, the auditing discrepancy would not be a problem.

Purposes placed in a hierarchical structure (the lattice) can now be defined more formally.

Definition 6 (Purpose Lattice) *A purpose lattice is a bounded lattice with elements representing singleton purposes from the enterprise's domain purpose set.*

A strict partial ordering exists between the elements of the purpose lattice, thus enabling the access control subsystem of a RDBMS to determine the relative suitability of a purpose in relation to other purposes.

More formally, a purpose lattice PL is defined as $(A, \leq)$, where A is the domain purpose set of the enterprise, and $\leq$ defines the ordering of the elements in the lattice. Elements in PL are ordered pairs with (x, y) indicating that $x \leq y$.

PL is also bounded, meaning that there is an element called 0 in A which is considered "weaker" than all the other elements in A. Thus, $\exists a \in A$ such that

$(a, y) \in PL, \forall y \in A$.

In the same fashion there is also an element called 1 *which is considered "more specific/stronger" than all the elements from* A*. Thus,* $\exists z \in A$ *such that* $(y, z) \in PL, \forall y \in A$

By definition (of elements in lattices in general), note that $i \leq i$*, or* $(i, i) \in A$*, for any* i *in the lattice.*

■

The ordering of purposes in the lattice allows reasoning about their relative suitability when provided as reasons for storing information or even as statements of intent when requesting access to information.

Having discussed purposes and the structure that will be used to present them, the key concept of this text, compound purposes and reasons, is now introduced.

3.5 Compound Purposes

A *compound purpose*, as the name suggests, is a purpose that is compounded from other purposes, that is, it is a purpose that is defined in terms of the conjunction or disjunction of other purposes; whereas a compound reason is a reason that is defined in terms of the conjunction or disjunction of other reasons, or purposes. Recall that a reason may simply be a purpose, a combination of purposes, or even a combination of other reasons.

This allows the combination of purposes that seem logically disconnected, but may be required as part of a particular business rule. For example, a particular purpose for storing an address may be for sending a parcel, or for sending a product catalogue. Although these purposes seem logically disconnected, it may be necessary for the enterprise to limit the sending of product catalogues to the event of sending a parcel to the recipient. This is an ideal situation for the privacy fundamentalist, who may only wish to receive communications from an enterprise when he engages in an interaction with them (buying an item from them).

Currently, most PETs enforce privacy by allowing the specification of several purposes during the binding phase. During the limitation phase there is no indication of which purposes should be mandatory during a request, or which should be optional. The EPAL, Byun, HDB, and other models only examine the data user's profile to determine if they have a subset of all the purposes bound to the datum.

Hyper-graphs do not provide an answer to the question of complex purposes either, as the hyper-graph is designed to indicate which *processes* should be mandatory or optional during an access request. Compound purposes strive to solve this problem by introducing a mechanism that allows the enterprise to cohesively state which purposes are mandatory for accessing information and which are not.

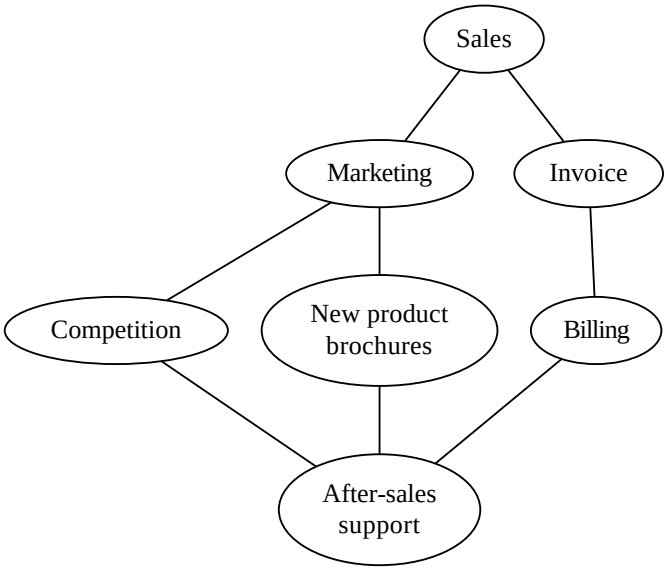

Figure 3.3: Example purpose lattice

Consider figure 3.3, which illustrates a sample purpose lattice. The reason for making use of purposes in the first place is to ensure that data is only released when the reason for using it is sufficient. Compound purposes depart from this idea in that they allow the data handler to state exactly which reasons *must* be provided when access is requested, which reasons may not be presented, and which reasons are optional during an access request. Tie this in with the relative suitability of purposes, and a model for an expressive privacy policy becomes possible. In particular access to an object Ω is only granted if the reason (purpose) supplied with the access request (Γ) dominates the purpose associated with the data (Φ), that is $\Phi \leq \Gamma$.

The reasons provided by the data user when requesting access to data can be encoded as part of their access token, or they can be supplied by the user as part of the query (see chapter 5 for a discussion of the advantages and methods of forcing data users to present their reasons for accessing information). In either case it becomes possible to associate a particular set of reasons (by using compounds) with a particular user.

Given the flexibility that compound purposes provide in this way, it becomes possible to reflect real-world scenarios quite closely, with little effort. One might define a purpose “because of a warrant” (for example an Anton Piller order [49]) and add it to the purpose lattice at the appropriate place in the lattice to allow access to data – purely because a warrant allows the data user access.

Because of the use of the lattice, one may also place the “because of a warrant”

purposes at the correct place in the lattice (in a just-in-time fashion), only allowing access to data that is requested by the warrant. Data protected for other reasons may still be safe in this case. It should be noted that the line between privacy protection and legislation is extremely fine [26].

Next the notation for defining compound purposes is introduced. It is important to note that the symbols are merely representative of the semantics behind the notion of compound purposes. As such they should accurately reflect what is meant by the compound, but they should not be confused with the *compound*. The reason for this will be made clear when it is shown how verification takes place using compound purposes (chapter 4).

There are three defined compounds:

1. *Or* compounds, which indicate that any of the purposes forming part of the compound *can* be used to gain access to information.

2. *And* compounds, which indicate that all the purposes forming part of the compound *must* be used to gain access to information.

3. Compounds with "and-not", which indicate that certain purposes *cannot* be used to gain access to information.

Also realise that these defined compounds can be used together, to create complex and rich expressive restrictions on the access to information.

In the succeeding sections these concepts just presented are discussed and *or compounds* are considered first.

3.5.1 *Or* Compounds

An *or* compound is defined by using the symbol $+$ as a binary operator between two or more (purpose) operands. In particular, a distinction is made between the semantics of the $+$ operator for purposes, and the $+$ operator for reasons. The former is indicated as $+_p$, while the latter is indicated as $+_r$.

Consider the purposes ϕ_1 and ϕ_4, with reasons $\rho_1 = \phi_1$ and $\rho_2 = \phi_4$. Now suppose a particular datum is stored for purpose $\phi_1 +_p \phi_4$. Semantically, this means that the data user can present either ρ_1 or ρ_4 as a reason for requesting access to information. Additionally, because of the hierarchical relationship between purposes and reasons, it also means that the data user can provide any reason that is "stronger/more specific" than ϕ_1, or any reason that is stronger than ϕ_4. It should be fairly obvious that the data user can also specify a reason that is stronger than both ϕ_1 and ϕ_4, and that they could alternatively specify a *compound reason*, and that, as long as it is stronger than either ϕ_1 or ϕ_4, it should be enough to allow access to information with bound purpose $\phi_1 +_p \phi_4$.

A data user might indicate that they want client information because they want to send parcels to some, and product catalogues to others. This is clearly a very loosely defined reason for wanting access to information. Suppose "Send parcel" is represented by mnemonic ρ_m, and "Send product catalogue" is represented by mnemonic ρ_n, then a statement of intent can be presented as $\rho_m +_r \rho_n$. Access control now happens on the presented reason; however, the "when" of granting access should be clearly understood. At this point it must be emphasised that a statement of intent has to be "stronger/more specific" than the (compound) purpose bound to the data, *in its entirety*, before access will be granted.

As a special point of interest, consider the case where the reason provided when requesting access to data is $\rho_m +_r \rho_n$, with the bound purpose as ρ_i. Access cannot be granted unless both ρ_m and ρ_n are stronger than ρ_i – otherwise access may be granted based on, say, ρ_n, when ρ_m is not suitable at all. This event is analogous to a SQL injection attack, in which the attacker will add a little tail to a query that is not properly checked (the well known "where 1=1").

In order to formally define what is understood under the suitability in terms of semantics of or compounds, the following axiom is given.

Axiom 1 Or *compound semantics*

1. $\phi_i +_p \phi_j \leq \phi_m \iff \phi_i \leq \phi_m \vee \phi_j \leq \phi_m$
   An or expression of purposes are only weaker than another purpose if at least one of the purposes in the or expression is weaker than the other purpose.

2. $\phi_i \leq \phi_n +_r \phi_m \iff \phi_i \leq \phi_n \wedge \phi_i \leq \phi_m$
   A purpose is only weaker than an or expression of purposes if it is weaker than both of the purposes in the expression.

3. $\phi_i +_p \phi_j \leq \phi_m +_r \phi_n \equiv \phi_i \leq \phi_m +_r \phi_n \vee \phi_j \leq \phi_m +_r \phi_n$ *(expanding using 1 and 2 above), if and only if* $\phi_i \leq \phi_m \wedge \phi_i \leq \phi_n \vee \phi_j \leq \phi_m \wedge \phi_i \leq \phi_n \vee (\phi_i \leq \phi_m \vee \phi_i \leq \phi_n) \wedge (\phi_j \leq \phi_m \vee \phi_j \leq \phi_n)$ *(also expanding using 1 and 2 above).*

■

The second binary operator (the and-compound operator) is discussed in the following section.

3.5.2 *And* compounds

An *and* compound (written as $\cdot_p$) demands that a data user's statement of intent dominate both the operands of the compound purpose bound to the datum. From the data user's perspective this indicates that they will use the data for a reason that satisfies both operands.

Each operand may, of course, be another compound purpose, but the syntax for more complex purposes will be defined later on. For the moment, the simple form of an *and* compound is considered. For instance, if a particular datum is stored for purpose $\phi_1 \cdot_p \phi_2$, then only reasons that are simultaneously stronger than $\phi_1 \cdot_p \phi_2$ can be given in order to be granted access to data. For example, an e-mail address may be stored for purchase notification and billing information. Thus, only when a data user wants to send a purchase notification and billing information will the e-mail address be provided.

The semantics of the *and* compound is now presented.

Axiom 2 And *compound semantics*

1. $\phi_i \leq \phi_n \cdot_r \phi_m \iff \phi_i \leq \phi_n \wedge \phi_i \leq \phi_m$
   A purpose is only weaker than an and expression if it is weaker than both purposes in the and expression.

2. $\phi_i \cdot_p \phi_j \leq \phi_n \iff \phi_i \leq \phi_n \wedge \phi_j \leq \phi_n$
   An and expression of purposes is only weaker than another purpose if both purposes in the expression is weaker than the other purpose.

3. $\phi_i \cdot_p \phi_j \leq \phi_n \cdot_r \phi_m \iff (\phi_i \leq \phi_n \vee \phi_i \leq \phi_m) \wedge (\phi_j \leq \phi_n \vee \phi_j \leq \phi_m)$
   (expanded using 1 and 2 above).

■

3.5.3 Mixing Compounds

The final point of attention is the question of the semantics of mixing compounds on either side of the relational operator. An axiom to aid in understanding the intent with mixed compounds is now given below. These are simply expansions of the expressions using axioms 1 and 2.

Axiom 3 *Mixed compounds*

1. $\phi_i +_p \phi_j \leq \phi_n \cdot_r \phi_m \equiv (\phi_i \leq \phi_n \cdot_r \phi_m) \vee (\phi_j \leq \phi_n \cdot_r \phi_m)$

2. $\phi_i \cdot_p \phi_j \leq \phi_n +_r \phi_m \equiv (\phi_i \cdot_p \phi_j \leq \phi_n \wedge \phi_i \cdot_p \phi_j \leq \phi_m)$

Some more comments on axiom 3.2 are in order. It is easy to assume that any one of the reasons provided may be sufficient to be granted access to the protected data. However, remember that the statement of intent is implying that the data **may** be used for either reason. To avoid the equivalent of a SQL injection attack [93], the verification algorithm must assume that the data will be used for both, and will therefore verify the access request based on this assumption.

The *and-not* operator is discussed in the following section.

3.5.4 *And-not* Compounds

In many cases it becomes necessary to explicitly state what information will *not* be used for. For example, suppose an organisation stores e-mail addresses for marketing purposes, and would like to indicate that whatever marketing it does, it will not use e-mail addresses for sending general marketing information. A negative purpose can be used to indicate the organisation's promise to not send general marketing information.

A negative purpose is an enhancement of the compound purpose concept, and a definition is therefore required.

A negative authorisation in the security context [17, 11] is any authorisation which explicitly denies a subject access to an object. The concept of a negative purpose shares this idea of a negative authorisation. It is a way of stating a purpose for which data will *not* be stored and used – thus meaning that a compound purpose can explicitly list the purposes that *cannot be presented to gain access to data*.

Negative authorisations are sometimes used in so-called open systems, that is, systems in which everything that is not denied is permissible. Negative purposes do not share this aspect with negative authorisations. It does not make sense to indicate that a client's e-mail address, for example, will not be stored for sending of junk mail, thereby allowing the use of the address for every other purpose that the enterprise may have for it. This type of approach is counter intuitive, and contrary to the principles regarding purpose limitation as outlined by the OECD.

The operator used for negative purposes is indicated by the symbol $\cdot\neg_p$ (read "and-not").

Using negative purposes allows the creation of powerful storage and use policies, for example it allows the creation of exclusion policies or "upper and lower bound" policies. Suppose a single purpose ϕ_b is bound to a datum. Suppose further that the creator of this policy wishes to exclude certain purposes that dominate ϕ_b of being used to access the datum (recall that purposes subsume each other). Using the *and-not* operator, the creator of the policy can clearly indicate which children purposes should be excluded. This means that the exclusion of a purpose x also excludes the children of x from being used.

This operator is beneficial for the following reasons:

- It allows the creator of a privacy policy to exclude children that may be common to two separate purposes. For example, suppose a datum has bound purpose ϕ_k. Furthermore suppose ϕ_l shares a common child with ϕ_k and that the policy creator wishes to indicate that the datum will in no way be used for any purposes relating to ϕ_l. This can be indicated by specifying $\phi_k \cdot \neg_p \phi_l$ as the compound purpose.

- In particular, it becomes possible to indicate that a datum will be used for one purpose, and one purpose only.

The creation of policies using this operator is beneficial in that it will allow an enterprise to restrict the purposes for a particular datum, thereby allowing a user to clearly see if the datum will be used for more general purposes, or for more specific purposes.

An important exception to the rule, however, is that the greatest lower bound should never be removed from the bound purposes of a datum. Since this specific purpose will be most likely be used to signify a "master" purpose for accessing data, it should remain a valid purpose for the data. For example, the greatest lower bound purpose might be "obligation to legislature"[1].

Discussion of the semantics of the *and-not* operator is left until 4.2.3.1, which follows the section on the other operators.

It is obvious that a compound purpose such as $\phi_1 \cdot \neg_p \phi_1$, basically stating the data is stored for ϕ_1 and not for ϕ_1, is nonsensical. Ideally the creation of this form of conflicting purpose should be stopped even before it is recorded as part of a privacy policy implementation. Even so, the definition of the *and-not* operator causes this type of conflicting purpose to be evaluated, and the result is the exclusion of ϕ_1 from being a legal purpose for accessing data Unfortunately, examining this type of purpose does incur the use of processor time.

This section leads to the extension of the initial indication of access control using compound purposes. Access to an object Ω is only granted if $\Phi \leq \Gamma$ and Γ does not contain purposes explicitly listed as negative purposes in Φ.

[1] Although this purpose is seemingly vague it demonstrates the principle that a datum should be accessible if the data controller is required to do so by law. The fact that access to data based on legal requirements should not be an implicit act is also acknowledged, as well as the fact that appropriate steps should lead up to the access of data. However, such a purpose in the lattice as a greatest lower bound clearly indicates that all data can be used in order to comply with legal responsibility.

3.6 Notation

Another important aspect of using compounds is that the correct method for expressing the compound is used. Axioms which clearly stipulate what an individual (enterprise) would mean when writing down a compound purpose expression have now been given. However, the text would be remiss if formal notational rules for writing down compound expressions, both reasons and purposes, were not given.

For clarity, the representation of compound expressions in a simple Backus-Naur notation is provided in figures 3.4 and 3.5. Note that the difference between the compound purpose expression and the compound reason expression is that the *and-not* operator cannot be used in a compound reason expression (see section 4.2.3.1 for an explanation). In this syntax definition, *and*, *and-not*, and *or* represent the operators that were referred to previously.

$$\begin{aligned} Expr &= Expr\ OR\ Term \\ Term &= Term\ AND\ PURPOSE\ | \\ &\quad Term\ ANDNOT\ PURPOSE\ |\ PURPOSE\ |\ (\ Expr\ ) \end{aligned}$$

Figure 3.4: Compound purpose syntax

$$\begin{aligned} Expr &= Expr\ OR\ Term \\ Term &= Term\ AND\ REASON\ |REASON\ |\ (\ Expr\ ) \end{aligned}$$

Figure 3.5: Compound reason syntax

In the following section the use of compound purposes is considered based on a simple example.

3.7 Using Compounds

To clarify exposition it is assumed that $\cdot\neg_p$ has highest precedence, followed by, $\cdot_p$, and then the $+_p$. The discussion commences with an example to illustrate how one would possibly use a compound purpose.

Example 1 *Suppose an enterprise has, amongst others, three purposes in its Purpose Lattice (PL):*

1. ϕ_x = *"Update personal information"*.
2. ϕ_y = *"Sell new products"*.
3. ϕ_z = *"Update portfolio"*.

Also assume that there is no relation between any of these purposes. The enterprise can encode the following policies regarding storage of telephone numbers (only a subset of the possible policies that may be constructed using these three purposes and the aforementioned operators is shown).

1. *Telephone numbers are stored so that we may call you to update your personal information and your portfolio:* $\phi_x \cdot_p \phi_z$
2. *Telephone numbers are stored so that we may call you to update your personal information, or your portfolio, or so that we may sell you a new product:* $\phi_x +_p \phi_y +_p \phi_z$
3. *Telephone numbers are only stored for updating personal information and not for updating a client's portfolio:* $\phi_x \cdot \neg_p \phi_y$

■

Consequently a data user may wish to access the stored data (which has the purposes bound to it as provided in the previous example.

Example 2 *Assume the data user wishes to access the data:*

1. *I want to call clients to update their personal information and portfolios:* $\phi_x \cdot_r \phi_z$*. In this example, the reasons and purposes match perfectly.*
2. *I want to call a client to update their personal information or to* tell *them about a new product:* $\phi_x +_r \phi_w$*. Here* ϕ_z *must be weaker than* ϕ_w*, or else access will not be granted.*

From this example, the flexibility of using compound purposes should be quite evident.

3.8 Summary

The use of purposes in a privacy context can greatly enhance the capabilities of the implementation of the access control policy. This chapter showed that purposes can be organised by using a lattice. Purposes placed in the lattice are considered to be a subset of a finite set of purposes, and each purpose can therefore be labelled uniquely. By making use of the hierarchy formed by a lattice, more general purposes are placed near the upper bound of the lattice. Children purposes converge on the lower bound and are considered to be more specific, or better purposes for

wanting to access information. Thus users can provide better purposes in order to access information stored for more general purposes.

A notation for combining purposes to form compound purposes was provided, and axioms that define the semantics of compound purposes were formulated.

In the following chapter, a method for validating access requests that make use of compound purposes is described. Current verification algorithms cannot be used for access requests using compound purposes, and this algorithm provides a step in that direction.

Chapter 4

Verification

Did you see me walk into your shop?
Yes.
Have you ever seen me before?
Never.
Then how do you know it was me?
– Nasrudin

In the previous chapter, the structure that is used to store purposes when using compound purposes and reasons was presented. In this chapter, a method is given that can determine if a provided statement of intent is sufficient to gain access to data that has an associated (or bound) statement of purpose.

