The Personal Model of Data
Towards a Privacy Oriented Information System
(extended abstract)
Joachim Biskup Hans Hermann Brüggenmann
Department of Computer Science, University of Hildesheim
Samelsonplatz 1, D-3200 Hildesheim, West Germany

Abstract
Privacy is considered as the individual's right of informational self-determination. Whereas previous approaches to security in information systems mainly focus on military needs our personal model of data has deliberately been designed to support privacy. In the model all individuals, users as well as data subjects, are represented by encapsulated objects. The data part of an object corresponds to the knowledge of the individual (about him/her and his/her relationship to other persons), the operation part corresponds to the possible actions of the individual. A person holds two independently distributed kinds of rights: it is acquainted with other persons, and possesses authorities on roles. Acquaintances can be dynamically granted and revoked; authorities are statically received during creation according to class membership; combined rights can be made temporarily available. The privacy policy confines a person to its acquaintances by query modification and refuses evaluation of expressions if the person cannot exhibit appropriate authorities. While the model is basically object oriented we can conveniently describe an application by non first normal form tuples and relations, and the data manipulation language is high-level and relational. An expression is evaluated in three stages: navigation in the set of surrogates of persons, asking for knowledge, and finally normalization, prime value processing and output preparation. A prototype implementation of selected parts of the model is based on a kernel concept.

1. Introduction
Privacy as Right of Informational Self-Determination
During the last two decades computer based information systems have become increasingly important for our societies. It was soon observed that extensively used information systems may constitute a great challenge for the individual's right of self-determination. Therefore several countries provided first safeguards for an individual against an invasion of personal privacy by enacting special laws, for instance [13, 24]. Since then concerned parties and citizens have continually demanded the enhancement of both legislation and technical means to enforce it.

In the Federal Republic of Germany the Federal Constitutional Court explicitly postulates privacy as the individual's right of informational self-determination. Its so-called 'Volkszählungsurteil' (judgment of census) [9] is based on the following guiding principles: "On the conditions of modern data processing safeguard for an individual against unlimited collection, maintenance, use and dissemination of personal information is comprised by the general right of privacy as stated in section 2, subsection 1 in connection with section 1 subsection 1 of the Constitutional Law. The constitutional rights so far guarantee the permission of an individual to determine on principle himself on the disclosure and use of his personal information. Exemptions from this right of 'informational self-determination' are only permitted in case of a predominant public policy need."

This concept of privacy is best explained from a sociological point of view (see for instance [44]) considering an individual as acting in various (social) roles. A role is described by a pattern of behavior with respect to a class of persons.

Then privacy means that an individual should
- independently dispose of his/her roles according to his/her right of self-determination; and
- have confidence that (public and private) institutions are strictly respecting the intended separation of his/her roles.

Unfortunately, the technical goal of computer based information systems, in particular if they are supported by computer networks, is to integrate data from various applications into one system in order to avoid redundancy and problems of inconsistency and to enhance integrity, availability and productivity. Hence we may be faced with a deep conflict of opposing values. Now computer scientists are challenged to contribute (partial) solutions to the evolved problem of designing privacy oriented information systems.

Privacy and Conventional Information Systems
Basically, a conventional information system is a highly developed centralized reference book implemented on a computer. Its internal structure is oriented towards gathering homogenous data into files such that the intended operations can be efficiently executed. As a result the personal information pertaining to a specific individual can be spread throughout the system, and vice versa, the relevance of a data item to a specific individual can be difficult to determine. This organization is consistent with the reference book paradigm: the owner of the reference book knows (and is permitted to know) all information about the concerned individuals.

However, in practice there does not exist (and there must not exist!) a single omnipotent owner but there are various users working on different tasks. The privacy issue requires that every user is confined to those parts of the information system that are needed for his/her current task, i.e. for the social role he/she is just acting in. Thus we need technical barriers that make available only the appropriate window on the information system. Unfortunately, the traditional approaches to design such barriers rely on concepts of the internal structure of the information system which in general does not appropriately match with the structure of social roles. Therefore we propose to reject the reference book paradigm and to favor the personal knowledge paradigm: every person knows everything about himself and must be asked again and again because nobody can permanently remember anything else.

There are two basic approaches to determine the data part of a window on the information system:
- use the classification labels to define a subset of data items,
- use the query language to define a view.

