SQL's Revoke with a view on Privacy

Wynand JC van Staden
Information and Computer Security Architectures (ICSA) Research Group
University of Pretoria
Pretoria
South Africa
0001
wvs@wvs.za.org

Martin S Olivier
Information and Computer Security Architectures (ICSA) Research Group
University of Pretoria
Pretoria
South Africa
0001
http://www.mo.co.za

ABSTRACT

Protecting access to data that can be linked to an individual (or personal identifiable information (PII)), thereby seeking to protect the individual's privacy can be accomplished through legislation, organisational safeguards, and technology. Of particular interest and the focus of this paper is the technological means by which data is protected, in particular we are considering the mechanisms of purpose binding and limitation which facilitate the organisational safeguards. Purpose binding allows an enterprise to specify their purpose with collected data, and purpose limitation controls access to information based on these purpose bindings.

Technologies that implement the aforementioned safeguards of PII forms a subset of a set of technologies commonly referred to as Privacy Enhancing Technologies (PETs). Many legacy systems do not employ these safeguards, even though it can be accomplished by providing "wrapper" technologies which reside on top of these legacy systems.

This article continues work done by the authors in which extensions to SQL was proposed in order to integrate PETs with structured databases. The extensions showed that access to data through SQL can be controlled non-intrusively, and that the general discretionary access control model provided by many database management systems can still be enforced. In our previous work the extensions were limited to the SQL *grant* and *select* statements.

In this article we propose a model for revoking privileges from database users, and thus consider the SQL *revoke* statement. We also show that the general principles of revoking privileges remain true for our proposed model. We also briefly consider extensions to the commands from the Data Manipulation Language (DML) that was not considered, being *insert, delete*, and *update*.

Keywords

Access Control, Compound Purposes, Privacy, Purpose Binding, SQL

Categories and Subject Descriptors

H.2.0 [**Information Systems**]: Security, Integrity and Protection; K.4.1 [**Computing Milieux**]: Computers and Society—*Privacy*

General Terms

Languages, Security

1. INTRODUCTION

Purpose binding (associating purposes with data) allows the owner of a database – typically an enterprise – to state for which purposes data will be used. This ideally means that data should only be released by the Database Management System (DBMS) if the subject that is requesting the data provides adequate reasons for wanting the data. The current research in this field is focused on the two components of privacy implementation: specification of purposes, and associating these purposes with the data, as well as verifying access requests.

In most cases the solution that is presented associates a set of purposes that can and cannot be used by a subject with that subject's profile. An access request initiates a process in which the Access Control System (ACS) examines the subject's profile to determine if they have the appropriate purposes to access the information. This means that subjects will get access to information without clearly specifying why they want it – allowing them to deny knowledge in the event of wrong-doing. Organisational safeguard may very well hold the perpetrator accountable for his actions, however, we are considering a technological solution that will link a subject's statement of intent with his request for access to the data. In this way, audit trails become more verbose, and a clear statement about the infraction can be made in the event that it does take place.

In a previous article [19] the authors argued that subjects must actively state why they want data, and those statements must be compared against their profiles. This required extensions to the SQL `select` statement. It is also generally accepted that privacy related data cannot be accomplished by using a Discretionary Access Control (DAC)

model. This means that many of the current *legacy* systems cannot provide privacy protection to the individuals on which they hold data, save for standard security mechanisms. The authors' previous work also illustrated that it is possible to support these legacy systems, and even extend access control to be more semantically rich, and fine-grained by extending the *grant* statement.

Our work introduced a Hybrid-DAC (HDAC) model for access control [19]. In this model, the System Security Officer (SSO) will bind purposes to data, and then *grant* subjects access to that data. Thus, there is still a central authority which decides on the access privileges, but subjects can now pass these privileges on to other subjects. To access objects, subjects must provide the correct reasons for doing so.

This new access control model would be incomplete if it were not possible to revoke those access privileges that were granted to subjects. Thus it must be considered how the *revoke* statement would operate in this context, and what a syntax for such a *revoke* statement would look like.

