AMES GRANT
IN-82-CR

THEFT OF INFORMATION IN THE TAKE-GRANT PROTECTION MODEL

224739

P-27

Matt Bishop

Technical Report PCS-TR88-137

Theft of Information in the Take-Grant Protection Model

Matt Bishop

Department of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

ABSTRACT

Using the information transfer extensions to the Take-Grant Protection Model, the concept of ''theft of information'' is defined and necessary and sufficient conditions for such theft to occur are presented, as well as bounds on the number of actors involved in such theft. Finally, the application of these results to reference monitors are explored.

1. Introduction

The terms *security* and *safety* are often used interchangeably; in fact, they are not synonyms. The term ''safe'' applies to an abstract model; its initial state is called ''safe'' if it is not possible to reach a new state in which a right can be transferred. The term ''secure'' applies to a nonabstract system; it requires not only that the abstract model of the system be safe, but also that the nonabstract system correctly implement the abstract model. Harrison, Ruzzo, and Ullman showed that in general, it cannot be determined whether or not a system is safe [4].. The Take–Grant model, developed as a theoretical model to test the limits of this result [5], describes a simpler type of system, called a *mono–operational system* (because each command performs a single primitive operation). For such systems, there is an algorithm that decides whether a given mono-operational system and initial state is safe for a generic right. In fact, safety in the Take–Grant system is not only decidable even if the number of objects which can be created is unbounded, but it is decidable in time linear in the size of the graph. In[7] , the results were extended to include the theft of rights; in[2] , the notion of information flow was introduced and necessary and sufficient conditions for information sharing were formulated.

This paper extends this work. In the next two sections, we review the rules governing transfer of rights and information within the model, as well as some of the consequences of those rules. Next, we define, and present necessary and sufficient conditions for, the theft of information; following that, we present bounds on the number of actors needed for information to be shared (or stolen). We then discuss the implications of our work, and show how it can be applied in practise to reference

This work was supported in part by grant NAG 2-480 from the National Aeronautics and Space Administration to Dartmouth College, and a Burke award from Dartmouth College.

monitors. Finally, we suggest areas for future research.

2. Transfers of Authority

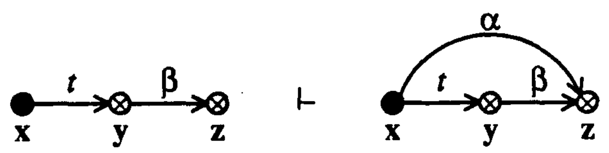
Let a finite, directed graph called a *protection graph* represent a system to be modelled. A protection graph has two distinct kinds of vertices, called *subjects* and *objects*. Subjects are the active vertices, and (for example) can represent users; they can pass information and authority by invoking *graph rewriting rules*. Objects, on the other hand, are completely passive; they can (for example) represent files, and do nothing.

In protection graphs, the subjects are represented by ● and objects by ○. Vertices which may be either subjects or objects are represented by ⊗ Pictures are very often used to show the effects of applying a graph rewriting rule on the graph; the symbol $\vdash$ is used to mean that the graph following it is produced by the action of the graph rewriting rule on the graph preceding it. The symbol $\vdash^*$ represents several rule applications. The term *witness* means a sequence of graph rewriting rules which produce the predicate or condition being witnessed, and a witness is often demonstrated by listing the graph rewriting rules that make up the witness (usually with pictures.)

The edges of a protection graph are labelled with subsets of a finite set R of rights. Suppose that $\{r,w,t,g\} \subseteq R$, where r, w, t, and g represent *r*ead, *w*rite, *t*ake, and *g*rant rights, respectively. When written as labels on a graph, the set braces are normally omitted.

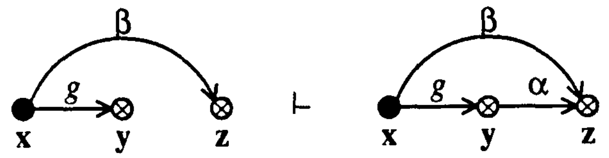
The Take–Grant Model permits users with certain rights to transfer rights from one vertex to another. The rules governing the transfer of rights are called *de jure* rules and are as follows :

take: Let **x**, **y**, and **z** be three distinct vertices in a protection graph G_0, and let **x** be a subject. Let there be an edge from **x** to **y** labelled γ with $t \in \gamma$, an edge from **y** to **z** labelled β, and $\alpha \subseteq \beta$. Then the *take* rule defines a new graph G_1 by adding an edge to the protection graph from **x** to **z** labelled α. Graphically,



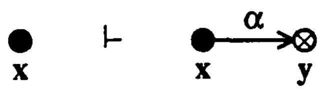
The rule is written: "**x** takes (α to **z**) from **y**."

grant: Let **x**, **y**, and **z** be three distinct vertices in a protection graph G_0, and let **x** be a subject. Let there be an edge from **x** to **y** labelled γ with $g \in \gamma$, an edge from **x** to **z** labelled β, and $\alpha \subseteq \beta$. Then the *grant* rule defines a new graph G_1 by adding an edge to the protection graph from **y** to **z** labelled α. Graphically,



