*Journal of Universal Computer Science, vol. 17, no. 3 (2011), 426-450*
*submitted: 28/6/09, accepted: 28/1/11, appeared: 1/2/11 © J.UCS*

# On Compound Purposes and Compound Reasons for Enabling Privacy

Wynand van Staden, Martin S. Olivier
(Information and Computer Security Architectures Research Group
University of Pretoria, Pretoria, South Africa
wvs@wvs.za.net, martin@mo.co.za)

**Abstract:** This paper puts forward a verification method for *compound purposes* and *compound reasons* to be used during purpose limitation.

When it is absolutely necessary to collect privacy related information, it is essential that privacy enhancing technologies (PETs) protect access to data – in general accomplished by using the concept of purposes bound to data. Compound purposes and reasons are an enhancement of purposes used during purpose limitation and binding and are more expressive than purposes in their general form. Data users specify their access needs by making use of compound reasons which are defined in terms of (compound) purposes. Purposes are organised in a lattice with purposes near the greatest lower bound (GLB) considered weak (less specific) and purposes near the least upper bound (LUB) considered strong (most specific).

Access is granted based on the verification of the statement of intent (from the data user) against the compound purpose bound to the data; however, because purposes are in a lattice, the data user is not limited to a statement of intent that matches the purposes bound to the data exactly – the statement can be a true reflection of their intent with the data. Hence, the verification of compound reasons against compound purposes cannot be accomplished by current published verification algorithms.

Before presenting the verification method, compound purposes and reasons, as well as the structures used to represent them, and the operators that are used to define compounds is presented. Finally, some thoughts on implementation are provided.

**Keywords**: Privacy Enhancing Technologies, Databases, Purposes, Purpose Lattices, Compound Purposes
**Categories**: E.m, K.4.m

## 1 Introduction

From a trust perspective it is important for enterprises to ensure that they act in a privacy conscious manner when accessing and working with an individual's personal information or personal identifiable information (PII). Systems that support privacy of individuals by introducing mechanisms which protect PII are collectively referred to as privacy enhancing technologies (PETs).

Of particular interest (and the focus in this paper) are the mechanisms that PETs employ in order to ensure the privacy of individuals (specifically the means of protecting privacy through the use of purposes). These PETs provide *purpose binding and limitation* capabilities and operate in two fundamental phases:

firstly, the enterprise is forced to state his purpose for collecting a certain datum. Secondly, access to these data elements is protected by requiring that the *data user* states his intent with the data when requesting access to that datum. Access to the data is granted only if the data user's intent (*reason for using the data*) matches the bound purpose. The first phase (purpose binding) is referred to as the specification phase and the second phase is referred to as the limitation phase [Fischer-Hübner 2001].

Purposes used in the binding and limitation phases are at their core considered to be singular in nature, as is evident from examples published in many papers [Ashley et al. 2003, Byun et al 2005, Massacci et al. 2005]. This means every purpose represents exactly *one reason* for using data.

In order to indicate that the enterprise has more than one purpose for a particular datum, most PETs allow the specification of multiple purposes. However, these purposes are considered in isolated form during the limitation phase. For example: when specifying the singular purposes that represent *To Send Invoice* and *To Send Parcel* to a shipping address, either of the two purposes can be used as a reason for accessing the information. These purposes are thus *singular* in nature and will hereafter be referred to as *singletons*.

How would the enterprise indicate that it will only use the shipping address of the individual to *send a parcel* ***and*** *invoice together*? Using singletons, one would have to create another singleton that embodies both these intentions, or *multi-purpose* singletons.

This use of multi-purpose singletons is not desirable for two reasons. Firstly, the policy implementor will have to ensure that he is not duplicating an existing multi-purpose. Moreover, there may be other effort involved such as having to request the addition of a new purpose to a central authority within the entreprise. Secondly, the amount of administrative effort may be extensive. Additionally, if purposes are in a hierarchical structure, it becomes laborious to determine exactly where this new multi-purpose will fit into the hierarchy.

A final comment on multi-purpose singletons: it is likely that a particular singleton will be present as a component of many such multi-purpose singletons and being able to use singletons in combinations is thus a natural way of thinking about purposes and requires further investigation.

The verification method presented in this paper is supported by three principles. Firstly, a way to express the combinations of purposes. Secondly, the data user can specify their intent when accessing information. And finally the semantics of the purpose combinations are clearly defined.

### 1.1 Contribution

This paper contributes to data storage (with respect to privacy) in two ways. Firstly, the semantics of the notation for expressing compound purposes and reasons are clearly defined. Secondly, a method for performing verification during access control when using these combination purposes is provided.

By wrapping access requests to a database inside a component that can parse a compound purpose statement and perform the proposed verification, existing PETs can utilise the methods presented here. The authors have also shown elsewhere [Van Staden and Olivier 2007] that privacy contracts can be much more customisable using compound purposes – especially from an online presence perspective. Data owners can state an acceptable range of purposes for their data, and the enterprise can indicate their most general and most specific purposes with data.

### 1.2 Structure of the paper

The rest of the paper is organised as follows: section 2 provides background information and presents related work. Section 2.2 presents the principles of compound purposes and reasons and examines the distinction between purposes and reasons. Section 3 introduces the notion of negative purposes which are an extension of compound purposes. Section 4 provides an outline of the approach to verification and the operators for constructing compound purposes. Section 5 provides detail on the nature of suitable purpose sets (SPSs) used during verification. Section 6 provides the final detail on verification. Sections 7, and 8 discuss the compound purpose and reason operators that are used during verification in more detail. Section 9 provides some thoughts on implementation, and finally section 10 provides concluding remarks.

The appendices provide theorems and proofs for the properties of the functions presented in section 4.

