5 AN INFORMATION-FLOW MODEL FOR PRIVACY (INFOPRIV)

Lucas C.J. Dreyer and Martin S. Olivier

Abstract: Privacy is concerned with the protection of personal information. Traditional security models (such as the Bell-LaPadula model) assume that users can be trusted and instead concentrate on the processes within the boundaries of the computer system. The InfoPriv model goes further by assuming that users (especially people) are not trustworthy. The information flow between the users should, therefore, be taken into account as well. The basic elements of InfoPriv are entities and the information flow between them. Information flow can either be positive (permitted) or negative (not permitted). It is shown how InfoPriv can be formalised by using graph theory. This formalisation includes the notion of information sanitisers (or trusted entities). InfoPriv is concluded with a discussion of its static and dynamic aspects. A Prolog prototype based on InfoPriv has been implemented and tested successfully on a variety of privacy policies.

5.1 INTRODUCTION

We live in an information age in which more and more personal information about the individual is stored in various database systems. These databases are maintained by health, financial and government-related institutions. It is of paramount importance to protect personal information from misuse. Refer to [5] for a discussion of privacy concerns and options for protecting information privacy on the National Information Infrastructure of the US.

Reports have been released about unauthorised disclosures of information in large information systems such as the NCIC (National Crime Information

S. Jajodia (ed.), *Database Security XII*

Centre) [1] and those of the IRS (Internal Revenue Service) of the USA [4]. These reports further support the privacy concern.

The IRS has a number of security mechanisms in place to address the problem. These include the Electronic Audit Research Log (EARL) for monitoring and detecting unauthorised browsing of tax-related information. However, the General Accounting Office [4] concluded that EARL is limited in detecting the unauthorised viewing of tax-related information by IRS employees (called 'browsing').

It follows from the above that traditional security mechanisms are generally insufficient to ensure the privacy of information in large systems. We develop an information-flow model (named InfoPriv) here that may be used to model privacy policies. The basic building blocks of InfoPriv are entities and the information flow between them. Only entities are used in InfoPriv as opposed to the users and entities (objects) of traditional security models.

Entities are viewed as information containers and indirect information flow can occur between entities. We define negative information flow as a way of preventing indirect information flow. Entities and the potential information flow between them translate directly to directed graphs called 'information can-flow graphs'. The entities form the vertices and the potential information flow forms the arcs.

In the next section we discuss the basic privacy principles. We define the Principle of Completeness and use it to unify the ideas of users and entities. The rest of the paper is roughly divided into three parts: Static Aspects of InfoPriv, a Formalisation and an introduction to the Dynamic Aspects of InfoPriv.

The Static Aspects of InfoPriv are concerned with entities and the potential information flow between the entities. Potential information flow is further divided into positive and negative information flow. InfoPriv will be formally presented in terms of graph theory next, followed by the Dynamic Aspects of InfoPriv. This paper will finally be concluded.

5.2 THE INFOPRIV MODEL

The purpose of this section is to describe a model called InfoPriv that is suitable for modelling privacy. InfoPriv and its underlying principles will be developed and justified throughout the rest of this paper by means of examples. We will start this section by defining the Principle of Completeness and describe how it may be used to relate security and privacy issues and how to model the privacy of a system.

5.2.1 Principles of privacy

The basic principle of privacy is that information should only be stored in a system for well-defined purposes [2, 3]. For instance, any information collected by the Internal Revenue Service (IRS) must only be related to and used for tax purposes. Any other use of IRS information is considered to be a violation of privacy.

We extend this principle by introducing the Principle of Completeness. The Principle of Completeness (PoC) states that the privacy of information can be better protected by having an improved understanding of the context of the information. The context of information is defined as the way in which the information will be used. An example of a context is the set of rules in an organisation that govern which employees should have access to payroll information. The PoC will now be illustrated by means of an example.