The flexibility that compound purposes and reasons impart would be of little value if it were not possible to perform verification of access requests that made use of compound reasons against the purposes that are bound to a datum [98]. Current verification algorithms are compound purpose and reason agnostic, and are thus unsuitable for access control when making use of compound purposes and reasons.

The verification algorithm is presented in two phases for clarity: firstly, the structures that support the verification algorithm, along with the formal validation of their operation, and secondly, the algorithm itself. The verification algorithm is in fact a combination of transformations that operate on the purposes that are bound to a datum, and the reasons that are given as part of an access request. The transformations do not modify the access request, or the purposes bound to the datum in any way, and it is proved that these transformations preserve the semantics of the access request, as well as the purposes bound to data.

The rest of the chapter is structured as follows: Section 4.1 provides definitions, and a discussion of the structures that are required to accomplish verification. Sections 4.2 and 4.3 provide a detailed discussion of compound operators

and how the introduced structures support their operation. Several proofs are provided to show that the operators and their supporting structures result in expressions that are sufficient for verification. The method through which verification is accomplished after the operators have been applied to an expression is examined 4.4. Section 4.5 contains an explanation of the run-time complexity of the verification technique, and techniques that may be used to speed up the computation of verification.

4.1 Verification Structures

In this section, definitions of the operators that function on reasons and purposes are provided, as well as the structures that are used for verification. The operators are defined in such a way that verification becomes a simple task, with some overhead restricted to the binding phase – this ensures that, from a practical point of view, once binding has been completed, access control is still fast.

The complex business of determining if a provided statement of intent is enough to gain access to data given the purpose that the enterprise stated it would use the information for dissolves into asking the very simple question: Is a (compound) reason Γ "good enough" when compared to a (compound) purpose Φ? In fact, this is such an important question that it is sensible to provide a simple notation for the question as well as the other computations and results that surround the question, and the effort to answer it. For this purpose, *the question* is written as $\Phi \leq \Gamma$?

The strategy for answering the question in this chapter is to make the question easy to answer, and then providing an answer. The way in which to make the question easier to answer is to convert the compounds that are part of the question to simpler structures that can be used to easily compute the answer. And so, from a semantic point of view, the ideal is to ask, $\Phi \leq \Gamma$? However, from a practical point of view, it is definitely better to ask a question that is easy to answer computationally. It is for this reason that the question is rather asked as: $\Phi' \subseteq \Gamma'$? Calculating if a set is a subset of another set is much simpler than attempting a rigorous analysis of a lattice to determine if all the relations are satisfied to ensure that $\Phi \leq \Gamma$.

To be able to ask if $\Phi' \subseteq \Gamma'$, both Φ', and Γ' have to be constructed from their representative statement of intent, and the compound purpose bound to the datum.

Along with the definition of each operator, it is proved that the resultant Γ' and Φ' are fully representative of the expressions they denote, and that verification therefore is done correctly.

The work done during verification is presented with some examples to help elucidate the goals of the construction of Γ and Φ, and the intermediate steps in

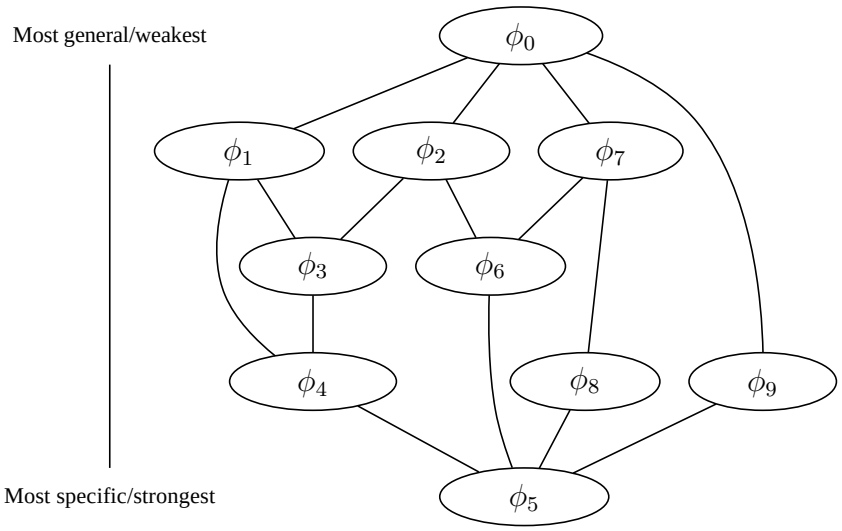

Figure 4.1: A simple purpose lattice

the process. A simple PL is provided in figure 4.1 in order to set the scene.

A basic example is now provided in order to explain what an enterprise may wish to accomplish with compound purposes. The apparent complexity of the scene is limited, in order to focus on the operation and function of the solution, rather than cluttering the reader's view of the process with too much information. Applying the proposed method to complex situations will not be difficult.

Example 3 *Suppose a compound purpose was specified as* $\phi_1 \cdot_p \phi_2 +_p \phi_7$*. Thus, a statement of intent that is*

1. *suitable for* ϕ_1 *and* ϕ_2 *simultaneously, or*
2. *suitable for* ϕ_7*, or*
3. *suitable for* ϕ_1 *and* ϕ_2 *and* ϕ_7 *simultaneously.*

will be suitable for gaining access to the datum.

This means that the set: Φ' *for* $\phi_1 \cdot_p \phi_2 +_p \phi_7$ *is* $\{\{\phi_1, \phi_2\}, \{\phi_7\}, \{\phi_4\}, \{\phi_3\}, \{\phi_4, \phi_6\}, \{\phi_4, \phi_8\}, \{\phi_4, \phi_6, \phi_8\}, \{\phi_6\}, \{\phi_8\}, \ldots\}$ *will be sufficient. For the sake of brevity a full listing of possible purposes is not made here.*

Suppose a data user presents $\phi_4 \cdot_r \phi_6 +_r \phi_8$ *as a statement of intent.* Γ' *representing his request is* $\{\{\phi_4, \phi_6\}, \{\phi_8\}\}$.

In this example, since $\{\{\phi_4, \phi_6\}, \{\phi_8\}\} \subset \{\{\phi_1, \phi_2\}, \{\phi_7\}, \{\phi_4\}, \{\phi_3\}, \{\phi_4, \phi_6\}, \{\phi_4, \phi_8\}, \{\phi_4, \phi_6, \phi_8\}, \{\phi_6\}, \{\phi_8\}, \ldots\}$*, access can be granted.*

■

In the example above the purposes used (which will be used to define compound purposes and reasons) should obviously be part of an ontology defined by the enterprise (a language such as EPAL allows an enterprise to define its own set of purposes). This set of enterprise-defined singleton purposes is referred to as the DPS for the enterprise, and these purposes are arranged in a lattice by the enterprise's privacy officer, or other responsible party.

The structure that represents Γ' is called the *reason set* for Γ, and the structure that represents Φ' is called the Suitable Purpose Set (SPS) for Φ. The construction of Γ' is in fact very simple, and a detailed discussion of that topic will be provided in section 4.3. The construction of Φ' requires a few more steps which are presented hereafter.

The SPS for Φ is a set of sets, wherein each element of the set represents a term of the original expression, and collectively the set represents the expression (consider example 3 again). Each element therefore is a conjunction, and the set itself is a disjunction.

For readability, the parts that constitute the SPS are presented before the SPS is considered in its entirety. Thus the conjunctions that form part of the SPS are first discussed. These conjunction sets are called *purpose sets*.

4.1.1 Purpose Sets

Purpose sets have two properties that make them suitable for use as representative elements of a compound purpose, and thus sufficient to use during verification. These two properties, *unambiguity* and *sufficiency*, are explored in this order. Sufficiency is further divided into two subproperties.

The reason these properties are desirable will become clear once the properties themselves have been identified, and the theory behind them enunciated.

4.1.1.1 Unambiguity

Simply stated, *unambiguity refers to the property that a set contains only elements for which there is no relation in the PL.* Suppose two purposes ϕ_i and ϕ_j are bound to a datum, ω, and $\phi_i \leq \phi_j$. The purpose set associated with ω will contain all of the purposes that can be presented to be granted access to ω. That is, a reason that is suitable for (dominates) ϕ_i, **or** ϕ_j must be presented to gain access to the datum. However, since ϕ_j dominates ϕ_i, any reason that is suitable for ϕ_j will also be suitable for ϕ_i.

The question now becomes: can ϕ_i simply be dropped from the purpose set? When one considers the possibility of failure the answer becomes easy – presenting a reason that dominates ϕ_i, but not ϕ_j may be a violation of the requirement that any statement of intent presented by the data user must be suitable for all

purposes bound to that datum. As such, if the user must present a reason that is suitable for both purposes bound to the datum, then an access violation would occur, and access must be denied.

If it were the case that a reason that is suitable for either one of the bound purposes may be presented, then no violation would occur. However, since a *purpose set* is a conjunction of purposes, the former case is relevant, resulting in a violation of the said requirement that presented statements of intent are suitable for the bound purposes. In this case, the weaker purpose must not be present in the purpose set. The work in this text makes no assumption, and offers no algorithm for removing the superfluous purpose from the purpose set. For the moment, it is assumed that the privacy policy officer ensures that no such violations occur.

The unambiguity property is thus formally as follows:

Definition 7 (Unambiguity Property) *A purpose set X is* unambiguous *if none of the elements dominate each other, and the set is not empty.*

Thus, if and only if $\forall x_i, x_j \in X,\ x_j \not\leq x_i$, with $i \neq j$, then X is unambiguous.

The exception to this rule is when $i = j$, with $|X| = 1$, such a set is defined to be unambiguous.

■

In the following section the property of sufficiency is discussed.

4.1.1.2 Sufficiency

The second property is sufficiency. Sufficiency is used to describe the degree to which a purpose set contains purposes which are suitable for a (compound) purpose. As mentioned in the introductory section, two subproperties of sufficiency are defined.

The first, single sufficiency, describes purpose sets which contain singleton purposes (or only one singleton purpose) suitable for some singleton purpose. The second, complete sufficiency, is an amended singly sufficient purpose set, in that the purpose set contains purposes which dominate more than one singleton purpose.

Singly sufficient purpose sets are typically present in cases where a term from a compound purpose expression comprises a single purpose. It is possible that an SPS may consist of only singly sufficient sets, in which case it will be an SPS for a singleton purpose.

Definition 8 (Single Sufficiency) *Given a purpose lattice PL, a purpose set P is singly sufficient for a singleton purpose ϕ_i, if there is a lattice S which is a sublattice of PL, such that $\forall p \in P,\ \exists(\phi_i, p) \in S$. And $\forall(a, p) \in S, a = \phi_i$.*

■

Completely sufficient purpose sets are typically present in cases where a conjunction of purposes is present in the compound purpose expression. A completely sufficient set is thus a purpose set which contains purposes which will dominate all of the purposes a term of the compound purpose expression. It is important to note that any purpose from the purpose set must dominate all the singleton purposes from the compound purpose expression in question.

More formally, the property of complete sufficiency may be shown as follows:

Definition 9 (Complete Sufficiency) *Suppose set A holds all the singleton purposes from a term in a compound purpose expression.*

A purpose set P is completely sufficient if and only if it is the case that for all p in P it must be the case that $a \leq p$ for all a in A. That is, $(a, p) \in PL, \forall a \in A, p \in P$.

■

How do negative purposes feature in these ideas? Simply put, a purpose set cannot be sufficient in any way if it contains purposes that are children of purposes specified as negative purposes in the term from the compound purpose against which it is being compared. To this end, the Complement to Suitable Purpose Set (CSPS) is introduced in the following section.

4.1.2 The Complement to Suitable Purpose Set

Recall that a negative purpose is used to indicate that a certain purpose or several purposes cannot be used to gain access to some datum. To simplify verification, negative purposes, as well as those purposes which dominate those negative purposes are placed in a set known as the CSPS. The CSPS for an SPS Φ' is written as $\overline{\Phi'}$.

During verification $\overline{\Phi'}$ is examined to determine that the *statement of intent* does not contain a negative purpose. This is done by ensuring that no purpose listed as a reason is present in the $\overline{\Phi'}$.

It is envisaged that the CSPS is populated during the binding phase, avoiding unnecessary computation during the limitation phase.

4.1.3 Suitable Purpose Sets

The unambiguity and sufficiency properties, as well as the CSPS have now been proved sufficient for the compound purpose representation. It has also been explained how the above can be used during purpose limitation. A normative definition of the SPS (which is crucial for verification to be done correctly) is now given.

The definition of the SPS combines the above properties and structures in order to provide a structure that is clearly defined, and a mechanism for verification to be done relatively easily.

Definition 10 (Suitable Purpose Set) *An SPS for Φ is a set of purpose sets such that:*

1. *Every purpose set in the SPS is **unambiguous**, and*
2. *Every purpose set in the SPS is **sufficient** for Φ (either singly, or complete), and*
3. *All possible **unambiguous** and **sufficient** purpose sets for Φ are present in the SPS,*
4. *No element from any of these purpose sets is either included in the purpose expression as a negative purpose, or dominates purposes included in the purpose expression as negative purposes.*

The last two properties are collectively called the completeness property *of an SPS.*

■

Now that the SPS has been properly defined, one may ask how such a structure is constructed. The SPS for a compound (or a singleton) expression is constructed through the actions of the operators that were first introduced in section 3.5. These operators are used notationally to provide a means for the enterprise to express its intent with stored data; they are also used for constructing the SPS which is central to the ideas presented in this text.

In the following section the compound purposes that are used to construct the SPS are explained in more detail; the *and-not* operator is also explained which is necessary to populate the CSPS.

4.2 Compound Purpose Operators

In this section the mechanism for constructing the SPS is expounded: the compound purpose operators. The key concept behind the SPS is the creation of I-Sets, or *initial sets*. I-Sets provide a well-defined starting point from which to construct potentially complex SPSs that may result from evaluating a compound purpose expression. The other reason I-Sets are so important from the perspective of verification is the fact that I-Sets are in fact SPSs (see section 4.2.1) from which

more complex SPSs are constructed – and they are relatively simple to construct initially.

The resulting strategy for performing verification is decidedly uncomplicated. Since an I-Set is expected to contain all the valid purpose set entries for a singleton purpose (a property of an SPS), all singleton purposes in Φ are converted to I-Sets. The resulting I-Sets are SPSs which are passed to the operators in the compound expression; the end result being that Φ is transformed into Φ', which will be used in the verification.

All compound purpose operators are proved to be property preserving in the following sections; thus, if the operands to an operator are SPS, then the result of the operation is an SPS. Thus the result will be an absolute representation of the purposes that may be presented to gain access to protected data.

4.2.1 I-Sets

When a compound purpose *expression* Φ is examined for verification purposes, it will typically be in the form $\Phi_0 +_p \Phi_1 +_p \ldots +_p \Phi_n$, with Φ_k being either a singleton purpose, or another compound expression.

An I-Set is a structure that is an absolute representation of a singleton purpose. In particular, it represents an absolute in terms of the purposes that can be presented to gain access to a datum protected by a singleton purpose. Intuitively it must therefore be an SPS. To help elucidate the direction this discussion is moving, consider (from figure 4.1), for example, an I-Set for purpose ϕ_7. Since ϕ_6, ϕ_8, and ϕ_5 are all suitable for ϕ_7, it is clear that the I-Set for ϕ_7 will include them. However, since ϕ_6 and ϕ_8 are "simultaneously" suitable to access data protected by ϕ_7, their combination must also be present in the I-Set, since a data user may provide a statement of intent exactly thus. The I-Set for ϕ_7 can therefore be summarised as $\{\{\phi_7\}, \{\phi_6\}, \{\phi_5\}, \{\phi_8\}, \{\phi_6, \phi_8\}\}$. Notice that combinations such as $\{\phi_6, \phi_5\}$ are not in the I-Set for ϕ_7 as their inclusion would violate the unambiguity property.

The verification method in this chapter, and therefore also, by implication, the compound purpose operators, are designed to operate on SPSs. Internally, the notational definition of a compound purpose is transformed into a suitably representative expression consisting of SPSs, and operators that are suited to handling them. In the course of discussing I-Sets, it will also be proved that I-Sets are in fact specialised forms of SPSs. This fact is advantageous since no special initial states have to exist for the operators which (as has been stated) are only defined to operate on SPSs.

The I-Set, in order to be an SPS, must be unambiguous, sufficient, and complete. Also recall that since an I-Set is a representation of a singleton purpose, it should contain only singly sufficient purpose sets (section 4.1.1).

The I-Set for a singleton purpose ϕ_i can be constructed by taking a set I of all the purposes that dominate ϕ_i, and constructing a power-set from that set. A constraint is placed on the contents of the elements of the power-set: they must be unambiguous. Proof 1 shows that the result of such a computation produces an SPS.

Definition 11 (I-Set) *An I-Set for a singleton purpose ϕ_i is a set that contains purpose sets, such that every element in a purpose set from the I-Set dominates ϕ_i, and no element in a purpose from the I-Set dominates another element from that purpose set.*

The I-Set is easily constructed algorithmically by finding the set of all the purposes that dominate ϕ_i and taking the power-set of that set. Lastly, all elements that are dominated by another element in the same purpose set are excluded from the result.

More formally:

1. *Take $X = \{x | \phi_i \leq x,$ with $x, \phi_i \in DPS\}$*
2. *Let $D = \mathbb{P}(X)$ then the I-Set for ϕ_i is $I_S(\phi_i) = \{Y \in D | \forall y_i, y_j \in Y, y_j \not\leq y_i \wedge i \neq j \wedge Y \neq \emptyset\}$.*

■

It has already been stated that an I-Set is an SPS, and as can be intuitively noted from the definition provided above, it should be so. However, a formal proof of the statement is provided.

Theorem 1 (I-Set Validity) *An I-Set $I_S(\phi_i)$ is an SPS for ϕ_i.*

■

Proof 1 (I-Set Validity) *It can be proved that an I-Set is an SPS by showing that it possesses the three properties of an SPS: unambiguity, sufficiency, and completeness.*

Sufficiency*: By step 1 (of I-Set construction, definition 11), X is singly sufficient for ϕ_i.*

Unambiguity *For step 2 (of I-Set construction), since $\mathbb{P}(X)$ produces a set of sets from a singly sufficient set, the resulting sets cannot contain elements which do not dominate ϕ_i, and therefore must be singly-sufficient; that is, $\forall X' \in P(X)$ it must be the case that $X' \subseteq X$. The restriction from step 2 ($\forall y_i, y_j \in Y, y_j \not\leq y_i \wedge i \neq j$) creates a set of unambiguous sets. The empty set is discarded to ensure that the unambiguity property is met fully.*

Completeness*: Showing that an I-Set is complete proceeds in two steps. Firstly, it is shown that no more unambiguous sets can be added to the I-Set, and secondly, that no sufficient sets can be added to the I-Set. The implication is that no purposes from the DPS are missing from the I-Set which represents the absolute representation for a singleton purpose.*

Take any purpose set $D' \in I_S(\phi_i)$. D' is thus sufficient and unambiguous.

If $\exists c \in X$ such that $D' \cup \{c\}$ does not violate the unambiguity property for ϕ_i then $D' \cup \{c\}$ must already be part of $I_S(\phi_i)$ by step 2 of the I-Set construction algorithm.

It is clear that $\forall v \in (DPS \setminus X), D' \cup \{v\}$ violates the sufficiency property (from the definition of an I-Set). Thus nothing that was not initially part of X can be added to D'.

If, however, $\exists c \in DPS$, where $D' \cup \{c\}$ does not violate the sufficiency property, it must be the case that $c \in X$, since X contains all the purposes that dominate ϕ_i. However, it has already been shown that if $c \in X$ then $D' \cup \{c\} \in I_S(\phi_i)$.

Since construction of the I-Set commences with a singleton purpose, and no singleton purpose is added to the CSPS for the singleton purpose, it must be the case that $\overline{\phi_i'} = \emptyset$. Therefore no purposes from the purpose sets in $I_S(\phi_i)$ can be in $\overline{\phi_i'}$.

By definition 10 an I-Set is thus an SPS for a singleton purpose, since all the purpose sets of the I-Set are unambiguous, singly sufficient, and the I-Set is complete.

■

At this point it is possible to provide definitions for the operators that have been mentioned from a use perspective. The first operator that will be introduced is the *and* operator.

4.2.2 And

Based on the notational definition of the *and* operator, the data user should only supply purposes that dominate the compound expression. Thus, when defining the operation of the *and* operator, it must be guaranteed that it produces a valid SPS from its operands.