This paper therefore continues previous work done by the authors [18, 19], by considering the syntax for the SQL *revoke* statement. Extensions to the other data-manipulation statements are also provided in order to provide a more verbose audit trail.

The paper 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 it from another subject). It does, however, suggest a step in the direction of access control using purposes in current database technology.

This paper contributes by completing the HDAC model in which subjects can now not only grant privileges to other users, but also revoke those privileges. It does so by considering the mechanism for removing a purpose expression from another expression, and showing how it can be ensured that revoking privileges do not afford more access rights to subjects, thereby keeping the system state valid.

The rest of this paper is structured as follows. Section 2 provides background information as well as work that is related to the work presented in this article. Section 3 discusses the mechanisms for revoking privileges from users in the HDAC model, and also presents proposed extensions to the *revoke* statement to accomplish this. Section 4 considers other issues surrounding extensions to the insert, delete, and update statements. Finally section 5 provides concluding remarks.

2. BACKGROUND AND RELATED WORK

Allowing users to communicate over the internet, and allowing them to subscribe to services rendered by enterprises seems a commonplace activity today. It is also this commonplace activity which has the privacy fundamentalists raising their eyebrows. It has long been anticipated that technology would allow the collection of vast amounts of information on people. This anticipation is made evident by the amount of research that has been put into the protection of data, as well (more recently) the protection of private information.

As early as the 1980's the Organisation for Economic Cooperation and Development (OECD) published their principles of fair use, outlining the "fair use" of private information [14]. Based in those principles, the IPC/Registratiekamer [10] collaborative effort between the Canada and the Netherlands produced a document outlining the principles of a system that would support the privacy of individuals. Technologies that implement these principles are commonly referred to as Privacy Enhancing Technologies (PETs).

A Privacy Enhancing Technology (PET) by its definition should protect the privacy of the individual through one of four mechanisms. These are anonymity, pseudonimity, unlinkability, and untraceability [16]. These restrictive mechanisms means that a system should only in rare occasions be able to link an individual to their privacy related information (or Personal Identifiable Information (PII)).

Examples of systems that provide anonymity are anonymous mailers such as mix-minion [6]. Mix-minion is based on an earlier type of PET first introduced by David Chaum in the early 1980's [4]. Mix-minion can also provide a level of untraceability, and unlinkability (with sender/receiver variations) [9, 16]. Another example, perhaps more well known is TOR [8], which allows anonymous web-browsing. Flocks [15] also falls into the same category as TOR.

It is sometimes necessary for the enterprise to store information on individuals for conducting business. For example an express delivery company cannot conduct business if they do not store physical addresses for their clients. They may provide some protection through pseudonyms, but sooner or later someone is going to need a physical address. In these cases protection of privacy is offered through the use of purpose binding and purpose limitation [9], as outlined in the document by the OECD [14].

2.1 Purpose Binding

Purpose binding refers to the act of associating a purpose with each datum, that is stating what the enterprise will use the data for. Purpose limitation is the act of only providing access to information which will be used for the purpose stated.

Many examples of technology that supports purpose bindings already exist. Perhaps the best known is that of Platform for Privacy Preferences (P3P) [5]. Unfortunately P3P does not offer a mechanism to enforce the promises made by P3P policies. And so was born the Enterprise Platform for Privacy Preferences (EP3P) [17], which as a mechanism should integrate into technology and enforce the privacy promises made. The EP3P as a standard has already led to the design of the Enterprise Privacy Authorisation Language (EPAL) [2]. The Extensible Access Control Markup Language (XACML) [12] also includes mechanisms for purpose binding.

2.2 Purpose Limitation

An example of a system that implements the purpose limitation phase is the Hippocratic Database Management System (HDBMS) [1]. The HDBMS protects information by limiting access to information based on the purposes that form part of a database subject's profile when access to data is

requested. New research on the HDBMS also includes auditing of queries to try and determine if data was used for inappropriate reasons. A model for controlling access in a Role Based Access Control (RBAC) scenario has also been proposed [3].