Consider two people: John Smith and Sarah Parker. Further assume that Sarah Parker works for the IRS and is the sister-in-law of John Smith. There is obviously a conflict-of-interest if Sarah has to process John's tax-related information (employees of the IRS are not permitted to 'browse' their relatives' tax information). However, specifying this requirement in a computer system that is based on a traditional security model is not so easy since traditional security has been designed around specific requirements. For instance, a very large number of security classes have to be used when applying the Bell-LaPadula model to this situation.

Using the PoC may solve the problem of Sarah having access to John's tax information. A security system that adheres the PoC should permit constraints of the form 'no person should have access to a relative's tax information' to be stored in addition to the normal security requirements (such as 'Sarah has access to John's tax information').

An ideal implementation of the PoC is a security system that permits general privacy policies to be incrementally refined until it can be proven (from the privacy policy) that no unintended information flow can occur. Unintended information flow is information flow that is not prohibited by the privacy policy. However, the SSO (System Security Officer) may be under the (wrong) impression that the privacy policy does indeed prohibit the above-mentioned information flow. This is particularly applicable to complex policies.

Note that the specific structure and form of a privacy policy is outside the scope of this paper. We assume in this paper that a privacy policy is a set of general statements of the form "Employees of the IRS should not have access to their relative's tax information" or "Jane Ullman is permitted to determine John Smith's salary". This paper is intended to describe how a privacy policy can be analysed once it is modelled in terms of InfoPriv.

The rest of this section is devoted to the development of the InfoPriv model by making use of the PoC.

5.3 STATIC ASPECTS OF INFOPRIV

The static aspects of a system are those parts or aspects that do not change over time. For example, a motorcar consists of a body, an engine and four wheels. The relationships between these parts stay the same even if it is impossible to determine where the car will travel over time. In the case of InfoPriv the static aspects consist of entities and the potential information flow between them. Note that we assume in this paper that the entities and the potential information flow between them (hence the privacy policy) stay fixed during the

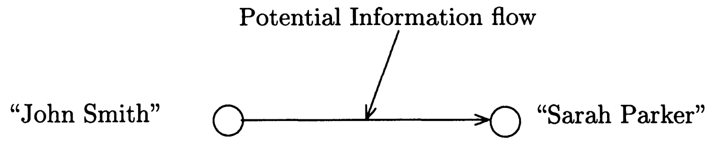

Figure 5.1 Potential information flow between entities.

lifetime of a system. This assumption is made due to space restrictions and the evolution of the privacy policy is the subject of [9].

Traditional security models make a distinction between (human) users and entities [12, 13, 14]. The main distinction between a user and an entity is that the user is viewed as external to the (computer) system while the entity is internal to the system. We mean by 'internal to the system' that the system contains the entity and has total control over it. Examples of entities 'internal to the system' are database tables, files and processes.

'External to the system' may be interpreted as meaning that the system only has sufficient information about a user as to permit interaction between it and the entities in the system. For instance, typical information that will be recorded for the user 'Sarah Parker' includes her password, privileges and other security related information. The system cannot force Sarah to keep certain information secret.

The first major difference between InfoPriv and traditional security models is that no distinction is made between users and entities in a computerised system. According to InfoPriv a computer system only consists of entities and the information flow between those entities. Refer to [8] for information about the lattice model that is also based on information flow. An entity can be viewed as an actor that interacts with various other actors throughout its entire lifetime. Examples of entities are "John Smith", "Sarah Parker" and "Employee Table". We will use quotation marks to indicate entities.

The two most important properties of an entity are that it should be uniquely identifiable and that it has memory (it is, therefore, an information container). "John Smith" knows various things such as where he lives, what his salary is and who his spouse is. He may even know things about other entities such as his wife's name and salary and the names of his children.

An integral part of InfoPriv is the interaction between entities. This interaction can be modelled by using information flow. Consider the following entities: "John Smith" and "Sarah Parker". Assume that "John Smith" may talk to "Sarah Parker". We say that information may flow between John and Sarah. A convenient way to depict this (potential) flow of information is to make use of a directed graph where the vertices represent the entities and the arcs represent the potential information flow between the entities. This is shown in Figure 5.1.