## 2 Background and related work

A system used to protecting an individual's privacy through the principle of information self-determination must support several established principles [Fischer-Hübner 2001], and these principles are outlined in several well-known documents (that outline the requirements of a privacy aware system) such as the organisation for economic cooperation and development (OECD)'s principles of fair use [OECD 1980], and a report by the Dutch IPC [Hes et al. 1998].

In the light of the explosion of privacy invading technologies available [Clarke 2008, WikiPedia2010, Bernat et al. 2008] a strong case exists for the adoption of PETs and many countries have adopted these principles in legislative form [Oberholzer and Olivier 2006]. The author's country of residence, for example, provides legislation that allows individuals to query information stored on them [SA Proatia 2002], to query where organisations obtained their information [SA ECT Act 2002].

From the principles in the aforementioned documents PETs can be placed into two broad categories (the paper does not attempt to provide a finer grained classification such as provided by the OECD [oecd2002]). The broad classification provided here is based on systems that protect privacy by protecting *what we do*, and systems that protect privacy by protecting *who we are.*

*What we do* protection relates to systems that protect privacy of communications (in essence who we are talking to, what we are saying, and so on). These systems typically employ mechanisms that will attempt to ensure *anonymity, pseudonymity, unlinkability*, and *unobservability* [Pfitzmann and Hansen 2007]. Examples of such systems are systems based on Chaum's MIXes [Chaum 1981], such as onion routers [Fischer-Hübner 2001], Tor [Dingledine et al. 2004], Mix-Minion [Danezis et al. 2003], and Flocks [Olivier 2005]. This paper regards *what we do* data that has been stored as *who we are* data – this data typically becomes historical data that describe the person about whom the data is collected – in essence it is profiling data.

*Who we are* protection systems are those that protect privacy by protecting information on individuals that is stored for a longer (ostensibly an infinite) period of time. The motivation behind collecting and storing information in this manner may be for legislative purposes (such as the financial intelligence center act (FICA) [SA FIC Act 2001], and the regulation of interception of communications act (RICA) [SA RIC Act 2002] in the author's country of residence), or because at least some information on an individual is necessary in order to conduct business – a postal address, for instance, in the event that a courier company has to deliver a parcel to an individual.

When PII is collected, the privacy of the data owner may be compromised, therefore the enterprise's systems should deal with the information in a responsible manner. For example: a system could allow the data owner to inspect the privacy promises made by the enterprise regarding their PII and allow them to accept or reject those promises. Such technologies are already deployed: platform for privacy preferences (P3P) [Cranor et al. 2002], for example, that deals with the collection of data in an online environment and can aid in the automation of this inspect/accept/reject action. Unfortunately P3P is not a technology that enforces the promises made by the enterprise: a trust relationship between the data-owner and data-collector is implicit; also P3P has not really experienced main-stream adoption (one is not forced to use P3P when moving around online). In order to ensure that these promises can be enforced, Karjoth et al. [Karjoth and Schunter 2002] have proposed the enterprise platform for privacy preferences (EP3P) which is a "superset" of P3P, and allows the creation of technologies that can enforce the privacy policies made by enterprises. A first example such a technology is enterprise privacy authorisation language (EPAL) [Ashley et al. 2003].

As far as policies with respect to purposes and privacy protection is concerned, EPAL allows the specification of purposes and also allows these purposes (which are placed in a hierarchy) to dominate each other. A data user may use a parent (super) purpose of child purposes only when they have authorisation to use all the child purposes of a parent purpose.

A more dramatic move to enable technologies to protect PII is the Hippocratic database put forward by Agrawal et al. [Agrawal et al. 2002]. There have been several extensions proposed to the concepts presented in the seminal Hippocratic database paper [Oberholzer and Olivier 2006, Massacci et al. 2005]. Agrawal et al. have also shown that the Hippocratic database lends itself to auditing through trivial extensions of SQL [Agrawal et al. 2004].

Massacci et al. [Massacci et al. 2005] present the organisation of tasks into directed acyclic graphs (DAGs) in which edges represent an *AND* relationship or an *OR* relationship with their parent *task* (the nodes in the graph). The primary difference between their work and the work presented in this paper is that their system is task oriented, and purposes are related to tasks. Tasks are what the enterprise needs to do in order to reach goals. If an enterprise wishes to ship a parcel, for example, it needs to know some address for shipping purposes. This can be a post office box or a street address (hence the and/or relationships). The notion of compound purposes is thus not entertained by Massacci et al.

The privacy and identity management for europe/everyone (PRIME) project is a technological framework that aims at providing the necessary tools (that compliments legislation) to accomplish privacy and identity management [Camenisch et al. 2005, Hansen et al. 2008]. Identity management as part of data protection is necessary for accountability [OECD 2007]. The framework itself provides three categories of technology for accomplishing this: firstly, allowing individuals the ability of information self-determination. Secondly, the ability for the enterprise to specify what they will do with information and a way for individuals and enterprises to negotiate an acceptable agreement regarding the use of the individual's data; and finally, the use of automated mechanisms to accomplish all the aforementioned. With this in mind, if the work presented in this article had been aimed at the PRIME framework it would fall into the "automated mechanism" group of technologies. It does not allow the individual to specify their privacy preferences, but it does protect the individual's information through the use of purpose limitation.

The authors have already presented work which shows how simple extensions to the SQL syntax (through the use of a pre-processor) can enable the data user to explicitly state their purposes with data [Van Staden and Olivier 2006]. The work shows that placing purposes in a lattice can enable a *hybrid-DAC model*, and also shows how compound purposes can be incorporated in the design.

Byun et al. [Byun et al 2005] use role based access control (RBAC) to represent hierarchies of purposes instead of explicitly listing them in a lattice. Moreover, it still only allows the binding of purposes using singletons. This paper enhances the Byun model (bar the role-based approach) and many models which propose the use of singletons, by allowing the expression of more complex purposes for use during the binding phase.