The first approach, referred to as the (pure) multilevel security approach, is based on the following constructs: there are users, data items (for instance: fields, records or
domains, relations on the database layer or files on the operation system layer), and a lattice of security levels; a user has a security level as clearance; a data item has a security level as classification. The basic security policy requires that the clearance c of a user must be at least as high as the classifications of the accessible data items, i.e. the user sees the subset of data items with classification less than or equal to c. This policy may be enforced by access restriction (implemented by a kernel [27, 8, 3] or a stream filter [20, 32, 43, 58, 50]) or by encryption [17, 1, 16, 29, 36, 45, 18, 46].

The multilevel security approach suffers from several disadvantages. Most importantly the variety and nature of social roles are not appropriately reflected by the lattice structure of (the usually used) security levels and it is difficult to express the sensitivity of relationships between data items and to adapt classifications if new tasks evolve. In order to moderate these disadvantages one can employ refinements of the pure approach or additional features (see for instance, in the naval surveillance system layer), and as consequences of the accessible data items, i.e. the user sees the subset of data items with classification less than or equal to c. This policy may be enforced by access restriction (implemented by a kernel [27, 8, 3] or a stream filter [20, 32, 43, 58, 50]) or by encryption [17, 1, 16, 29, 36, 45, 18, 46].

The multilevel security approach suffers from several disadvantages. Most importantly the variety and nature of social roles are not appropriately reflected by the lattice structure of (the usually used) security levels and it is difficult to express the sensitivity of relationships between data items and to adapt classifications if new tasks evolve. In order to moderate these disadvantages one can employ refinements of the pure approach or additional features (see for instance, in the naval surveillance system layer), and as consequences of the accessible data items, i.e. the user sees the subset of data items with classification less than or equal to c. This policy may be enforced by access restriction (implemented by a kernel [27, 8, 3] or a stream filter [20, 32, 43, 58, 50]) or by encryption [17, 1, 16, 29, 36, 45, 18, 46].

The multilevel security approach suffers from several disadvantages. Most importantly the variety and nature of social roles are not appropriately reflected by the lattice structure of (the usually used) security levels and it is difficult to express the sensitivity of relationships between data items and to adapt classifications if new tasks evolve. In order to moderate these disadvantages one can employ refinements of the pure approach or additional features (see for instance, in the naval surveillance system layer), and as consequences of the accessible data items, i.e. the user sees the subset of data items with classification less than or equal to c. This policy may be enforced by access restriction (implemented by a kernel [27, 8, 3] or a stream filter [20, 32, 43, 58, 50]) or by encryption [17, 1, 16, 29, 36, 45, 18, 46].

The multilevel security approach suffers from several disadvantages. Most importantly the variety and nature of social roles are not appropriately reflected by the lattice structure of (the usually used) security levels and it is difficult to express the sensitivity of relationships between data items and to adapt classifications if new tasks evolve. In order to moderate these disadvantages one can employ refinements of the pure approach or additional features (see for instance, in the naval surveillance system layer), and as consequences of the accessible data items, i.e. the user sees the subset of data items with classification less than or equal to c. This policy may be enforced by access restriction (implemented by a kernel [27, 8, 3] or a stream filter [20, 32, 43, 58, 50]) or by encryption [17, 1, 16, 29, 36, 45, 18, 46].

The second approach, referred to as the (pure) view approach ((55, 51, 54, 10, 35, 42)), is based on the following constructs: there are users, views defined in some high level query language, and access rights (which may also be considered as security level as clearance); a user possesses access rights to views. The basic security policy requires that the user sees only data items that are contained in his views.

The merits of the view approach strongly depend on the expressive power of the view definition language and on an adequately chosen database schema. By a sophisticated design a rather close match between the (data) needs of a social role and (the data part of) a view seems to be achievable. However, the poor structural concepts of the relational model of data can lead to clumsy constructions. For our personal model of data, presented in this paper, we argue for the employment of special structural concepts besides the usual relational ones in order to facilitate defining appropriate views.

Of course the indicated approaches, concepts, policies and mechanisms can be mixed in many different ways. An advanced example is described in [12, 21, 2, 22].

Determining the data part of a window primarily deals with the read and (to a lesser extent) with the write access to data items. But in order to model social roles we must have more elaborate and more specific operations at our disposal. In principal both the multilevel security approach and the view approach could be extended to cover more than the elementary read and write access by adapting ideas of object oriented programming. However, in combination with the relational model of data there are no distinguished units (data items, views) that could serve as objects. The personal knowledge paradigm (mentioned above) suggests that a person with its knowledge would be an ideal object.