In examples like theHDBMS purposes are typically represented as strings, and have no relationship to other purposes. Other examples notably by Byun et al [3], XACML [12], Fischer-Hübner [9], and Karjoth et al [11], and Van Staden et al [18] do place purpose in hierarchies of some sort. Byun, XACML, and the EPAL all propose that access to a datum bound to a purpose can only be gotten if the subject may use all the child purposes of that purpose. Fischer-Hübner, Karjoth et al, and Van Staden et al proposed that purposes in the hierarchical form can subsume each other, that is given a purpose x, for any purpose y lower in the hierarchy, and for which a path exists between x and y, y is considered to subsume x. This means that any datum protected by x, can also be accessed by presenting y. This allows that placement of general purposes near the top of the hierarchy and specific purposes near the bottom of the hierarchy. Single purposes such as x and y which represent an atomic concept are referred to as singleton purposes.

2.3 Purpose Lattices and Compound Purposes

The notable difference between Karjoth and Van Staden's work is that the latter author places purposes in a lattice (see figure 1). The lattice mechanism allows the creation of compound purposes [18]. A compound purpose is a purpose which is constructed from existing purposes in that lattice through the use of special operators, first presented in [18]. Compound purposes allows the creation of access control

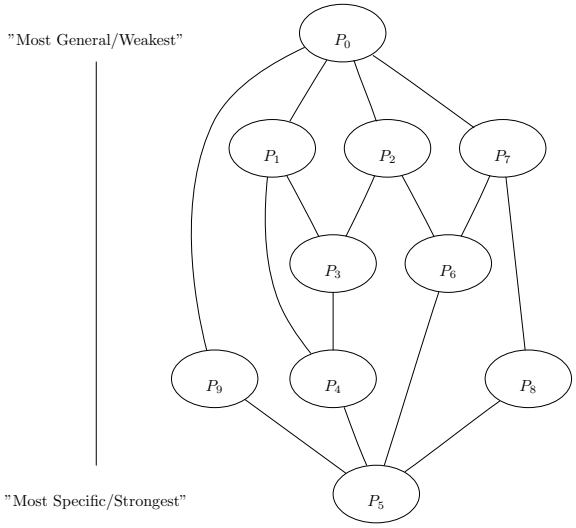

Figure 1: A simple purpose lattice

policies which indicate the dependencies between the purposes for which data is stored. For example, one can specify that a credit card number will only be used for "billing *or* refunding". Given the fact that purposes can subsume each other this means anything at least as specific as either billing or refunding (or both) can be presented to gain access to a datum. Thus, the data will be released for any of those purposes.

With compound purposes one can even state that an address is stored for "invoicing *and* shipping". That means data is only used when an invoice is sent along with a parcel, never in just one case. Any purpose at least as specific as both invoicing and shipping can be presented as a reason to get access to the information. In this paper we will use the term purpose to refer to the intended use for which an organisation has collected data and the term *reason* to refer to the intended use for which the data has been requested.

The operators used to construct compound purposes are $\cdot_p, +_p, \cdot_r, +_r$, and $\cdot_p \neg$. The reason for the different operators for reasons $(\cdot_r, +_r)$, and purposes $(\cdot_p, +_p, \cdot_p \neg)$ is found in the verification mechanism for compound purposes. The verification mechanism falls outside of the scope of this paper and will be reported on elsewhere. The $+_{p|r}$ represents the operator representing an *or* compound, and the $\cdot_{p|r}$ represents the *and* compound operator. The $\cdot_p \neg$ operator is used for exclusion, that is, it is used to indicate what purposes may not be used as reasons when requesting access to information. We show this in the following example.

