Dynamic Aspects of the InfoPriv Model

Lucas C.J. Dreyer & Martin S. Olivier

Department of Computer Science, Rand Afrikaans University,
PO Box 524, Auckland Park, Johannesburg, South Africa 2006
lucas.dreyer@eskom.co.za
molivier@rkw.rau.ac.za

Abstract

This paper describes the dynamic aspects of the InfoPriv model. The dynamic aspects are concerned with the actual information flow as well as the change of the privacy policy over time. An information can-flow graph represents the potential information flow between entities, thereby describing only the static aspects of InfoPriv. The dynamic aspects are divided into two categories: dynamic information flow and the evolution of the static aspects. "Dynamic information flow" refers to the simulation of the actual information flow as it occurs over time. Information flow is, therefore, permitted or denied based on the past information flow of a system. The "evolution of the static aspects" refers to the change of the privacy policy over time. For instance, it is necessary to give entities access to more information during their lifetime. An algorithm is presented that extends the can-flow graph without introducing unauthorised information flow.

1. Introduction

An increasing amount of personal information is being stored in large databases. The privacy of the individual can be greatly compromised if this information is not properly protected against misuse. However, traditional security models and mechanisms are insufficient to guarantee privacy in a computer system. This has been clearly demonstrated by the unauthorised disclosure of tax-related information in the IRS [5] and the NCIC of the USA [6].

We proposed an information-flow model (named InfoPriv) that can be used to model the privacy of a computer system [3]. The basic constructs of InfoPriv are entities and the information flow between the entities. This is in contrast to the distinction between subjects and entities in traditional security models [7]. Entities are defined as information containers and indirect information flow can occur between entities. Negative information flow (denied information flow) is used to control indirect information flow.

This paper describes the dynamic aspects of InfoPriv. Dynamic aspects are concerned with information flow as it occurs (instead of the potential information flow). We give an algorithm that can be used for simulating the information flow over time. We further extend this algorithm by introducing the notion of uncertain information flow between entities such as people.

Changes of the privacy model need to be addressed by a privacy model in order to guarantee the privacy of information. We give an algorithm in this paper that can be used to implement the evolution of the static aspects of InfoPriv over time. This algorithm extends the privileges of an entity (the information that the entity can receive) without introducing unauthorised information flow.

Note that implementation issues such as the scalability of the algorithms are outside the scope of this paper. Some implementation issues have been addressed in [4] by means of a Prolog prototype.

Section 2 gives an overview of the InfoPriv model of privacy. Section 3 contains a discussion of the information flow over time. Section 4 is concerned with the evolution of the static aspects.

2. Background

The purpose of this section is to give a brief overview of the InfoPriv model. Note that a complete discussion of InfoPriv can be found in [3] and we will only outline its main features in this section.

Entities and information flow will be discussed in the first part of this section. We will show how the potential information flow between entities can be represented with an information can-flow graph. The second part of this section is concerned with reachability sets. A reachability set is a way of representing indirect information

explicitly. Many of the InfoPriv algorithms can be greatly simplified by pre-calculating the reachability sets of entities.

2.1 Entities and information flow

The basic elements of InfoPriv are entities and the information flow between the entities. An entity is defined as a uniquely identifiable container of information about itself and other entities. Examples of entities are John Smith, Employee Table, File1 and Printer1. The information flow between entities is caused by interaction between the entities. We will now illustrate entities and information flow by means of an example.

Consider the Internal Revenue Service (IRS) of the USA. Assume that Sarah Parker and Jane Ullman are working for the IRS. Jane Ullman is assigned to process the tax information of an individual named John Smith. Assume further that Sarah Parker is the sister-in-law of John Smith. Employees are prohibited from seeing tax-related information about their relatives [5]. Information is, therefore, not permitted to flow directly from “John Smith’s Tax” to “Sarah Parker”.

However, this situation is more complicated when Sarah and Jane are friends. It now becomes possible that Jane will disclose John's tax information to Sarah. Information can, therefore, flow indirectly from John Smith's Tax to Sarah Parker and mechanisms are needed to control such indirect information flow. Negative information flow is used to for this purpose.

Figure 1 uses a directed graph to depict the above situation. Note that this type of graph is called an information can-flow graph (briefly referred to as a 'can-flow graph').