A central requirement in the method presented is the placement of purposes in a lattice, which is discussed in more detail in the following section.

### 2.1 Purposes in a lattice

The work presented in this paper continues work presented earlier in [Van Staden and Olivier 2005]. A quick discussion will provide the necessary context.

A *compound purpose* is a purpose that relies on singleton purposes that the enterprise defined previously. However, singleton purposes are stored in a bounded *lattice*. A lattice is a set that has elements, and defines some partial ordering on the elements in the set (it is possible to compare some of the elements to each other to order them, although not all elements can be compared in this way). A bounded lattice is a lattice that has a GLB (one element smaller than all the other elements) as well as a LUB (one element greater than all the other elements.

A purpose in the lattice dominates another purpose in the lattice (if the relation $x \leq y$ exists). Any purpose that dominates another purpose bound to data is considered sufficient for accessing that data. For example, an email address may be stored for *marketing purposes*. An access request to the email address for *sending a product catalogue* may be granted if *sending a product catalogue* dominates *marketing* in the lattice; thus these two purposes can be compared, and the last is greater than the first. It is this ability to compare purposes to eachother that allows a data user's statement of intent to more closely reflect their true intent with data.

The following definition provides a formal definition of a lattice that stores purposes (a purpose lattice).

**Definition 1 Purpose Lattice.** A purpose lattice (PL) is a bounded, partially ordered set (poset) $(A, \leq)$ and $\forall(a, b) \in A, a, b \in DPS$. Where domain purpose set (DPS) is the set of purposes that the organisation will use. $\exists x \in DPS$ such that $(x, y) \in A \forall y \in DPS$. $x$ is known as element 0, or the GLB. Also $\exists z \in DPS$ such that $(y, z) \in PL \forall y \in DPS$. $z$ is known as element 1, or the LUB.

$(a, a)$ is taken to mean $a \leq a$, and will always be present in the lattice for all purposes in the lattice.

Since the lattice is bounded, element 0 is dominated by all the other purposes in the lattice and is the *most general purpose* for storing information; 1 dominates all purposes in the lattice and is known as the *most specific* purpose in the lattice.

Providing element 0 ensures that there will always be a very weak (or general) purpose associated with a datum, and 1 ensures that there will always be a single strong (very specific) purpose that can be used to gain access to a datum.

A compound purpose and reason is defined by forming *conjunctions* and *disjunctions* from *singleton* purposes in the PL.

## 2.2 Compound Purposes and Compound Reasons

In this section the notation for defining a statement of intent as well as the purpose to bind to data is presented. This notation was presented elsewhere [Van Staden and Olivier 2005]; however, it is necessary to provide a short summary to provide proper context.

The first point to clarify is the distinction between compound *purposes* and compound *reasons*. These are effectively the same, but the verification method presented requires that a distinction be made. The reason for this distinction will become apparent in section 6 where the functioning of the operators is discussed in detail. The following sections provides details on the notation used, and the semantics of compound purposes and reasons.

### 2.2.1 Compound Purposes

A compound purpose expression is defined using purposes and three binary operators.

$\cdot_p$ is used to indicate that access to data is only granted if the data will be used for both purposes represented by its operands – and thus the data user must explicitly provide both reasons in his statement of intent.

$+_p$ is used to indicate that access to data is only granted if the data will be used for either one of the two purposes (or both) represented by its operands – the data user must therefore explicitly provide one (or both) of the reasons in his statement of intent.

Finally, $\cdot\neg_p$ is used to indicate that certain purposes [see 3] cannot be presented to access the data – the data user's statement of intent cannot contain these excluded purposes.

To clarify exposition it is assumed that $\cdot\neg_p$ has higher precedence than $\cdot_p$ which has higher precedence than $+_p$.

An example is provided to clarify the envisaged use of compound purposes.

*Example 1.* Suppose an enterprise has, amongst others, three purposes in their PL: $\phi_x$ = "Update Personal Information", $\phi_y$ = "Sell New Products", and $\phi_z$ = "Update Portfolio".

Suppose that no relation exists between any of these purposes. The enterprise can encode the following policies regarding the storage of telephone numbers

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. $\phi_x \cdot \neg_p \phi_y$

■

### 2.2.2 Compound Reasons

A compound reason is used exclusively during the verification phase and is defined using two operators: the first, $\cdot_r$ (and/conjunction), is used to indicate that the requested datum will be used for both purposes represented by the operands,and the second, $+_r$ (or/disjunction), is used to indicate that either one of the purposes will be used for the data; for exposition it is assumed that the $\cdot_r$ has the higher precedence.

*Example 2.* Suppose a data user wishes to indicate his intent for accessing telephone numbers [see example 1]. Any one of the following can be used.

1. I want the telephone number because I am updating the client's personal information as well as his portfolio: $\phi_x \cdot_r \phi_z$
2. I want the telephone number because I am either updating the client's personal information, or his portfolio: $\phi_x +_r \phi_z$

In the last request, the data user might not know beforehand if the client has any portfolio changes that has to be made, thus the user indicates that the information is requested for doing one or the other (or maybe even both).

■

Before continuing the discussion on verification, the paper now turns to the concept of negative purposes which has not been presented previously, and that was mentioned in section [2.2.1].

## 3 Negative Purposes

In this section the concept of compound purposes is extended to include the new idea of *negative purposes*. In many cases it becomes necessary to exclude certain purposes from the purposes bound to data. An organisation may store email addresses for marketing purposes, for example, and may also want to indicate that whatever marketing they do, they will not use email addresses for sending *general* marketing information. A negative purpose can be used to achieve this and is somewhat similar to *negative authorisations* (not those used in open systems [Castano et al. 1994]).

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

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 $\phi_k$. Furthermore, suppose $\phi_l$ shares a common child with $\phi_k$ and that the policy creator wishes to indicate that the datum will in no way be used for any purposes relating to $\phi_l$. This exclusion 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.