Example 1. Consider figure 1 which might represent a purpose lattice for an enterprise. Compound purposes allows the Database Administrator (DBA) to bind a purpose such as $\phi_9 \cdot_p \phi_1 +_p \phi_7$. Access to this datum is only granted if the subject presents a reason at least as strong as the stated compound purpose. The subject can,for example specify $\phi_3 +_r \phi_6$ to gain access. This means that the subject will use the data for purpose ϕ_3 or for ϕ_6. Only if the data can be used for those purpose (either one alone, or both together) will it be released. Had the subject stated $\phi_3 \cdot_r \phi_6$ it would have meant that he will use the information for both purposes. □

The use of the or-compound signifies that the subject may be unsure beforehand for which purposes a datum will be used. For example, a short term insurance agent requests a client's telephone number for "updating client information" or "for selling a new insurance policy". Until the agent actually phones the client he will not know if the client's details has to be updated.

When a purpose y subsumes another purpose x, or is sufficiently specific to gain access to a datum we write $y \geq x$. A compound purpose, can of course also be sufficiently specific to gain access to a datum. In the above example it is clear that $\phi_3 \cdot_r \phi_6 \geq \phi_9 \cdot_p \phi_1 +_p \phi_7$. Verification using compounds, as well as formal definition of the operators will be reported elsewhere.

The use of a lattice and compound purposes, also allows the customisation of privacy agreements [20] as first proposed by Oberholzer et al [13]. An enterprise can publish their least specific purpose Φmin for data, as well as their most specific purpose Φmax. The customer can then customise his privacy agreement with the enterprise by providing a compound purpose Φc such that $\Phi max \geq \Phi c \geq \Phi min$. The enterprise can thus access the information only if they pro-

vide a strong enough reason (at least as good as Φc) for doing so. The enterprise can specify Φmax as a way of indicating for what purposes they will use the data in order to conduct day to day business. If the customer specifies anything stronger, then the enterprise will not be able to conduct business.

2.4 Active Specification of Purposes and the HDAC

In previous work [19], the authors argued that subjects should not simply get access to data because they have the appropriate reasons in their subject profile. The whole point of protecting privacy is to create verbose audit trails (thus the risk in misusing data lies with the person requesting the data), and to make it difficult to access data for inappropriate reasons. Thus, we proposed extensions to the SQL *select* statement (presented in definition 1 from [19]) to allow users to specify their reasons for accessing information.

Definition 1. 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 DBMS, and r_i is a reason. □

Since access control policies in the database should be actionable through the Database Management System (DBMS) [7] we also extended the *grant* statement (definition 2, also from [19]) to allow the purposes which subjects may present to be specified. Thus a subject is given a set of purposes as part of his profile, but he must actively state what he is using data for.

Definition 2. GRANT <privileges> $[for_1]$ on <object> to <subject>
[with grant option $[for_2]$][for_3]

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

By extending the *grant* statement, we also showed that is was possible to construct a HDAC model. Generally protecting PII is not considered possible with a DAC model. However, using the notion of lattices and carefully constructing verification mechanisms we showed that one can allow subjects the opportunity to pass rights onto other subjects. The prime requisite is that the purpose privileges each subject receives must be at least as restrictive as the grantor's. This is accomplished by ensuring that the purposes the grantee (subject receiving the privileges) receives (Φs) are dominated by the grantor's privileges (Φg), that is $\Phi_g \geq \Phi s$ (where Φ is a compound purpose). This means that a grantee will never have access to data that is protected with more specific purposes than the grantor has. Generally, we expect very specific purposes for use to be bound to sensitive data. In other words, as the chain of grants progresses, each subject down the chain will ultimately only be able to access less data, or the same data as the subject that granted him access.

In both the *select* and *grant* statements, it was shown that the "for" clauses could be omitted with well defined results [19]. In those cases, the least upper bound of the lattice is used as the reasons for performing the actions. The special keyword *default* was also introduced to be used in the place of object names $o_1, o_2, \ldots$ in the *select* definition. This keyword indicates the default reason for which a subject performs an action. Omitting the "for" clauses thus actually results in a rewrite of the statement with the "for" clauses using the least upper bound of the lattice as the default reason.