Assume that each operand of the operator in question is understood to be a valid SPS. Before commencing evaluation of any compound purpose expression, it is to be understood, as has been stated previously, that each of the singleton purposes in the expression will be converted to an I-Set. It has been proved already that I-Sets are valid SPSs. This means that each set from each operand is suitable for each operand respectively. It is therefore clear that combinations of sets from the SPS for the first and second operands will be a suitable answer to the question

of which sets should be present in the result. Two further questions arise from this reasoning. Firstly, in what way should these sets from the first and second operands be combined? Secondly, are these all the possible sets that should be in the result, and will there be any sets, or elements from these sets, that do not belong in the result?

The first question can be answered by ensuring that the combination of two sets results in unambiguous sets. This requirement implies that the normal set union operator would not suffice. The reason for this has to be answered in two phases. Firstly, simply creating a union of the two SPSs will not produce the correct SPS, since the SPS will now contain sets which may only be suitable for one of the operands. Secondly, this means that the elements of the SPSs should be added together; however, special care has to be taken to ensure that adding the purpose sets from both SPSs together does not violate the properties of the SPS.

As a consequence of the above restrictions that are placed on the operation of the *and* operator, a new operator for combining sets is defined (definition 12). This operator guarantees that the result of a combination operation is an unambiguous set.

The second question is answered by recalling that each operand contains all the possible sets that are considered valid for that operand. Since the *and* operation does not introduce sets not already present in either of the SPSs,there should be no purpose sets which should not be present, and all the sets that should be present must in fact be present. A proof that this is in fact the case is presented in proof 2.

One final consideration of the *and* operator is that it may reintroduce elements in purpose sets which were explicitly removed. In the case of the mechanism in this text, such an element may have been removed from an operand's purpose sets by way of the *and-not* operator. If this element was in fact present in a purpose set of the other operand, a violation of the completeness property will occur. In this case the *and* operator relies on the existence of a blacklist, or closed list; the CSPS is used in this instance to ensure that the said anomaly does not come to pass.

The combine operator is defined formally as follows:

Definition 12 (Combine Operator $\sqcup$) *The $\sqcup$ operator is recursively defined as follows:*

$$\{\} \sqcup \{a_0, \ldots, a_n\} = \{a_0, \ldots, a_n\} \tag{4.1}$$

$$\{a_0, \ldots, a_n\} \sqcup \{b_0, \ldots, b_m\} = \{a_1, \ldots, a_n\} \sqcup X \tag{4.2}$$

X is consequently defined for three cases.

1. *The first deals with a_0 not having any relation to any element b_i. Thus, $\forall b_i$ with $a_0 \not\leq b_i \wedge b_i \not\leq a_0$ it is clear that $X = \{b_0, \ldots, b_m, a_0\}$.*

2. *The second case ensures that purpose sets remain unambiguous, by ensuring that the inserted element is not less suitable than any element already in the purpose set. Thus, if* $\exists\, b_i$ *such that* $a_0 \leq b_i$ *then* $X = \{b_0, \ldots, b_m\}$.

3. *The final case also deals with ambiguity, by ensuring that all the purposes that the inserted element dominates are removed from the purpose set. Thus, if* $\exists b_i$ *such that* $b_i \leq a_0$*, then take* $Y = \{b_i | b_i \leq a_0\}$*, then* $X = (\{b_0, \ldots, b_m\} \setminus Y) \cup \{a_0\}$.

■

As a result of the definition of the combine operator, it is now possible to provide a definition of the *and* operator, which will ensure that the result it delivers is an SPS.

Definition 13 (*And* Compound) $\Phi'_1 \cdot_p \Phi'_2 = \{x \sqcup y | x = j \setminus \overline{\Phi'_2}, j \in \Phi'_1$ *and* $y = k \setminus \overline{\Phi'_1}, k \in \Phi'_2\}$

■

Up to this point, it has been indicated that the *and* operator is property preserving. That is, it accepts two SPSs as operands, and produces a single SPS as a result. This resulting SPS can again be used as an operand to another operator. A formal proof for this statement is now provided.

Theorem 2 (*And* Compound Validity) *The* and *operator, given two SPSs as operands, produces an SPS which is an absolute representation for the operands. That is, the resulting SPS will contain only purposes, or combinations of purposes which are suitable for use as statements of intent for* both *purposes as represented by the operands passed to the* and *operator.*

■

Proof 2 (*And* Compound Validity) *Assume that* Φ_1*' and* Φ_2*' are both SPSs.* Φ_1*' and* Φ_2*' represents the SPSs for* Φ_1 *and* Φ_2 *respectively. Take* X'' *to be the result of* Φ_1*'* $\cdot_p$ Φ_2*'.*

Proof of this statement is presented in three phases, much as was done with the proof of the I-Set. Proving that the three properties hold not only proves that the result is an SPS, but also that it is an SPS that is suitable as a result of applying the and *to its operands.*

Unambiguity*: By definition of the* $\sqcup$ *operator, the sets in the resulting SPS must be unambiguous.*

Sufficiency*: Take* $U \in \Phi'_1$ *and* $V \in \Phi'_2$*. Since all sets in* X'' *are composed as* $U \sqcup V$*, it is clear that every* Y *in* X'' *is at least suitable for either* Φ_1*,* Φ_2 *or both.*

Completeness*: Firstly, suppose X'' is not complete. Then there must exist a purpose set Y which is unambiguous and sufficient, which can be added to X''. This means that either $Y \in \Phi'_1$, or $Y \in \Phi'_2$, or by the definition of the* and *operator $Y = K \sqcup L$. In the latter case, K must be suitable for Φ_1, or Φ_1 and Φ_2, and L must be suitable for Φ_2, or Φ_1 and Φ_2. Thus $K \in \Phi'_1$, or $K \in \Phi'_1$ and $K \in \Phi'_2$. Also, $L \in \Phi'_2$, or $L \in \Phi'_1$ and $L \in \Phi'_2$. However, by definition of the* and *operator, if the previous statements are true, then Y must already be in X''.*

Secondly, Φ'_i cannot contain any elements from $\overline{\Phi'_i}$ (definition of the and-not *operator in section 4.2.3.1). It is, however, possible for Φ_1 (not the SPS) to contain purposes that are elements in $\overline{\Phi'_2}$. This will happen when Φ_1 permits a purpose n to be used, but Φ_2 prohibits its use. The definition of the* and *operator, however, ensures that all elements present in the second operand's CSPS are removed from the elements in the first operand, and vice versa. This ensures that no element present in a CSPS from any of the operands can be present in the result of the $\sqcup$ operator.*

Thus, the and *operator produces a valid SPS for its operands.*

■

It is trivial to show that the result of the *and* operator enforces the semantics of the and statement as defined in axiom 2. By comparing each statement of the axiom to the result of an *and* computation, it is possible to determine if the *and* calculation matches what is expected from the *and* compound, and a proof is omitted.

The following section deals with the operation of the *or* operator.

4.2.3 Or

The *or* operator allows the use of data for more purposes (as specified in the purpose expression), so in a sense it can be seen as less restrictive than the *and* operator. Because it is less restrictive it will intuitively have more purpose sets that form part of a final SPS.

The *or* operator ensures that the resulting SPS contains purpose sets which are sufficient for either one of the operands or both. Informally, this means that any purpose set that is sufficient for the first operand must be part of the resulting SPS, as well as any purpose set that is sufficient for the second operand. Moreover, the resulting SPS must also contain purpose sets which are sufficient for both operands (simultaneously).

The result of the *or* operator is thus easily accomplished in view of the *and* operator discussed above: include the sets from both operands individually, and include the result of performing an *and* operation on the operands.

Because the *or* operator includes sets from either operand individually, as well as a combination of purpose sets from both, the previously alluded to intuitive view of the operator is in fact true. A formal definition of the operation of the *or* operator is provided:

Definition 14 (*Or* Compounds) $\Phi'_1 +_p \Phi'_2 = \Phi'_1 \cdot_p \Phi'_2 \cup \Phi'_1 \cup \Phi'_2$

■

This definition is nothing more than the one previously introduced. Of course, it has to be ensured that the result of the *or* operator produces the correct SPS for its operands. As such, the validity of the *or* operation is considered next.

Theorem 3 (*Or* Compound Validity) *An* or *compound produces a valid SPS for its operands. This SPS is also the absolute representation for the purposes and combination of purposes that may be used to gain access to data which is protected by the operands passed to the* or *operator.*

■

Proof 3 (*Or* Compound Validity) *The proof for the* or *compound follows from the* and *compound proof. Assume that* Φ'_1 *and* Φ'_2 *are SPSs for* Φ_1 *and* Φ_2 *respectively.*

Let X'' *be the result of* $\Phi' +_p \Phi'$.

Unambiguity*: The result of* $\Phi'_1 \cdot_p \Phi'_2$ *is unambiguous; moreover,* Φ'_1 *and* Φ'_2 *are unambiguous already since they are valid SPSs.*

Sufficiency*: As with the* and *operator, the result of* $\Phi'_1 \cdot_p \Phi'_2$ *is sufficient, and so are* Φ'_1 *and* Φ'_2 *(per definition). Thus* X'' *consists of unambiguous and sufficient sets.*

Completeness*: To show that the SPS is complete it is only necessary to remember that the* and *operator produces all the purposes sets that are required for both operands – thus it is ensured that statements of intent involving both operands are catered for. What is missing from the result, however, is the purpose sets which represent each individual operand (since it will be perfectly legal to provide only purposes (or combinations thereof) which are suitable for only one of the operands. Those purpose sets are not added as part of the* and *step in the process. Therefore, they are trivially added to the resulting SPS by performing a standard set-union operation on the resulting SPS and the two operands.*

Since the and *operation step produces all possible purpose sets for a conjunctive statement, and the union step adds all possible purpose sets for the individual operands, it is clear that the resulting SPS contains all the (and no more) possible purpose sets for the expression.*

Once again it is trivial to show that X'' *does not contain any purposes from* $\overline{X}$ *(see proof 2), and the proof is omitted.*

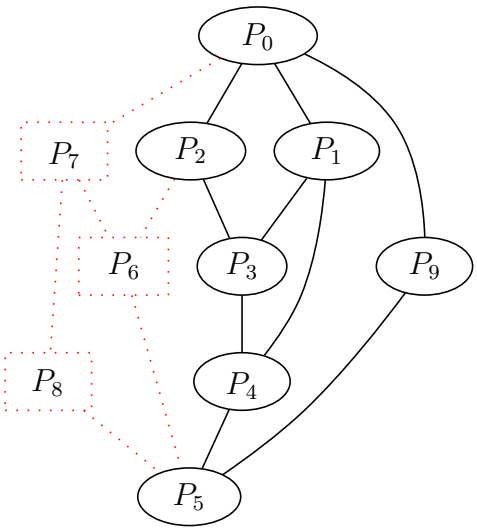

Figure 4.2: Purposes marked for removal in a negative purpose

Thus, the or-operator produces a valid SPS from its operands.

■

Again, it is trivial to show that the operation of the *or* operator matches the expectation from the or compound's semantics as provided in axiom 1, and a discussion is not provided.

This ends the discussion of the two standard operators for compound purposes: *and* and *or*. The following section commences discussion of the *and-not* operator, which is used to construct the CSPS for an expression.

4.2.3.1 And-Not

Functionally the *and-not* operator should ensure that no purpose that is suitable for its second operand is present in the SPS for the first operand. This ensures that those purposes are removed and that they are no longer "considered suitable".

Consider, for example, a compound purpose associated with a piece of data as "$\phi_2 \cdot \neg_p \phi_7$". Figure 4.2 illustrates the purposes that are to be removed to validate the compound statement.

The *and-not* operator simply removes all those purposes which are suitable for the second operand from the first operand's SPS. These purposes that were removed are then added to the CSPS for the first operand. This allows the completeness test (introduced in definition 10) to be performed without having to add purposes from the lattice which would not have violated the unambiguity and sufficiency properties of the SPS.

Recall that the least upper bound of the lattice should never be removed from the lattice. An important aspect of the *and-not* operator is the fact that the least upper bound of the lattice is in fact never removed from the first operand's SPS. Since this purpose will be a "powerful" purpose, ideally used in limited cases for tasks such as database maintenance, or tasks that companies are required to perform by law, it is necessary to ensure that it remains a legitimate way of accessing information. Ideally, only the SSO or senior DBA will be authorised to use this particular purpose.

From the informal definition provided above it is clear that the *and-not* operator takes a valid SPS, removes requested purposes from the structure, and places those in the CSPS. It does not add any new purposes, and does not remove any purposes other than requested. It is therefore trivial to prove here that the *and-not* operator produces a valid SPS for its operands, and a proof to this effect is presented in proof 4.

An important part of compound purpose expressions is the ability to define negative purposes. This is done using the *and-not* operator. Negative purposes are removed from purpose sets as specified in the following definition.

Definition 15 (*And-Not* Operator) *For any* and-not *compound* $\Phi \cdot \neg_p \phi$*:* and-not *purposes can be removed from the SPS using the following steps:*

1. $L = \{x | \phi \leq x \wedge x, \phi \in DPS\}$
2. $L' = L \setminus \phi_{max}$*, where* ϕ_{max} *is the least upper bound of the purpose lattice.*
3. $\Phi' = \{X | X = Y \setminus L', \forall Y \in \Phi'\}$
4. $\overline{\Phi} = \overline{\Phi} \cup L'$

■

Theorem 4 (*And-Not* Validity) *The* and-not *operator produces an SPS from its operands.*

Proof 4 (*And-Not* Validity) Unambiguity*: Since the* and-not *operator removes purposes from a purpose set, it is not possible for it to introduce ambiguous sets to the SPS.*

Sufficiency*: The* and-not *operator does not remove any purpose sets from the first operand, save for one case: where the second operand is the only element of the purpose set. In the case where no sets are removed from the SPS, it is clear that it will contain all the possible purpose sets to be an absolute representation of the operand. In the case where the singleton purpose is removed, the following*

holds: Since there is a $D \cup \{c\}$ (with $D = \{\}$) which can be added to X'' which does not violate the ambiguity, or sufficiency properties, it is clear that X'' is not complete. However, since $\{c\} \in \overline{\Phi}$, it is not possible to add $D \cup \{c\}$ to X'' without violating the completeness property of X''.

Thus, the and-not *operator produces a valid SPS for its operands.*

■

In the following section, the compound reason operators are considered in mote detail. These operators are used as part of the statement of intent in an access request.

4.3 Compound Reason Operators

Now that the operators that are used to construct the structures that represent the purposes bound to data have been introduced, the operators that construct the structures used as part of an access request (the data user's statement of intent) are discussed. Since the result of the compound purpose operators is the SPS, which is a set of sets, it is intuitively clear that the goal of the compound reason operators is to convert the statement of intent into a set of sets representing the access request. This set is trivially compared to the SPS to determine if access should be granted – based on the intent of data use, whether or not the data user has access to read data is still to be determined by the access control subsystem (see the next chapter for more detail on this).

The compound reason operators are also defined to operate on sets. An example will help clarify the approach proposed in this section.

Example 4 (Converting a Compound Reason) *At the start of evaluation of a compound reason the singleton purposes that form part of the compound reason are first converted into simple sets of the form $\{\{\phi\}\}$.*

Consider, for example, the simple compound reason $\Gamma = \phi_9 +_r \phi_2 \cdot_r \phi_7$. All the singleton purposes are transformed into simple sets which will be passed as operands to the reason operators, thus $\Gamma' = \{\{\phi_9\}\} +_r \{\{\phi_2\}\} \cdot_r \{\{\phi_7\}\}$.

■

The purposes that are specified in the reason expression are transformed regardless of their existence, and the fact that they may dominate each other. No assumption is made to the effect that the system will attempt to rectify any mistakes in the user's intent statement, or give warning about incorrect purposes being used. Users that specify a reason for requesting access to data must be absolutely correct in their statement of intent. In this way it can be ensured that users can be held accountable for their actions [100].

At this point, the operators can be considered in more detail. Before the operators perform any transformation on their operands, the operands are first (as is the case with compound purposes) converted to sets. The rules for conversion can be summarised as follows: Firstly, consider every operand for reason operators to be a set of sets. Initially this is accomplished by placing a purpose (that is written as part of the reason expression) as a single element set, within a set. For example, the reason r_i is taken as the operand r_i. Now, the *and* operator produces the union of all the elements from the first operand with the elements of the second operand. The *or* operator produces the union of its two operands.

For exposition purposes, it is once again assumed that $\cdot_r$ has higher precedence than $+_r$.

Definition 16 (And Operator (Reasons)) $\Gamma'_1 \cdot_r \Gamma'_2 = \{x \cup y | x \in \Gamma'_1 \wedge y \in \Gamma'_2\}$

■

Note that the *or* operator limits the combination of purposes that are transformed. Since it cannot be ensured that the user will use the data for all the purposes that are stated as part of the compound reason (users may lie, after all), they cannot be combined as was done for the *and* operator. Consider the access request and test: $\phi_1 \cdot_p \phi_2 \leq \phi_1 +_r \phi_2$, with $\phi_1 \cdot_p \phi_2$ the binding and $\phi_1 +_r \phi_2$ the statement of intent. Clearly the data user is indicating that he will use the data for either one of the two stated purposes. It is obvious that access cannot be granted, since semantically what the user is saying is that he will be using the information for either one of the two purposes *or* both. The policy clearly states that the data will only be used for both purposes, and should thus not be released. Therefore $\phi_2 \cdot_p \phi_2 \not\leq \phi_1 +_r \phi_2$ as described in Van Staden and Olivier [99].

Definition 17 (*Or* Operator Reasons) $\Gamma'_1 +_r \Gamma'_2 = \Gamma'_1 \cup \Gamma'_2$

■

This concludes the discussion of the operators that perform the translations from a compound purpose and reason expression to a form that will allow verification to take place. The following section covers exactly that.

4.4 Final Verification

Final verification of the compound reason against the compound purpose is done by ensuring that the reasons that were presented in the statement of intent are a subset of the SPS of the compound purpose. Since the SPS of the compound purpose holds all the valid combinations of purposes which can be presented in order to gain access to a datum protected with the particular compound, it follows

that the transformed reason set must be a subset of the SPS in order for access to be granted. A proof for verification is given in section 5.

It is thus now possible to clearly explain what is meant by suitable "in some way" first mentioned in section 4.1. A compound reason is "suitable in some way" for a compound purpose if it contains only valid combinations of purposes which can be legally presented to gain access to a datum protected with the compound purpose.

A brief and informal comparison of the proposed verification and what one would intuitively expect of a verification mechanism to do is now made.

Verification of the statement of intent is lastly accomplished by doing a simple test to see if the set that represents the compound reasons is a subset of the SPS for the compound purpose.

Theorem 5 (Verification of Compounds) $\Phi \leq \Gamma \Leftrightarrow \Gamma' \subseteq \Phi' \wedge \forall A \in \Gamma', A \cap \overline{\Phi} = \emptyset$.

■

Proof 5 (Verification of Compounds) *From the definitions of I-Sets and SPSs, it has been shown that Φ' is complete. That is, it contains all the possible combinations of purposes that can be used to gain access to information from a privacy perspective.*

If $\Gamma' \subseteq \Phi'$, then Γ' contains only purposes and combinations of purposes which are suitable for Φ, moreover, since none of the purposes in the sets in Γ' are in the CSPS for Φ it logically follows that $\Phi \leq \Gamma$.

■

By carefully constructing the structures that are absolute representations of the compound purposes bound to data, as well as the structures that represent the data user's statement of intent, it becomes possible to correctly control access to data from a privacy perspective. Equally important, is the ability to prove that access control is taking place correctly.

In the following section, the important question regarding the apparent run-time complexity of calculating compound purposes and reasons is considered.

4.5 Verification Complexity

The work in this chapter offers the reader the idea of using sets to verify compound reasons against compound purposes. An important aspect of using any form of algorithm to is to consider the run-time complexity of such an algorithm. The efficiency of the algorithm will determine firstly, if it should be used at all, and secondly, if anything should be done to mitigate the effects of a slow algorithm on

a desired (perhaps slow) result. In this section, the merits of the algorithm above are discussed, including the run-time complexity, and measures to be put in place for efficient use of the verification technique. The particular focus is the run-time complexity for the verification phase, since this is what has the biggest impact on the usability of the verification mechanism.