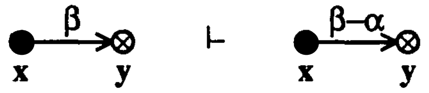
The rule is written: ''x grants (α to z) to y.''

create: Let x be any subject in a protection graph G_0 and let α be a subset of R. *Create* defines a new graph G_1 by adding a new vertex y to the graph and an edge from x to y labelled α. Graphically,



The rule is written: ''x creates (α to new vertex) y.''

remove: Let x and y be any distinct vertices in a protection graph G_1 such that x is a subject. Let there be an explicit edge from x to y labelled β, and let α be any subset of R. Then *remove* defines a new graph G_1 by deleting the α labels from β. If β becomes empty as a result, the edge itself is deleted. Graphically,

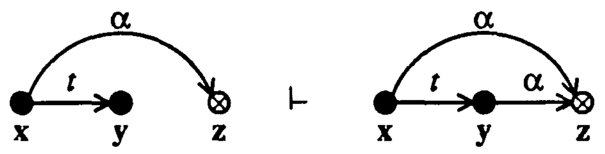


The rule is written: ''x removes (α to) y.''

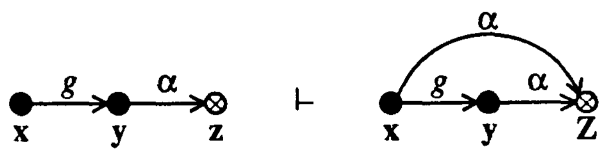
The edges which appear in the above graphs are called *explicit* because they represent authority known to the protection system.

Note that there is a duality between the take and grant rules when the edge labelled t or g is between two subjects. Specifically, with the cooperation of both subjects, rights can be transmitted backwards along the edges. The following two lemmas[5] demonstrate this:

Lemma 1:



Lemma 2:



As a result, when considering the transfer of authority between subjects, neither direction nor label of the edge is important, so long as the label is in the set $\{t,g\}$.

The first question that comes to mind is under what conditions can rights be shared? To answer this question, we first need to examine some characteristics of take-grant graphs.

Definition: A *tg–path* is a nonempty sequence $v_0, \ldots, v_k$ of distinct vertices such that for all i, $0 \le i < k$, v_i is connected to v_{i+1} by an edge (in either direction) with a label containing t or g.

Note that the vertices in a tg–path may be either subjects or objects.

Definition: Vertices are *tg–connected* if there is a tg–path between them.

Definition: An *island* is a maximal tg–connected subject–only subgraph.

Any right that one vertex in an island has can be obtained by any other vertex in that island. In other words, an island is a maximal set of subject–only vertices which possess common rights.

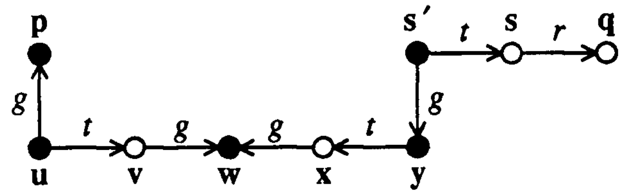
With each tg–path, associate one or more words over the alphabet $\{ \vec{t}, \overleftarrow{t}, \vec{g}, \overleftarrow{g} \}$ in the obvious way. If the path has length 0, then the associated word is the null word ν.

Definition: A vertex v_0 *initially spans* to v_k if v_0 is a subject and there is a tg–path between v_0 and v_k with associated word in $\{ \vec{t}, \vec{g} \} \cup \{\nu\}$.

Definition: A vertex v_0 *terminally spans* to v_k if v_0 is a subject and there is a tg–path between v_0 and v_k with associated word in $\{ \vec{t}^* \}$.

Definition: A *bridge* is a tg–path with v_0 and v_k both subjects and the path's associated word in $\{ \vec{t}^*, \overleftarrow{t}^*, \vec{t}^*\vec{g}\overleftarrow{t}^*, \vec{t}^*\overleftarrow{g}\overleftarrow{t}^* \}$.

An initial span is a tg–path along which the first vertex in the path can transmit authority; a terminal span is a tg–path along which the first vertex in the path can acquire authority. A bridge is an edge along which a right can be passed, possibly by using lemma 1 and 2 as well as the *de jure* rules. As a note, a bridge is said to be *directed away from* v_0. The following diagram illustrates these terms:



islands:	$I_1=\{p,u\}, I_2=\{w\}, I_3=\{y,s'\}$
bridges:	u,v,w and w,x,y
initial span:	p with associated word: ν
terminal span:	s′,s with associated word: $\vec{t}$

The following predicate formally defines the notion of *transferring authority*:

Definition: The predicate *can•share*(α, **x**, **y**, G_0) is true for a right α and two vertices **x** and **y** if and only if there exist protection graphs $G_1, \ldots, G_n$ such that $G_0 \vdash^* G_n$ using only *de jure* rules, and in G_n there is an edge from **x** to **y** labelled α.