The HDAC consists of a central authority which grants access to subjects, and in turn allows them to grant the access onward. Thus it differs from the standard DAC model in that subjects do not own objects in the database, but they can at their discretion grant other subjects access to data (provided they have access themselves).

3. REVOKING PRIVILEGES IN THE HDAC MODEL

In this section we consider the action of revoking privileges from subjects in the HDAC model presented in section 2, and elsewhere [19]. Since the model presented here fits tightly into standard DBMS technology, revoking is accomplished in the same fashion as in current DBMSs, through the use of the SQL *revoke* statement. We afford revoking capability to subjects 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*), and special permissions to grant privileges onward, our *revoke* is a bit more complex.

The extended *revoke* must allow the revoker (the subject revoking a privilege) to remove all the privileges that the standard *revoke* does. In addition to this it must also allow the revoker to remove the privilege of presenting certain purposes when requesting access to information. It must also allow the revoker to remove the privilege of presenting certain purposes when granting privileges onward.

A final note about the aims of the extended *revoke*. Firstly, it must allow a subject, who is not concerned with privacy related data, to use the *revoke* as per usual. Thus, it 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 subject s, who had *grant* option, then that privilege must be revoked from all subjects the received the privilege from s.

3.1 Revoking Privileges in the HDAC model

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

3.1.1 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 subject's profile.

This representation allows simple removals of purposes from purpose expressions. The representation relies on the expansion of the purpose expression. Expansion means that a purpose expression will be in a simple form such as $C_1 +_p C_2 +_p \ldots C_n$, where every C_i is of the form $\phi_{x_1} \cdot_p \phi_{x_2} \cdot_p \ldots \phi_{x_m}$. This form allows us to consider an expression as a set of sets; every element in the set represents a term from the expression. The set representation allows easy reasoning about the validity of the technique presented here.

Our set representation for a purpose expression of the form $C_1 +_p C_2 +_p \ldots C_n$ (after expansion) is taken to be $\{\{\phi_{x_1}, \phi_{x_2}, \ldots, \phi_{x_n}\}, \ldots\}$. Every element of the set is a representation of a term formed using the conjunction operator ($\cdot_p$). Converting an expression to this set form can be done explicitly using well known and published computer algebra techniques; the expression is expanded, and terms are placed in sets, using the $+_p$ operator as a "field" separator. Finally, the $\cdot_p$ operator within the elements of the set is used as a "field" separator.

After the removal has been completed, it is possible to use the same techniques to convert the expression set back into an arithmetic form. Elements within the set are separated by the $+_p$ operator, and elements within the elements of the set are separated by the $\cdot_p$ operator. Once again using computer algebra techniques the expression can be factorised for easy maintenance.

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 purposes from purpose expression. When indicating that a (compound) purpose C has to be removed from a purpose expression P, we write $P \oslash C$.

3.1.2 Removing Singleton Purposes

Removal of a singleton purpose from a purpose expression is in essence extremely simple given our set representation of the purpose expression. This removal is presented more formally in definition 3.

Definition 3. 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

$$E_s \oslash \phi_i = \{Y | Y = X \setminus \{\phi_i\}, \forall X \in E_s\} \quad (1)$$

□

3.1.3 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 big 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 4

Definition 4. 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

$$E_s \oslash E_R = \{Y | Y = X \setminus Z, \forall X \in E_s \wedge \forall Z \in E_{R_s}\} \quad (2)$$

□

3.1.4 Valid Revokes

The removal of one expression from another expression is the important mechanism for managing the *revoke* in the HDAC. A second critical part is that of ensuring that a *revoke* leaves the system in a valid state. This implies that the removal, although legal with respect the definitions, might cause an incorrect resulting purpose expression. This incorrect expression could provide the subject with more access than he initially had.