Note that we refer to a graph that depicts potential information flow (such as Figure 5.1) as an information can-flow graph or can-flow graph for short. There are two important advantages in using a can-flow graph to represent a privacy policy. First, graphs are a natural way of representing information, especially the potential information flow in a system as well as constraints on that flow. The second advantage is that a large base of graph algorithms can be used to analyse a can-flow graph for possible illegal information flow [7].

The potential flow of information in Figure 5.1 can be realised by either "John Smith" writing information to "Sarah Parker" or "Sarah Parker" reading information from "John Smith". This symmetry between reading and writing forms the basis of the second major difference between InfoPriv and traditional security models. Reading and writing are traditionally viewed as separate operations, especially for discretionary security models [13]. This is no longer the case in InfoPriv and reading and writing are replaced by information flow altogether. The potential information flow of Figure 5.1 can, therefore, be seen as either "John Smith" writing to "Sarah Parker" or "Sarah Parker" reading from "John Smith".

We justify the symmetry between reading and writing by considering again the distinction between users and entities of traditional security models. Users are viewed as outside a computer system and it is, therefore, not logical for entities inside the system to 'write' information to the users. It makes much more sense from that perspective to say that a user reads information from an entity instead of saying that the entity has written information to the user. A write operation is, therefore, an operation that is performed upon an entity inside a computer system. InfoPriv assumes no distinction between users and entities (both users and entities are viewed as entities modelled inside the world) so the difference between reading and writing is less important.

The last point of discussion for this section is the question of what constitutes an entity and what not. Consider "John Smith" again. It is quite intuitive to visualise "John Smith" as an entity since John can be uniquely identified. What about John's salary? Does it qualify as an entity or should it be an attribute of John? The answer to this question is that it depends on the situation (also depending on the definition of an attribute that is covered in the formalisation of InfoPriv).

It is perfectly valid to view John's salary as a number that should be stored as an attribute of John since it cannot be uniquely identified. However, it is also valid to store John's salary in a separate entity provided that this entity can be uniquely identifiable. John's salary would be stored in a separate entity when it is necessary to control the flow of information to and from John's salary separately from John. Figure 5.2 depicts a separate entity ("John Smith's Salary") that stores John's salary.

Figure 5.2 further shows that information can flow from the entity "John Smith's Salary" to "John Smith" and via "John Smith" to "Sarah Parker". This is the notion of indirect information flow that will be discussed in the following section.

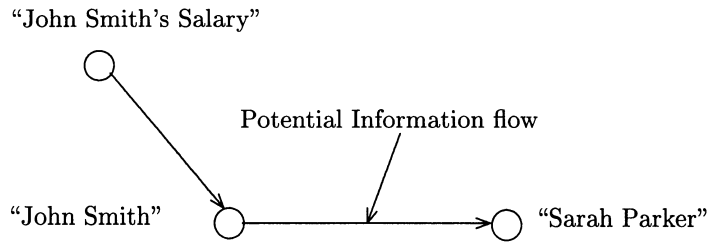

Figure 5.2 Storing of an entity's information in another entity.

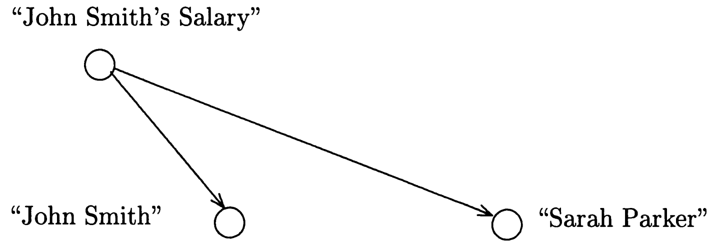

Figure 5.3 Direct (potential) information flow.