In short, if **x** can acquire α rights to **y**, then *can•share*(α, **x**, **y**, G_0) is true. The theorem which establishes necessary and sufficient conditions for this predicate to hold is [6]:

Theorem 3: The predicate *can•share*(α, **x**, **y**, G_0) is true if and only if there is an edge from **x** to **y** in G_0 labelled α, or if the following hold simultaneously:

(i) there is a vertex $s \in G_0$ with an s-to-y edge labelled α;

(ii) there exists a subject vertex $\mathbf{p}'$ such that $\mathbf{p}'=\mathbf{p}$ or $\mathbf{p}'$ initially spans to $\mathbf{x}$;

(iii) there exists a subject vertex $\mathbf{s}'$ such that $\mathbf{s}'=\mathbf{s}$ or $\mathbf{s}'$ terminally spans to s; and

(iv) there exist islands $I_1, \ldots, I_v$ such that $\mathbf{p}'$ is in I_1, $\mathbf{s}'$ is in I_v, and there is a bridge from I_j to I_{j+1} $(1 \le j < v)$.

Finally, if the right can be transferred without any vertex which has that right applying a rule, the right is said to be stolen. Formally:

Definition: The predicate *can•steal*(α, **x**, **y**, G_0) is true for a right α and two vertices **x** and **y** if and only if there there is no edge labelled α from **x** to **y** in G_0, there exist protection graphs $G_1, \ldots, G_n$ such that $G_0 \vdash^* G_n$ using only *de jure* rules, in G_n there is an edge from **x** to **y** labelled α, and if there is an edge labelled α from s to **q**, then no rule has the form "**s** grants (α to **q**) to **z**" for any $\mathbf{x} \in G_j$, $(1 \le j < n)$.

Esentially, this says that *can•steal* is true if *can•share* is true, the right did not exist initially, and no owners of the right in the initial graph gave it away. Necessary and sufficient conditions for this to be true are [7]:

Theorem 4: The predicate *can•steal*(α, **x**, **y**, G_0) is true if and only if the following hold simultaneously:

(i) there there is no edge labelled α from **x** to **y** in G_0;

(ii) there exist a subject vertex $\mathbf{p}'$ such that $\mathbf{p}'=\mathbf{p}$ or $\mathbf{p}'$ initially spans to **x**; and

(iii) there is a vertex **s** with an edge from **s** to **q** labelled α in G_0; and

(iv) *can•share*(t, **p**, s, G_0) is true.

These rules apply only to the transfer of *rights*. But information may be transferred without any transfer of rights. Let us now examine this question.

3. Transfers of Information

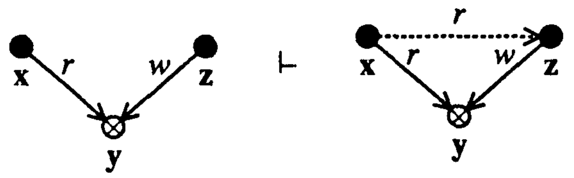
The *de jure* rules control the transfer of authority only; they say nothing about the transfer of information. The two are clearly different; for example, if a user is shown a document containing information which he does not have authority to read, the information has been transfered to the user. The *de jure* rules do not model cases like this. Instead, we use a different set of rules, called *de facto* rules, to derive paths along which information may flow.

In order to describe transfers of information, we cannot use explicit edges, because no change in authority occurs. Still, some indication of the paths along which information can be passed is necessary. Hence, we use a dashed line, labelled by r, to represent the path of a potential *de facto* transfer. Such an edge is called an *implicit* edge. Notice that implicit edges cannot be manipulated by the *de jure* rules, since the *de jure* rules can affect only authorities recorded in the protection system, and implicit edges do not represent such authority.

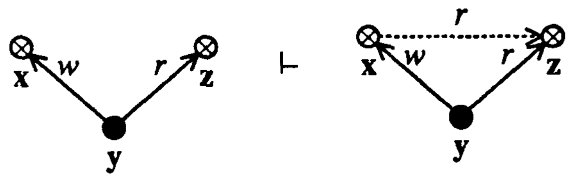
A protection graph records all authorities as explicit edges, so when a *de jure* rule is used to add a new edge, an actual transfer of authority has taken place. But when a *de facto* rule is used, a path along which information can be transferred is exhibited; the actual transfer may, or may not, have occurred. It is impossible to tell this from the graph, because the graph records authorities and *not* information. For the purposes of this model, however, we shall assume that if it is possible for information to be transferred from one vertex to another, such a transfer has in fact occurred.

One set of proposed *de facto* rules was introduced in[2] to model the transfer of information. Although these are not the only rules possible, their effects have been explored, and so we shall use them.