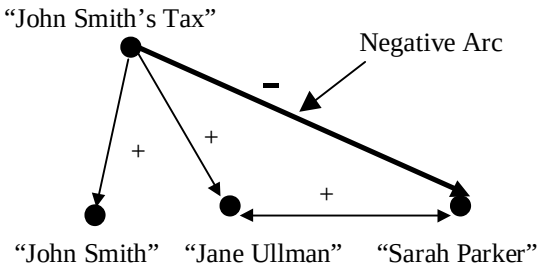

Figure 1 Negative information flow

The positive arc from "John Smith's Tax" to "John Smith" indicates that information is permitted to flow from "John Smith's Tax" to "John Smith". The same applies to the positive arcs between "John Smith's Tax" and "Jane Ullman" and between "Jane Ullman" and "Sarah Parker". Note that information may flow from "John Smith's Tax" to "Sarah Parker" via "Jane Ullman". This is referred to as indirect information flow.

The negative arc (the thick arc) indicates that information may not flow directly or indirectly from "John Smith's Tax" to "Sarah Parker".

This situation can be represented formally by a graph $G = (V, \rightarrow)$ where V is the set of entities and $\rightarrow$ is the set of directed arcs (positive and negative) between the entities. The positive arcs represent the possible information flow (see [1] for a description of graphs and graph theory) while the negative arcs indicate denied information flow.

Note that quotation marks will be used to refer to an entity (for instance “John Smith’s Tax”). Further note that the value of “John Smith’s Tax” can be stored as an attribute of “John Smith”. However, it is necessary in this instance to store John’s tax information separately in order to control what entities have access to it. Refer to [3] for a discussion of attributes in InfoPriv.

2.2 Reachability

It is important to clarify full and limited reachability of a can-flow graph before discussing the dynamic aspects. The full reachability set (abbreviated to 'full-reach') of a vertex contains all the vertices that are connected to that vertex by means of one or more paths. For instance, full-reach ("John Smith's Tax") = {"John Smith", "Jane Ullman", "Sarah Parker"}. Refer to [2] for an overview of path-finding algorithms that can be used to determine the full-reachability set of an entity.

The limited reachability set (simply referred to as 'reachability' or 'reach') of a vertex contains all the vertices that can receive information from that vertex (taking negative arcs into account). For instance, reach ("John Smith's Tax") = {"John Smith", "Jane Ullman"}. "Sarah Parker" is not in reach ("John Smith's Tax") because of the negative arc from "John Smith's Tax" to "Sarah Parker" in Figure 1.

The algorithm shown in Figure 2 can be used to determine the reachability sets of all the entities in the can-flow graph. The reachability sets of the entities in Figure 1 are shown in Table 1.

This concludes the introduction to InfoPriv and information flow. This section was mainly concerned with the static aspects of a privacy policy since a can-flow graph indicates potential information flow. The rest of this paper is devoted to the dynamic aspects of InfoPriv.

3. Dynamic information flow

The purpose of this section is to give an overview of techniques that can be used to keep track of the actual

information flow between entities over time. A dynamic analysis implies that the information flow between entities will be analysed as is it happens.

FindReach (E, Σ, V, u, reach-set)

Input: E is the set of positive arcs, Σ the set of negative arcs, V the set of vertices and $u \in V$
Output: reach-set will contain the set of vertices that are reachable from u (taking negative arcs into account)

S: a stack, initially empty

```
1    push u on S
2    while S is not empty
3        do pop a vertex u1 from S
4        for all vertices v ∈ V
5            do if ((u1, v) ∈ E) and (v ∉ reach-set))
6                then
7                    if (no vertex in reach-set is connected
                        to v by a negative arc) and
                        (u is not connected to v by a
                        negative arc)
8                        then add v to the reach-set
9                            push v on S
```

Figure 2 Finding the reachability set of a vertex

Table 1 Reachability sets for all the entities

Entity	reach (Entity)
"John Smith"	{}
"John Smith's Tax	{"John Smith", "Jane Ullman" }
"Jane Ullman"	{"Sarah Parker", "Jane Ullman"}
"Sarah Parker"	{"Jane Ullman", "Sarah Parker"}

3.1 The has-reached relation

We mentioned in Section 2.1 that the basic constructs of InfoPriv are entities and the information flow between the entities. Each entity has its own view of the world and this view is altered as the result of interaction between entities (see [3] for a description of an entity's "view of the world"). The entity "Jane Ullman" may start of as an "empty" entity with only knowledge about itself. "Jane Ullman" can now start to interact with other entities such as "Sarah Parker" and "John Smith's Tax". For example, information can flow from "John Smith's Tax" to "Jane Ullman" informing "Jane Ullman" of the tax-related information of "John Smith" in the process.

Note that after "Jane Ullman" has obtained the tax of "John Smith" it may happen that the tax of "John Smith" change later on. "Jane Ullman" will only know the old value of "John Smith's Tax" until she is informed of the new tax information. The has-reached relation is based on these ideas and will now be formalised.