An important exception to the usage of $\cdot\neg_p$ is that it should not be possible to specify the LUB (element 1) as the second operand to $\cdot\neg_p$. 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: the LUB purpose might be "Obligation to Legislation", for example. A discussion of the semantics of the and-not operator is provided in [7.3].

A compound such as $\phi_1 \cdot \neg_p \phi_1$ is clearly nonsensical and the creation of this form of compound should ideally be detected and stopped before the binding phase. However, the definition of the and-not operator will ensure that data cannot be accessed for $\phi_1$.

With all of the above in mind, access to data is only granted if the statement of intent is sufficient for the compound purpose bound to the data, and the statement of intent does not contain any of the purposes that were explicitly excluded in the compound purpose statement using $\cdot\neg_p$.

In the following section the approach to verification using compound purposes is discussed in more detail.

## 4 Verification Approach

In this section the approach taken to the verification of a statement of intent against bound purposes is discussed. In particular, the goals of the verification system, as well as the elements needed to accomplish the goals are provided.

The current means for using purposes during access control assume purposes bound to data to be represented using sets and that the purposes in the data user's profile are present in these sets. The approach followed here is similar, the only difference being that the statement of intent and bound purpose are expressed using the notation presented in section [2.2].

The bound purpose expression thus represents all the purposes (and their combinations) that may be present in a statement of intent. In particular, a bound purpose expression can be thought of as a statement specifying that a data user may present purpose $\phi_a$, or they may present $\phi_b$ and $\phi_c$, for example.

Verification is thus accomplished by examining the statement of intent and ensuring that all the conjunctions expressed there-in, are present in the set of all possible conjunctions for the bound purpose expression. If this is so, then access is granted, if not, access is denied. In short what is needed to support the approach taken here is:

1. A suitable representation of purposes that form part of the compound purpose statement,
2. A suitable representation of reasons that form part of the statement of intent.
3. A computationally inexpensive method for determining if the statement of intent is enough to grant access to data that has a bound compound purpose bound to it.

To accomplish the goals listed above, the following approach is followed:

1. convert the compound purpose statement into a set of purposes that represent all the possible statements that may be presented.
2. convert the statement into a set of purposes,
3. determine if the set of purposes derived from the statement of intent is a complete subset of the set associated with the compound purpose. Set calculations can be accomplished easily, and some thoughts on implementation of sets for verification is provided in section [9].

The set of purposes associated with the compound purpose statement is called the SPS (written $\Phi'$), and the set associated with the statement of intent is called the reason set (RS) (written $\Gamma'$).

An example based on the lattice presented in figure 1 is presented to help elucidate the goals of constructing $\Gamma'$ and $\Phi'$.

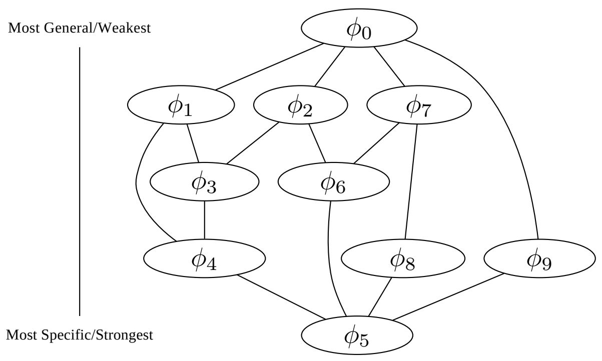

**Figure 1:** Example Purpose Lattice

*Example 3.* Suppose a compound purpose is specified as $\phi_1 \cdot_p \phi_2 +_p \phi_7$. Thus, a statement of intent that is suitable for $\phi_1$ and $\phi_2$ simultaneously, or suitable for $\phi_7$, or suitable for $\phi_1$ and $\phi_2$ and $\phi_7$ simultaneously will be suitable for gaining access to the datum.

This means that the set: $\Phi'$ 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 sake of brevity a full listing of possible purposes is not presented here).

Suppose a data user presents $\phi_4 \cdot_r \phi_6 +_r \phi_8$ as a statement of intent. $\Gamma'$ 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.

■

The creation of the SPS requires some additional preparation work (for many reasons, one of these being speeding up of the computation), which is discussed in the following section.

## 5 SPSs

In this section, the principles behind the SPS is considered in detail. This is accomplished by examining (and defining) the structures that are required to derive SPSs, as well as the construction of SPSs – since the construction of the SPS depends on the compound expression used, the construction is discussed alongside the definition of the operators used to express a compound statement.

A *purpose set* forms the foundation for an SPS: it is the simplest way of expressing a conjunction of purposes that may be presented to get access to a datum. Combining a purpose set with other purpose sets using disjunctions provides the complete SPS and consequently the complete set of conjunctions of purposes that may be presented the get access to data (for the expression that this SPS represents).

In the following section the purpose set itself is discussed in much more detail.

## 5.1 Purpose Sets

Purpose sets have two properties that make them suitable for use as a representative element of a compound purpose, and sufficient to use during verification: firstly, the *unambiguity*-property, and secondly, the *sufficiency* property.

### 5.1.1 Unambiguity

An unambiguous set is a set that contains only elements for which there exists no relation in the lattice. A simple example illustrates the benefit of unambiguity. Suppose, for example, that the purpose expression $\phi_i \cdot_p \phi_j$ is bound to a datum and that $\phi_i \leq \phi_j$. To gain access to the datum a reason that is suitable for $\phi_i$, as well as $\phi_j$ must be presented. Since $\phi_j$ already dominates $\phi_i$, any reason that is suitable for $\phi_j$ will also be suitable for $\phi_i$. Hence the weaker purpose need not be present in the purpose set representing the compound purpose.

**Definition 2 Unambiguity-property.** A purpose set $X$ is *unambiguous* if none of the elements dominate each other. 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.