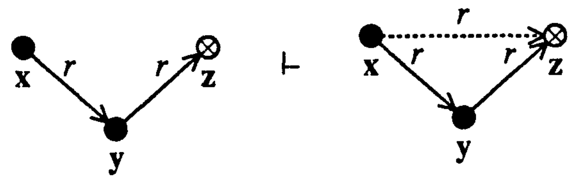
post: Let **x**, **y**, and **z** be three distinct vertices in a protection graph G_0 and let **x** and **z** be subjects. Let there be an edge from **x** to **y** labelled α, where $r \in \alpha$, and an edge from **z** to **y** labelled β, where $w \in \beta$. Then the *post* rule defines a new graph G_1 with an implicit edge from **x** to **z** labelled $\{r\}$. Graphically,



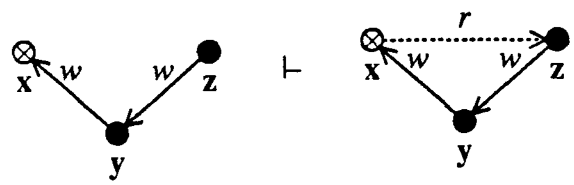
pass: Let **x**, **y**, and **z** be three distinct vertices in a protection graph G_0, and let **y** be a subject. Let there be an edge from **y** to **x** labelled α, where $w \in \alpha$, and an edge from **y** to **z** labelled β, where $r \in \beta$. Then the *pass* rule defines a new graph G_1 with an implicit edge from **x** to **z** labelled $\{r\}$. Graphically,



spy: Let **x**, **y**, and **z** be three distinct vertices in a protection graph G_0, and let **x** and **y** be subjects. Let there be an edge from **x** to **y** labelled α, where $r \in \alpha$, and an edge from **y** to **z** labelled β, where $r \in \beta$. Then the *spy* rule defines a new graph G_1 with an implicit edge from **x** to **z** labelled $\{r\}$. Graphically,



find: Let **x**, **y**, and **z** be three distinct vertices in a protection graph G_0, and let **y** and **z** be subjects. Let there be an edge from **y** to **x** labelled α, where $w \in \alpha$, and an edge from **z** to **y** labelled β, where $w \in \beta$. Then the *find* rule defines a new graph G_1 with an implicit edge from **x** to **z** labelled $\{r\}$. Graphically,



Note that these rules add implicit and not explicit edges. Further, as these rules model information flow, they *can* be used when either (or both) of the edges between **x** and **y**, or **y** and **z**, are implicit.

3.1. Information Flow in a Graph with Static Rights

Now, consider the conditions necessary for a potential *de facto* transfer to exist in a graph.

Definition: The predicate *can•know•f(***x**, y, G_0) is true if and only if there exists a sequence of graphs $G_1, \ldots, G_n$ $(0 \le n)$, such that $G_i \vdash^* G_{i+1}$ $(0 \le i < n)$ by one of the *de facto* rules and in G_n either aN **x**-to-**y** edge labelled r exists or a **y**-to-**x** edge labelled w exists and if the edge is explicit, its source is a subject.

Intuitively, *can•know•f*(**x**, **y**, G_0) is true if and only if **x** has the authority to read **y**, **y** has the authority to write to **x**, or an implicit edge from **x** to **y** can be added by means of the *de facto* rules. Note the duality of read and write. If **x** can write to **y**, then **y** effectively can read **x**. All **x** has to do is write to **y** any information that **y** wants to see. This duality will play an important role in later results.

Definition: An *rw–path* is a nonempty sequence $\mathbf{v}_0, \ldots, \mathbf{v}_k$ of distinct vertices such that for all i, $0 \le i < k$, $\mathbf{v}_i$ is connected to $\mathbf{v}_{i+1}$ by an edge (in either direction) with a label containing an r or a w.

With each rw–path, associate one or more words over the alphabet $\{ \overrightarrow{r}, \overleftarrow{r}, \overrightarrow{w}, \overleftarrow{w} \}$ in the obvious way; for instance, the protection graph

$$\begin{tikzcd} \bullet \arrow[r, "r"] & \circ & \bullet \arrow[l, "w"'] & \circ \arrow[l, "rw"'] \end{tikzcd}$$

has associated $\overrightarrow{r}\overleftarrow{w}\overleftarrow{r}$ and $\overrightarrow{r}\overleftarrow{w}\overleftarrow{w}$. If the path has length 0, then the associated word is the null word ν.

Definition: An rw–path $\mathbf{v}_0, \ldots, \mathbf{v}_k$, $k \ge 1$, is an *admissible rw–path* if and only if it has an associated word $a_1 a_2 \cdots a_k$ in the regular language $(\overrightarrow{r} \cup \overleftarrow{w})^*$, and if $a_i = \overrightarrow{r}$ then $\mathbf{v}_{i-1}$ is a subject and if $a_i = \overleftarrow{w}$ then $\mathbf{v}_i$ is a subject.

Note that there cannot be two consecutive objects on an rw–admissible path. Given these definitions,

Theorem 5: Let **x** and **y** be vertices in a protection graph G_0. Then *can•know•f*(**x**, **y**, G_0) is true if and only if there is an admissible rw–path between **x** and **y**.

3.2. Information Flow in a Graph with Changing Rights

These results can be extended to include both *de jure* and *de facto* rules. To do so, we must define terms combining characteristics of those used in both the *de jure* and *de facto* developments.

Definition: The predicate *can•know*(**x**, **y**, G_0) is true if and only if there is a sequence of protection graphs $G_1, \ldots, G_n$ such that $G_0 \vdash^* G_n$ and in G_n either a **x**-to-**y** edge labelled r exists, or a **y**-to-**x** edge labelled w exists and, if the edge is explicit, its source is a subject.