In all cases a *revoke* must ensure that the result of the *revoke* action does not provide the subject with more access privileges than he initially had. This means that his 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 at least as restrictive as the revoker's.

Consider the simple grant graph presented in figure 2. Suppose that A received privileges α_0, and passed on $\alpha_{a\to b}$ to B, and $\alpha_{a\to c}$ to C. Suppose that these privileges were all legally granted onward with the *grant* as defined by us [19]. It is thus clear that $\alpha_0 \geq \alpha_{a\to b}$, and that $\alpha_0 \geq \alpha_{a\to c}$.

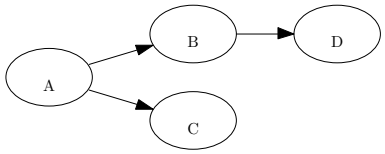

Figure 2: Simple Grant Graph

For a *revoke* to be valid, the state of the system depicted by the grant graph must still be such that $\alpha_0 \geq \alpha_{a\to b}$, and that $\alpha_0 \geq \alpha_{a\to c}$. We refer to this state as the least privilege property. Definition 5 provides the Least privilege property.

Definition 5. Any purpose privileges received by a subject B from subject A, must be such that subject B's privileges are at least as restrictive as subject A's. In other words, if subject A had privileges α_a and granted $\alpha_{a\to b}$ to B, then $\alpha_a \geq \alpha_{a\to b}$ must hold.

Removing a purpose from a purpose expression during a *revoke* will not guarantee this property to hold.

Theorem 1. Removing a purpose from a purpose expression during a *revoke* does not guarantee the Least Privilege Policy to hold. □

PROOF. We can show this to be true in a simple example. Consider a purpose privilege such as $\phi_1 \cdot_p \phi_2$, which indicates that the subject must present both reasons when accessing a datum. Suppose the subject that granted this privilege no

longer wishes the recipient of the aforementioned privilege to be able to present ϕ_2 when accessing data, issues $\phi_1 \cdot_p \phi_2 \oslash \phi_2 = \phi_1$. However, the subject can now get access to any data stored for purpose ϕ_1. This is clearly not a safe state ($\phi_1 \not\geq \phi_1 \cdot_p \phi_2$). □

To prevent this access violation, the revoke constraint is introduced (definition 6).

Definition 6. 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 $E_{rev} \geq (E \oslash E_r)$

There is also a difference in the use of purposes for stating why privileges are granted onward, and the privileges that are granted. This difference will be highlighted in the following section.

3.2 The Extended Revoke Syntax

The *revoke* statement must thus allow us to revoke two things: Firstly, it must allow either a *revoke* of a complete privilege (such as *delete*, *select*, or *update*), or a *revoke* of a sub-reason from reason. Secondly, it must allow the revoking of the grant option from a subject. The SQL 2003 draft standard is used as a basis for the syntax of the extended revoke statement (definition 7).

Definition 7. `<revoke> ::= REVOKE [ GRANT OPTION [FOR <for1>] FOR ] <privilege> [ FOR <for2> ] [, <privilege> [ FOR <for3 ] ] [, ... ] 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 4 it is clear that $E \oslash E =$, indicating that no reason my 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 equation 4 can be used to modify the reasons for which the revokee may grant onward.
3. If $\text{for}_{2..n}$ is omitted, then all the reasons for accessing data using the specified privilege is revoked, thus the privilege itself is revoked.
4. If $\text{for}_{2..n}$ is a valid compound reason, then as with the GRANT OPTION statement, equation 4 can be used to modify the reasons for which a subject may use with a particular privilege.

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 subject cannot get 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 subject 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 $\text{for}_{2..n}$.

Based on the revoke restriction, we thus have the following. If subject A granted subject B a specific access mode on data using the extended grant statement for a specific purpose (for ease of discussion we assume read access), and now wishes to revoke those purposes (either just some of them or all of them), then it is clear that $A_{for_1} \geq (B_{for_1} \oslash for_2)$ 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.