5.3.1 Indirect and negative (potential) information flow

Indirect information flow occurs when an entity is capable of disclosing information that it received from other entities (refer to the example of John Smith's salary). This is in contrast to direct information flow where the possible information flow in a system is explicitly defined. An example of direct information flow would be "information can flow from John Smith's Salary to John Smith and from John Smith's Salary to Sarah Parker". Assume for the moment that only direct information flow can occur. The privacy policy of Figure 5.2 would then be represented as depicted in Figure 5.3.

Note that information may be prevented to flow from one entity (say "John Smith's Salary") to another entity (say "Sarah Parker") by removing the corresponding arc from the can-flow graph (see Figure 5.3).

Indirect information, however, is more complex and prevents one from determining the information flow between entities by a simple examination of the can-flow graph. A mechanism is needed by which information can be explicitly prevented from flowing between two specific entities. We will use the notion of negative information flow for this purpose.

Suppose that we want to prevent indirect information flow between "John Smith's Salary" and "Sarah Parker" in Figure 5.2. A convenient way of depicting this is by making use of special arcs (called negative arcs) that represent

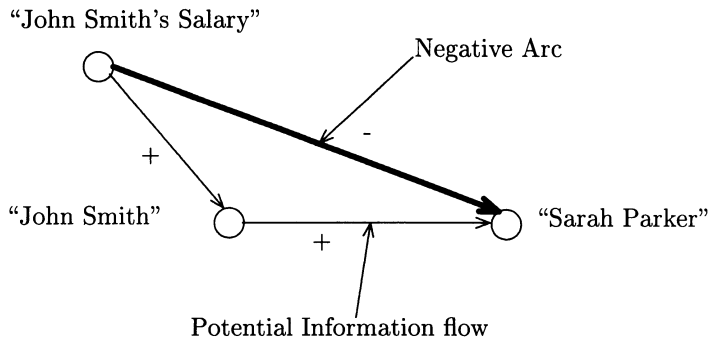

Figure 5.4 Negative (potential) information flow.

illegal information flow. Figure 5.4 depicts positive and negative arcs. Thick lines and negative labels indicate negative arcs.

The positive arcs between "John Smith's Salary" and "John Smith" and between "John Smith" and "Sarah Parker" indicate positive (potential) information flow. Positive (potential) information flow is permitted by the privacy policy and can be direct or indirect.

The negative arc between "John Smith's Salary" and "Sarah Parker" indicates that information is not permitted to flow either directly or indirectly from "John Smith's Salary" to "Sarah Parker". Mechanisms will be discussed by which negative arcs can be implemented (refer to the Dynamic Aspects of InfoPriv).

We conclude this section by introducing the idea of trust and information sanitisers. Current security models are sometimes too strict concerning the flow of information between entities [11]. Assume that "John Smith" and "Sarah Parker" work in the same department in a company. They need to exchange information in order to do their tasks. However, according to information-flow models such as the Bell-LaPadula model [6] information is prohibited from flowing from "John Smith" to "Sarah Parker" in order to prevent John from disclosing his salary to Sarah. This is unrealistic since most organisations let colleagues communicate with each other with the clear understanding that payroll information should not be discussed. People are, therefore, trusted not to disclose certain information.

One way of permitting John to talk to Sarah without disclosing his salary is to make John an information sanitiser. This is done by dividing him into logical sub-entities with each sub-entity corresponding to information that John discloses to specific entities. Figure 5.5 shows "John Smith" as an information sanitiser [11]. Note that this division of "John Smith" into sub-entities is according to the PoC. The more accurately we can model the way in which John Smith compartmentalises and discloses information, the higher is the level of privacy that the security system may provide.

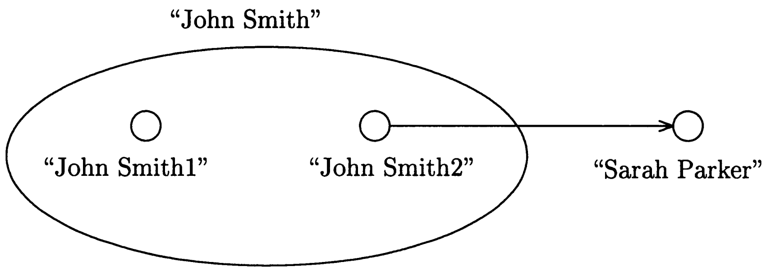

Figure 5.5 An information sanitiser.