This is merely *can•know•f*(x, y, G_0) without the restriction on the types of rules used.

Definition: An *rwtg–path* is a nonempty sequence $v_0, \ldots, v_k$ of distinct vertices such that for all i, $0 \le i < k$, v_i is connected to v_{i+1} by an edge (in either direction) with a label containing a *t*, *g*, *r*, or a *w*.

With each rwtg–path, associate one or more words over the alphabet $\{ \overrightarrow{t}, \overleftarrow{t}, \overrightarrow{g}, \overleftarrow{g}, \overrightarrow{r}, \overleftarrow{r}, \overrightarrow{w}, \overleftarrow{w} \}$ in the obvious way.

Definition: The vertex v_0 *rw–initially spans* to v_k if v_0 is a subject and there is an rwtg–path between v_0 and v_k with associated word in $\{ \overrightarrow{t}^* \overrightarrow{w} \}$.

Definition: A vertex v_0 *rw–terminally spans* to v_k if v_0 is a subject and there is an rwtg–path between v_0 and v_k with associated word in $\{ \overrightarrow{t}^* \overrightarrow{r} \}$.

Definition: A *bridge* is an rwtg–path with associated word in the regular language

$$B = \{ \overrightarrow{t}^* \cup \overleftarrow{t}^* \cup \overrightarrow{t}^* \overrightarrow{g} \overleftarrow{t}^* \cup \overrightarrow{t}^* \overleftarrow{g} \overleftarrow{t}^* \}$$

(Note that this is the same as the definition given earlier in this section.) A *connection* is an rwtg–path with associated word in the regular language

$$C = \{ \overrightarrow{t}^* \overrightarrow{r} \cup \overleftarrow{w} \overleftarrow{t}^* \cup \overrightarrow{t}^* \overrightarrow{r} \overleftarrow{w} \overleftarrow{t}^* \}$$

The next result characterizes the set of graphs for which *can•know* is true:

Theorem 6: *can•know*(x, y, G_0) is true if and only if there exists a sequence of subjects $u_1, \ldots, u_n$ in G_0 ($1 \le n$) such that the following conditions hold:

(i) $u_1 = x$, or u_1 rw–initially spans to x,

(ii) $u_n = y$, or u_n rw–terminally spans to y,

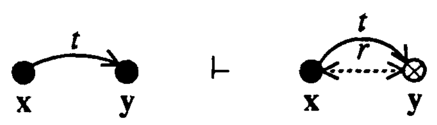
(iii) for all i, $1 \le i < n$, there is an rwtg–path between u_i and u_{i+1} with an associated word in $B \cup C$.

In order to appreciate these results, let us now look at some examples of the uses of the rules and theorems; these will be useful in deriving our later results.

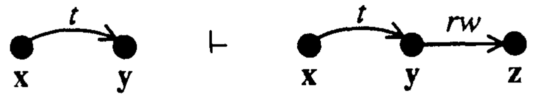
4. Some Examples of Combined *de jure* and *de facto* Rule Applications

In this section we present results which not only are good examples of how the graph rewriting rules and the theorems in the previous sections are used, but also which will be quite useful in our later work. The first two results are quite basic, and state that that if one subject has take or grant rights over another subject, either can (with the co-operation of the other) read information from the other. More formally:

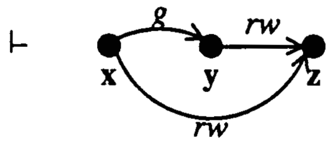
Lemma 7:



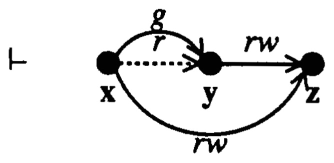
Proof: First, y creates (*rw* to new vertex) z:



Then, **x** takes (*rw* to **z**) from **y**:

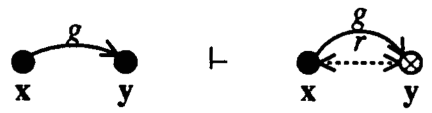


Finally, **x** and **y** use the post rule:

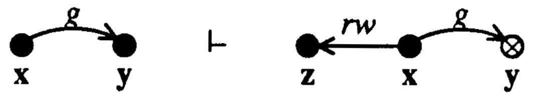


Note that the direction of the implicit read edge depends on which rights **x** and **y** use. If **x** writes to **z** and **y** reads from **z**, the implicit edge (information flow) goes from **x** to **y**. If, on the other hand, **y** writes to **z** and **x** reads from **z**, the implicit edge (information flow) goes from **y** to **x**. □

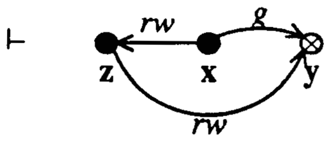
Lemma 8:



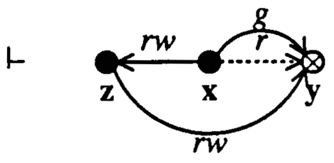
Proof: First, **x** creates (*rw* to new vertex) **z**:



Then, **x** grants (*rw* to **z**) to **y**:



Finally, **x** and **y** use the post rule:



As with the previous lemma, note that the direction of the implicit read edge depends on which rights **x** and **y** use. □

Next, we consider the following question: suppose we have a graph with three vertices; two subjects **x** and **y**, and another vertex **z** which may be either a subject or an object. There is a path from **x** to **z** and a path from **y** to **z**; these are the only paths in the graph. Furthermore, these paths may be initial, terminal, rw-initial, or rw-terminal (any combination is possible.) Our problem is to derive witnesses to *can•know*(**x**, **y**, G) for those combinations of paths for which that predicate is true, and prove that predicate is false for the others.