The mentioned approaches to design barriers focus mainly on the requirements of military or related organizations [23]. Accordingly some of the discussed deficiencies already originate from a mismatch between military needs and the idea of privacy. Recently a similar observation was made on the difference between military and commercial information systems [11].

Intent of the Paper
We present a general report on our approach to solve the problem of privacy oriented information systems. Many details and extensions are outside the scope of this paper or in need of further research. The report is based on extensive research experiences in specifying the structure of such a system, including the underlying data model and the privacy policy, as well as on the insight gained from a prototype implementation of selected parts of the specification [5, 28, 31, 37]. Our system is called DORIS (Datenschutz-orientiertes Informationsystem).

2. General Design

Intuitive Concepts
A data model offers the basic features to represent formally the (mini)-world (of an application). In the personal model of data the world is supposed to consist only of persons. In order to accentuate the technical aspects we use the pronoun "it" in speaking about a person of the model; for real world individuals, however, we use the pronouns "he/she". Any data is attached to a person representing its knowledge. Any action is performed by a person. Thus individuals (data subjects) as well as the various users within the agencies maintaining the information system, for instance employees serving clients, system administrators or privacy officers, are modelled in the same way. Note that a person may be a substitute for somebody who is both a data subject and a user. Of course there exists data without any connection to any special person, e.g. physical laws, data from a sensor. Although these data can be modelled in our system we do not consider such data in this paper.

A person knows anything (that is relevant for the application) about itself and, possibly, about its relationships to other persons. It is not permitted to remember permanently anything else. In particular no person can possess its own centralized reference books within the system.

A person is acquainted with other persons. The set of acquaintances constitutes its (social) environment. The person is confined to its environment and it is not capable to act beyond this limit. It can send messages to its acquaintances querying about their knowledge or asking to perform an operation, say to update their knowledge.

Then an acquaintance reacts to the message if the sender exhibits an appropriate authority. The authority of a person with respect to an acquaintance depends on the role that it is just practising. In principle, it may be allowed to practise different roles, but only one role may be used at a given time.

A person remembers the messages that it is sending or receiving. Thus all transactions within the system can be traced by asking the affected persons.

A person is a member of a group. All persons of a group have the same knowledge schema, operation declarations, authorities and roles.

The privacy policy of DORIS is based on two independent features that we call the rights of a person: a person holds individually received acquaintances and collectively obtained authorities on roles. The login procedure of DORIS identifies any user of the system as a person within the system that will serve as his/her substitute. If the user requires to perform a transaction by stating a query or update expression of the data manipulation language then the system enforces the privacy policy by two rules. The evaluation of the expression is modified such that messages are sent only to the acquaintances of the substitute. The evaluation of the expression is totally refused if the substitute is not able to exhibit one authority with respect to each involved group.
If a person creates a new person then, at least, the creator is acquainted with its creation and the creation is acquainted with itself. Subsequently further acquaintances can be granted and revoked. As all other operations, creating, granting and revoking require suitable authorities.

A person may temporarily make available some of its rights to other persons in order to assist them in exceptional situations. In this case the person specifies a fixed combination of acquaintances and authorities. If the assisted persons will benefit from the availability then they must explicitly refer to the original right holder which, in contrast to granting of acquaintances, retains complete control over its rights.

System administrators and privacy officers are persons with suitable authorities. They are intended to perform the usual actions that are required to ensure the regular services of the system, for instance schema definition, initialization or recovery. Privacy officers should be able to enforce regulations of privacy legislation. In particular they are intended to help an individual to gain access to the information pertaining to him/her which is contained in the system in a form comprehensible to him/her. Furthermore they are permitted to evaluate the remembrances of the persons for control purposes. Since neither administrators nor privacy officers alone should become too powerful their rights are determined by cooperative politics that force them to cooperate.

Technical Concepts

The technical design of DORIS is based on careful adaptations of concepts developed for object oriented programming, relational databases, and capability based operating systems (see for instance [30,14,56], [57,15,38], and [19, 25, 26, 40, 49, 59]).

A person is treated like an object. Every person is created as an instance of a class which we call group. As usual the group declaration describes the internal structure of all of its instances and defines the possible operations on its instances. Furthermore the group declaration contains the roles in which its instances may be involved and the authorities which its instances hold. The groups form a hierarchy. Besides having its own components a group inherits the structure, operations, roles, and authorities from all its supergroups. The root of the hierarchy, the group ALL, describes those properties that all persons should share. The leaves of the hierarchy describe the most specialized kinds of persons.