### 5.1.2 Sufficiency

Sufficiency is used to describe the degree to which a purpose set contains purposes that are suitable for a (compound) purpose. Two forms of sufficiency are defined: singly-sufficient and completely sufficient.

Singly sufficient purpose sets are purpose sets in which every element from the set dominates a specific other singleton purpose. An SPS can consist of only singly sufficient sets in which case it will be an SPS for a singleton purpose – which leads to the concept of single sufficiency.

Before single sufficiency is defined it is necessary to provide the definition of a sub-lattice (which supports the concept of single sufficiency).

**Definition 3 Sub-lattice.** A sub-lattice is a mathematical structure that describes a lattice that is contained within another lattice.

$S$ is a sub-lattice of $L$, if and only if $S \in L$, and $S$ is a proper bounded lattice. ■

**Definition 4 Single Sufficiency.** Given a purpose lattice $PL$, a purpose set $P$ is singly sufficient for a singleton purpose $\phi_i$, if there exists a lattice $S$ which is a sub-lattice of $PL$, such that $\forall p \in P, \exists(\phi_i, p) \in S$, $\forall(a, p) \in S, a = \phi_i$, and $\forall(x, y) \in PL \setminus S, x \neq \phi_i$.

■

Completely sufficient purpose sets are purpose sets such that all purposes in the set dominate at least one purpose in a compound expression; and that all purposes in the compound expression is dominated by some purpose in the purpose set. More formally:

**Definition 5 Complete Sufficiency.** Given a term $A$ from a compound expression, with a set $A_T$ representing all the purposes from this term which were not specified as negative purposes and the set $A_N$ representing all the purposes that were specified as negative purposes in this term.

A purpose set $P$ is completely sufficient if and only if it is the case that for all $p$ in $P$, $a \leq p$ for all $a$ in $A_T$.

■

A purpose set cannot be sufficient in any way if it contains purposes that dominate purposes specified as negative purposes in the term from the compound purpose against which it is being compared.

To ease verification, negative purposes and purposes that dominate purposes specified as negative purposes are placed in a structure called the complement to suitable purpose set (CSPS) discussed in more detail in the next section.

## 5.2 CSPS

Recall that a negative purpose is used to indicate that a certain purpose or several purposes cannot be used to gain access to a 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 $\Phi'$ is written as $\overline{\Phi'}$.

During verification $\overline{\Phi'}$ is examined to ensure that the *statement of intent* does not contain a negative purpose. CSPS is populated during the binding phase, avoiding unnecessary computation during the limitation phase.

The CSPS is used as a black-list (or closed-list) of purposes that may not appear in any *purpose set* in the SPS. [Section 7.3] provides a definition of the *and-not* operator that is defined for placing purposes in the CSPS.

### 5.3 Definition of the SPS

The unambiguity and sufficiency properties, along with the CSPS that was introduced provide a foundation for the SPS that was presented in example 3. A formal definition of the SPS is now presented.

**Definition 6 suitable purpose set.** An SPS for $\Phi$ 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 $\Phi$ (either singly, or complete), and
3. All possible **unambiguous** and **sufficient** purpose sets for $\Phi$ are present in the SPS, and
4. No element from any of these purpose sets are either included in the purpose expression as a negative purpose, or dominate purposes included in the purpose expression as negative purposes.

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

■

The introduction of the SPS and its definition provides the foundation for the verification method. At this point the construction of the SPS from a compound purpose expression must be examined in more detail. Hence, the operation of the compound purpose operators presented in section 2.2 is defined (their presentation in the mentioned section was only notational in nature).

To keep the operation of the operators consistent their operands are limited to SPSs. However, since (notationally) the compound expression consists of singletons as operands to the operators, it is necessary to provide a method for converting a singleton to an SPS. This very first step in the verification process is known as *creating Initial SPS Sets* or *I-Sets* and is discussed in the following section.

### 5.4 I-Sets

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

An *I-Set* is an appropriate operand for the compound operators. It is appropriate since each *I-Set* is an SPS [see A.1].

**Definition 7 I-Set.** A *I-Set* for a singleton purpose, $\phi_i$, is a set that contains purposes sets such that every element in each purpose set dominates $\phi_i$ and no two elements in a purpose set dominate each other.

The I-Set is easily constructed algorithmically as follows:

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

■

By this definition an *I-Set* is thus a set of unambiguous, completely sufficient purpose sets, making it an SPS [see A.1] for a proof of this statement).

The construction of a power set is of order $O(2^n)$. Thoughts on keeping the run-time for the creation of *I-Sets* minimal is presented in section 9.

The compound purpose expression, thus contains only SPSs as operands, and compound purpose operators that operate on these operands.

## 6 Verification

Final verification of the compound reason against the compound purpose is done by ensuring that the reasons that were presented are a subset of the SPS of the compound purpose, and is formally defined below.

**Definition 8 Verification of Compounds.** Access to a datum protected by a (compound) purpose $\Phi$ is granted when the RS representing the statement on intent is a subset of the SPS representing $\Phi$.

■

The previous sections provided information on the goals of the verification method as well as the definitions of the structures that are needed to support verification. The following section now provides the definitions of the operators that are used to create SPSs from their operands.

## 7 Compound Purpose Operators

In this section the detail of the operators for compound purposes is presented.

There are two requirements from the discussions presented in the previous section:

1. The compound purpose operators must produce an SPS from it's operands.

2. The compound purpose operator must produce an SPS that is representative of its semantic definition, for it's operands.

Recall that by definition an SPS will contain all the possible conjunctions of purposes that dominate the expression the SPS represents. Therefore, if the operator's two operands are SPSs, and it produces a valid SPS, both the above requirements have been met.

A discussion of each of the compound operators is now provided to determine if the above requirements are indeed met.

## 7.1 And ($\cdot_p$)

$\cdot_p$ must produce an SPS that will require the data user to present a purpose (or a conjunction of purposes) that dominates both operands of $\cdot_p$.