The easiest way to do this is to consider each combination of paths. First, if the path from **x** to **z** is rw-terminal, it does not matter whether or not **z** is a subject or an object; if the former, the word associated with the **x**-to-**z** path is not in the set $B \cup C$, and if the latter, the word associated with the **x**-to-**y** path is not in the set $B \cup C$ either. In those four cases *can•know*(**x**, **y**, G) is false by condition (iii) of theorem 6. If the path from **y** to **z** is rw-initial, similar reasoning shows that, again, *can•know*(**x**, **y**, G) is false.

Let us now consider the remaining cases one by one. Throughout the remainder of this discussion, we will assume that initial, terminal, rw-initial, and rw-terminal paths all are of length 1. We may do so without loss of generality because if the path is longer, all edges but the first are take edges, and so by repeated applications of the take rule the vertex at the source of the directed path may obtain an edge with the rights of the last edge in the path. We shall also not draw the pictures as we did in the two previous lemmas.

x-*to*-**z** *terminal*, **y**-*to*-**z** *terminal*

First, if **z** is an object, the word associated with the path between **x** and **y** is not in $B \cup C$ and therefore the predicate is false. But if **z** is a subject, the following is a witness:

(1) **z** creates (*rw* to new vertex) **v**.

(2) **x** takes (*r* to **v**) from **z**.

(3) **y** takes (*w* to **v**) from **z**.

(4) **y** and **x** use the post rule to obtain an implicit *r* edge from **x** to **y**.

This verifies *can•know*(**x**, **y**, G). Note that all three vertices **x**, **y**, and **z** must act.

x-*to*-**z** *terminal*, **y**-*to*-**z** *initial*

In this case it is not relevant whether **z** is an object; the following witness works in either case.

(1) **y** creates (*rw* to new vertex) **v**.

(2) **y** grants (*r* to **v**) to **z**.

(3) **x** takes (*r* to **v**) from **z**.

(4) **y** and **x** use the post rule to obtain an implicit *r* edge from **x** to **y**.

This verifies *can•know*(**x**, **y**, G). In this case, only **x** and **y** need act.

x-*to*-**z** *terminal*, **y**-*to*-**z** *rw-initial*

If **z** is an object, the word associated with the path between **x** and **y** is not in $B \cup C$ and therefore the predicate is false. But if **z** is a subject, the following is a witness:

(1) **z** creates (*rw* to new vertex) **v**.

(2) **x** takes (*r* to **v**) from **z**.

(3) **x** and **z** use the post rule to obtain an implicit *r* edge from **x** to **z**.

(4) **y** and **x** use the post rule to obtain an implicit *r* edge from **x** to **y**.

This verifies *can•know*(**x**, **y**, *G*). As in the first case we looked at, **x**, **y**, and **z** all need to act.

x-*to*-**z** *terminal*, **y**-*to*-**z** *initial*

The following witness works whether or not **z** is a subject:

(1) **x** creates (*rw* to new vertex) **v**.

(2) **x** grants (*w* to **v**) to **z**.

(3) **y** takes (*w* to **v**) from **z**.

(4) **y** and **x** use the post rule to obtain an implicit *r* edge from **x** to **y**.

This verifies *can•know*(**x**, **y**, *G*). Again, note that only **x** and **y** need act.

x-*to*-**z** *terminal*, **y**-*to*-**z** *rw-initial*

If **z** is an object, the word associated with the path between **x** and **y** is not in $B \cup C$ and therefore the predicate is false. But if **z** is a subject, the following is a witness:

(1) **z** creates (*rw* to new vertex) **v**.

(2) **x** takes (*r* to **v**) from **z**.

(3) **x** and **z** use the post rule to obtain an implicit *r* edge from **x** to **z**.

(4) **y** and **x** use the post rule to obtain an implicit *r* edge from **x** to **y**.

This verifies *can•know*(**x**, **y**, *G*). As in the first case we looked at, **x**, **y**, and **z** all need to act.

x-*to*-**z** *initial*, **y**-*to*-**z** *terminal*

The following witness works whether or not **z** is a subject:

(1) **x** creates (*rw* to new vertex) **v**.

(2) **x** grants (*w* to **v**) to **z**.

(3) **y** takes (*w* to **v**) from **z**.

(4) **y** and **x** use the post rule to obtain an implicit *r* edge from **x** to **y**.

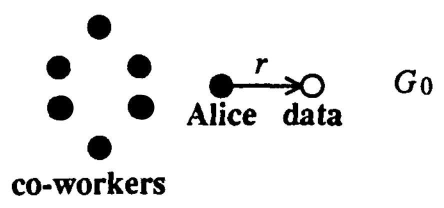
This verifies *can•know*(**x**, **y**, *G*). Again, note that only **x** and **y** need act.