Definition 1: The *has-reached relation* Γ maps each entity to the set of entities that have "reached" it. $\Gamma(\text{entity}_m) = \{\text{entity}_1, \text{entity}_2, \ldots, \text{entity}_n\}$ denotes the set of entities that have reached entity_m. We mean by "entity_i has reached entity_j" that information from entity_i has reached entity_j (at a specific moment). Note that the has-reached relation is initialised so that each entity maps to itself.

Consider the can-flow graph depicted in Figure 1. The Γ relation is initialised so that each entity maps to itself: For instance, Γ("John Smith's Tax") = {"John Smith's Tax"}, Γ("John Smith") = {"John Smith"} etc. This indicates that each entity's view of the world only includes itself.

Consider now what happens when "Jane Ullman" and "Sarah Parker" talk to each other. Jane will receive information from Sarah so that Γ("Jane Ullman ") = {"Jane Ullman ", "Sarah Parker"}. Sarah will receive information from Jane so that Γ("Sarah Parker") = {"Sarah Parker ", "Jane Ullman"}. Remember that Jane has not processed John's tax information at this stage so that the information exchange between Jane and Sarah is no problem.

Assume that information flows from "John Smith's Tax" to "Jane Ullman" as she is processing John's tax forms. The has-reached set of "Jane Ullman" will now contain "John Smith's Tax" as well as "Sarah Parker": Γ("Jane Ullman") = {"Jane Ullman", "Sarah Parker", "John Smith's Tax"}.

Any information flow from "Jane Ullman" to "Sarah Parker" that occurs at this stage will require the has-reached set of "Sarah Parker" to be changed to {"Sarah Parker", "Jane Ullman", "John Smith's Tax"}. This is not permitted because the negative arc in Figure 1 prohibits information to flow from "John Smith's Tax" to "Sarah Parker".

We can see from the above that the has-reached relation can be used to control the dynamic information flow (information flow as it occurs over time). An analysis of the can-flow graph of Figure 1 alone only shows that information from "John Smith's Tax" is able to reach "Sarah Parker" via the positive arcs. However, the negative arc indicates that such information flow is not permitted. We are left to either prevent information to flow from "John Smith's Tax" to "Jane Ullman" or prevent Jane and Sarah from talking to each other.

Figure 3 shows an algorithm that implements dynamic information flow by means of the has-reached relation. Note that the has-reached relation is transitive since

information that reached an entity is able to flow to other entities. The following section will extend the dynamic information flow of InfoPriv by introducing uncertain information flow.

DoFlow (E, Σ, V, u, v, success)

Input: E is the set of positive arcs, Σ the set of negative arcs, V the set of vertices, u, v ∈ V and information flow from u to v is attempted

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

```
1    for all u1 ∈ Γ (u) do
2        if ((u, v) ∉ E) or ((u1, v) ∈ Σ) then
3            success ← False
4            exit
5    Γ (v) ← Γ (v) ∪ Γ (u)
6    success ← True
```

Figure 3 Algorithm to do information flow

3.2 Uncertain information flow

It was mentioned previously that Jane Ullman and Sarah Parker are friends. They are people and their behaviour is less predictable than entities in a computer system. Hence we cannot always determine the exact information flow between them. However, we can make the pessimistic assumption that Jane and Sarah will share everything they know. This is referred to as 'uncertain information flow' since it is impossible to predict in advance whether information will actually flow between Jane and Sarah.

A privacy policy can be extended to model uncertain information flow by adding uncertain arcs to the can-flow graph. The uncertain arcs can be seen as forming a separate graph from the can-flow graph of the privacy policy. We will call this the *Uncertain Graph* and Figure 4 shows the Uncertain Graph for the privacy policy of Section 2. The uncertain arcs are depicted by dotted lines. Note that "Anne Summers" is a friend of both "Jane Ullman" and "Sarah Parker".

Assume that no information flow has occurred: Γ ("John Smith's Tax") = {"John Smith's Tax"} and Γ ("Jane Ullman") = {"Jane Ullman"}. When information flows from "John Smith's Tax" to "Jane Ullman" the has-reached set of "Jane Ullman" will be modified to {"Jane Ullman", "John Smith's Tax"}.

However, the uncertain arcs in Figure 4 indicate that all the information that flows to "Jane Ullman" will automatically flow to "Anne Summers" and "Sarah Parker". The can-reach sets of "Anne Summers" and "Jane Parker" will then be updated to:

Γ ("Anne Summers") = {"Anne Summers", "Jane Ullman", "Sarah Parker", "John Smith's Tax"} and Γ ("Sarah Parker") = {"Sarah Parker", "Jane Ullman", "Anne Summers", "John Smith's Tax"}.