There are two key phases of the presented work to consider for performance. Firstly, the construction of I-Sets, and secondly the application of the operators and the verification.

Before the complexity of the methods is considered in more detail, it is important to note that one can discuss complexity with regard to a particular choice of implementation. A rudimentary implementation will have a severe performance impact on the system, much as a bubble sort is on average much slower than a standard merge sort algorithm – this is not to say that sorting should not be attempted at all.

With a little forethought and some creativity, it is possible to enhance the performance of the system dramatically. Throughout the discussion, it is assumed that sets will ultimately be presented as a string of bits. There are well-known advantages to using bit-strings to represent sets [53] and further discussion is excluded from this text. Another requirement for enhanced performance of the system is the presence of *dominance sets*.

Definition 18 (Dominance Sets) *A dominance set for a purpose ϕ_i is a set which holds all the elements that ϕ_i dominates. More formally, the dominance set Δ_{ϕ_i} for ϕ_i is a set such that all the elements from the set are dominated by ϕ_i (also recall that $\phi_i \leq \phi_i$). However, since all dominance sets will include the Greatest Lower Bound (GLB) of the lattice, it is removed, with no loss of information.*

Each dominance set has a complement ($\Delta_{\phi_i}^{-1}$), which will be the set of all the purposes that dominate that particular purpose.

The creation of dominance sets will typically take place during the maintenance phase of the system, and will therefore not impact the purpose limitation phase.

Constructing a power-set from any other set algorithmically can be accomplished in $O(2^n)$. The restrictions on the particular power-set for a I-Set are of linear time so calculations become $O(n \times 2^n)$ which is still $O(2^n)$. This exponential time algorithm only makes sense for a small number of purposes in the DPS; it will hamper construction of I-Sets, and consequently potential adoption of the methods proposed in this text. Thus, constructing the I-Set is not a practical solution, moreover, storing the I-Set for later use is also not a good idea.

It is, however, possible to accomplish verification computationally and some thoughts are presented here to illustrate how it can be accomplished, followed by the expected runtime complexity.

It is possible to determine if a set from Reason Set (RS) (one of the reasons presented for accessing data) would be part of the created SPS by examining the compound purpose expression (bound to the data), using the operator definitions provided in this chapter, and the dominance set. Thus the following rule will hold for determining if a purpose will be in the I-Set of another purpose.

Evaluation Rule 1 *A purpose will end up in the I-Set of another purpose if it dominates that purpose. This purpose will thus be present in the dominance set for that purpose.*

■

Determining if a purpose will end up in another purpose's I-Set is only half the problem: it must be possible to determine if the purpose (and eventually a compound purpose) will end up in the SPS of a compound purpose expression. Again, the definition of the operators provide the way to determine the answer, and the following rules will hold:

Evaluation Rule 2 *A purpose will be in a set in the SPS resulting from an* or *operation if the purpose dominates either one of the operands of $+_p$. Since it will be in the I-Set for either one of the operands (see definition 14) it must be in the result of the* or *operator.*

■

Evaluation Rule 3 *A purpose will be in a set in the SPS resulting from an* and *operation if the purpose dominates* both *operands of the $\cdot_p$ operator (see definition 13).*

■

Evaluation Rule 4 *A purpose will not be in the SPS for the first operand, if the operator is $\cdot\neg_p$ and the purpose is present in the CSPS.*

■

Finally, determining if a set consisting of multiple purposes (a set from RS) will be in the SPS for an expression the following generalization can be used:

Evaluation Rule 5 *A conjunction of purposes will be in the I-Set of another purpose, if all purposes in the conjunction dominate said purpose, and none of the purposes in the conjunction dominate the other.*

■

Evaluation Rule 6 *A conjunction of purposes will be in the SPS for an* or *expression if all purposes in the conjunction dominate at least one of the operands for the $+_p$ operator.*

■

Evaluation Rule 7 *A conjunction of purposes will be in the SPS for an* and *expression if each of the operands are dominated by at least one purpose in the conjunction.*

■

Evaluation Rule 8 *A conjunction of purposes will not be in the SPS for and* and-not *expression if any of the purposes in the conjunction are in the CSPS.*

■

Using the above rules, a compound expression can be evaluated by analysing each term of the expression to determine if there is a set from the RS that will be in its SPS.

Example 5 *Given the compound expression presented in example 3:* $\phi_1 \cdot_p \phi_2 +_p \phi_7$*, and the statement of intent:* $\phi_4 \cdot_r \phi_6 +_r \phi_8$*. The system has to determine if the statement of intent's RS (*$\{\{\phi_4, \phi_6\}, \{\phi_8\}\}$*) is good enough.*

It will be good enough if $\{\phi_4, \phi_6\}$*, and* $\{\phi_8\}$ *is in the SPS of the expression. This is determined using the generalizations put forward above. The system starts by examining a set from RS (since sets are unordered, it does not matter which set is chosen first). The example proceeds with* ϕ_4 *and* ϕ_6*.*

1. *Since* ϕ_4 *dominates* ϕ_1 *and* ϕ_6 *dominates* ϕ_6 *the SPS for* $\phi_1 \cdot_p \phi_2$ *will contain* $\{\phi_4, \phi_6\}$ *(based on the definition of the* $\sqcup$ *operator).*

2. *The evaluation can stop at this point for the* $\phi_4 \cdot_r \phi_2$ *statement of intent since the only remaining operator is* $+_p$ *and the system has already determined that, should the I-Sets for* ϕ_1 *and* ϕ_2 *be constructed, and the SPS for* $\phi_1 \cdot_p \phi_2$*, the set* $\{\phi_4, \phi_2\}$ *will be in it.*

3. ϕ_8 *does not dominate* ϕ_1 *or* ϕ_2*, however, it does dominate* ϕ_7*. Based on the definition of the* $+_p$ *operator, the entire* $I_S(\phi_7)$ *will be "unioned" with the SPS for* $\phi_1 \cdot_p \phi_2$*: thus the set* $\{\phi_8\}$ *will be present in the SPS for the entire expression.*

This means that all sets from the RS will be present in the SPS for the expression, and that the statement of intent is therefore suitable for the compound expression bound to the data.

■

Overall evaluation of the expression thus depends on the number of elements in the RS, and the number of terms and factors in the compound expression, as well as the effort involved in determining if a purpose dominates another purpose.

Since the dominance set for each purpose is readily available, and should be implemented as a bit-string, determining if a purpose is in the dominance set of a purpose is reduced to a bit-wise operation which should run in constant time – it could potentially run in linear time if a large set is to be represented using several bytes.

The number of elements in the RS will be relatively small (it is after all an expression to indicate the statement of intent – and it is reasonable to assume that this will be between one and ten elements). The number of terms and factors in the compound expression will be significantly less than the constructing the SPS for the expression.

The resulting complexity will therefore be $O(lk)$ complexity in the worst case with l being the number of elements in the RS, and k the number of terms and factors in the expression[1].

4.6 Summary

Compound purposes present interesting and rich ways in which privacy policies can be expressed. As such, privacy aware systems can be afforded a semantically more rich mechanism for implementing access control to PII. This chapter formally presented the notion of compound purposes and reasons that can be used to protect the PII of data subjects. The concept of negative purposes, as well as the algorithm for verification of access requests that contain statements of intent in compound-purpose form against compound purposes bound to data was formally introduced. It was also proved that the verification algorithm is sufficient to perform access control.

In addition to the above discussion, the run-time complexity of the verification algorithm was also examined, and it was shown that it adds no significant overhead to access control. This allows the efficient use of the verification algorithm when servicing an access request.

The following chapters of the text take a look at practical use of the compound purpose and reason idea by examining their potential use as part of an existing data storage and retrieval system.

[1] A term is taken to be a conjunction, and a factor is taken to be a disjunction

Chapter 5

Extensions to SQL

> If you tell the truth you don't have to remember anything.
> – Mark Twain

The rigorous definition of a new verification mechanism for controlling access to PII is rendered useless if no proposal is proffered for integrating the verification system into some existing set of systems. In this text, the identified system that may benefit from such an enhancement is the RDBMS – part of technological safeguards that an enterprise may employ in order to accomplish data protection through purpose limitation. The data user will interact with the RDBMS using the well-defined language SQL. During this interaction it is important that the data user at all times clearly indicate their intent with any PII that they attempt to access.

This rather strict approach will ensure that there is a high level of accountability when auditing the access logs of the system. No data user will be able to repudiate their attempt to access information and their intent with the information, which is easily done when a user is granted access to data because their profile had the right permissions.

It is clear from this informal introduction that it now becomes necessary to provide a means for data users to explicitly state their intent with information. In this chapter, enhancements to the SQL language (and by implication a standard RDBMS) are tabled which would provide compound purposes to an RDBMS, from the end-user perspective, that is, the compound purpose mechanism is considered from the data user's perspective, rather than from a technical implementation viewpoint.

The enhancements provide the ability to force the data user to explicitly indicate what their intent with information is, as opposed to simply being granted access to information by virtue of the correct purposes being part of their user profile. This is important if correct access control is to be exerted on the protected

information. Many legacy systems do not provide an acceptable mechanism for enforcing access control based on purpose limitation, and this chapter provides a guideline as to what such a mechanism should look like.

Specifically this chapter focuses on the way in which access to data is controlled with SQL, how access to data is requested, and how modification of data is handled. It provides a non-intrusive access control system called the Hybrid Discretionary Access Control (HDAC) (users who are not charged with privacy related tasks are not affected), and shows that the HDAC correctly enforces access control in the same fashion as the Discretionary Access Control (DAC) in RDBMSs.

The rest of this chapter is structured as follows: Section 5.1 provides a gentle introduction to the philosophy behind extending a data access and manipulation language in order to add reason specification. The process of controlling access in a typical RDBMS with the HDAC is considered in section 5.2. Section 5.3 provides some examples, and the syntactical definition for the proposed extensions. A mechanism is examined in which data users may be allowed to grant access to other users. Using the structures supplied by compounds, it is shown how the HDAC is created. An algorithm is then provided for determining whether access can be granted to the requested objects. One of the particular aims of this work is to extend an existing system in a non-intrusive manner; and section 5.4 provides detail on allowing data users who are not charged with accessing PII to request access – even in the event that they need to access PII. Section 5.5 covers the act of revoking access to data that may have previously been granted using grant statements. Some implementation notions are mentioned in section 5.6, and section 5.7 provides brief comments on extending the insert, update, and delete statements.

5.1 Introduction

Current PETs protect access to information by binding purposes to objects (data) – the specification phase, and only allowing access to data if the data user's statement of intent can be determined to be sufficient in context of the purposes for which the data was stored.

There are many examples in literature (see chapter 2) on how purposes should be associated with data, and what a statement of intent from the data user's perspective should be. Note that statements have already been made in this work regarding the nature of associating purposes with data. That is the whole premise for using compound purposes in the first place. This principle is thus not considered further in this chapter. What this chapter is concerned with is the way in which the data user's statement of intent is presented to the Access Control

Subsystem (ACS).

Current ACSs attempt to hide away the presence of privacy related decisions from the data user. Privacy is more important than this and more complex – attempting to hide this information from the data user could ultimately be a great risk. The way in which privacy related issues are hidden away from the person using the data is by simply associating a set of purposes from the DPS with the data user's profile. This profile travels with the data user, and any access request made by the user has the associated set of purposes that is typically presented (behind the scenes) to the ACS.

Recent work done on the limitation phase [14] proposes the association of purposes with a data user's profile (or with a role in RBAC). Access is granted if a data user's profile contains the "correct" purposes for the data to which they are requesting access. This method is implemented in a Mandatory Access Control (MAC) model, with a central authority deciding for which purposes data is stored, and which purposes are associated with the data user – thus no ownership of data.

It is clear that data users are now using data without explicitly stating their intent with information: the system is doing this for them. In this case, there can be no accountability. A data user may gain access to information, and use this information in no way related to the original stated purpose. There is no consequence for using the data in the incorrect manner, seeing as all data users got the information they were looking for, and did not agree to be bound by some agreement per data to which they were requesting access. In this scenario, purposes are little more than additional "access modes" that are used to control access to data. Just as the enterprise storing the data is to be held accountable for its use of the data, so too must the employee of the enterprise be held accountable for his use of the data. Thus the mechanism that allows the employee access to the data should also provide him with a way of stating his purpose with the data, allowing proper access control, and facilitating the creation of verbose audit trails.

In this chapter, extensions to the most widely used mechanism by which access is gained to data (SQL) are considered in some detail. If such extensions are to be useful they must be compatible with current SQL standards, and should not compromise the protection of data. The work here focuses primarily on the *application* of data, and thus considers the extensions in two phases.

The first phase considers extensions to the `grant` statement, which enables the implementation of an access control model which is a hybrid of DAC [42] and MAC [8] [99, 100]. This hybrid model relies on the "no ownership" of data of MAC to protect privacy, and the delegation of access control of DAC by data users to provide flexibility. It thus avoids the central reason that DAC is not considered plausible in a privacy environment [38], namely ownership of data. In the second phase, an extension to the primary data access clause, *select*, is considered.

Revoke, *insert*, *delete*, and *update* are also considered in detail to provide a

complete picture of access control within the proposed model. Revoking normal access modes in the extended model should be no different from the normal model. Revoking purposes becomes more complex, and it must be ensured that revoking a purpose from a privilege results in a more restrictive privilege.

The work here does not consider the issues surrounding the current *revoke*, such as the fact that revoking privileges does not ensure that the grantee no longer has those privileges (as they might have received them from another data user). It does, however, suggest a step in the direction of access control using purposes in current database technology.

The proposed extensions are flexible enough for implementation as they do not change the utilisation of the relevant SQL statements significantly. The extensions are clauses that can be specified when necessary or omitted – the semantics of the extensions are defined in both cases.

The primary intention is not to change the SQL specification, but to consider a possible syntax for such extensions, their semantics, and impact on protecting access to data. While it is not possible to provide a comprehensive solution to privacy issues, the extensions proposed in this text take a step in the direction of integrating PETs with existing database technologies.

It is possible to construct a "preprocessor" which manages these extensions; reading purposes from the database and doing "pre-verification", after which the extensions are removed from the original SQL statement, and the statement is passed to the RDBMS for normal access control.

Work done on the verification phase [14] using RBAC requires that granting of permissions is controlled by a central authority. LeFevre et al. [55] use query rewriting to limit disclosure – also requiring a central authority to control authorisation. In most work, a data user has a set of associated purposes in their profile. These purposes are extracted by the ACS of the RDBMS, and matched against the purposes bound to an object, and access is granted *if the data user's profile contains any purposes that are bound to the object*. Byun et al. [14] allow the specification of *prohibited* purposes to make access control more restrictive.

The concept of extending SQL is not new and has, for example, been proposed by Rosenthal et al. [86] to extend the grant statement to limit the privileges that a data user receives, allowing more flexibility in DAC in the relational environment, and ensuring that privileges are limited correctly.

5.2 Straw-man HDAC Model

In this section the process of controlling access in a typical RDBMS with the HDAC is considered in more detail. The HDAC is founded on the same principles as for the normal DAC. Data users may delegate access to data to other users, only

if the access they are granting onward is at least as or more restrictive than the access they themselves have. In compound purpose terminology this translates to stating that access may only be granted onward, if the access being granted onward relies on weaker purposes for accessing information.

Ultimately the DBA "owns" all the data in the database, and may grant other data users the right to access information, in very much the same fashion as the standard access control model does. Users that are granted access to data, will receive a set of (compound) purposes that they may present in order to access data. At their discretion they may pass these (compound) purposes onward to other data users. The only limitation will be that the (compound) purposes they pass onward must be weaker than those purposes they themselves have. This ensures that information enjoying a certain *level* of protection (as a result of the compound purpose bound to its relative suitability) will still enjoy the same level of protection even after access to it has been delegated to a different user.

In a typical setting, the following is the standard interaction that a data user will have with the RDBMS employing the HDAC (see figure 5.1 – the numbers in parenthesis refer to the following numbered list).

1. The data controller binds purposes to storage locations (effectively binding the purpose to each datum stored in that storage location). This binding provides the purposes for which the data will be used, and may be expressed in a syntax derived from the semantics discussed in previous chapters. This internal representation may be translated into a privacy policy by hand, or by an automated method [1]

2. A data owner, who wishes to subscribe to some service offered by the enterprise, provides their requested information to the data controller. The data owner adjusts the privacy agreement to suit their expectations of privacy management (see chapter 6).

3. The DBA (sanctioned by a security policy) grants certain data users the privilege to use certain purposes from the DPS during queries.

4. A data user requests access to information and states their intent with the information.

5. The ACS verifies that the data user may present the stated reasons as a reason for accessing a particular datum.

[1] In fact, since the bindings make use of mnemonics, it is reasonable to assume that verbose strings may exist describing each purpose in detail, and a policy may be generated automatically from this.

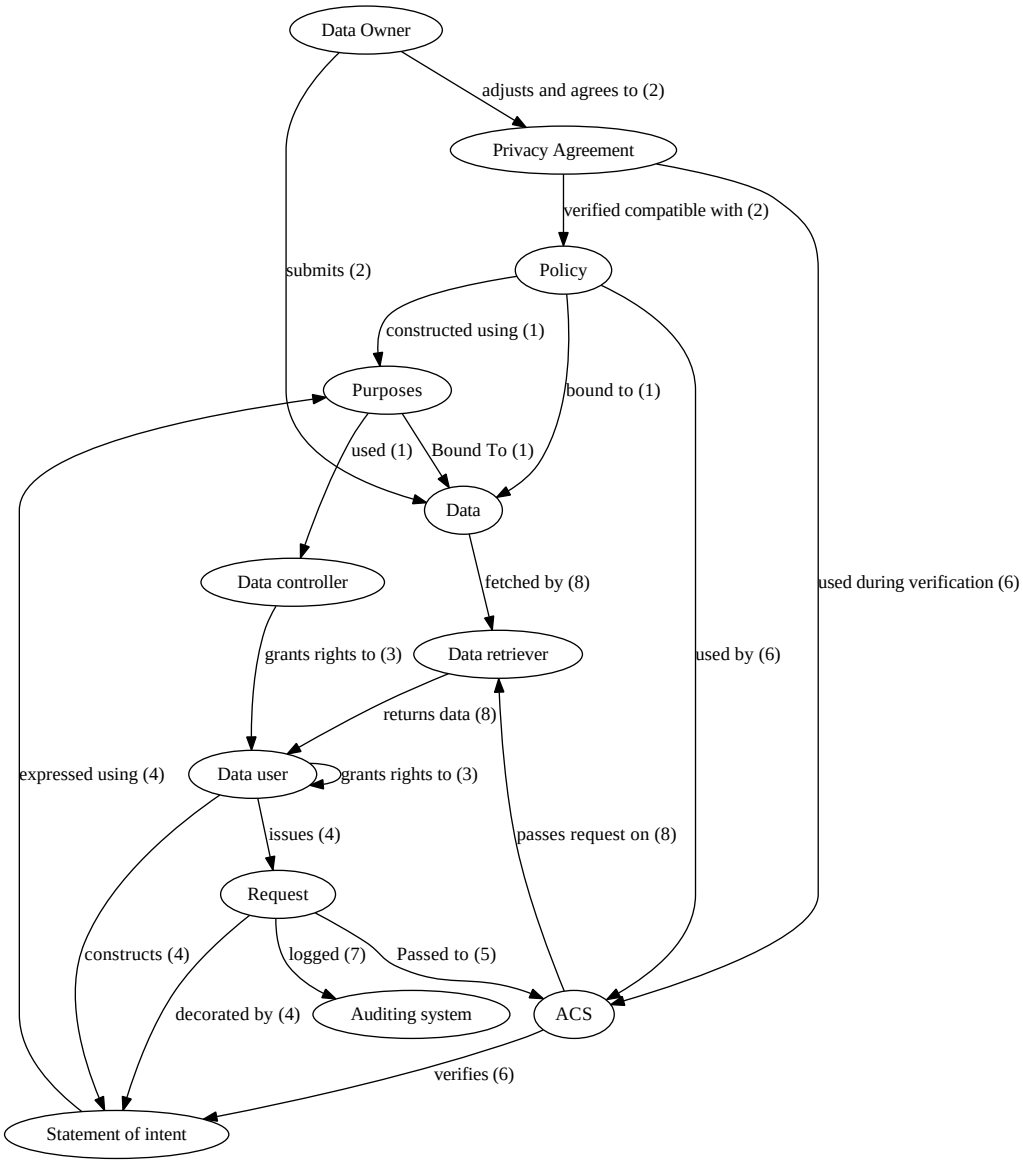

Figure 5.1: HDAC model

6. The ACS verifies that the stated intent dominates the bound purposes, and if so, passes the access request, now stripped of purposes to the RDBMS to determine if the user may perform the requested action on the data (select, insert, update, delete).