To accomplish this, assume that each operand of the and operator is a valid SPS. It is therefore clear that conjunctions of sets from the SPSs of the first and second operands will produce a set of sets which may constitute an SPS. The interesting problem is determining how to combine the sets from the operands to produce a valid SPS.

An intuitive approach would be to place the union of each set from the first operand with each set from the second operand in the result. However, this may produce ambiguous sets in the result. Consequently a new operator for combining sets is defined (definition 9). This operator guarantees that the result of a combination (*union*) operation is an unambiguous set.

**Definition 9 Combine Operator $\sqcup$.** $\sqcup$ is recursively defined as follows:

$$\{\} \sqcup \{a_0, \ldots, a_n\} = \{a_0, \ldots, a_n\} \tag{1}$$

$$\{a_0, \ldots, a_n\} \sqcup \{b_0, \ldots, b_m\} = \{a_1, \ldots, a_n\} \sqcup X \tag{2}$$

$X$ is defined for three cases.

1. $X = \{b_0, \ldots, b_m, a_0\} \Leftrightarrow \forall b_i | a_0 \not\leq b_i \wedge b_i \not\leq a_0$.
2. $X = \{b_0, \ldots, b_m\} \Leftrightarrow \exists b_i \mid a_0 \leq b_i$.
3. $X = (\{b_0, \ldots, b_m\} \setminus \{b_i | b_i \leq a_0\}) \cup \{a_0\} \Leftrightarrow \exists b_i \mid b_i \leq a_0$.

■

The combine operator produces a set which is unambiguous, and the definition of the *and* operator (definition 10) makes use of $\sqcup$ to produce an SPS from it's operands. A proof that $\cdot_p$ produces a valid SPS is presented in [section A.2].

**Definition 10 And Compound.** $\Phi'_1 \cdot_p \Phi'_2 = \{x \sqcup y | x = j \setminus \overline{\Phi'_2}, j \in \Phi'_1$ and $y = k \setminus \overline{\Phi'_1}, k \in \Phi'_2\}$

■

The following section presents the definition of the *or* operator.

## 7.2 Or ($+_p$)

$+_p$ 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 $\cdot_p$. Because it is less restrictive it will, intuitively, have more purpose sets that form part of the resultant SPS.

The 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 $+_p$ is thus easily accomplished in view of $\cdot_p$: include the sets from both operands individually, and include the result of $\cdot_p$ on the operands.

**Definition 11 Or Compounds.** $\Phi'_1 +_p \Phi'_2 = (\Phi'_1 \cdot_p \Phi'_2) \cup \Phi'_1 \cup \Phi'_2$

■

A proof of the validity of the result as an SPS is presented in [section A.3].

## 7.3 Andnot ($\cdot\neg_p$)

Functionally, $\cdot\neg_p$ should ensure that no purpose that is suitable for the 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".

It simply removes all those purposes which are suitable for the second operand from the first operand's SPS, and adds them the CSPS for the first operand. This allows the completeness test (introduced in definition 6) without having to add purposes from the lattice which would not have violated the unambiguity and sufficiency properties of the SPS.

**Definition 12.** For any andnot-compound $\Phi \cdot \neg_p \phi$:

1. $L = \{x | \phi \leq x \wedge x, \phi \in DPS\}$
2. $L' = L \setminus \{1\}$.
3. $\Phi' = \{X | X = Y \setminus L', \forall Y \in \Phi'\}$

4. $\overline{\Phi} = \overline{\Phi} \cup L'$

■

An important aspect of $\cdot\neg_p$ is the fact that the least upper bound of the lattice is 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 the enterprise are required to perform by law, it is necessary to ensure it remains as a legitimate way of accessing information.

From the definition it is clear that $\cdot\neg_p$ 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 that the andnot operator produces a valid SPS for its operands, and a proof is omitted.

## 8 Compound Reason Operators

The operators that are used to construct the SPS for a particular compound purpose expression have now been introduced. In this section the operators that will be used to construct RSs is considered in more detail.

The compound reason operators converts a statement of intent to a representative (and logically equivalent) set. At the start of evaluation of a compound reason the singleton purposes that form part of the compound reason are converted into sets of the form $\{\{\phi\}\}$; these sets are used as the operands to the compound reason operators. Consider for example the simple compound reason $\Gamma = \phi_9 +_r \phi_2 \cdot_r \phi_7$. All the singleton purposes are transformed into sets which will be passed as operands to the operators, thus $\{\{\phi_9\}\} +_r \{\{\phi_2\}\} \cdot_r \{\{\phi_7\}\} \equiv \Gamma$.

The purposes that are specified in the reason expression are transformed regardless of their existence, and the fact that they may dominate each other. It is not necessary for the system to rectify mistakes in the user's intent statement – 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 [Van Staden and Olivier 2006].

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

Note that $+_r$ 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, they cannot be combined as was done for the $\cdot_r$. Consider the access request and test: $\phi_1 \cdot_p \phi_2 \leq \phi_1 +_r \phi_2$. The data user is indicating that they will use the data for either one of the two stated purposes. In this case access cannot be granted, since the user is stating that the data will be used for either one of the two purposes, *or* both. The policy, on the other hand, states that the data will only be used for both purposes. Therefore, by definition: $\phi_2 \cdot_p \phi_2 \not\leq \phi_1 +_r \phi_2$ [Van Staden and Olivier 2005].

$\cdot_r$, and $+_r$ are now presented more formally in definitions 13 and 14.

**Definition 13 And Operator (Reasons).** $\Gamma'_1 \cdot_r \Gamma'_2 = \{x \cup y | x \in \Gamma'_1 \wedge y \in \Gamma'_2\}$ ■

**Definition 14 Or Operator Reasons.** $\Gamma'_1 +_r \Gamma'_2 = \Gamma'_1 \cup \Gamma'_2$ ■

All the operators for compound purposes have now been presented formally, as well as the algorithm for performing verification. The following section provides some thoughts on implementation aspects of the verification algorithm.