Note that "John Smith's Tax" has reached "Sarah Parker". This is unauthorised information flow according to Figure 1 and should be prevented. Figure 5 shows an algorithm that can handle dynamic information flow of uncertain graphs.

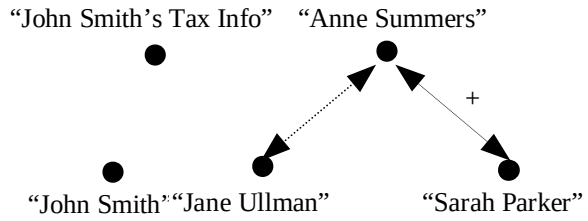

Figure 4 An uncertain Graph

DoFlow (E, Σ, V, U, u, v, success)

Input: E is the set of positive arcs, Σ the set of negative arcs, V the set of vertices, U is the set of uncertain arcs, u, v ∈ V and information flow from u to v is attempted

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

```
1    FindFullReach (U, V, v, reach-set)
2    for all u1 ∈ Γ (u) do
3        for all w ∈ (reach-set ∪ {v}) do
4            if ((u, v) ∉ E)) or ((u1, w) ∈ Σ then
5                success ← False
6                rollback
7                exit
8            Γ (w) ← Γ (w) ∪ {u1}
9    success ← True
10   commit
```

Figure 5 Extended algorithm to do information flow

Line 1 calculates the vertices in the uncertain graph that can be reached from v and stores them in reach-set (refer to the description of full and limited reachability in Section 2.2). Lines 2 to 7 determine whether any of the vertices in reach-set (including v) cannot receive information from either u or any vertex in Γ (u). If this is the case the information flow will be prohibited and the has-reached sets of all the entities will be restored with

the *rollback* command. Line 8 updates the has-reached sets of v and all the vertices in the uncertain graph that can be reached from v with all the vertices that have reached u (including u itself).

This concludes our discussion of the dynamic or actual information flow. The rest of this paper is dedicated to the dynamic aspects of the privacy policy itself (how it changes over time).

4. Evolution of the Static aspects

Dynamic information flow was discussed in the previous section. It is concerned with the analysis of information flow as it occurs while the privacy policy was assumed to be static. This is not realistic in practical applications since the privacy (and security) requirements of systems change constantly. Existing users regularly need different privileges and new users need to be added to the system. This section is concerned with the change of the Privacy Policy and ways in which this change can be done without introducing unauthorised information flow. This is referred to as the "Evolution of the Static Aspects".

Consider the lattice of Figure 6. The entity named 'subject' can read information from lower entities in the lattice such as Entity1 and Entity3. It can write information to higher entities such as Entity3 and Entity4. Refer to [7] for a discussion of lattice-based security models. Note that no distinction is made between subjects and entities in InfoPriv, only entities are used.

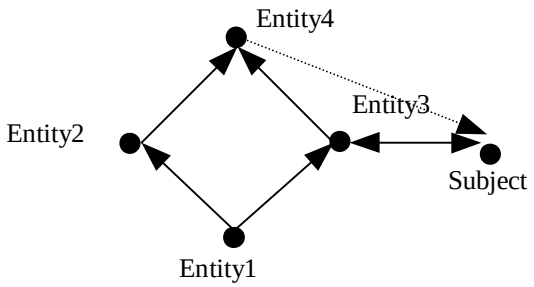

Figure 6 An example lattice

It can be seen in Figure 6 that information flows upward (between the entities) while information can flow between "Entity3" and "Subject". Suppose that we want information to flow between "Entity4" and "Subject", thereby 'promoting' "Subject" in the lattice (the dashed arrow indicates this promotion).

A way of thinking about the 'promotion' of an entity in a lattice is to see that the entity follows one of many alternative paths upwards in the lattice. For example, information can first flow between "Entity1" and "Subject1", then between either "Entity2" and "Subject" or between "Entity3" and "Subject" (but not both cases). At the highest level information will flow between "Entity4" and "Subject". Note that at any moment there will only be a single bi-directional arc between "Subject" and any one of the entities in the lattice. An informal algorithm is given in Figure 7 that can be used to 'promote' an entity in a can-flow graph.