7. Access requests are logged to an auditing system.

8. If the request passes the verification by the ACS, the data is returned to the

data user.

In the following section the extensions to SQL for expressing statements of intent are delineated.

5.3 Extending SQL for Specifying Reasons

At this point, the proposed extensions to SQL are expounded. These extensions are defined to be optional subclauses of the `grant`, and `select` statements. In all cases the syntax of the extensions allows the subclauses to be omitted providing a level of ease of use. To ensure predictable and well-defined behaviour of these proposed extensions, it is also necessary to ensure that in cases where the extended syntax elements are omitted, the SQL statement is executed with expected results. To ensure this, the semantics of the extensions are defined to ensure predictable behaviour if they are omitted – in principle the default behaviour is to use the weakest purpose as a reason for accessing information, thereby ensuring that only insensitive information is released.

Even though one of the tenets of using compound purposes is that users should always specify their intent with information, the omission of these statements is supported to ensure that the mechanism remains non-intrusive, thereby allowing users that perform tasks that have no privacy impact to continue to request information as per usual.

The first SQL statement that is considered is the `grant` statement. This statement is arguably the highest impact statement with respect to protecting access to information.

5.3.1 The `grant` Statement

The classic grant/revoke model first proposed by Griffiths et al. [42, 10] creates a data user profile which stores the permissions a data user has on an object. It is important to realise that the purposes that a data user may specify for accessing data and the purposes they eventually will state as part of their access request are two separate issues to consider.

Insofar as the act of storing purposes that a data user may specify as part of their statement of intent is similar to many of the models presented here, it is important to remember that the extensions provide a way for the data user to explicitly state what they intend to do with information. It therefore adds an additional level of auditing ability. A user is allowed to use a certain set of reasons for accessing information, and states his actual intent using that set of reasons. If the data user deviates (uses purposes not granted to him) access is denied, and

an audit trail now exists indicating this attempt to use information for the wrong reasons. Stating correct reasons and then misusing the data also has severe implications, since the data owner and data controller now have an audit trail showing that a data user was up to mischief with their information.

The extended `grant` statement presented here is intended to add only two pieces of information: the reasons that the grantee (data user) may present for accessing the particular object forming part of the grant statement, and the reasons for which the grantee may grant onward (definition 19). Keeping with the notion of privacy protection, the new grant statement also allows a data user to indicate their reasons for granting access to another user.

Definition 19 (Extended GRANT) *GRANT <privileges> [for_1] on <object> to <subject>* [*with grant option* [for_2]][for_3]

With for_i = for $< reason_1, reason_2, \ldots, reason_n >$.

■

The first of these *for* subclauses (for_1) is all the reasons for which the grantee may access the specified object. The reasons that can be specified here by the granter depend on the reasons afforded to him.

The second *for* subclause (for_2) is associated with the *grant-option* clause. This indicates the reasons for which the grantee is allowed to grant onwards. The third *for* subclause (for_3) is the reasons for which the granter is granting onward. Note that the granter's reasons for granting onward should be more specific than (dominate) those he received.

Also note that for_1 is independent of $for_{\{2|3\}}$, but that the grantee's for_1 must at least be as restrictive as the granter's for_1 (see 5.3.2).

Every *for* clause is a comma separated list of reasons for which an action may take place (accessing an object, or granting onward).

5.3.2 Granting Onward

The grant option in the DAC model allows a data user to grant other data users rights on objects that they themselves have access to. These rights must be as restrictive as the rights of the granter.

The extended grant statement ensures that the grantee does not receive more access than the granter – it ensures that the grantee's reasons for accessing and for granting onward are sufficiently restrictive.

5.3.2.1 Granting reasons for granting onward

A grantee's grant-onward reasons are restricted by ensuring that they are more specific than the granter's grant-onward reasons. Suppose RS_2 is the reason

that σ_1 received from σ for passing on as part of a grant option. If σ_1 issues a grant statement, passing the grant option to σ_2 as: `GRANT` π `FOR` RS'_1 `ON` ω `TO` σ_2 `WITH GRANT OPTION FOR` RS'_2 `FOR` RS'_3`;` Then it is required that $RS'_2 \leq RS_2$. It is obvious that $RS'_2 \leq RS_3$ must also hold.

5.3.2.2 Granting reasons for accessing objects

More sensitive objects will be protected by making use of reasons closer to the Least Upper Bound (LUB) of the purpose lattice, as these purposes are more restrictive (specific). Thus, in order to adhere to the grant statement's "narrowing" property, granting reasons onward requires that those reasons that are granted are dominated by the reasons the granter has.

Theorem 6 *(Grant Onward)*

If, for every reason R_i that a granter passes onward he has a reason R_j such that $R_i \leq R_j$, the grantee's access will not be less restrictive than the granter's.

■

Proof 6 *(Grant Onward)*

Suppose σ is granting reasons onward on ω, and that ω_P is the (compound) ***purpose*** *bound to ω.*

Furthermore, suppose RS_1 is the reason that σ received, and that RS'_1 is the reason he wants to grant onward.

If σ is only able to access portions of ω for which $RS_1 \geq \omega_P$, then clearly if $RS_1 \geq RS'_1$, σ_1 will only gain access to portions of ω for which σ has access, or less.

Thus, if $\forall R_i \exists R_j, R_i \in RS'_1, R_j \in RS_1$ such that $R_i \leq R_j$, then $RS_1 \geq RS'_1$, and the grantee will never be able to access information that the granter does not have access to, or for reasons other than those which the granter has.

■

5.3.3 Accessing Objects

Access to objects can occur by either using an SQL Data Manipulation construct or by executing a procedure. Since the focus of this work is on protecting data through compound reasons and purposes, the protection of a procedure using reasons and purposes does not really fall in the scope of the work. However, some comments are provided. Firstly, since the procedure will be a series of SQL Data

Manipulation statements which should each be decorated with a statement of intent (and subject to reason inference when the statement is omitted) limiting access to data is still enforced. Secondly, it is possible to extend the procedure execution syntax (`exec someProcedure ...`) to include a statement of intent *viz.* `exec someProcedure <params> FOR` for_1. The parser would then replace all the for_1 clauses in the select, update or delete statements with the statement of intent specified as a clause in the execute procedure statement.

There are, in essence, two problems with extending the execute procedure syntax as put forward in the previous paragraph. Firstly, procedures are normally precompiled to enhance performance of the query engine – the changes may require recompilation of the procedure each time, rendering the solution worthless. Secondly, the procedure may have been written with specific statements of intent in mind, and ad hoc changes may deviate from the original intent of the procedure. Even so, stored procedures may be an interesting extension, but will not be considered further here.

Requesting access to data through the normal constructs results in an access verification for every single construct that is found as part of the statement (query). This makes it possible to nest queries without having to redesign and implement the ACS. It is possible to merge certain types of queries [79], but this is done largely for faster query execution, and is thus not considered here. It is assumed that queries can be considered in isolated form.

Every access request requires that the data user provide their reason for accessing data. In the same fashion as the grant statement, the data access statements are augmented with an optional *for* clause, but there is an important difference between the grant statement and these statements.

Data access queries may include references to various (base or virtual) tables, and columns from these tables. Each one of these objects could have a purpose associated with it. Thus the data user must specify a reason for accessing each object, which is accomplished by having the *for* clause provide a list of "key=value" pairs. Each pair consists of an object (key) and the reason for accessing that particular object. The data access statement is thus defined as per definition 20.

Definition 20 (Extended SELECT) *SELECT* θ *[for* $< o_1 = r_1, o_i = r_2, \ldots >$*]*

Where θ *represents the body of the statement,* o_i *is an object that is known and protected by the RDBMS, is present in the body of the select statement, and* r_i *is a reason presented for accessing* o_i*.*

Suppose a data user wishes to get the names of all the customers in the database for "Public relations" purposes. The query is presented as (for example purposes it is assumed that purposes are presented as strings): "`select name from cust for <name="PR", cust="PR">`"

It is possible that a data user could request access to several objects. Specifying a reason for each explicitly can become tedious, so it is possible to omit an object from the list – with the specific understanding that the ACS will automatically associate the GLB of the purpose lattice as the reason for accessing that object, thereby assuming the data user is accessing the object for the most general reason. This automatic association also serves the purpose of protecting access to information on a least privilege basis. If a data user specifically omits their reasons for accessing a certain object, then there is no "fall" through event which would cause them to gain access to that object: access to every object must be qualified.

If several objects are accessed for the same reason, the special keyword "default" is introduced. A data user can use this "object" to specify their default reason for accessing objects. For example, in the query "`select name from cust for <default="PR">`" the ACS interprets the access request to mean that `name` and `cust` are accessed for the reason "PR". If the "default" object is not set explicitly, the greatest lower bound of the purpose lattice will be associated with it (see 5.4).

5.3.3.1 Reason Inference

Subjects often access a table and columns from that table for the same reason. The data user can omit the reason for accessing the table, and the ACS can infer this reason by examining the reason specified for the columns.

If multiple columns from the same table are accessed, a compound reason for the table (consisting of the reasons specified for the columns of the table) is formed by using the **and** operator (defined in Van Staden and Olivier [99]).

In the same way that SQL requires each object in a SQL query to be uniquely identifiable (using fully qualified names in case of ambiguity), the ACS [91] must be able to associate a reason with every object. Fully qualified names should be used in cases of ambiguity.

5.3.4 Controlling Access

A simple algorithm is presented in listing 5.1, which can aid in the verification process based on the extensions that have been given. Before the algorithm is provided, it is necessary to define several auxiliary functions.

1. *reason(o)*: returns the reason that is associated with an object as specified in the query; if no reason is associated, the *null* value is returned. *null(o)* returns true if o is a null value.

2. $and_r(r_1,r_2)$: forms an **and** conjunction of the reasons r_1 and r_2.

Listing 5.1: Algorithm for determining access

```
1  PDACS(s, Olist)
2    default = def(Olist)
3        result = true
4        for o in Olist
5          if null(reason(o)) then
6              r = default
7              for o2 in getColumnsOf(o, Olist)
8                  if null(reason(o2)) then
9                      r = and_r(r, default)
10                 else
11                     r = and_r(r, reason(o2))
12                 endif
13             endfor
14         else
15             r = reason(o)
16         endif
17         result = result & verify(s, o, r)
18       endfor
19       return result
20     end
```

3. *getColumnsOf(o, Olist)*: returns a list of all the objects in the list *Olist* which are columns of the first parameter *o* – if *o* is not a table, an empty list is returned.

4. *verify(s,o,r)*: returns true if *s* can access *o* for reason *r*.

5. *def(list)*: will return the "default" value if it is defined in *list*, otherwise it returns the GLB of the purpose lattice.

The Purpose Driven Access Control Subsystem (PDACS) function takes a list of objects (specified in the query) and the ID of a data user. It finds the default reason, iterates through the list of objects, and attempts to find a reason for every object in the list. If no reason is found for an object, it is assumed that the object is a table. In this case PDACS will try to infer the reason for accessing the table by finding the columns of that table that are listed in the query, their associated reasons, and constructing a compound reason for the table.

During each iteration verification for a specific object takes place, and is "anded" with past results. Any "false" returned by *verify* will thus result in ac-

cess being denied completely.

5.4 Omitting the *for* Clauses

Since the *for* subclauses are optional, users may continue using the `grant` and `select` statements as per usual. In those cases where any of the *for* clauses are omitted the ACS modifies the statement to include the *for* clauses with *default* values. As stated previously, this ensures that access control takes place in a well-defined way. Unless otherwise specified (by using the *default* keyword) the default reason used when the *for* clauses are omitted is the weakest purpose in the PL. This will force data users to present a statement of intent, since they will only gain access to insensitive data.

Default values for `grant` can be inferred from the reasons associated with the granter. Where a granter has provided no specific reasons, the greatest lower bound of the purpose lattice is substituted.

Suppose data user σ is granted access to object ω with grant statement (in which RS_i represents a list of reasons): `GRANT SELECT FOR` $RS_{1:\sigma}$ `ON` ω `TO` σ `WITH GRANT OPTION FOR` $RS_{2:\sigma}$ `FOR` RS_3`;`

If σ wishes to grant access on ω to data user σ_2, he can issue the grant statement: `GRANT SELECT ON` ω `TO` σ_2 `WITH GRANT OPTION;` The ACS will insert the missing *for* clauses in the granter's statement, based on those given to the granter, changing the statement to: `GRANT SELECT FOR` $RS_{1:\sigma}$ `ON` ω `TO` σ_2 `WITH GRANT OPTION FOR` $RS_{2:\sigma}$ `FOR` $RS_{2:\sigma}$`;`

In the case where the for clause of the `select` is omitted, the ACS associates the most general purposes in the purpose lattice with the "default" object. For example, "`select name from cust;`" is changed to "`select name from cust for <default="noreason">`" (provided that the most general purpose is "noreason").

5.5 Revoking Privileges in the HDAC Model

In this section the action of revoking privileges from data users in the HDAC model referred to previously and elsewhere [100, 97] is considered in more detail. Since the model presented here fits closely into standard RDBMS technology, revoking is accomplished in the same fashion as in current RDBMSs, through the use of the SQL *revoke* statement. Revoking privileges are afforded to data users by extending the *revoke* statement to handle the added attributes in the HDAC model. Whereas the standard *revoke* only removes access modes (read, write, modify, and *delete* [29]), and special permissions to grant privileges onward, the revoke state-

ment here is necessarily more complex. This complexity is a direct result of the additional information that a *grant* statement adds to the access control mechanism. This additional information has to be considered carefully when revoking privileges from a data user.

Not surprisingly, one would expect the extended *revoke* to allow the revoker (the data user revoking a privilege) to remove all the privileges that the standard *revoke* does. In addition to this, it must also allow the revoker the ability to revoke two privileges: firstly, the privilege of presenting certain purposes when requesting access to information is to be revoked, and secondly, it must allow the revoker the ability to revoke the privilege of presenting certain purposes when granting privileges onward from the revokee.

A final note about the aims of the extended *revoke*. Firstly, it must allow a data user, who is not concerned with privacy related data [2], to use the *revoke* as per usual. Thus, *revoke* is well defined for cases where purpose privileges are omitted from the *revoke* statement. Secondly, it must avoid dangling privileges. In other words, when a privilege is revoked from a data user s, who had a grant option, then that privilege must be revoked from all data users the received the privilege from s.

Before considering syntactical extensions to the *revoke* statement, the actions of a *revoke* statement are considered in terms of privileges. The removal of purposes from purpose expressions that form part of a data user's privilege set is considered before revoking the standard privileges (access modes and grant options).

5.5.1 Revocation Semantics

The accepted standard definition for any revoke action in the RDBMS model is to remove given privileges in such a way that it appears as though the target of the revoke action had never received the given privileges to begin with. This implies two important facts. Firstly, a revoke action need not concern itself with finding reasons that dominate the revoked reason and removing those from the data user's perspective. The reason for this stems from the fact that a data user is only allowed to use the reasons specified during the grant. The data user is never afforded the right to use purposes or reasons that dominate the granted purposes (reasons).

Secondly, revoking privileges will almost always take place in a scenario involving a chain of grants that has been received. To this end, the revoke action must ensure that removal of a privilege leads to a state which is equivalent to the data user never having received the privilege being revoked, and that the act of revoking does not provide the data user with more privileges than they initially

[2] For example, a data user working with data that does not relate to PII at all.

had.

The remainder of this section will cover those two important aspects in greater detail. However, before embarking on this discussion, some foundational work is needed to understand the way in which the listed goals are to be met.

The important fact to remember at this point is the meaning behind the revocation of a particular purpose. There is a definite difference between a revoke statement such as "data user may no longer use ϕ_i on its own", and "data user may no longer use ϕ_i at all". Consider, for example, a data user with the privilege to use ϕ_i on its own but also the privilege to use $\phi_i +_p \phi_j$. A statement which will revoke ϕ_i, will only remove ϕ_i from the data user's allowed purposes that may be presented. However, since the data user may still present $\phi_i +_p \phi_j$, they may potentially be granted access to information that they should not have access to (based on this privilege, the data user may use the data for reason ϕ_i or ϕ_j). It is intuitively clear that a principle of "saving the data owner" should be employed, and the privilege $\phi_i +_p \phi_j$ should be modified to prohibit the data user from accessing the information.

In the following section, the purpose expression form is considered in some detail to understand how the removal of purposes from a set of privileges may be accomplished.

5.5.2 Purpose Expression Form

It is important to note that the operators for compound purposes are defined in such a way that it is easy to represent the purpose expressions in a simple format as part of the data user's profile (the fact that this may be done in a bit-string form was already alluded to in section 4.5). The significant aspect of this representation is that it will be easily to manipulate in set forms.

Before considering removal of purposes in general, the removal of a singleton purpose from a purpose expression is first considered. This provides the basis for the general removal of a purpose from a purpose expression. The notation for indicating that a (compound) purpose C has to be removed from a purpose expression P, is written as $P \oslash C$.

5.5.2.1 Removing Singleton Purposes

Removal of a singleton purpose from a purpose expression is in essence extremely simple given the proposed set representation of the purpose expression. This removal is presented more formally in definition 21.

Definition 21 (Removing a Singleton from an Expression)
Suppose the singleton purpose ϕ_i is to be removed from E. If E is considered in its set form (written as E_s, and recall that E_s is a set of sets), then

$$E_s \oslash \phi_i = \{Y | Y = X \setminus \{\phi_i\}, \forall X \in E_s\} \tag{5.1}$$

■

5.5.3 Removing Compounds

Removing compounds relies on the same technique as removing singleton purposes: an expanded expression that is represented as a set of sets (E_s). The main difference in technique now lies in the fact that the item to be removed may itself be a purpose expression. The expression to be removed from E (called E_R) has to be expanded and placed in set representation as well, E_{R_s} (also represented as a set of sets). Compound purpose removal is presented more carefully in definition 22.

Definition 22 (Removal of Compounds from Expressions)
Suppose a purpose expression E_R is to be removed from E. E and E_R are converted to set representations, E_s, and E_{R_s} respectively. Then

$$E_s \oslash E_R = \{Y | Y = X \setminus Z, \forall X \in E_s \wedge \forall Z \in E_{R_s}\} \tag{5.2}$$

■

5.5.4 Valid Revokes

The removal of one expression from another is the important mechanism for managing the *revoke* in the HDAC. A second critical aspect is ensuring that a revoke leaves the system in a valid state. This implies that the removal, although legal with regard to the definitions, might cause an incorrect resulting purpose expression. This incorrect expression could provide the data user with more access than they initially had. This means that their access mode rights must at least be as restrictive as the revoker's, and that his purpose privileges (for both granting and accessing data) must be at least as restrictive as the revoker's.

Suppose data user A received privileges ϕ_1, and passed on $\phi_2 +_r \phi_3$ to data user B, and ϕ_1 to data user C. Suppose these privileges were all legally granted onward with the *grant* as depicted in figure 5.2.

For a revoke to be valid, the state of the system depicted by the grant graph must still be such that any privileges (purposes) revoked cannot leave the system in a state in which grantees have greater access privileges than granters. A system that satisfies this requirement is said to conform to the *least privilege property*.

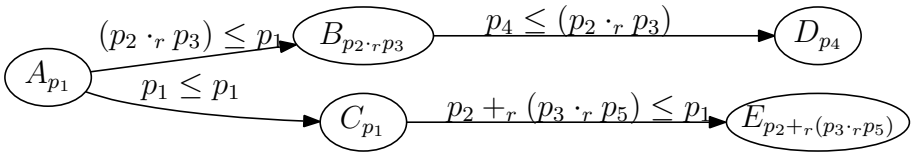

Figure 5.2: Grant graph in the HDAC

Definition 23 (The Least Privilege Property) *Any purpose privileges received by a data user B from data user A must be such that data user B's privileges are at least as restrictive as data user A's. In other words, if data user A had privileges α_a and granted $\alpha_{a \to b}$ to B, then $\alpha_{a \to b} \leq \alpha_a$ must hold.*

However, in the same fashion in which granting onward may cause the illegal flow of information, removing a purpose from a purpose expression during a revoke will not guarantee this property to hold. This is purely because the removal of a purpose from a purpose expression may fundamentally change the semantics of that expression to such an extent that information that was previously inaccessible now becomes accessible.