Figure 5.5 shows that "John Smith" consists of two logical entities: "John Smithl" and "John Smith2". "John Smithl" is the part of "John Smith" that contains confidential information while "John Smith2" contains public information. "John Smith" is permitted to select information from "John Smithl" and 'place' it in "John Smith2" thereby publishing it (no information flow occurs from our point of view since we cannot tell what John is thinking).

Definition 1: An information sanitiser is an entity that is permitted to selectively disclose confidential information to other entities. Such an entity is modelled as consisting of logical sub-entities with each sub-entity containing information that is intended for disclosure to specific entities. Note that an information sanitiser models trust.

So far we have given an informal overview of InfoPriv and will proceed to formalise it in the following sections.

5.4 FORMALISATION

We will now develop InfoPriv formally by using graph theory. A can-flow graph can be defined as the tuple G = (E,→) where E is the set of entities and → is the set of arcs (or potential information flow between the entities).

Figure 5.4 depicts a can-flow graph with E = {"John Smith's Salary", "John Smith", "Sarah Parker"} and → = {<"John Smith's Salary", "John Smith", +1>, <"John Smith", "Sarah Parker", +1>, <"John Smith's Salary", "Sarah Parker", -1>}. Note that '+1' indicates a positive arc while '-1' indicates a negative arc. We will use angled brackets '<', '>' throughout the rest of this paper to indicate tuples.

Each entity is associated with a set of values (i.e. John Smith is 30 years old and lives in 10 Bourbon Street). We define the set V of all the entity-values, A the set of all attribute names and the function ϕ to map entities and their attributes to their corresponding values.

Definition 2: The set V consists of all the values. For example, V = {10000, "10 Bourbon Street", ...}.

Definition 3: The set A consists of all the attribute names. For example, A = {"value", "salary", "designation", "age", "address", ...}.

Definition 4: The function ϕ: E $\times$ A $\rightarrow$ V maps the entities and their attributes to corresponding values. If ϕ is represented by the set of triples {<"John Smith", "Age", 30>, <"John Smith", "Address", "10 Bourbon Street" >} then we can conclude that John Smith's Age is 30 (as stored in the attribute "Age" of entity "John Smith") and his address is "10 Bourbon Street".

We stated in the previous section that indirect information flow can occur between entities. Indirect information flow will now be formally defined in terms of the full-reachability function ψ.

Definition 5: The full-reachability function ψ: E$\rightarrow$ $\wp$ (E) maps each entity to the set of entities that can be reached from it in the can-flow graph. Formally, for an entity e_i

$$\psi'(e_i) = \{e_j \mid (e_j \in \mathrm{E}) \land (< e_i, e_j, +1 > \in \rightarrow)\}$$

$$\psi(e_i) = \psi'(e_i) \bigcup_{e_j \in \psi'(e_i)} \psi(e_j)$$

For instance, the full-reachability set of "John Smith's Salary" is {"John Smith", "Sarah Parker"} since its value can flow to "John Smith" and to "Sarah Parker" (via "John Smith"). Note that ψ ignores negative arcs.

Information is permitted to flow from an entity to any entity in its full-reachability set provided that a negative arc does not prevent this flow. For instance, if ψ ("John Smith's Salary") = {"John Smith", "Sarah Parker"} then information can flow from "John Smith's Salary" to "John Smith" but not to "Sarah Parker" since the arc <"John Smith's Salary", "Sarah Parker", -1>$\in\rightarrow$.