Now that we have seen the basic model, let us consider the question of theft of information.

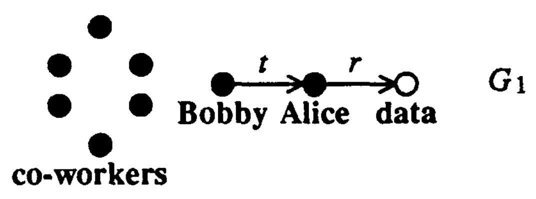
5. Snooping, or the Theft of Information

Up to this point, we have been considering cases where all vertices cooperate in sharing information, so all *de facto* rules may be applied with impunity. Suppose this is not true; suppose a vertex which has the right to read information from another vertex flatly refuses to pass the information along. Under what conditions can a vertex which does not have read rights over a second vertex obtain information from the second vertex?

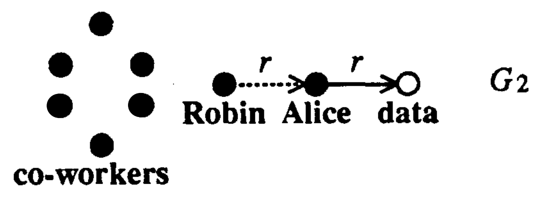
An example will help show what the problem is. Suppose Alice works for a firm which has proprietary information that its competitors need desperately to see. Alice, who works with this information quite a bit, has the authority to read the documents containing the proprietary information whenever she likes, with the understanding she is not to pass this sensitive data to anyone else, including co-workers. The situation, in Take Grant terms, is:



Any documents as sensitive as those which Alice consults must be kept under lock and key. Alice's company has a large vault, which is opened by a key that Alice has. One of her co-workers, Bobby, is not cleared to read these documents and does not have a key to the vault. While passing Alice's desk, he notes a key lying on top of it. Were Bobby to take that key, he would be "taking" Alice's right to read the documents, because she could no longer open the vault; in effect, he would have illicitly obtained the right to read those documents. He could also pass this information on to someone else. This is an example of Alice's sharing (albeit unknowingly) her right to read the documents:



Because he is honest, Bobby does not take the key, but merely suggests to Alice that she be a bit more careful. Later in the day, Alice takes a sensitive document out of the vault, goes back to her desk, and begins to read the document. Unfortunately, Robin, who sits directly behind Alice in her office, can see what Alice is reading just by looking over Alice's shoulder. In Take Grant terms, this situation is:



By the spy rule, Robin can read anything Alice can (the Robin-to-Alice edge, being unauthorized, is implicit); hence, *can•know*(Robin, "proprietary data", G_2) is true as long as Robin can look over Alice's shoulder; if Alice read the document elsewhere, such as in the vault, Robin would no longer be able to read the document over Alice's shoulder, so the spy rule would not be applicable since there would be no (implicit or explicit) Robin-to-Alice edge. Notice the difference between *can•know*(Robin, "proprietary data", G_2) and *can•know*(Bobby, "proprietary data", G_1); in the latter case, the *can•know* is true whether or not Alice cooperates by (knowingly or unknowingly) allowing her shoulder to be looked over. The *can•know* predicate fails to capture this distinction.

We define a new predicate, called *can•snoop*. This predicate will be true if *can•know* is true and no-one who has any rights over the information being snooped for cooperates with the snooper. For example, *can•snoop*(Robin, "proprietary data", G_2) is false, since Alice has to pass the information to Robin (by letting Robin look over her shoulder, in this example), whereas *can•snoop*(Bobby, "proprietary data", G_1) is true, since Bobby could see the documents whether or not Alice cooperated, once Bobby had "taken" them.

More formally,

Definition: The predicate *can•snoop(***p**, **q**, G_0) is true if, and only if, *can•steal*(r, **p**, **q**, G_0) is true or there exists a sequence of graphs and rule applications $G_0 \vdash_{\rho_1} \cdots \vdash_{\rho_n} G_n$ for which all of the following conditions hold:

(a) there is no explicit edge from **p** to **q** labelled r in G_0;

(b) there is an implicit edge from **p** to **q** labelled r in G_n; and

(c) neither **q** nor any vertex directly connected to **q** is an actor in a grant rule or a *de facto* rule resulting in an (explicit or implicit) read edge with **q** as its target.

Before we state necessary and sufficient conditions for *can•snoop* to be true, let us examine the definition more closely. The predicate is rather clearly the *de facto* analogue of *can•steal*, just as *can•know* is the *de facto* analogue of *can•share*. If **p** can steal read rights to **q**, clearly no-one who owns those rights over **q** can prevent **p** from obtaining information from **q**. Similarly, if **p** has authority to read **q**, it would strain the meaning of what we are trying to define to say *can•snoop*(**p**, **q**, G_0) is true. In G_n, if *can•steal*(r, **p**, **q**, G_0) is false, note that any read edge from **p** to **q** must be implicit. And for the purposes of this discussion, we will assume that **q** will not cooperate (either wittingly or unwittingly) with any snooping; it would be equally reasonable to assume that **q** would cooperate, in which case what follows must be modified somewhat.