Theorem 7 (Invalidity of Removals) *Removing a purpose from a purpose expression during a revoke does not guarantee the that the least privilege property will hold.*

■

Proof 7 *This can be shown to be true in a simple example. Consider a purpose privilege from figure 5.2 $\phi_2 \cdot_p \phi_3$. Suppose data user B no longer wishes user D to be able to present ϕ_3 when accessing data, and decides to revoke this privilege.*

The resulting privilege now allows D to access certain data with less restriction than B, since $\phi_2 \not\leq \phi_2 \cdot_p \phi_3$. In fact, it is also true that $\phi_2 \cdot_p \phi_3 \not\leq \phi_2$.

The result is thus undefined, and the data user will now be able to gain access to data other than that which was intended.

■

This failure could result in a violation of privacy for all data owners in the database. To prevent this access violation, the revoke constraint is introduced (definition 24). This constraint will ensure that a revoke is only effected in the system if it can be verified that the privileges resulting from the revoke does not violate the least privilege property (definition 23).

Definition 24 (Revoke Constraint) *A revoke can only take place if the result of the removal of purposes E_r from the purpose expression E is at least as restrictive as the revoker's purpose privileges E_{rev}, thus $(E \oslash E_r) \leq E_{rev}$*

■

The revoke constraint will ensure that the state of the system is consistent with the original DAC model from a grant/revoke perspective. The only difference is that the user issuing the revoke statement may now get a warning or error message that indicates that the revoke was not successful.

Now that the operation of the revoke statement has been presented, as well as the caveats surrounding its usage, the following section deals with the extended revoke syntax for SQL.

5.5.5 The Extended Revoke Syntax

The revoke statement must thus allow the revoking of two things: firstly, it must allow a revoke of either a complete privilege (such as *delete*, *select*, or *update*), or a sub-reason, and secondly, it must allow the revoking of the grant option from a data user. The SQL 2003 draft standard [63] is used as a basis for the syntax of the extended revoke statement[3] (definition 25).

Definition 25 (Extended Revoke Syntax)
<revoke> ::= REVOKE [GRANT OPTION [FOR <for$_1$>] FOR] <privilege> [FOR <for$_2$>] [, <privilege> [FOR <for$_3$]] [, ...] FROM <grantee>

■

Based on the new definition introduced here, the following can be accomplished:

1. By omitting for$_1$, the revoker can completely revoke the GRANT OPTION from the revokee. From 22 it is clear that $E \oslash E = \emptyset$, indicating that no reason may be presented to perform the desired action; in such a case the privilege is removed completely.

2. If for$_1$ is a valid compound reason, then definition 22 can be used to modify the reasons for which the revokee may grant onward.

3. If for$_{2..n}$ is omitted, then all the reasons for accessing data using the specified privilege is revoked, thus the privilege itself is revoked.

4. If for$_{2..n}$ is a valid compound reason, then as with the GRANT OPTION statement, definition 22 can be used to modify the reasons for which a data user may use with a particular privilege.

[3] As far as could be determined the portion of the revoke statement that is used has not changed between the 2003 draft, the 2003 final, and 2008 final versions.

When omitting the *for* clauses the statement in question is rewritten to include the *for* subclauses, with the reason specified to be "default=α", where α is the greatest lower bound of the purpose lattice. This ensures that the data user cannot gain access to data that is not protected by more specific reasons, and thus forces him to actively specify his reasons for manipulating the data.

Reasons that can be listed in for_1 from the revoke statement do not ever access PII. They are used to indicate for which purposes a data user may grant access onward. As such, they can thus be removed from for_1 without restriction, that is, the least purpose privilege property does not apply as with $for_{2..n}$.

Based on the revoke restriction, the following becomes apparent: if data user A granted data user B a specific access mode on data using the extended grant statement for a specific purpose (for ease of discussion read access is assumed), and now wishes to revoke those purposes (either just some of them or all of them), then it is clear that $(B_{for_1} \oslash for_2) \leq A_{for_1}$ must hold. Where A_{for_1} is the reasons which A may present when accessing the data, and B_{for_1} is the reasons B may present when accessing the data.

5.6 Implementation Thoughts

Even though this work does not concern itself with an implementation at this stage (the questions that drives the work presented here – see section 1 – is answered in full by the defined model), some implementation thoughts on the meta-data for the HDAC is provided in this section.

The extra access privilege attributes that are added to the data user's profile in the hybrid access control model demands a change in the table that is used to store access profiles. This change does not require a complete redefinition of the profile table. It only requires that some of the value types change.

The standard model will typically store a time-stamp, the granter's ID, the object to which access is granted, the access mode granted, and a *grant* option value. The access mode in the standard model allows the profile to indicate whether the user can *read*, *update*, *insert*, or *delete* values from the table, and is typically a simple array that indicates which access modes the data user has. The grant option is a simple true/false value which indicates if the data user may grant onward.

The hybrid model (naturally) requires more information to be stored. Only the access mode and grant option columns need modification. The access mode is changed from an array that stores access modes to an array that stores tuples. These tuples indicate the access mode and the purposes that may be presented on those access modes. For example, an access mode entry may look like this: $(read, \phi_1 \cdot_p \phi_2 +_p \phi_6)$. Currently there would seem to be very little benefit for access modes other than *read* and *delete* to have associated purpose privileges (see

section 5.7), but an implementation as presented here leaves room for extension if needed.

The grant option column is also changed into a simple array which stores the reasons that may be presented when granting onward.

After the revoke has been effected on a data user, it is necessary to revisit each of the privileges that the said data user had passed on and to revoke those privileges from the recipients, provided that those rights were not granted by another user. This approach is exactly the same as the normal revoke, and a detailed discussion is therefore unnecessary.

5.7 Insert, Update, and Delete

The proposal for extensions to *grant*, *revoke*, and *select* are a direct requirement of allowing the database and database users to handle PII more carefully. An obvious question to ask would be: should the other data manipulation statements be extended in the same fashion?

The primary goal of using reasons to control access to the database is to control the application of data; it is arguable whether or not updating data is really an act of data application. However, for completeness sake, it is conceded that the enterprise can use meta-reasons for audit purposes, and therefore trivial extensions to the *insert*, *delete*, and *update* statement are considered briefly.

5.7.1 Insert

The purpose of the insert statement is to store data in the database. The data being stored is, of course, data on some data owner. From a privacy viewpoint, the enterprise storing the data states its reasons for doing so as part of its privacy policy. Thus, any data user inserting data into a table which contains PII does so under the directive of the privacy policy. Thus, each datum is stored for its bound purpose determined by the privacy policy of the enterprise, and a reason need not be given for storing each datum.

It may be necessary for internal auditing purposes to hold the data user accountable for his action of inserting the data into the database. In such a case it is beneficial to extend the insert statement in a minor way to allow the data user who is inserting the data to specify his reasons doing so.

It is proposed that these reasons will form a subgraph of the purpose lattice which is not intended to be used as part of a general privacy policy as published by the enterprise. This is because these reasons for inserting are for a clear audit trail only and are therefore very specific. It does not, for example, make sense to publish a privacy policy which states that telephone numbers are stored so that

they may be deleted at some point – this type of statement clearly falls in the realm of obligation management, which is not considered in this text.

For sake of completeness a small extension to the insert which allows a data user to indicate his reason for inserting data is presented. This statement may be stored as part of an audit log.

Definition 26 *<insert statement> [for reason$_1$]*

■

5.7.2 Update

The *update* statement can be considered in the same light as the *insert* statement. Data of a personal nature will only need updating in order to have the most up to date version of it. Thus, there is only a single reason to update data, and as such it is not worth extending the *update* statement.

With regard to updating the privacy agreement between the enterprise and the data data user note the following: From work reported on elsewhere [101] (also see chapter 6) it has been proposed that an enterprise stores a minimum as well as a maximum purpose for storing data. The data owner provides a custom acceptance level for his data. This custom acceptance level falls somewhere between the minimum and maximum purposes. The data owner's data can only be accessed if a reason suitable for the custom acceptance level is provided.

Updating this custom acceptance level allows the data owner to change his agreement with the enterprise. Again, this is viewed as updates to have the latest profile of the data owner.

5.7.3 Delete

The *delete* statement seems the only logical choice when considering a data manipulation statement for extension (aside from *grant*, *select*, and *revoke*). Data may be deleted for many purposes, for example obligation management, that is the retention deadline for the data has been reached. The law might require the data to be deleted, and so forth.

A small change in the *delete* statement is consequently presented, and it is envisaged that it may be desirable to protect PII by stating for which purposes it may be deleted. A data user must thus specify why he is deleting information. In the event that the information is not protected by "delete purposes", it may just be a good way of providing an audit trail.

Definition 27 (Delete) *<delete> ::= DELETE FROM <table> [for <reason>] where <search condition>*

■

The above definition should also be compatible with the notion of omitting the *for* clause: the least upper bound can be used as a reason for deleting information.

Also note the following: the *where* subclause of the delete statement will access data. In view of this it is important to consider if a reason has to be specified for accessing the data in the *where* subclause.

If a reason for accessing the data listed is the *where* subclause, it may be argued that a data user can launch an inference attack on information he cannot necessarily read. However, since the delete statement is destructive in nature, his actions will appear in a simple audit: many customer records will disappear in his attempt to gather information.

An extension that requires a reason for accessing data specified in the *where* subclause is therefore not proposed.

5.8 Summary

In this chapter extensions to the de facto standard language for accessing and storing information for enabling privacy aware RDBMSs were considered. The extensions are useful since they allow the protection of PII using compound purposes. This protection is afforded by forcing data users to explicitly state their intention when accessing information.

The proposed process of forcing data users to explicitly state their intent with data also allowed the creation of a hybrid access control model for protecting PII. In this model (HDAC), data users can pass access control on to other data users without violating privacy principles. Specifically, data users cannot pass on more rights than they receive, and the lattice that is employed to manage purposes, and their relationships to each other ensures that PII will never be used outside of the terms (see chapter 6) as agreed to by the data owner and the data controller.

The proposed extensions covered access to data through the *select* statement. When using this command the data users must specify their intent with each object of information that they are attempting to access. Through the *grant* statement data users may at their discretion pass access rights on to other data users without a central authority controlling the passing on of access rights. The grant statement allows a data user to state which reasons the recipient may use when accessing information. Special care is taken to ensure that the granter does not (and cannot) pass on more access rights than he himself has.

Finally, to complete the model, since granting access is part of the model, it also includes the ability to *revoke* previously granted rights. The act of revoking rights also ensures that there are no side effects which would allow the data user whose rights are being revoked more access. It is necessary to ensure this because of the semantics of compound purposes and reasons.

Extensions to the *update* statement are not considered desirable at this point; however, the proposed implementation allows these extensions to be added at a later stage.

Extensions to the *delete* statement were considered briefly in order to establish a foundation for possible future work. The extension to the *delete* statement is only considered to be a "nice to have" and not essential to the operation of the HDAC model.

Extensions to the *insert* statement were considered briefly in order to provide an auditing device. It is of course possible to use the insert statement to indicate a custom acceptance level as defined in Van Staden and Olivier [101].

In order to minimise impact the *revoke* statement can be used as usual, without utilising the extensions. In this case it is assumed that all privileges are revoked from a data user. This allows a form of backward compatibility.

The impact of using these extensions is low, as the clauses can be safely omitted with well-defined results, allowing data users to continue using the SQL statements as usual.

The following chapter examines the use of compound purposes and reasons to facilitate the creation, and management of privacy agreements.

Chapter 6

Privacy Agreements with Compounds

"You said they'd be left in the city under my supervision!" – Lando
"I have altered the deal. Pray I do not alter it any further." – Vader

Responsibly controlling PII creates an undeniable requirement that some trust relationship must exist between the data controller and the data owner. This trust relationship can be embodied as a privacy agreement between the controller and owner – a verbalising of the understanding between two parties, so to speak.

The agreement allows the data owner to exercise some control over his PII and enables the data controller to use PII in a fair manner which could ultimately lead to an image of fairness in the marketplace, resulting in better relationships with customers, and numerous other benefits (a cynical view may be that the data controller avoids a run-in with the authorities as a result of "playing fair").

In this chapter *privacy agreements* as defined by Oberholzer and Olivier [69] (also see section 2.4.4) are considered from a compound purpose and reason perspective. The original proposal of Oberholzer is an elegant way of constructing agreements, and it allows the user to have a "strong presence in the data controller's hallways" as it were.

The original proposal of *privacy agreements* can benefit from compound purposes by virtue of the PL employed in the compound purpose model, as well as the ease with which new and flexible purposes are constructed when using the compound operators. These benefits manifest as the freedom to avoid suspending (freezing) privacy agreements whenever the data controller's environment changes. Freezing a privacy agreement results in renegotiation of the agreement before service delivery by the data controller can be provided.

Another benefit is that the user is afforded the opportunity to dictate the relative suitability (see section 3.4.1) of purposes that must be presented when access

to his information is requested.

To understand how compound purposes may be used to augment privacy agreements, *privacy agreements* as described by Oberholzer and Olivier are examined in more detail (section 2.4.4 provided a short summary). After the discourse on privacy agreements, compound purposes are applied to privacy agreements, and the benefits stated in the previous paragraph are examined.

The rest of this chapter is structured as follows: section 6.1 provides the promised detailed background information on *privacy agreements*. Sections 6.2 and 6.3 introduce and explain the concept of acceptance levels which are used by the data owner to restrict access to his data. Section 6.4 explores the invalidation (freezing) of agreements when the parameters of the agreements are adjusted, and argues that it is not always necessary to freeze agreements.

6.1 Introduction

The conducting of day-to-day business for many enterprises requires the use of PII. As an example, consider a Financial Institution (FI) such as a bank. Banks require data on all of their clients, and cannot conduct business without PII. Even so, the collection of PII demands that the FI act responsibly. In order to engender trust, in general, a data controller will publish a privacy policy to state its intent with the collected data – nearly all web-sites have a link to their privacy policy somewhere on their website. By now, this text has firmly established that a data controller must only use PII for the purposes published in its privacy policy.

In many cases, however, data controllers publish privacy policies to ensure that they are following due diligence. The content of the policy is extremely draconian. The data controller publishes a policy that will benefit it more than the data owner. The data owner, on the other hand, wishes to exercise maximal control over his collected data, resulting in a natural conflict of interest.

The content of the privacy policy resembles a “do-or-die” approach to data collection: “either accept our terms or go away”. Ostensibly this approach has caused many a potential customer (particularly privacy fundamentalists) to simply avoid doing business with data controllers such as this.

It is easy to see that a data controller that wishes to engender trust will voluntarily subscribe to the guiding principles as outlined by a governing or watchdog body such as the OECD. However, in spite of their good intentions, their privacy policies will still most likely be structured in such a way as to benefit them exclusively – resulting in the “do-or-die” approach as was just highlighted.

Oberholzer and Olivier [69] propose the use of privacy contracts which categorise agreements in one of four levels: levels 0 to 3. The reasons for the distinction in levels is based on the mandatory and optional nature of purposes – some are

mandatory, i.e. the data controller cannot conduct business if the data protected by the mandatory purposes is not agreed to by the data owner, and some are optional – data that is not essential to the day-to-day running of the data controller's business.

They also argue that data is related to transactions, and uses the term "transaction" in favour of "agreement". The chapter continues in the same fashion.

Level 0 transactions are considered mandatory transactions and will be included in every privacy agreement between the data controller and the data owner. Data owners must agree to these transactions (the data being collected, and the purposes for the data). The data controller is saying that the data is required for the stated purposes in order to conduct day-to-day business. These types of transactions are what is referred to as "do-or-die" – specifically with regard to agreeing to the policy or not. A responsible data controller will typically be extremely careful with the data and purposes that are part of level 0 transactions.

Level 1 to 3 transactions are considered optional transactions. Each level varies with in the actual customisation that can be done by the data owner. The data owner need not agree to any of the transactions in the optional levels, and the data controller is guaranteed that the data required for business activities are accessible already. The data owner will typically define the agreement between himself and the data controller on one of the said levels (including the mandatory level 0).

Level 1 transactions support those individuals who are only marginally concerned with privacy – transactions can only be agreed to based on mandatory purposes for the transactions. This also reflects the so-called "do-or-die" approach, but only for the purposes specified for data use.

Level 2 transactions allow the user to stipulate the purposes for which data can be used, and the data controller may only use the data for the purposes defined by the data owner. This level supports the privacy pragmatist in setting a privacy agreement with the data controller.

Level 3 transactions cater specifically for the privacy fundamentalist. Privacy agreements at this level allow the data owner to state the data as well as the purposes for which the data may be used. This level provides an advanced configuration choice for the more technically inclined.

The concept of privacy agreements is again very nearly dependent on absolutes: on the one hand, level 0 agreements are mandatory, and the data owner must agree to the purpose for which the data controller will store the data. On the other hand, level 3 agreements provide a lot of flexibility in specifying what data may be stored, and the purposes for storing them.

By using the PL, it is possible to augment the Oberholzer privacy agreement. Instead of specifying mandatory agreements and free-hand configurations, the data controller uses the PL as a tool in defining the most general purpose it may

have for data, as well as the most specific purpose. The data owner, who acts responsibly with his data, may not agree with the most general use for data as specified by the data controller. He chooses his own minimal purpose from the PL for which the data controller may use his data.

Unfortunately, the data owner may also be very strict and stipulate that his data may only ever be used for the most specific purpose (element 1) in the PL. For the data controller, this may mean that conducting day-to-day business with the data owner's data is no longer possible. To ensure that this scenario is avoided, the privacy aware system that employs privacy agreements with a PL ensures that the data owner cannot specify purposes that dominate the most specific purpose as was defined by the data controller. In this way, as long as the data owner's defined purposes fall between the most general and most specific purposes as specified by the data controller, the data controller can conduct business, and the data owner is assured that his data is not being misused (and his privacy violated).

By using the PL and compound purposes, the classic Oberholzer privacy agreements may still be modelled (see above), and some other benefits are also offered. Firstly, the principle of freezing agreements is offered, but it is also possible to avoid freezing contracts as defined by Oberholzer and Olivier. The PL provides the opportunity to keep rendering service under an agreement even though some of the parameters in the data controller's and data owner's landscapes have changed. Thus, once a contract exists between the data controller and the data owner, service can continue to be rendered under the contract until it is renegotiated by both parties.

This change in landscape may result in certain purposes being removed from the PL, or the data owner becoming more strict about access to his data.

This fluidity with which privacy agreements can still be offered under certain circumstances also allows the data controller to have multiple privacy agreements with multiple data owners, each of which is tailored to suit the data owner's needs, while still allowing the data controller access to the data owner's data (within limits).

In the following section acceptance levels are examined in more detail. Acceptance levels are the foundation on which the augmentation of privacy agreements is built. They allow the data controller to specify their most general, and most specific purpose and ultimately allow the data owner to specify his desired use for his data.

6.2 Acceptance Levels

Privacy agreement levels from the model proposed by Oberholzer and Olivier may require a large amount of customisation by the data owner (selecting all the

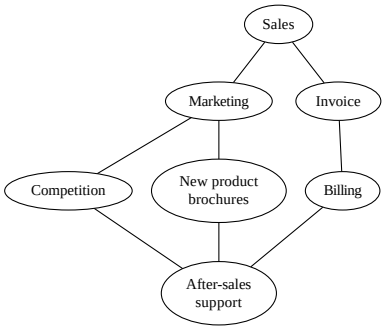

Figure 6.1: A small sub-purpose lattice

elements for levels 1, 2 and 3). To reduce the administrative effort, a single view of privacy agreements is presented to the data owner. This will reduce the number of privacy levels to effectively two, a level 0 and level 1.

To accomplish this single view customisation, the data controller publishes a minimum set of purposes for which the data owner's data will be used, known as the Minimal Acceptance Limit (MinAL). For example, the data controller might state that it will use e-mail addresses for "marketing" **or** "invoicing" (figure 6.1). The privacy fundamentalist who feels that these purposes are too unrestricted can set his preference levels as more restrictive, such as just "invoicing". The data controller can now no longer use any marketing related purposes to access data, since it may only specify a purpose which is stronger than "invoicing". On the other hand, the privacy pragmatist, who is willing to risk the occasional spam mail may leave his setting at the minimum as published by the controller.