To conclude, the static aspects of InfoPriv may be used for a static analysis of a privacy policy (hence the can-flow graph). Potential unauthorised information flow may be identified and the can-flow graph can be modified to prevent such information flow from occurring during the lifetime of the system. Refer to [10] for a description of an InfoPriv Workbench that may be used for this purpose.

The next section will contain a description of the dynamic aspects of InfoPriv.

5.5 DYNAMIC ASPECTS

The previous section formalised the Static Aspects of InfoPriv. A can-flow graph is really a representation of how information can flow between entities (the static aspects). We correspondingly refer to the can-flow graph as a can-flow relation [14]. Consider the can-flow graph of Figure 5.2 again. The only information contained in Figure 5.2 is that information may flow from "John Smith's Salary" to "John Smith" and information may further flow from "John Smith" to "Sarah Parker".

We cannot deduce from a can-flow graph what information flow will actually occur and in what order. For instance, information may first flow from "John Smith" to "Sarah Parker" followed by information flow from "John Smith's Salary" to "John Smith". This will not lead to the disclosure of John's salary to Sarah Parker (assuming that "John Smith" does not know his salary at first).

However, by analysing the static aspects we can only make the pessimistic assumption that Sarah Parker will eventually be able to determine John's salary.

This unauthorised information flow can be prevented by either preventing the direct information flow from "John Smith's Salary" to "John Smith" or by preventing the direct information flow from "John Smith" to "Sarah Parker". John obviously has to know his salary, so the direct information flow from "John Smith" to "Sarah Parker" has to be prevented (John and Sarah may be placed in different departments). Note that numerous graph traversal algorithms do exist by which paths through a directed graph can be found [7]. We will not discuss these and possible conflict resolution algorithms further. Refer to [10] for a discussion of conflicts and conflict resolution of a can-flow graph.

The other alternative is to determine the actual information flow during run-time. This constitutes the dynamic aspects of InfoPriv. A few observations about the 'world' are in order before we describe the dynamic aspects of InfoPriv. We assume that the world consists of entities where each entity has its own viewpoint of the world. These viewpoints change as a result of interactions between entities. For instance, the personnel manager at John Smith's company may have decided to change John's salary without notifying John. The viewpoint of "John Smith's Salary" is different from John's since he only remembers an earlier value of his salary. However, the personnel manager can inform John of his new salary or John can see it on his salary statement at the end of the month, thereby changing his viewpoint of the world.

Dynamic aspects may be conveniently modelled by extending the definition of the ϕ function. Each entity has an associated ϕ function instead of one global ϕ function. For instance, the ϕ function for "John Smith" will be denoted by $\phi_{John\ Smith}$ where $\phi_{John\ Smith}$ represents John Smith's view of the world. The change of $\phi_{John\ Smith}$ over time models the change of John Smith's viewpoint over time (hence John's dynamic aspects).

Definition 6: The function ϕ_{Entity} represents Entity's viewpoint of the world. The change of ϕ_{Entity} represents the dynamic aspects of "Entity".

We will now define how information flow between two entities will influence their viewpoints. Assume that $\phi_{John\ Smith's\ Salary}$ = {<"John Smith's Salary", "value", 20000>}, $\phi_{John\ Smith}$ = {<"John Smith's Salary", "value", 10000>} and information flows from "John Smith's Salary" to "John Smith". "John Smith" will now know the new value of his salary. After the information flow $\phi_{John\ Smith}$ will be modified to $\phi_{John\ Smith}$ = {<"John Smith's Salary", "value", 20000>}.

Note that entity attributes can be changed by using information flow. Assume the privacy policy of Figure 5.6. The value of "John Smith's Salary" can be changed by the entity "Personnel Manager" since information can flow from it to "John Smith's Salary". Assume that $\phi_{Personnel\ Manager}$ = {<"John Smith's Salary", "value", 10000>}. "Personnel Manager" can now change his 'memory' to $\phi_{Personnel\ Manager}$ = {<"John Smith's Salary", "value", 20000>}. Information flow from "Personnel Manager" to "John Smith's Salary" will update John Smith's salary to 20000.