The internal structure of an instance (person) must be qualified to represent the knowledge of a person. Like the schema definition of a relational database, the group declaration determines the attributes and their types. As types we allow prime types (e.g. INTEGER, STRING, ...), record types, and set types where the elements are either prime or records. If all types are prime then the knowledge structure of one person is just an encapsulated tuple and the tuples of all instances of one group form a homogeneous relation. If all types are sets of records then we can consider the knowledge structure of one person as a complete encapsulated relational database (since then attributes of the group declaration correspond to relation names and field identifier correspond to attributes of the database).

Thus the designer of an application may apply a wide range of options. The variety of the global models covers a single relational database (for each group a relation) as well as a big collection of relational databases (for each person a set of relations). Nevertheless all possible models can be uniformly treated by the concept of non first normal form, NF^2 for short, relations of depth at most 2. Using this concept the knowledge structure of one person is an encapsulated NF^2 tuple.

Some system defined operations are applicable to instances (persons) of all groups. The basic operations are: tell (parts of the knowledge); insert, delete, modify (parts of the knowledge); create, dispose (a person); grant, revoke (acquaintances). Further operations can be defined within a group declaration. All operations are only invoked by the system in evaluating a query or update expression.

An instance (person) is uniquely identified within the system by a surrogate which is a random number secretly created by the system and never revealed to any user.

Although for each individual his/her private information is separately encapsulated by a single object, namely the substituional person, the data manipulation language is set-oriented and essentially as powerful as the relational one. A query expression (and similarly an update expression) has three levels and is evaluated in three stages accordingly. At the first stage, evaluation always starts with the surrogate that identifies the person on behalf of which the query is processed; then by navigating by means of group names within the environment, i.e. the set of acquaintances, some sets of surrogates are determined which can be further manipulated by selections (according to conditions on the knowledge) and the other usual relational operators (product, intersection, union, difference, projection); as a result one or more relations on surrogates are produced which of course remain hidden to the user and under the strict control of the system. At the second stage the system asks the involved persons about some parts of their knowledge; if necessary the answers (i.e. values, tuples, relations) are normalized and subsequently for each surrogate relation they are combined for a relation on prime values. At the third stage, these relations can again be manipulated by usual relational operators yielding the final result shown to the user.

The privacy mechanism must enforce the policy: at the first stage it must confine navigation to the environment; at the first (if selections are involved) and the second stage it must check whether the substituional person holds appropriate authorities and, if any check fails, abort the evaluation. Furthermore, for a multi-user system, the privacy mechanism must strictly separate the working areas of the users at all stages.

There are several possibilities to implement the system depending on the degree of distribution. The system may be centralized running on a single processor or totally distributed connecting a collection of personal microcomputers. In any case, surrogates serve as capabilities for accessing persons, and a kernel should be the only component allowed to perform the operations on persons. Thus any action of a user must be mediated by the kernel which checks authorities, handles relations on surrogates, translates surrogates into physical addresses, invokes operations on persons and delivers relations on prime values back to a user. Furthermore the kernel must safeguard the declared authorities.

3. Terminology

In order to model an application on the conceptual level, we use NF^2 tuples and relations (see for instance [15]). An NF^2 tuple is built up from columns which may have further subcolumns each of which contain a single value or a set of values. An NF^2 relation is a collection of NF^2 tuples. The relational operators projection, selection, and unnesting are denoted by \( \pi_A(R) \), \( \sigma_C(R) \), and \( \mu_A(R) \).

4. Data Model

Person Tuples

For each person we set up an NF^2 tuple with the column names
A value for
- SURROGATE ("the person with surrogate ...") is a random number secretly created by the system and never revealed to any user. It uniquely identifies the tuple within the system all the time.
- KNOWS ("the person knows...") will be structured according to the application in hand. It has several subcolumn names A1,...,An which are the person's attributes. They are defined in the declaration of the group to which the person belongs or of its supergroups.
- ACQUAINTED ("the person is acquainted with ...") is a set of random numbers used as surrogates for the environment of the person.
- ALIVE ("the person is alive or else cancelled") is a boolean one. If the value is false, then in most cases the person will have to be totally cancelled in the future, however for the purpose of control this procedure has been postponed.
- AVAILABLE ("the person has made available the rights ...") is a set of records describing rights made available.
- REMEMBERS ("the person remembers the messages ...") is a set of records describing messages sent or received.