## 9 Implementation Thoughts

The paper presents the reader with the principle of using sets to verify compound reasons against compound purposes. In this section some thoughts on a possible approach to implementation is provided. These thoughts are provided to show that the runtime complexity of a system that uses compound purposes during the binding and limitation phase is acceptable.

Sets, and set operations are assumed to be accomplished by using a string of bits. Each bit corresponds to a single purpose from the DPS. The mapping of a bit's position to the purpose it represents is arbitrary, and no limitation need be placed on it. Using this representation, union operations are done using a *bitwise or*, set difference operations are accomplished using a combination of *exclusive or*, and *bitwise and*'s.

Assume also that as a special requirement, there will be a set that holds the elements that dominate a particular purpose for each purpose in the DPS, called the *dominance set.*

**Definition 15 Dominance Sets.** A dominance set for a purpose $\phi_i$ ($\delta_{\phi_i}$) holds all the purposes that are dominated by $\phi_i$, including $\phi_i$ (recall that $\phi_i \leq \phi_i$).

Since all dominance sets will include the GLB of the lattice, it can be removed without any loss of generality.

Dominance sets can easily be created and maintained during the lifetime of the PL: whenever a new purpose $\phi_y$ is inserted in the PL, it's dominance set is the union of the dominance sets for all the purposes that it dominates. Additionally, the new purpose may dominate other purposes – the new purpose is therefore added to the dominance sets of all the purposes that it dominates.

The dominance set is used to speed up the construction of *I-Sets*; and this technique is discussed in the following section.

### 9.1 Constructing I-Sets

Constructing an I-Set requires that all the sets with purposes that dominate $\phi_i$ be found. These sets can easily be constructed using an iteration over $\delta_{\phi_i}$. Since all sets are represented using bit-strings, one can cycle through a numeric iteration and perform a *bit-wise and* with the dominance set thereby deriving the set of sets that dominate $\phi_i$. Each iteration will produce a set $(O)$ that will eventually be a purpose set $(P)$, however, at this point each $O$ may be ambiguous.

To ensure that each $O$ is unambiguous, it is sufficient to remove the purposes that may dominate other purposes. Hence, $P_m = a \cup O_m \setminus \delta_a, \forall a \in O_m$.

### 9.2 Evaluation of compound purposes

It is intuitive that a purpose bound to data is not supposed to change frequently. Therefore a quick method for ensuring optimised verification is to have the verification system evaluate a compound purpose as needed, and to store the result for later use. It is even possible to store the result of an access request – the system will then do a lookup on the result of a verification involving a specific statement of intent, as well as a specific compound purpose. The use of bit-sets and bit operations, will also allow the runtime performance of the functions defined in this paper to be sped up significantly. A lack of space,however, precludes a detailed analysis of the run-time complexity of the functions.

## 10 Conclusion

Compound purposes are a unique and desirable way of extending expressiveness and flexibility of privacy aware systems that protect access to PII. In this paper a quick review of the principles behind compound purposes and reasons was provided, the notion of a negative purpose was introduced, and it was also shown how verification for compounds can be accomplished by introducing the SPSs, as well proving that the operators that are used to construct SPSs generate valid SPSs that can be used for verification. Reconstructing the SPS when the purposes change was not considered and will be reported on elsewhere.

## References

[Agrawal et al. 2004] R. Agrawal, R. Bayardo, C. Faloutsos, J. Kieman, R. Rantzau, and R. Srikant. Auditing compliance with a hippocratic database. In *Proceedings of the 30th VLDB Conference*, 2004.

[Agrawal et al. 2002] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.

[Ashley et al. 2003] P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise privacy authorisation language (EPAL 1.1). Technical report, International Business Machines Corporation, 2003.

[Bernat et al. 2008] L. Bernat, N. Mansfield, and A. Carblanc. RFID: OECD policy guidance, a focus on information security and privacy. Technical report, OECD, 2008.

[Byun et al 2005] J.-W. Byun, E. Bertino, and N. Li. Purpose based access control of complex data for privacy protection. In *SACMAT'05*, Stockholm, Sweden, June 2005. ACM.

[Camenisch et al. 2005] J. Camenisch, A. Shelat, D. Sommer, S. Fischer-Hübner, M. Jansen, H. Kraseman, R. Leenes, and J. Tseng. Privacy and identity management for everyone. In *DIM'05*, Fairfax, Virginia, USA, November 2005. ACM.

[Castano et al. 1994] S. Castano, M. Fugini, G. Martella, and P. Samarati. *Database Security*. ACM Press, 1994.

[Chaum 1981] D. L. Chaum. Untraceable electronic mail, retrun addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[Clarke 2008] R. Clarke. *Business Cases for Privacy Enhancing Technologies*, chapter 7, pages 135–155. IRM Press, 2008.

[Cranor et al. 2002] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. The platform for privacy preferences (P3P1.0) specification. Technical report, W3C, Available at http://www.w3.org/TR/P3P/, 2002.

[Danezis et al. 2003] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[Dingledine et al. 2004] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[Fischer-Hübner 2001] S. Fischer-Hübner. *IT-Security and Privacy: Design and Use of Privacy-Enhancing Security Mechanisms*. Springer-Verlag, 2001.

[Hansen et al. 2008] M. Hansen, A. Schwartz, and A. Cooper. Privacy and identity management. *IEEE Security and Privacy*, 2:38–45, 2008.

[Hes et al. 1998] R. Hes and J. Borking, editors. *Privacy Enhancing Technologies: The Road to Anonimity*. Dutch DPA, revised edition, 1998.