From figure 6.1 the MinAL is thus $\phi_1 +_p \phi_2$, and the data owner's preference is ϕ_2. Note that $\phi_1 +_p \phi_2 \leq \phi_2$.

The obvious problem with this approach is that the data owner might be too strict with his preference level, for example the data owner might set his preference level to "Only for access demanded by law" (not shown in the figure). In this case, the data controller may be unable to conduct business, as it cannot gain access to data (for sending out invoices). To prohibit such a draconian approach by the data owner, the data controller employs a Maximal Acceptance Limit (MaxAL), which specifies an upper bound on the purposes for data.

A MaxAL indicates that set of purposes which marks the most specific reason for using a datum. Any purposes more specific than those purposes will prohibit the data controller from conducting day-to-day business.

The data controller thus associates two purpose expressions with each piece of data, a MinAL and a MaxAL. The data owner is allowed to adjust his preferences between these two published levels of use. His preferences are taken to be stronger than the MinAL, assuring him that their data will not be used frivolously, and as

long has his preferences are not stronger than the MaxAL, the data controller will be able to use the data to conduct business.

For example: a data controller states its MinAL to be ϕ_i, and its MaxAL to be ϕ_j. The data owner can set his personal preferences ϕ_u anywhere between these limits (inclusive). Thus, a valid agreement between the data owner and the data controller hinges on the *agreement inclusion principle*, as given in definition 28.

Definition 28 (Agreement Inclusion Principle) *For any agreement to be valid, and for any data controller to be able to make use of the data it collects, it must always be the case that:*

$$\phi_i \leq \phi_u \leq \phi_j \tag{6.1}$$

A small change to the privacy meta data schema as proposed by Agrawal et al. [4] will allow the data controller to record these acceptance limits (table 6.3) for a particular piece of data.

In cases where the data is imperative for day-to-day operation, it is still possible for the data owner to take an active role in specifying the uses for his data. He is thus not strong-armed into an agreement. The data controller also benefits, in that it still has access to the data for normal business functions, it promotes a relationship of trust between the data owner and itself, and may receive access to the data for less restrictive purposes.

In cases where the data submitted by the data owner will be used to provide fringe services, the user may be interested in adjusting this level of opt-in as well. Because the data is not necessary to conduct business, the data controller may not care if the data owner states that his data may only be used on very rare occasions, such as in the event of a law-enforced inspection of the database.

In this case MaxAL can be set to the most specific purpose from the lattice for accessing the data, allowing the data owner to choose this purpose as his preference. The data controller still acts in good faith and specifies a MinAL, and allows the data owner to adjust his preferred acceptance levels. If the purpose for the data is specified as ϕ_i, then the data owner's opt-in specification ϕ_u must still be such that $\phi_i \leq \phi_u \leq \phi_z$, where ϕ_z is the data controller's most specific reason for accessing any data.

To support the notion of acceptance levels the table schema is shown in tables 6.1 through 6.3.

6.3 Privacy Agreement Levels

This section provides a discussion on modelling the Oberholzer privacy agreement using acceptance levels. Because of the flexibility given the number of transaction

Table	attributes
Privacy Policies Table	policyID, MinAL, MaxAL table, attribute, external recipients, retention
Agreements Table	ownerID, policyID, Custom Acceptance Level Valid Flag, Version

Table 6.1: Privacy meta data schema

policyID	MinAL	MaxAL	Table	Attribute	Ex. Recipients	Retention
1	ϕ_0	ϕ_6	customer	name	x	y
2	$\phi_1 \cdot_p \phi_2$	ϕ_6	customer	address	x	y
3	ϕ_3	ϕ_4	customer	credit card	x	y
4	ϕ_4	ϕ_4	customer	credit card	x	y

Table 6.2: Privacy policies table

levels is effectively reduced to two. The following table shows how the Oberholzer agreement and the model presented here line up.

Using the PL to augment the privacy agreement levels provides that additional sense of control to the data owner, in that he gets to choose the purposes for which his data is used – not only that, he can see which purposes dominate each other, and can base his decision on the relative weakness/strength of a purpose. Moreover, because of compound purposes, he can define a strong and complex acceptance level for the use of his data, forcing the data user to carefully construct access requests to his data.

6.4 Agreement Invalidation

Whenever a data controller changes its privacy policy, a new agreement between it and the data owners must be reached. Using the model in this chapter it is also possible that the data owner may decide to adjust his preference levels – a function that should be provided by the data controller *bona fide*.

dataownerID	policyID	Custom AL	Valid Flag	Version
x_1	1	$\phi_1 +_p \phi_5$	true	y
x_2	3	ϕ_4	false	y

Table 6.3: Agreements table

Oberholzer Level	Compounds Level	Description
Level 0,1,2	Level 0	Data controller defines MinAL and MaxAL as well as the data used. Data owner adjusts his acceptance level. Data controller can still conduct day-to-day business, and the data owner controls the purposes for his data.
Level 3	Level 1	Data controller defines MinAL and MaxAL. Data owner chooses data used, and defines his acceptance level.

6.4.1 Data Owner Changes to the Agreement

Changes by the data owner mean that the data controller may have more or less access to the data owner's data. Any change in the data owner's preferences naturally indicate a change in the agreement that is undertaken by the data owner and the data controller. This will result in a new agreement having to be "undersigned" by both parties. However, since the user changes his preferences, and as long as his preferences remain within those allowed by the data controller (between the MinALs and MaxALs), it can be assumed that the data controller has a "safe" agreement with the data owner to have access to the information which will not hinder day-to-day business.

From the data owner's perspective, since he is the one changing preferences, and since the data controller is running a PET which will ensure that his data will not be misused, accepting the changing of the agreement can be automated.

6.4.2 Enterprise Changes

Unfortunately the changes to the privacy agreement between the data controller and owner cannot be automated in the same fashion as was done where the data owner changed the terms of the agreement. This is necessarily true for an important reason. Since the data controller conducts business by using PII from data owners, it is essential that any change in the data controller's stance towards PII purposefully require the data owner's involvement. Moreover, the data owner must be afforded the opportunity to review the terms of the agreement before the data controller may continue using PII.

In the event that the data controller changes the terms of the privacy agreement, the contract between the controller and data owner should be considered

"frozen" [69]. In this text the term "invalidated" will be used to indicate that the agreement between the data controller is no longer considered valid, and that the data controller may no longer use any of the data which falls under the invalidated agreement. The data controller must ensure that a valid agreement exists between itself and the data owner before it will be able to continue using the data owner's data. It is at this critical point that the technological safeguards should protect the data owner's interests, and prohibit the data controller from accessing the stored PII.

The data controller can invalidate an agreement in one of two ways: it may change its MaxAL or MinAL, or modify its PL. The agreement is invalidated because the data owner's acceptance levels may now fall outside of the "interval" between the MinAL and MaxAL

In either case, the agreement can no longer be considered valid, as the data owner's levels might prohibit the conducting of day-to-day business. However, the data controller cannot simply adjust the data owner's custom levels, as this would allow it to access his data with more general purposes. If the MaxALs dominate the data owner's preference levels (after adjustment), then the data owner might wish to adjust his levels to the "maximum" allowed again – the same argument can be offered for adjusting the MinAL.

As mentioned, the data controller may modify its purpose lattice, and thus introduce new purposes which dominate the preference levels as set by the data owner. Purposes can also be removed from the purpose lattice. Addition and removal of purposes from the lattice creates a new lattice which may completely change the data owner's expectation of the controller's use and purpose for his PII. In such cases, it is once again important that the controller allow the data owner to review changes to the agreement, and opt out if he does not agree with them.

As a special case, purposes that are part of any agreement may not be removed from the lattice. This action would place an agreement in an *undefined* state. To avoid these situations the system should perform a restricted delete: when a purpose is removed from the lattice, the system will first verify that no agreement is subject to that purpose. This technique can typically be used to clean up the purpose lattice and remove purposes that are not used.

A final remark on the deletion of entries in the purpose lattice: it is possible that the data controller may consider a purpose and all its children as unnecessary. The system should therefore support a "deep" removal of a purpose from the lattice. Thus the targeted purpose and all its children will be removed.

A "cascaded" removal of a purpose from the lattice is analogous to a cascaded delete from a relational database. Where the database deletion removes the entries that violate integrity, the cascaded delete removes the targeted purposes from the entries in the privacy policy table.

It is natural to consider that any change to the purpose lattice may be sig-

nificant from the controller's viewpoint. That is, the change implies a complete change in the controller's approach to conducting business. A change may also be small enough that the controller may consider previous agreements valid under the old purpose lattice, and the old MinAL and MaxAL. In such cases, a snapshot of the agreement as it stands between the parties may be kept, and this version of the agreement will be considered the binding agreement.

6.4.3 Versioning of the Policy and Purpose Lattice

Changes to the purpose lattice need not invalidate an agreement. A data owner that subscribes to a service provided by a data controller under a particular agreement may continue to receive services provided by the data controller, as long as precise details regarding the version of the privacy policy under which the agreement took place are kept.

The basis for a privacy policy in the model presented here is the purpose lattice. As long as versions of the purpose lattice can be stored, that is, changes to the lattice are recorded, and a particular version can be reconstructed accurately, the privacy policy can be "versioned". Versioning of the lattice can be accomplished in much the same fashion as performing a difference calculation between two files. An agreement with a new data owner is always done under the latest version of the purpose lattice.

By labelling an agreement, it is possible to version an agreement directly. Consider the fourth entry in table 6.2: it applies to the same object in the database, but has a different policyID.

6.4.4 Multiple Agreements

It is plausible that the data controller can have multiple agreements with the data owner, for example an agreement between the data owner and the data controller regarding the physical address, and an agreement regarding the data owner's credit card details. These agreements can then be grouped under one umbrella agreement. The reason for having many agreements can be easily justified. Suppose data controller changes its policy regarding credit card details. Such a shift need not invalidate the agreement the data controller had with the data owner regarding his address.

This is especially true in the case where a minor change in the purpose lattice suddenly invalidates a data owner's agreements with the data controller completely, effectively cutting him off from services.

6.4.5 Invalidation

Whenever a change in policy originates from the data controller, it can be considered either mandatory or optional. Mandatory changes require that all agreements be invalidated, and optional changes only require a versioning of the lattice or policy (as recorded in the database).

Requests to access data of invalidated agreements will not be granted, and will result in a *conflict miss*. Before data can be accessed the agreement will have to be validated again.

Requests to access data of versioned agreements will result in the appropriate version of the lattice or policy being loaded, after which verification of the access request will take place based on those versions.

6.5 Summary

In this chapter the concept of privacy agreements as presented by Oberholzer and Olivier [69] was augmented and extended. The management of a privacy policy (or agreement) between the data owner and the data controller was considered in detail. The idea of privacy agreement management by using the purpose lattice, and how the purpose lattice itself can allow for a finer level of customisation were explored.

The notion of MinALs and MaxALs was introduced. These elements allow a data owner to take control over the use of his data while enabling the data controller to retain access to the data owner's data for day-to-day business tasks – enabling information self-determination as discussed in section 2.1.

Purpose lattices, MinALs, and MaxALs sufficiently support privacy agreement levels, although the levels are effectively reduced to two – data controllers provide their MinALs and MaxALs, and the data owner stipulates the data he is willing to share, and his acceptance levels.

There are many reasons for a privacy agreement between the data controller and the data owner to be invalidated. These are related to either the shifting of the agreement levels (MinAL and MaxAL), and adding new purposes, or removing purposes from the purpose lattice. In cases where these changes are fundamental, it may be that the agreement is invalidated, and the data controller may no longer make use of data, until the data owner has reviewed the new proposed agreement and agrees to it. The data owner may, of course, also opt out of the proposed agreement.

There are many other cases where the privacy agreement need not be invalidated: the data controller may consider the changes to the purpose lattice to be of such little impact, that the agreement between itself and the data owner is "ver-

sioned". Access to data is then governed by the old agreement. Once again, if the agreement is at some stage considered impractical (from the controller's point of view) the controller may request that the old agreement be abandoned, and that the data owner review the new (current) policy, and either agree or decline to agree to the policy.

To allow the management of the agreements and the policies, an enhanced database schema for an RDBMS was also proposed.

It has been shown that policy management with purpose lattices and compound purposes is feasible, and provides a great amount of flexibility. Policy management in this fashion provides the final requirement for privacy enabled as set out at the start of this text: that of information self-determination.

This final chapter thus concludes the goals as was set at the start of the text. The following chapter provides concluding remarks, as well as ideas that are suitable for future work in this field as identified through the research conducted.

Chapter 7

Conclusion

Adde parvum parvo magnus acervus erit
– Ovid.

When discussing the protection of privacy, two types of systems should be considered. The first type of system protects the actions of individuals, such as their communication with other individuals, and other systems protect data that describe the individuals.

The work in this text classifies these two types of systems as systems that protect what we do, and those that protect who we are. Systems that protect what we do ensure that attackers cannot determine the actions or identity of an individual from amongst a group of individuals. Systems that protect who we are ensure that the data that describes an individual (or PII), their name, identity number, address, credit card number, and so on, is protected from misuse.

The focus of this research was systems that protect who we are. In particular, emphasis was on the association of purposes with data, and enforcing access control using the purposes associated with data. The nature of these purposes, and exactly how they are used during access control was examined. This examination reveals several problems with the way in which purposes are used, and these problems were addressed in the content of this text.

One of the key problems with systems that protect who we are is that they make use of purposes without regard for their relation to other purposes. During the association of purposes with data (the binding phase), a group of purposes is typically heaped on the datum. No indication is given if some purposes are optional when protecting access to the data, or if all purposes are mandatory, and so on.

Another key problem is that purposes are placed in a data user's profile (the individual accessing data) as a way of associating credentials with their profile, without considering that data users should be forced to state why they are access-

ing data. Forcing data users to state their intent with data makes them responsible for their actions with data, provides auditable actions, and ensures that the data users cannot repudiate their reason for using the data.

The final key problem identified is that the semantics of the purposes are typically very narrow in the action that they represent. For example, a purpose may be "Sending e-mail", or "Sending catalogue". There is no way of indicating that data is used for "Sending e-mail **and** Sending catalogue". Such complex (or compound purposes) are typically created by creating yet another purpose to represent the complex meaning. Purposes from these systems are called *singular* purposes, since each represents a *singular purpose* that the data may be used for.

To solve the problems identified, the notion of compound purposes and reasons was introduced in this research. These are purposes that can be combined using specially defined operators to form more complex purposes in an *ad hoc* fashion. Several axioms can be used to precisely define the semantics of compound purposes. Compound reasons are used as a statement of intent during an access request.

The verification of compound reasons (as statements of intent during access requests) was examined, and it was proved that verification can be accomplished (with no significant overhead during the verification of the access request). The operators that were introduced as part of the axioms for compound purposes was defined, and their actions on their operands (the purposes) was also defined. Furthermore, special operators for compound reasons was defined, and it was proved that the operators can be used to accurately create structures that are equivalent to the semantics as laid out in the axioms of the compounds. As a final thought on verification, the run-time complexity of the verification algorithm was discussed along with some key optimisation ideas.

Placing purposes in a data user's profile and granting access to data because they "happened to have" purposes in their profile that matched the purposes associated with data is hardly the correct approach when protecting sensitive information. This is one of the key faults with the current incarnations of privacy aware systems. To solve this problem, extensions to a current technology were proposed that is used to store and access large amounts of data: a typical RDBMS. In particular, the mechanism used to access and store information in an RDBMS was extended.

A new hybrid of an accepted access control model was introduced along with the extensions of SQL. This access control model avoids the typical problem that is identified when working with privacy related data – there can be no delegation of trust, and data must be centrally owned. The new model (HDAC) allows data users to grant other users access to data and to revoke granted access, and ensures that access to data is tightly controlled and that no user can gain access to sensitive information if they were not specifically granted access.

An identified benefit of compound purposes and reasons is that they provide a way of augmenting a previously proposed concept of *privacy agreements* (see chapter 6). The augmentation of privacy agreements allows data controllers to enter into agreements with data owners that do not have to be renegotiated when the environment of the data controller or data owner (their view on privacy or what the intent with data is) changes.

The questions posed in the introduction chapter have been answered, i.e. how to make the expression of purposes more rich and flexible, whether this new expression of purposes can be included in existing technologies non-intrusively, and finally, whether these more expressive purposes can be used to provide a sufficient mechanism for constructing and enforcing privacy agreements between the data controller and the data owner. The initial goals of the work have therefore been met.

7.1 Reflection

There are always areas in research that either provide answers that are not clear, answers that only lead to more questions, and areas that provide no answers at all. It is important to step back and reflect on proposed ideas in order to identify shortcomings, and possible future areas of work. This section provides some reflection on the work presented.

With any access control model, or work in security, one of the fundamental aspects is determining if there are channels in the model that allow the unauthorised flow of data from high security classification to a lower security classification – the classic Bell-LaPadula model [8]. Although the HDAC was proved to be internally logically consistent, a thorough analysis of the least privilege property (definition 23) should be done.

The definite benefit of the extensions to SQL is the early definition of the HDAC, which provides decentralised control of access to information (which made the DAC so popular).

The model in this work also allows extensions to an existing model (the Oberholzer privacy agreements dealt with in chapter 6). The model thus reaches beyond the sphere in which it is presented, and contributes to other solutions in an easy fashion.

Another aspect is, of course, the question whether any extensions to a widely used technology such as SQL will ever be incorporated into the standards that define it. It may well be the case that the extensions in chapter 5 will never see the light of day, and that the model for access control carefully set out in chapter 4 is consequently stunted. If one considers that access requests may come in many

forms (not only with SQL), for example XPath[1], then there is definitely an arena for the proposed access control model.

The association of purposes with data almost always creates an obligation with the use of the data. An obvious void in the text is the consideration of obligations, and how they relate to compound purposes and the extensions to SQL and privacy agreements using compounds. However, they will almost certainly prove to be either extremely simple – since so much has already been done on obligations and obligation management. On the other hand they may prove to be so interesting that another document this size can be written on the subject. In either case, obligations are left for future work, and are listed in the "future work" section below.

7.2 Future work

The idea of compound purposes, their verification, using them in SQL and privacy agreements has opened up some new avenues of thought with regard to using purposes in a privacy aware system (or a PET). Additional work can be done to expound the field of compound purposes, and a non-exhaustive list is provided. The items flow directly from some of the open questions identified in the text, but others (from some more subtle source) may surely be added.

The first item for further consideration is of obligation management using compound purposes: the broad question of obligation management becomes an interesting one, when one considers that obligation statements may also be made part of a compound expression, and that the proposed method for managing privacy agreements may allow a data owner to place their own obligations on data controllers. What would these expressions look like, and how would one perform verification around them? Also, how would one determine if the violation of the obligation invalidates the agreement?

Secondly, since purposes and reasons will be used by a data controller (internally), barriers for compound purpose acceptance will certainly be created in privacy aware systems if purposes are defined per data controller. As section 3.3 explains, a protocol converter can do the trick. However, a better way to go would be to consider providing standardised purposes organised into domains of interest. For example, one may have a standard PL defined for the banking industry, and another for the medical profession (or industry). Protocol conversion will only be done in the rare cases where a particular enterprise has extended the standard PL with some custom purposes.

Thirdly, changes to the privacy policy may force significant changes to the

[1] http://www.w3c.org/TR/xpath

purpose lattice. These changes may inadvertently introduce overlap in the purposes extracted from the policy. Ontologies may provide a way of automating the detection of these overlaps, and as future work, annotating purposes using ontologies could provide some interesting results. Moreover, ontologies may provide a way of easing integration of PLs between different enterprises, and that alone warrants further investigation.

The fourth item for consideration is the extension of the SQL syntax for procedures to allow the use of compound purposes. Some limitations and issues is the recompilation of procedures each time a statement of intent is passed to it, and the fact that the procedure as a unit may represent a specific approach to accessing data (it may contain several select statements, for example, each with very specific statements of intent), finding a good way of allowing a data user to modify the statement of intent in a meaningful way will be an interesting problem to consider.

In the fifth place, the model presented here could also be fitted into the RBAC model. This removes the need for the HDAC in its current form, and this impact will have to be carefully considered, as well as how these compound purposes will impact the RBAC model.