Collecting the tuples for all persons we get an NF2 relation called PERSON that is variant with respect to the KNOWS column and that has SURROGATE as key column. Note that the PERSON relation exists only on the conceptual level and may be implemented in any suitable form.

For the purpose of defining semantics, since only persons that are 'alive' will regularly react on messages, we introduce the following abbreviation:

\[ \text{LIVING \_PERSON := } \sigma_{\text{ALIVE TRUE}}(\text{PERSON}) \]

Catalogue
Each person is created as an instance of a group. On the conceptual level there is a relation CATALOGUE with the two column names SURROGATE and GROUP that summarizes the membership of persons in groups.

World
Neglecting roles and authorities for the moment, the current status of an application is fully described by an NF2 relation WORLD which is the natural join of PERSON and CATALOGUE.

Groups
All persons sharing the same attributes, operations, roles, and authorities form a group. In a group declaration the designer determines the details of these features using the following syntax:

\[ \langle \text{group declaration} \rangle ::= \langle \text{GROUP} < \text{group identifier} > \rangle ; \]

\[ \langle \text{SUPERGROUP} < \text{group identifier} > \rangle ; \]

\[ \langle \text{ROLES} < \text{role} > ; \rangle \]

\[ \langle \text{AUTHORITIES} < \text{authority} > ; \rangle \]

\[ \langle \text{OPERATIONS} < \text{procedure} > ; \rangle \]

In the data manipulation language GROUP identifiers are used for navigating, and they are part of authority declarations.

The SUPERGROUP declaration selects a different group as immediate supergroup. If for group G we select supergroup H then we write G < H. Let \( \leq \) be the reflexive, transitive closure of the relationship <, \( \leq \) must form a tree-like hierarchy (connected, acyclic). The unique top element is the group ALL and describes those properties that all persons should share. A group inherits the attributes, roles, authorities, and operations from its supergroups.

For the purpose of defining semantics, we shall deal with the persons belonging to a specified group \( g \) or a subgroup of it:

\[ \text{MEMBER}(g) := \sigma_{\text{SUR}} \sigma_{\text{GRS}}(\text{CATALOGUE}) \]

The ATTRIBUTES declaration describes the structure of knowledge that all instances of the group should share.

The collection of group declarations can be conveniently represented by an NF2 relation SCHEMA.

Example:
The group EMPLOYEE and its supergroup ALL is described by

\[
\begin{align*}
\text{GROUP} : & \quad \text{EMPLOYEE} ; \\
\text{SUPERGROUP} : & \quad \text{ALL} ; \\
\text{ATTRIBUTES} : & \quad \text{employee-no} : \text{INTEGER} ; \text{salary} : \text{INTEGER} ; ... \\
\text{GROUP} : & \quad \text{ALL} ; \\
\text{SUPERGROUP} : & \quad \text{ALL} ; \\
\text{ATTRIBUTES} : & \quad \text{name} : \text{STRING} ; \text{address} : \text{STRING} ; \text{birthday} : \text{INTEGER} ; ... \\
\end{align*}
\]

Persons as Objects
Note that the relation PERSON exists only on the conceptual level. Actually for each person there exists an encapsulated object that is an instance of its group. The data part of this object is the person's NF2 tuple.

Example: We describe some persons of the group EMPLOYEE (for convenience surrogates are denoted by <ident>):

\[
\begin{align*}
\text{SURROGATE} : & \quad \text{<MILLER>} <\text{WILLIAMS}> <\text{BAKER}> \\
\text{GROUP} : & \quad \text{EMPLOYEE} \text{ EMPLOYEE} \text{ EMPLOYEE} \\
\text{KNOWS} : & \quad \text{name} : \text{MILLER} \text{ WILLIAMS} \text{ BAKER} \\
\text{address} : & \quad \text{LITTLE RD} \text{ MAIN ST} \text{ SUN BLVD} \\
\text{birthday} : & \quad 1952 03 21 1954 06 14 1951 10 25 \\
\text{employee-no} : & \quad 37 12 27 \\
\text{salary} : & \quad 40000 45000 42000 \\
\text{ACQUAINTED} : & \quad \{ <\text{MILLER}> \} \{ <\text{WILLIAMS}> , <\text{BAKER}> , <\text{MILLER}> \} \\
\text{ALIVE} : & \quad \text{TRUE} \text{ TRUE} \text{ TRUE} ... \\
\end{align*}
\]

The operation part of this object is constituted by a collection of system defined operations, possibly supplemented by the operations defined in the respective group declaration. We only sketch the intended meaning of the most important system defined operations:

- create : creates a new instance of a group
- dispose : sets the value of ALIVE to FALSE
- tell : returns the value of a subcolumn of KNOWS
- insert : transforms the value of a subcolumn of KNOWS
- delete : }
- modify : returns the value of ACQUAINTED
- acquire : transforms the value of ACQUAINTED
- available : returns the value of AVAILABLE
- make_available : transforms the value of AVAILABLE
5. Data Manipulation Language

A user can retrieve or manipulate the knowledge, acquaintances, alive status, availabilities, and remembrances of some persons. In this section we deal with these facilities without caring about the privacy policy.