4. IMPLEMENTATION THOUGHTS

The extra access privilege attributes that are added to the subject'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 grantor'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, write, *insert*, or *delete* values from table, and is typically a simple array that indicates which access modes the subject has. The grant option is a simple true/false value which indicates if the subject may grant onward.

The hybrid model requires (naturally) more information to be stored. Only the access mode, and grant option columns need modification. The access mode is changed from a simple 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 as follows: $(read, \phi_1 \cdot_p \phi_2 +_p \phi_6)$. Currently we see no reason for access modes other than read and *delete* to have associated purpose privileges (see section 4.1), 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.

4.1 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. It is also an obvious question to ask whether or not the other data manipulation statements should be extended in the same fashion.

We consider each statement in due course in the following sections.

4.1.1 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 subject. From a privacy viewpoint, the enterprise storing the data states their reasons for doing so as part of a their privacy policy. Thus, any subject 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 that the subject be held 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 subject who is inserting the data to specify his reasons for performing the action of inserting the data into the database.

We propose 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. The reason being that 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 paper.

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

Definition 8. `<insert statement> [ for reason`$_1$ `]` □

4.1.2 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 thereof. Thus, there is only a single reason to update data, and as such it is not worth extending the *update* statement.

With respect to updating the privacy agreement between the enterprise and the data subject note the following: From work we have reported on elsewhere [20] we have proposed that an enterprise stores a minimum purpose for storing data, as well as a maximum purpose for storing data. The data subject provides a custom acceptance level for his data. This custom acceptance level falls somewhere between the minimum and maximum purposes. The data subject'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 subject to change his agreement with the enterprise. Again, we view this as an update to have the latest profile of the data subject.

4.1.3 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 data has reached its maximum retention period. The law might require the data to be deleted, and so forth.

We thus present a small change in the *delete* statement. We envisage that it may be desirable to protect PII by stating for which purposes it may be deleted. A subject must thus specify why he is deleting information. In the event where the information is not protected with "delete purposes", it may just be a good way of providing an audit trail.

Definition 9. `<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 sub-clause of the delete statement will access data. In light of this we need 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 in the where-subclause, it may be argued that a subject 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.

We therefore do not propose an extension which requires that a reason for accessing data specified in the where-subclause be given.

5. CONCLUSION

In this article extensions to the *revoke* and *delete* SQL statements were considered. These extensions provide the closing pieces of a HDAC model in which a central authority assigns permissions to subject, who can in turn pass those permissions to other subjects. Aside from this ability, users actively specify their intent with data.

Extensions to the *insert*, and *update* statements are not considered desirable at this point in time, however, the proposed implementation allows these extensions to be added at a later stage. Extensions to the *delete* statement were briefly considered in order to establish a foundation for possible future work. The extension to the *delete* statement are only considered to be a "nice to have" and not essential to the operation of the HDAC model.

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

Future work on this subject involves reporting on a proof of concept implementation (currently being developed) as well as flow-analysis for possible attacks to the system.

6. REFERENCES

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

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] C. J. Date. *An introduction to database systems*, volume 1. Addison-Wesley, seventh edition, 2002.

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

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

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

[11] 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.

[12] OASIS Access Control TC. OASIS extensible access control markup language (xacml) version 2.0. Technical report, OASIS, February 2005.

[13] 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.

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

[15] 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.

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

[17] M. Schunter and P. Ashley. The platform for enterprise privacy practices. Technical report, IBM, 2002.

[18] 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.

[19] 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.

[20] W. J. van Staden and M. S. Olivier. Using purpose lattices to facilitate the customisation of privacy agreements. In *Accepted for publication in the proceedings of the Fourth International Conference on Trust and Privacy for Digital Business*, Regensburg, Germany, September 2007. Springer-Verlag.

W van Staden and MS Olivier, "SQL's Revoke with a View on Privacy," in *Proceedings of SAICSIT 2007 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, L Barnard and RA Botha (eds), 181-188, Fish River, South Africa, October 2007