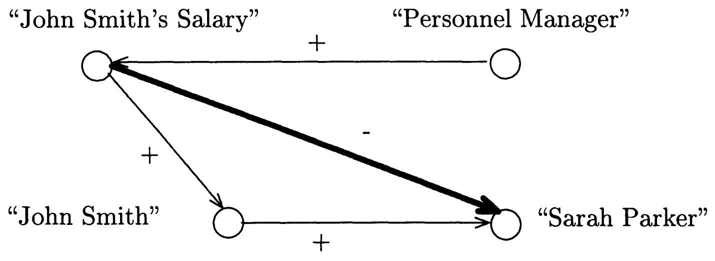

Figure 5.6 Changing of values by means of information flow.

Negative information flow can be implemented with the ϕ functions of the various entities. For instance, assume that we want to prevent the value of "John Smith's Salary" to reach "Sarah Parker". We can permit information to flow from "John Smith" to "Sarah Parker" as long as information has not flowed from "John Smith's Salary" to "John Smith" ($\phi_{John\ Smith}$ should, therefore, not contain a triple of the form <"John Smith's Salary", a, v> with a $\in$ A, v $\in$ V.

Note that it may still not be realistic to prohibit John Smith from talking to Sarah Parker after he has determined his salary. Information sanitisers can be used to model John Smith more realistically in terms of sub-entities (refer to the indirect information flow of InfoPriv).

The dynamic aspects (as discussed in this section) are summarised in the algorithm of Figure 5.7. Lines 1 to 5 test whether any of the entities of which information has reached entity t may not reach entity u. If this is not the case the ϕ function of entity u is updated according to ϕ_t (refer to the example of "John Smith's Salary" and the "Personnel Manager").

Note that negative arcs are not transitive. Assume that a negative arc indicates that information may not flow from entity A to entity B and a second arc indicates that information may not flow from entity B to entity C. This does not mean that information may not flow from entity A to entity B (there may be a positive arc from entity A to entity C). Only a direct negative arc from entity A to entity C will prevent information to flow from entity A to entity C.

A prototype of the dynamic aspects has been implemented in Prolog. This prototype has been quite successful in representing and analysing various privacy policies such as the IRS scenario mentioned previously. It is general enough to permit the specification of any privacy requirement that can be described in terms of Prolog predicates. We are in the process of testing the prototype for more general privacy policies such as workflow. The details of this prototype are outside the scope of this paper.

DoFlow ($\rightarrow_{pos}$, $\rightarrow_{neg}$, E, t, u, success)

Input: $\rightarrow_{pos}$ is the set of positive arcs, $\rightarrow_{neg}$ the set of negative arcs, E the set of entities and t, u $\in$ E (information flow from t to u is attempted)

Output: success is a flag that indicates whether the flow was successful or not

```
1   for all $< e_i, a_i, v_i > \in \phi_t$
2       where $e_i \in$ E, $a_i \in$ A and $v_i \in$ V do
3       if $< e_i$, u $> \in \rightarrow_{neg}$ then
4           success $\leftarrow$ False
5           exit
6   update $\phi_u$ according to $\phi_t$
7   success $\leftarrow$ True
```

Figure 5.7 Algorithm to implement the dynamic aspects of InfoPriv.

5.6 CONCLUSIONS

It is crucial to ensure the privacy of personal information in order to prevent the misuse of such information. Suitable privacy models, therefore, have to be developed by which the privacy policy of a system can be described and enforced. This paper has presented a model (called InfoPriv) for privacy that is based on the Principle of Completeness (PoC). This principle dictates that the level of privacy that can be guaranteed by a system depends on how thoroughly the context of the information can be modelled.

One of the results of the PoC is the unification of users and entities. Instead of making a distinction between the users of the system and the entities we chose to model only entities. Traditional data entities map directly to the entities as defined by InfoPriv while users map to entities with weaker guarantees (we cannot determine the precise information flow between people for instance). The entities (viewed as information containers in InfoPriv) together with the flow of information between them form the building blocks of InfoPriv.