Extend (E, Σ, V, u, v, success)

Input: E is the set of positive arcs, Σ is the set of negative arcs, V the set of vertices, v is the entity that should have access to u and u, v ∈ V

Output: success will be true if the extension was successful, false otherwise

Given: $W = \{u_1 \mid (u \in reach(u_1)) \wedge (v \in reach(u_1) \wedge ((u_1, v) \in E)\}$

```
1   success ← True
2   if W = ∅
3       then   success ← False
4              exit
5   for all u2 ∈ W do
6       if     (u2, v) ∈ E
7              then remove (u2, v) from E
8   add (u, v) to E
9   for all u2 ∈ reach (v) do
10      add (v, u2) to E
11  for all u2 ∈ E do
12      if     (u2 ∈ reach (v)) and
13             (u2 ∉ reach (u))
14             then add (v, u2) to Σ
```

Figure 7 An algorithm for extending the can-flow graph

Assume that we want "Subject" to be able to access information from "Entity4" (v is, therefore, "Subject" and u is "Entity4"). Lines 2 to 4 of the algorithm determine whether there is at least one entity from which information can flow to "Subject" and " Entity4". There should further be an arc from that entity to "Subject" in the can-flow graph (refer to the definition of the W set). The only entity that satisfies this criterion in Figure 6 is "Entity3". Since there is such an entity the test of lines 2 to 4 has succeeded and "Subject" can now be promoted so that it can send information to and receive information from "Entity4".

Lines 5 to 7 will remove arcs from the members of the W set to v. For the example of Figure 6 the arc from "Entity3" to "Subject" will be removed. Note that the arcs

originating from “Subject” will be unaffected at this stage.

The purpose of lines 11 to 14 is to make sure that the reachability set of u (“Entity4”) is not expanded. This is done by adding negative arcs to the can-flow graph. For instance, “Entity3” is in the reachability set of “Subject” but not in the reachability set of “Entity4”. A negative arc from “Subject” to “Entity3” is, therefore, added.

Note that lines 9 and 10 will explicitly add arcs to the can-flow graph for all the entities that are in the reachability set of “Subject”. According to Figure 6 “Entity3” and “Entity4” are in the reachability set of “Subject” so the arcs (“Subject”, “Entity3”) and (“Subject”, “Entity4”) should be added to the can-flow graph. The reason for this is that the negative arcs that are added in lines 11 to 14 may reduce the reachability set of “Subject” too much. This happens when some of the entities that can be reached indirectly from v get removed as well from v’s reachability set while they will not cause unauthorised information flow.

This concludes the discussion of the “evolution of the static aspects”. An algorithm was given that can be used to ‘promote’ an entity in the can-flow graph without causing unauthorised information flow. Note that this algorithm is not unique in that many alternative ways exist by which the can-flow graph can be modified. It is a topic of further research to determine what all the possible changes are of the static aspects and for what scenarios can they be applied (such as workflow).

5. Conclusions

A model called InfoPriv was proposed in [3] that can be used to model the privacy of a computer system. The basic constructs of InfoPriv are entities and the information flow between them. Negative information flow can be used to explicitly prevent information flow between specific entities. Policies (that are defined by using InfoPriv) translate directly to can-flow graphs, with the entities depicted by the vertices and the information flow depicted by the arcs.

A dynamic analysis of a can-flow graph is the simulation of the information flow as it occurs over time. Information flow from one entity (say “Entity1”) to another entity (say “Entity2”) can be permitted or denied on the basis of the information that “Entity1” has previously has had access to. This gives a finer-grained approach to the analysis of a privacy policy.

We further discussed changes of the privacy policy (and hence the can-flow graph) over time. This is referred to as the “evolution of the static aspects” (as opposed to the “dynamic information flow” which is a simulation of the actual information flow). We gave an algorithm that can be used to extend the privileges of an entity (the amount of information that the entity can receive) without introducing unauthorised information flow.

Further research needs to be done concerning the evolution of the static aspects of InfoPriv. It is important to find alternative algorithms for schemas such as workflow and more general privacy requirements.

6. References

[1] **C. Chartrand, L. Lesniak,** Graphs & Digraphs Third Edition, Chapman & Hall, London, 1996

[2] **T. H. Cormen, C. E. Leiserson, R. L. Rivest**, Introduction to Algorithms, McGraw-Hill Book Company, MIT, 1994

[3] **L. Dreyer, M. Olivier**, “An information-flow model for Privacy (InfoPriv)”, IFIP WG 11.3 Working Conference on Database Security, Chalkidiki, Greece, July, 1998

[4] **L. Dreyer, M. Olivier**, “A workbench for privacy policies”, Compsac 98 The Twenty-Second Annual International Computer Software and Applications Conference, Vienna, Austria, August, 1998

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

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

[7] **R.S. Sandhu,** “Lattice-Based Access Control Models”, IEEE Computer, 1993, pp. 9-19.