The login procedure of DORIS identifies any user as a person within the system that will serve as his/her substitute. It is represented by its surrogate. Since the semantics of an expression depend on this surrogate we use the notation

\[ \text{expression} \mid \text{expression} \mid \text{expression} \]

where

\[ p \]

is the user's surrogate.

An expression of the data manipulation language can be a query or an update. A query delivers a (normalized) relation into the (private) working area of the user and, as a side effect, it inserts appropriate control data into the REMEMBERS column of the involved persons. An update transforms the WORLD relation.

Queries

A query expression has three levels and is evaluated in three stages accordingly: navigation (in the CATALOGUE) and surrogate processing, asking (for KNOWS entries) and normalization, prime value processing and output preparation.

Basic constructs for navigation and surrogate processing are expressed as follows:

\[ \text{query} \mid \text{surrogate_expr} \mid \text{surrogate_expr} \]

A set operation has only the first and the respective second level. The first level is for navigation and surrogate processing. The second level is for performing an operation (corresponding to the operation of telling for queries). There are four kinds of system defined operations: updating operations (corresponding to the operation of updating for queries). The product of two relations is a set of surrogates. For each person involved persons about their knowledge and for immediate normalization of their answers if necessary. As a result a normalized relation on prime values is returned. It column names are taken from the specified list of components:

\[ \text{query} \mid \text{surrogate_expr} \mid \text{TELL} \mid \text{component} \]

We treat only some simple cases where the surrogate expression \( \text{surrogate_expr} \) delivers a set (unary relation). If all components are prime attributes, say A and B, then the semantics is defined by

\[ \text{query} \mid \text{surrogate_expr} \mid \text{TELL} \mid \text{component} \]

Here for each person \( \tau \) exactly one tuple with column names A and B is produced.

Example (continued): WILLIAMS now completes his question: I ACQUAINTED EMPLOYEE KNOW address \# MAIN ST TELL name, address

The answer is: (name: MILLER, address: LITTLE RD).

If an attribute is of a set type then we must normalize. If an attribute is of a record type then for every field identifier we produce a separate column name.

The third level is for further processing of prime valued relations and output preparation. Here we can select according to conditions, compute products, intersections, unions, and difference and project on specified columns as in a (normalized) relational database.

Updates

An update expression has only the first and the respective second level. The first level is for navigation and surrogate processing. The second level is for performing an operation (corresponding to the operation of telling for queries). There are four kinds of system defined operations: updating knowledge, creating or disposing persons, granting or revoking acquaintances, making available rights (see section 7).

6. Privacy Policy

Any person holds two independently distributed kinds of rights.

Acquaintances constitute the first kind. A person individually receives its acquaintances by the operations create and grant. It is strictly confined to its set of acquaintances that we call its (social) environment. Confinement is achieved by modifying the evaluation of a surrogate expression, i.e. of the first level of expressions of the data manipulation language. Therefore the semantics of ACQUAINTED are redefined (using the same notation as in section 4):

\[ \text{query} \mid \text{surrogate_expr} \mid \text{TELL} \mid \text{component} \]

Sets or relations of surrogates can be manipulated by the usual set operations: cartesian product, intersection, union, difference.

The second level of a query expression is for asking the involved persons about their knowledge and for immediate normalization of their answers if necessary. As a result a normalized relation on prime values is returned. It column names are taken from the specified list of components:
practise a role with respect to the persons of a group or to itself:

\[
< \text{authority} > ::= \langle \text{group identifier} \rangle \langle \text{role identifier} \rangle \langle \text{operation identifier} \rangle \langle \text{access condition} \rangle
\]

A role is a subcollection of the system defined or group declared operations that is referenced by a role identifier. An access condition restricts the usage of an operation.

\[
< \text{role} > ::= \langle \text{role identifier} \rangle \langle \text{operation identifier} \rangle \langle \text{access condition} \rangle
\]

Note that authorities and roles are declared at different places in general. An authority must be declared in the group of the agent of the user. A role must be declared in the group of the affected persons.