We further noted that indirect information flow can occur between entities. It is necessary to control the indirect information flow between entities and this is done by means of negative information flow. Negative information flow was defined as a way to prevent the actual flow of information between two entities.

An InfoPriv policy maps well to a graph with the entities forming the vertices while the potential information flow between the entities corresponds to directed arcs. We introduced the idea of an information sanitiser. An information sanitiser is an entity that is trusted to selectively disclose confidential information to other entities. InfoPriv was further formalised in terms of graph theory.

We proposed two ways of implementing negative information flow. The first involves a static analysis of the can-flow graph for conflicting information flows (positive information flow that is prohibited by negative information flow). The second method involves a dynamic analysis of the actual information flow during run-time.

We lastly discussed the dynamic aspects of InfoPriv. The definition of an entity was extended by adding a local view of the world to each entity instead of having a global or absolute world. The dynamic aspects, therefore, correspond to a change over the local views of all the entities. We showed how an entity can change the attribute of another entity by first altering its own 'memory' and initiating an information flow to the destination entity after that.

It was, therefore, attempted to illustrate in this paper how a privacy model can be derived by considering relevant privacy issues. The privacy issues, a full mathematical model of privacy and the implementation of the model warrant further research. Note that a Prolog prototype based on InfoPriv has been implemented and tested successfully on a variety of privacy policies.

Acknowledgments

We would like to thank Professor Sushil Jajodia for his generous contributions to this paper.

References

[1] Anonymous, General Accounting Office United States, National Crime Information Center: Legislation Needed to Deter Misuse of Criminal Justice Information, Document GAO/T-GGD-93-41, Washington, 1993

[2] Anonymous, European Union, Directive 95/46/EC on the Protection of Individuals With Regard to the Processing of Personal Data and on the Free Movement of such Data, 1995

[3] Anonymous, Information Infrastructure Task Force, Privacy Working Group, Principles for Providing and Using Personal Information, Washington, 1995

[4] Anonymous, General Accounting Office United States, IRS Systems Security: Tax Processing Operations and Data Still at Risk Due to Serious Weaknesses, Document GAO/AIMD-97-49, Washington, 1997

[5] Anonymous, Information Infrastructure Task Force, Information Policy Committee, Options for Promoting Privacy on the National Information Infrastructure, Washington, 1997

[6] DE Bell, LJ LaPadula, "Secure Computer Systems: Unified Exposition & Multics Interpretation", Technical Report MTIS AD-A023588, MITRE Corporation, 1975

[7] TH Cormen, CE Leiserson, RL Rivest, *Introduction to Algorithms*, McGraw-Hill Book Company, MIT, 1994

[8] DE Denning, "A lattice Model for Secure Information Flow", *Communications of the ACM*, **19**, 5, 236–243, 1976

[9] LCJ Dreyer, MS Olivier, "Dynamic Aspects of the InfoPriv Model", In: Proc. 9th *Database and Expert Systems Applications Dexa 98* (RR Wagner Editor), IEEE Computer Society, Los Alamitos, 340–345, 1998

[10] LCJ Dreyer, MS Olivier, "A workbench for privacy policies", In: Proc. 22th *Computer Software and Applications Conference Compsac 98* (E Hughes editor), IEEE Computer Society, Los Alamitos, 350–355, 1998

[11] S Hsieh, E Unger, R Mata-Toledo, "Using Program Dependence graphs for information flow control", *J. Systems Software*, 17, 227–232, 1992

[12] CP Pfleeger, *Security in Computing*, Prentice Hall, New Jersey, 1989

[13] F Rabitti, E Bertino, W Kim, D Woelk, "A model of authorization for next-generation database systems", *ACM Transactions on Database Systems*, **16**, 1, 1991 88–131.

[14] RS Sandhu, "Lattice-Based Access Control Models", *IEEE Computer*, 9–19, 1993