Finally, now that the foundational work has been done, a proof of concept will provide wonderful detail into the practical problems and key victories with the proposed model. An implementation will present areas for consideration that were only touched on in theory (such as run-time performance), and will allow practical solutions for future implementations in existing systems to be examined. Although the work done in this text has provided the groundwork for realising such an implementation, much remains to be done for an implementation: the Purpose Driven Access Control Subsystem (PDACS) and SQL parsing mechanism, a database management system control module (or integration into an one) to realise management of purposes and users, a user interface to allow data owners to customise privacy agreements, and the component to convert from one enterprise's DPS to another's.

Bibliography

[1] EU data protection directive 95/46/ec. Tech. rep., European Commission, October 1995. Available from: `http://ec.europa.eu/justice_home/fsj/privacy/law/index_en.htm`.

[2] Report on the OECD forum session on privacy-enhancing technologies (PETs). Tech. Rep. DSTI/ICCP/REG(2001)6/FINAL, OECD Committee for Information, Computer and Communications Policy, Paris, October 2001.

[3] AGRAWAL, R., BAYARDO, R., FALOUTSOS, C., KIEMAN, J., RANTZAU, R., AND SRIKANT, R. Auditing compliance with a hippocratic database. In *Proceedings of the 30th VLDB Conference* (2004).

[4] AGRAWAL, R., KIERNAN, J., SRIKANT, R., AND XU, Y. Hippocratic databases. In *Proceedings of the 28th VLDB Conference* (Hong Kong, China, 2002).

[5] ASHLEY, P., HADA, S., AND KARJOTH, G. E-p3p privacy policies and privacy authorisation. In *WPES'02* (Washington, November 2002).

[6] ASHLEY, P., HADA, S., KARJOTH, G., POWERS, C., AND SCHUNTER, M. Enterprise privacy authorisation language (EPAL 1.1). Tech. rep., International Business Machines Corporation, 2003.

[7] BAUER, K., MCCOY, D., GRUNWALK, D., AND SICKER, D. Bitblender: Light-weight anonymity for bittorrent. In *SecureComm* (2008).

[8] BELL, D. E., AND PADULA, L. J. L. Secure computer systems: unified exposition and multics interpretation. Tech. Rep. ESD-TR-75-306, The MITRE Corporation, Mar 1976.

[9] BERNAT, L., MANSFIELD, N., AND CARBLANC, A. RFID: OECD policy guidance, a focus on information security and privacy. Tech. rep., OECD, 2008.

[10] BERTINO, E. Data security. *Data and Knowledge Engineering 25*, 2 (1998), 199–216.

[11] BERTINO, E., BETTINI, C., AND SAMARATI, P. A temporal authorization model. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security* (1994), ACM Press, pp. 126–135.

[12] BONATTI, P. A., DAMIANI, E., DE CAPITANI, S., AND SAMARATI, P. A component-based architecture for secure data publication. In *Proceedings of the 17th Annual Computer Security Applications Conference* (2001), IEEE Computer Society, p. 309.

[13] BRANDI, W., AND OLIVIER, M. S. Extending p3p to facilitate proxies which pose as a potential threat to privacy. In *TrustBus* (2006), pp. 81–90.

[14] BYUN, J.-W., BERTINO, E., AND LI, N. Purpose based access control of complex data for privacy protection. In *SACMAT'05* (Stockholm, Sweden, June 2005), ACM.

[15] CAMENISCH, J., AND LYSYANSKAYA, A. A formal treatment of onion routing. In *Proceedings of CRYPTO 2005* (August 2005), V. Shoup, Ed., Springer-Verlag, LNCS 3621, pp. 169–187.

[16] CAMENISCH, J., SHELAT, A., SOMMER, D., FISCHER-HÜBNER, S., JANSEN, M., KRASEMAN, H., LEENES, R., AND TSENG, J. Privacy and identity management for everyone. In *DIM'05* (Fairfax, Virginia, USA, November 2005), ACM.

[17] CASTANO, S., FUGINI, M., MARTELLA, G., AND SAMARATI, P. *Database security*. ACM Press, 1994.

[18] CHAUM, D. L. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM 24*, 2 (1981), 84–88.

[19] CHAUM, D. L. The dining cryptographers problem: Unconditional sender and recipient untracability. *Journal of Cryptology 1* (1988), 65–75.

[20] CHAUM, D. L. Showing credentials without identification transferring signatures between unconditionally unlinkable pseudonyms. *AUSCRYPT, LNCS 453* (1990), 246–264.

[21] CHAUM, D. L. Achieving electronic privacy. *The Scientific American* (1992), 96–101.

[22] CLARKE, R. *Business cases for privacy enhancing technologies*. IRM Press, 2008, ch. 7, pp. 135–155.

[23] COHEN, B. The bit torrent protocol specification, 2008. Available from: `http://www.bittorrent.org/beps/bep_0003.html`.

[24] COTTRELL, L. Mixmaster and remailer attacks. Published Electronically. Available from: `http://www.obscura.com/~loki/remailer/remailer-essay.html`.

[25] CRANOR, L., LANGHEINRICH, M., MARCHIORI, M., PRESLER-MARSHALL, M., AND REAGLE, J. The platform for privacy preferences (P3P1.0) specification. Tech. rep., W3C, 2002. Available from: http://www.w3.org/TR/P3P/.

[26] DANEZIS, G., AND DIAZ, C. A survey of anonymous communication channels. Tech. Rep. MSR-TR-2008-35, Microsoft Research, January 2008.

[27] DANEZIS, G., DINGLEDINE, R., HOPWOOD, D., AND MATHEWSON, N. Mixminion: Design of a type iii anonymous remailer protocol. In *In Proceedings of the 2003 IEEE Symposium on Security and Privacy* (2003), pp. 2–15.

[28] DANEZIS, G., AND GOLDBERG, I. Sphinx: A compact and provably secure mix format. In *Proceedings of the 30th IEEE Symposium on Security and Privacy (S&P 2009)* (Oakland, California, USA, May 17–20 2009), IEEE Computer Society, pp. 269–282.

[29] DATE, C. J. *An introduction to database systems*, seventh ed., vol. 1. Addison-Wesley, 2002.

[30] DÍAZ, C., SEYS, S., CLAESSENS, J., AND PRENEEL, B. Towards measuring anonymity. In *Designing privacy enhancing technologies* (2002), H. Federath, Ed.

[31] DIAZ, C., TRONCOSO, C., AND SERJANTOV, A. On the impact of social network profiling on anonymity. In *LNCS 5134* (2008), Springer-Verlag, pp. 44–62.

[32] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium* (August 2004).

[33] EASTLAKE 3RD, D., SCHILLER, J., AND CROCKER, S. Randomness Requirements for Security. RFC 4086 (Best Current Practice), June 2005.

[34] Republic of South Africa: Electronic Communications and Transactions Act No 25 of 2002. Available from: http://www.info.gov.za/view/DownloadFileAction?id=68060.

[35] EDMAN, M., AND YENER, B. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Computing Surveys 42*, 1 (2009), 1–35.

[36] Republic of South Africa: Financial intelligence centre act 38 of 2001. Available from: `https://www.fic.gov.za/`.

[37] FISCHER-HÜBNER, S. *IT security and privacy: Design and use of privacy enhancing security mechanisms*. Springer-Verlag, 2001.

[38] FISCHER-HÜBNER, S., AND OTT, A. From a formal privacy model to its implementation. In *21st National Information Systems Security Conference* (Arlington, VA, USA, October 1998).

[39] GOEL, S., ROBSON, M., POLTE, M., AND SIRER, E. G. Herbivore: A scalable and efficient protocol for anonymous communication. Tech. Rep. 2003-1890, Cornell University, Ithaca, NY, February 2003.

[40] GOLDBERG, I., WAGNER, D., AND BREWER, E. Privacy-enhancing technologies for the internet. In *IEEE COMPCON '97* (1997).

[41] GOLDSCHLAG, D., REED, M., AND SYVERSON, P. Onion routing for anonymous and private internet connections. *Communications of the ACM 42* (1999), 39–41.

[42] GRIFFITHS, P. P., AND WADE, B. W. An authorization mechanism for a relational database system. *ACM Transactions on Database Systems (TODS) 1*, 3 (1976), 242–255.

[43] GRUBER, T. Ontology. *Encyclopedia of database systems* (2009).

[44] HANSEN, M., SCHWARTZ, A., AND COOPER, A. Privacy and identity management. *IEEE Security and Privacy 2* (2008), 38–45.

[45] HE, Q., AND ANTÓN, A. I. A framework for modeling privacy requirements in role engineering. In *Proceedings of the 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03)* (Austria, 2003), RefsQ, pp. 115–124.

[46] HENRY, R., HENRY, K., AND GOLDBERG, I. Making a nymbler nymble using VERBS. Tech. Rep. CACR 2010-05, University of Waterloo, May 2010.

[47] HES, R., AND BORKING, J., Eds. *Privacy enhancing technologies: The road to anonimity*, revised ed. Dutch DPA, Registratiekamer, Den Haag, August 2000.

[48] The health insurance portability and accountability act of 1996. Tech. rep., United States of America, 1996. Available from: `http://www.gpo.gov/fdsys/pkg/CRPT-104hrpt736/pdf/CRPT-104hrpt736.pdf`.

[49] HOFMAN, J. *Electronic evidence in South Africa*. LexisNexis Butterworths, London, UK, 2007.

[50] HOLT, J. E., AND SEAMONS, K. E. Nym: Practical pseudonymity for anonymous networks. Tech. Rep. 2006-4, Internet Security Research Lab, Brigham Young University, June 2006.

[51] KARJOTH, G., AND SCHUNTER, M. A privacy policy model for enterprises. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop* (June 2002), Springer-Verlag.

[52] KELLY, D. *A taxonomy for and analysis of anonymous communications networks*. PhD thesis, Air Force Institute of Technology, March 2009.

[53] KNUTH, D. *The art of computer programming, volume 4 fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams*. Addison-Wesley, 2009.

[54] KOUNGA, G., MONT, M. C., AND BRAMHALL, P. Extending xacml access control architecture for allowing preference-based authorisation. In *Trust, Privacy and Security in Digital Business*, S. Katsikas, J. Lopez, and M. Soriano, Eds., vol. 6264 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2010, pp. 153–164.

[55] LEFEVRE, K., AGRAWAL, R., ERCEGOVAC, V., RAMAKRISHNAN, R., XU, Y., AND DEWITT, D. Limiting disclosure in hippocratic databases. In *30th International Conference on Very Large Data Bases* (Toronto, Canada, 2004).

[56] LOESING, K. *Privacy enhancing technologies for private services*. PhD thesis, University of Bamberg, May 2009.

[57] LOUKIDES, G., AND GKOULALAS-DIVANIS, A. Privacy challenges and solutions in the social web. *ACM Crossroads 16*, 2 (2009), 14–17.

[58] MAIA, M., ALMEIDA, V., AND ALMEIDA, J. Identifying user behaviour in online social networks. In *First Workshop on Social Network Systems* (2008).

[59] MALLESH, N., AND WRIGHT, M. Countering statistical disclosure with receiver-bound cover traffic. In *Proceedings of ESORICS 2007, 12th European Symposium on Research in Computer Security* (Dresden, Germany, September 24 – 26 2007), J. Biskup and J. Lopez, Eds., vol. 4734 of *Lecture Notes in Computer Science*, Springer, pp. 547–562.

[60] MASSACCI, F., MYLOPOULOS, J., AND ZANNONE, N. Minimal disclosure in hierarchical hippocratic databases with delegation. Tech. Rep. DIT-05-051, Informatica e Telecomunicazioni, 2005.

[61] MCCOY, D., BAUER, K., GRUNWALD, D., KOHNO, T., AND SICKER, D. Shining light in dark places: Understanding the Tor network. In *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)* (Leuven, Belgium, July 2008), N. Borisov and I. Goldberg, Eds., Springer, pp. 63–76.

[62] MCLACHLAN, J., TRAN, A., HOPPER, N., AND KIM, Y. Scalable onion routing with Torsk. In *Proceedings of CCS 2009* (November 2009).

[63] MELTON, J. (ISO-ANSI working draft) foundation (SQL/foundation). Tech. rep., ISO-ANSI, 2003.

[64] MÖLLER, B. Provably secure public-key encryption for length-preserving chaumian mixes. In *Proceedings of CT-RSA 2003* (April 2003), Springer-Verlag, LNCS 2612.

[65] MONT, M. C., THYNE, R., AND BRAMHALL, P. Privacy enforcement with hp select access for regulatory compliance. Tech. Rep. HPL-2005-10, HP Laboratories Bristol, 2005.

[66] MONT, M. C., THYNE, R., AND PEARSON, S. A systematic approach to privacy enforcement policy compliance checking in enterprises. In *Third International Conference on Trust and Privacy for Digital Business* (Krakow, Poland, 2006), Springer-Verlag.

[67] MURDOCH, S. J. *Covert channel vulnerabilities in anonymity systems*. PhD thesis, University of Cambridge, December 2007.

[68] OASIS ACCESS CONTROL TC. OASIS extensible access control markup language (xacml) version 2.0. Tech. rep., OASIS, February 2005. Available from: `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`.

[69] OBERHOLZER, H. J., AND OLIVIER, M. S. Privacy contracts incorporated in a privacy protection framework. *International Journal of Computer Systems Science and Engineering 21*, 1 (2006), 5–16.

[70] OECD guidelines on the protection of privacy and transborder flows of personal data. Tech. rep., Organisation for Economic Co-operation and Development, 1980. http://www.oecd.org/document/18/0,2340,en_2649_34255_1815186_1_1_1_1,00.html.

[71] Inventory of privacy enhancing technologies. Tech. Rep. JT00119007, Working Party on Information Security and Privacy, Organisation for Economic Co-operation and Development, 2002.

[72] OLIVIER, M. S. A layered architecture for privacy-enhancing technologies. In *Proceedings of the Third Annual Information Security South Africa Conference (ISSA2003)* (Sandton, South Africa, July 2003), J. H. Eloff, H. S. Venter, L. Labuschagne, and M. M. Eloff, Eds., pp. 113–126.

[73] OLIVIER, M. S. Flocks: Distributed proxies for browsing privacy. In *Proceedings of SAICSIT 2004 — Fulfilling the Promise of ICT* (Stellenbosch, South Africa, October 2004), G. Marsden, P. Kotzé, and A. Adesina-Ojo, Eds., pp. 79–88.

[74] OLIVIER, M. S. Distributed proxies for browsing privacy — a simulation of Flocks. In *Research for a changing world — Proceedings of SAICSIT 2005* (White River, South Africa, September 2005), J. Bishop and D. G. Kourie, Eds., pp. 104–112.

[75] Payment card industry (PCI) data security standards, July 2009. Available from: `https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html`.

[76] PETTERSSON, J. S., FISCHER-HÜBNER, S., DANIELSSON, N., NILSSON, J., BERGMANN, M., CLAUSS, S., KRIEGELSTEIN, T., AND KRASEMANN, H. Making prime usable. In *Symposium On Usable Privacy and Security (SOUPS)* (Pittsburgh, PA, USA, July 2005).

[77] PFITZMANN, A., AND HANSEN, M. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology, July

2007. Available from: http://dud.inf.tu-dresden.de/Anon_Terminology.shtml.

[78] PFLEEGER, C. P. *Security in computing*. Prentice-Hall International, Incorporated, 1997.

[79] PIRAHESH, H., HELLERSTEIN, J. M., AND HASAN, W. Extensible/rule based query rewrite optimization in starburst. In *SIGMOD Conference on the Management of Data* (San Diego, California, 1992), ACM.

[80] Republic of South Africa: Promotion of Access to Information Act 2 of 2000, 2002. Available from: http://www.info.gov.za/view/DownloadFileAction?id=68186.

[81] RABATTI, F., BERTINO, E., KIM, W., AND WOELK, D. A model of authorisation for next-generation database systems. In *ACM Transactions on Database Systems* (March 1991), vol. 16, ACM, pp. 88–131.

[82] REED, M. G., SYVERSON, P. F., AND GOLDSCHLAG, D. M. Anonymous connections and onion routing. *IEEE Journal on Selected Areas in Communications 16* (1998), 482–494.

[83] REITER, M. K., AND RUBIN, A. D. Crowds: Anonymity for web transactions. *ACM Transactions on Information Systems Security 1* (November 1998), 66–92.

[84] Republic of South Africa: Regulation of Interception of Communications and Provision of Communication-related Information Act 70 of 2002. Available from: http://www.info.gov.za/acts/2002/a70-02/.

[85] RISSANEN, E., BROSSARD, D., AND SLABBERT, A. Distributed access control management - a xacml-based approach. In *ICSOC/ServiceWave* (2009), pp. 639–640.

[86] ROSENTHAL, A., AND SCIORE, E. Extending SQL's grant operation to limit privileges. In *Data and Application Security, Development and Directions, IFIP TC11/ WG11.3 Fourteenth Annual Working Conference on Database Security* (August 2000), B. M. Thuraisingham, R. P. van de Riet, K. R. Dittrich, and Z. Tari, Eds., Kluwer, pp. 209–220.

[87] SANDHU, R. S. Role-based access control. *Advances in Computers 46* (1998).

[88] SCHUNTER, M., AND ASHLEY, P. The platform for enterprise privacy practices. Tech. rep., IBM, 2002.

[89] SERJANTOV, A., AND DANEZIS, G. Towards an information theoretic metric for anonymity. *LNCS 2482, Designing Privacy Enhancing Technologies* (2002), 41–53.

[90] SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal 27* (July 1948), 379–423.

[91] SILBERSCHATZ, A., KORTH, H. F., AND SUDERSHAN, S. *Database Systems Concepts*, fourth ed. McGraw-Hill Higher Education, 2002.

[92] SYVERSON, P., STUBBLEBINE, S., AND GOLDSCHLAG, D. Unlinkable serial transactions. *Financial Cryptography. Lecture Notes in Computer Science 1318* (1997), 39–56.

[93] THOMAS, S., WILLIAMS, L., AND XIE, T. On automated prepared statement generation to remove sql injection vulnerabilities. *Inf. Softw. Technol. 51* (March 2009), 589–598.

[94] TSANG, P. P., AU, M. H., KAPADIA, A., AND SMITH, S. W. Blacklistable anonymous credentials: blocking misbehaving users without ttps. In *ACM Conference on Computer and Communications Security* (2007), pp. 72–81.

[95] TSANG, P. P., AU, M. H., KAPADIA, A., AND SMITH, S. W. PEREA: towards practical ttp-free revocation in anonymous authentication. In *ACM Conference on Computer and Communications Security* (2008), pp. 333–344.

[96] TSANG, P. P., KAPADIA, A., CORNELIUS, C., AND SMITH, S. W. Nymble: Blocking misbehaving Users in anonymizing networks. *IEEE Transactions on Dependable and Secure Computing (TDSC)* (September 2009).

[97] VAN STADEN, W., AND OLIVIER, M. S. Sql's revoke with a view on privacy. In *SAICSIT Conf.* (2007), L. Barnard and R. A. Botha, Eds., vol. 226 of *ACM International Conference Proceeding Series*, ACM, pp. 181–188.

[98] VAN STADEN, W., AND OLIVIER, M. S. On compound purposes and compound reasons for enabling privacy. *J. UCS 17*, 3 (2011), 426–450.

[99] VAN STADEN, W. J., AND OLIVIER, M. S. Purpose organisation. In *Proceedings of the fifth annual Information Security South Africa (ISSA) Conference* (Sandton, Johannesburg, South Africa, June 2005).

[100] VAN STADEN, W. J., AND OLIVIER, M. S. Extending SQL to allow active specification of purposes. In *Third International Conference on Trust and Privacy for Digital Business* (Krakow, Poland, 2006), Springer-Verlag.

[101] VAN STADEN, W. J., AND OLIVIER, M. S. Using purpose lattices to facilitate the customisation of privacy agreements. In *Proceedings of the Fourth International Conference on Trust and Privacy for Digital Business* (Regensburg, Germany, September 2007), Springer-Verlag.

[102] WRIGHT, J., STEPNEY, S., CLARK, J. A., AND JACOB, J. Formalizing anonymity: A review. Tech. Rep. YCS-2005-389, Department of Computer Science, University of York, June 2005.

[103] ZHOU, B., PEI, J., AND LUK, W. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *SIGKDD Explorer Newsletter 10*, 2 (2008), 12–22.