Example: In the declaration of the group EMPLOYEE the following roles are defined:

- hire (create [EMPLOYEE], insert [name, address, birthday, employee_no, salary])
- earn (tell [salary])
- change_in_salary (tell [salary], modify [salary])
- fire (dispose [EMPLOYEE])

The corresponding authorities are given in the declaration of group PERSONNEL-MANAGER:

- EMPLOYEE: hire, change_in_salary, fire
- and of group EMPLOYEE (every employee may read his own salary):

\[
1: \text{earn}
\]

The evaluation of an expression \(\exp\) on behalf of a substitutional person is totally refused if the person does not hold appropriate authorities for all involved operations. In this case the user is only notified that he/she is not permitted to issue the expression.

If a substitutional person with surrogate \(p\) issues an expression then \(p\)'s authorities are recorded in the current authority table. The authority checking mechanism enforces the policy that at most one role per group may be used. Furthermore note that authorities are only required for invoking operations on our objects. All other actions, in particular execution of relational operators (except selection during the first stage), may not be further supervised other than that the system must strictly separate the working areas of the users.

7. Making Rights Available

Acquaintances can be dynamically altered by the operations grant and revoke. Authorities on roles, however, basically remain fixed because they are stated in group declarations which are not likely to be changed frequently.

The initial distribution of rights should be designed according to the least privilege principle giving any person the weakest rights such that it can still perform its usual tasks. On the other hand, there must be provisions for exceptional or even unforeseen situations where individuals must act beyond their normal duties.

For this purpose we designed facilities for temporarily making available combined rights. The basic idea is as follows. If an individual represented by his/her substitutional person, called the sender, wishes that other persons, called receivers, may use one of its authorities on roles with respect to further persons, called the targets, then he/she can specify the receivers by a surrogate expression the targets by a surrogate expression the role by a role identifier the period by stating a date (limiting the availability) or a number (of permitted usages)

The syntax for such an expression is defined by

\[
< \text{update} > ::= \langle \text{binary surrogate exp} \rangle \text{MAKEAVAILABLE} \langle \text{role identifier} \rangle < \text{period} >
\]

The semantics are that the AVAILABLE column of the sender's NF2 tuple is transformed, invoking the system defined operation make_available, by adding the specified quadruple.

Moreover the sender needs a further authority for making available its authority \(gr.ro\). The simplest syntactic form of such a second order authority is to mark the basic authority \(gr.ro\), say by an asterisk, expressing that it can be made available.

Thus the sender keeps complete control over the rights that it makes available. Since they are never spread over the system, they can easily be withdrawn just by deleting the corresponding quadruple.

8. Procedures

The OPERATIONS entry in a group declaration offers the option to define procedures as local operations. A procedure consists of a privacy part and a body comprising DORIS expressions and the usual programming constructs (assignments, repetitional and conditional statements). An operation either transforms the NF2 tuple of the involved persons or it returns a surrogate relation or it returns a prime valued relation.

Since the details are still under investigation we only list our fundamental requirements (see [28] for further details):

- Operations should facilitate programming as in other information systems, for instance in POSTGRES [53].
- Operations should provide a mechanism to relax the privacy policy for special purposes in a controlled form, in particular confinement to the environment and the one-role-per-group rule may be partially circumvented.
- Operations should provide a mechanism to relax privacy as proposed for the operating systems HYDRA [59] or OSKAR [25]. Thus for instance operations can be used for delivering aggregated data for statistical purposes.

9. Remembrances and Control

The REMEMBERS entry in the relation PERSON contains information on the recent queries and updates of the system. This information is used for control which is done by the privacy officers. Control should be able to solve problems like:

"Which queries were put?".
"Which system defined operations were produced by a query"?
"What has recently been done by person \(X\)?".
"Give a survey of all attempts to access data without the appropriate authorities!".
"What has happened with the data of person \(Y\) who has obtained it? Who has changed it?".

Control is done exclusively by inspecting the REMEMBERS entries. Each query or update expression is recorded in the REMEMBERS entry of the substitute of the individual who has entered it. Furthermore all system defined operations produced by this query (or update) are recorded together with the authorities available at the time.
10. Initialization, System Administrators, and Privacy Officers

For any application suitable group declarations, especially appropriate authorities and roles must be designed. It is important to observe that the DORIS system can achieve its goal only if the particular design also is in the spirit of privacy legislation. For the DORIS system could be totally misused, say by installing a "one person world" that simulates a normal relational database.