[Karjoth and Schunter 2002] G. Karjoth and M. Schunter. A privacy policy model for enterprises. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*. Springer-Verlag, June 2002.

[Massacci et al. 2005] F. Massacci, J. Mylopoulos, and N. Zannone. Minimal disclosure in hierarchical hippocratic databases with delegation. Technical Report DIT-05-051, Informatica e Telecomunicazioni, 2005.

[Oberholzer and Olivier 2006] H. J. Oberholzer and M. S. Olvier. Privacy contracts incorporated in a privacy protection framework. *International Journal of Computer Systems Science and Engineering*, 21(1):5–16, 2006.

[oecd2002] Working Party on Information Security and Privacy. Inventory of privacy enhancing technologies. Technical Report JT00119007, Organisation for Economic Co-operation and Development, 2002.

[OECD 1980] Organisation for Economic Cooperation and Development. OECD guidelines on the protection of privacy and transborder flows of personal data. Technical report, Organisation for Economic Co-operation and Development, 1980.

[OECD 2007] At a crossroads: "personhood" and digital identity in the information society. Technical report, 2007.

[Olivier 2005] M. S. Olivier. Flocks: Distributed proxies for browsing privacy. In G. Marsden, P. Kotzé, and A. Adesina-Ojo, editors, *Proceedings of SAICSIT 2004 — fulfilling the promise of ICT*, pages 79–88, Stellenbosch, South Africa, October 2004.

[Pfitzmann and Hansen 2007] A. Pfitzmann and M. Hansen. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. Electronically Published, July 2007.

[SA ECT Act 2002] Republic of South Africa: Electronic Communications and Transactions Act No 25 of 2002. Published Electronically, Available from: http://www.gov.za, 2002.

[SA FIC Act 2001] Financial intelligence centre act 38 of 2001. Published Electronically. Available from `https://www.fic.gov.za/DownloadContent/LEGISLATION/ACTS/01.a38-01b.pdf`.

[SA Proatia 2002] Republic of South Africa: Promotion of Access to Information Amendment Act No 54 of 2002. Published Electronically, Available from: http://www.gov.za, 2002.

[SA RIC Act 2002] Regulation of interception of communications and provision of communication-related information act 70 of 2002. Published Electronically. Available from `http://www.info.gov.za/acts/2002/a70-02/`.

[Van Staden and Olivier 2005] W. J. van Staden and M. S. Olivier. Purpose organisation. In *Proceedings of the fifth annual Information Security South Africa (ISSA) Conference*, Sandton, Johannesburg, South Africa, June 2005.

[Van Staden and Olivier 2006] W. J. van Staden and M. S. Olivier. Extending SQL to allow active secification of purposes. In *Third International Conference on Trust and Privacy for Digital Bussines*, Krakow, Poland, 2006. Springer-Verlag.

[Van Staden and Olivier 2007] W. J. van Staden and M. S. Olivier. 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.

[WikiPedia2010] Open road tolling. Published Electronically. Available from `http://www.wikipedia.com/wiki/Open_road_tolling`.

## A Theorems, and Proofs

In this section formal definitions and proofs surrounding the work that was presented in previous sections are provided. This is done to avoid obstructing the reader's view with notation and reasoning when reading the concrete ideas presented in the previous sections.

### A.1 I-Sets

**Theorem 16 I-Set validity.** *An I-Set $I_S(\phi_i)$ is an SPS for $\phi_i$.*

■

*Proof.* 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 7), $X$ is singly-sufficient for $\phi_i$.

*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 $\phi_i$, 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 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 $\phi_i$ 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 dominates $\phi_i$. 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 6 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.

■

## A.2 And Operator Validity

**Theorem 17 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.* Assume that $\Phi_1$' and $\Phi_2$' are both SPSs. $\Phi_1$' and $\Phi_2$' represents the SPSs for $\Phi_1$ and $\Phi_2$ respectively. Take $X''$ to be the result of $\Phi_1$' $\cdot_p$ $\Phi_2$'.

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 $\Phi_1$, $\Phi_2$ or both.

*Completeness*: Firstly, suppose that $X''$ 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 $\Phi_1$, or $\Phi_1$ and $\Phi_2$, and $L$ must be suitable suitable for $\Phi_2$, or $\Phi_1$ and $\Phi_2$. 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 $X''$: either $Y$ was included as part of an operation with the $\sqcup$ operator, thus $Y \sqcup Z \in \Phi'_1$, or since $Y = K \sqcup L$, $Y$ must already be in $X''$.

Secondly, $\Phi'_i$ cannot contain any elements from $\overline{\Phi'_i}$ (definition of the and-not operator in section 7.3). It is, however, possible for $\Phi_1$ (not the SPS) to contain purposes that are elements in $\overline{\Phi'_2}$. This will happen when $\Phi_1$ permits a purpose $n$ to be used, but $\Phi_2$ 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.

■

## A.3 Or Operator Validity

**Theorem 18 Or Operator 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 get access to data to which is protected with the operands passed to the* or *operator.*

■

*Proof.* The proof for the or-compound follows from the and-compound proof. Assume that $\Phi'_1$ and $\Phi'_2$ are SPSs for $\Phi_1$ and $\Phi_2$ respectively.

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

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

*Sufficiency*: As with the *and* operator, the result of $\Phi'_1 \cdot_p \Phi'_2$ is sufficient, and so are $\Phi'_1$ and $\Phi'_2$ (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.

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

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

■

## A.4 Final Verification

**Theorem 19 Verification of Compounds.** $\Phi \leq \Gamma \Leftrightarrow \Gamma' \subseteq \Phi' \wedge \forall A \in \Gamma', A \cap CSPS_{\Phi} = \emptyset$ *With* $CSPS_{\Phi}$ *the CSPS for* $\Phi$.

■

*Proof.* Since the SPS of the compound purpose holds all the valid combinations of purposes which can be presented in order to get 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.