We propose some rules as a guideline.
1. In addition to the top group ALL there are at least two further groups. The group ADMINISTRATOR should obtain strong authorities on roles in order that its instances can perform the usual actions that are required to ensure the regular services of the system, for instance defining new groups or recovery. These instances, however, should hold only minimal sets of acquaintances. The group PRIVACY OFFICER should obtain only weak authorities on roles but it has the same access permissions, as it is possible that the system administrator is missing. Hence in many cases administrators and privacy officers should be forced to cooperate by combining their authorities.
2. Since every individual should be able to access the knowledge of his/her substituional person the group ALL should include appropriate declarations.
3. All group declarations should be public. There are special commands that help an individual to retrieve the declarations, in particular the matching AUTHORITY and ROLES entries he/she is interested in.

11. Implementation

Thus far we have only implemented some parts of our design in a prototype fashion, in particular query evaluation under our privacy policy has been considered. Roughly described the DORIS system consists of a hierarchy of three levels: analysis and evaluation of expressions, control, and kernel. A simulation of DORIS on a conventional relational DBMS is under development. As a final goal we envisage to construct a totally decentralized system (see e.g. [47], [34]).

12. Summary and Evaluation

We designed a privacy oriented information system based on the personal model of data. Individuals, users as well as clients, are represented by encapsulated objects called persons. The data processed by an object corresponds to the knowledge of the role part it corresponds to. The privacy policy is defined based on two kinds of rights: evaluation of an expression is confined to acquaintances by query modification and is refused if a person does not hold appropriate authorities on roles comprising the needed operations. Acquaintances can be granted and revoked; combined rights can be made temporarily available to other persons. Although the system is object-oriented the data manipulation language is high-level and relational.

With respect to evaluation on criteria and other protection issues of [50] our approach represents a closed approach since the system denies any access or operation unless it is explicitly permitted by acquaintances or authorities. It is of fine granularity: acquaintances permit access to persons as basic protection units; authorities on roles permit use of specific operations which can be further restricted by access conditions. It includes non-monotonic control by the one-role-per-group policy. It allows discretionary control by attaching access conditions to operations in role declarations. It solves the problem of multiple control predicates by the mechanism for the one-role-per-group policy. It responds to privacy violations by query modification if acquaintances are missing, and by refusal if authorities are missing. It locates security checking mainly inside a kernel which is the only component of the system that may invoke operations on persons.

Logging is based on the person's remembrances. System administrators and privacy officers obtain rights such that they must cooperate for auditing and controlling. We attempt to avoid Trojan Horse attacks by using a kernel. There are no provisions against covert channels. In disposing a person we do not immediately destroy a person but only set its ALIVE flag to FALSE. Thus we accept the risk of maintaining 'forbidden' data because we favor the possibility of belated control. We did not seriously consider the problem of the interaction between operating system and information system. Whenever it is possible we argue to run an information system without a general purpose operating system. Cryptographic techniques can be included for secondary storage. Historic-dependent access control is only dealt with in very restricted form: for complicated expressions the one-role-per-group policy prevents arbitrary combinations of authorities, and issuing several primitive expressions and combining the results afterwards is not supported by our system because persons cannot store answers to queries (but of course it can be done by other means).

Evaluating DORIS in a broader context we observe that an information system should not have at least the following characteristics: it provides a model of (a part of) reality; it is self-descriptive by means of a schema (data dictionary) for supporting integrity, recovery, access control, optimization etc.; it hides data persistently; it offers powerful (set oriented) data manipulation languages; it allows data sharing among many users. As already pointed out in the Introduction some of these features may be conflicting with the concept of privacy. Ideally we would like an individual to keep complete control on his/her personal information. Unfortunately, inside of a formal system full information flow control can hardly be achieved in a strict sense, see e.g. [19, 48]. Alternatively we try to rely on two more modest but actually realizable concepts: the personal knowledge paradigm, i.e. directly modeling the real world individuals with their knowledge and actions, and access control. Their merits and limits are further discussed in [4], respectively [7]. Here we only note that (at least when implemented in a totally distributed way) they drastically influence the above mentioned characteristics. In particular sharing is realized by communication [52] between persons (models of individuals). Correspondingly expressions of our data manipulation language comprise a first level of navigation and surrogate processing, as powerful and set oriented as relational algebra, that hides all details of communication from the user.

References


354