Theorem 9: For distinct vertices **p** and **q** in a protection graph G_0 with explicit edges only, *can•snoop(***p**, **q**, G_0) is true if and only if one of the following conditions holds:

(i) *can•steal*(r, **p**, **q**, G_0) is true; or,

(ii) all of the following hold simultaneously:

(a) there is no edge labelled r from **p** to **q** in G_0;

(b) there is a subject vertex **p**′ such that **p**′ = **p** or **p**′ *rw*-initially spans to **p**;

(c) there is a subject **q**′ such that **q**′ ≠ **q**, there is no edge labelled r from **q**′ to **q** in G_0, and **q**′ *rw*-terminally spans to **q**; and

(d) *can•know*(**p**′, **q**′, G_0) is true.

Informal argument: If *can•snoop* is true, and *can•steal* false, we have to show all parts of condition (ii) are true. Condition (ii)(a) follows from the definition. By part (b) of the definition, *can•know*(**p**, **q**, G_0) is true, from which condition (ii)(b) springs. Also, by theorem 6, condition (ii), we have **q**′. Combining this with the definition, it becomes clear that although **q**′ *rw*-terminally spans to **q**, **q**′ ≠ **q**, and there is no edge labelled r from **q**′ to **q** in G_0. The proof that *can•know*(**p**, **q**, G_0) is true involves proving that the first rule to add a read edge with target **q** is a take rule, and working backwards.

Going from the conditions to *can•snoop* is trivial.

Proof: (⇒) Let *can•snoop*(**p**, **q**, G_0) be true. If *can•steal*(r, **p**, **q**, G_0) holds, we are done. So, assume *can•steal*(r, **p**, **q**, G_0) is false.

Part (a) of the definition gives condition (ii)(a) of the theorem.

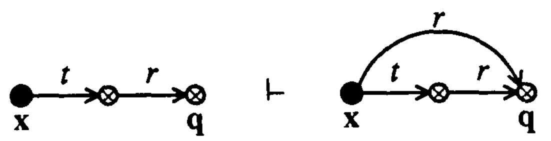
By part (b) of the definition, there is an implicit read edge from **p** to **q** in G_n, whence by definition *can•know*(**p**, **q**, G_0) is true; so, condition (ii)(b) of this theorem results from condition (i) of theorem 6.

By condition (ii) of theorem 6, there is a subject **q**′ such that **q**′ ≠ **q** or **q**′ *rw*-terminally spans to **q**. If **q** is an object, we can take **q**′ to be the **q**′ in condition (ii)(c) of this theorem. If **q** is a subject, by part (c) of the definition of *can•snoop*, it is not used in the sequence of rule applications witnessing *can•snoop*. Hence, in this case, **q**′ ≠ **q**; choose **q**′ in condition (ii)(c) of the theorem to be this **q**′. Thus, in either case, the **q**′ in condition (ii)(c) of this theorem is the same as the **q**′ as in condition (ii) of theorem 6.

Assume **q**′ and **q** are directly connected by an edge labelled r in G_0. Either *can•share*(t, **p**′, **q**′, G_0) is true [which means *can•steal*(r, **p**, **q**, G_0) is true, contradiction] or **q**′ must actively participate in a grant, pass, or spy rule application [contradicting part (c) of the definition of *can•steal*.] In either case, there cannot be an edge labelled r from **q**′ to **q** in G_0.

It remains to be shown that *can•know*(**p**′, **q**′, G_0) is true. Let $G_0 \vdash_{\rho_1} \cdots \vdash_{\rho_n} G_n$ be a minimum length derivation sequence, and let i be the least index such that $G_{i-1} \vdash_{\rho_i} G_i$, there is no explicit or implicit read edge from **x** to **q** in G_{i-1}, and there is an explicit or implicit read edge from **x** to **q** in G_{i-1}, where **x** is any vertex in G_{i-1}. That is, G_i is the first graph in which an edge labelled r with target **q** is added. Consider the rule ρ_i which caused this edge to be added. ρ_i cannot be a grant rule since, by part (c) of the definition of *can•snoop*, the owner of r rights to **q** will not will not grant them to anyone else. ρ_i cannot be a pass, spy, or find rule, since by part (c) of the definition of *can•snoop*, the owner of r rights to **q** will not pass information from **q** to anyone else. ρ_i cannot be a post rule since by part (c) of the definition of *can•snoop*, **q** will not pass information from itself to anyone else. As neither the create nor the remove rules add edges to existing vertices, ρ_i cannot be either. Hence, ρ_i must be a take rule.

We therefore have:

ρ_i: 

Recalling that *can•know*(**p**, **q**, G_0) is true, by theorem 6 we see that *can•know*(**p**′, **q**, G_0) is true. Apply theorem 6 again. By this theorem, there is a subject **q**′ such that **q**′ ≠ **q** or **q**′ *rw*-terminally spans to **q**. Noting that there is no direct edge labelled r from **q**′ to **q** in G_0, we take **q**′ = **x** in theorem 6 and in this theorem, whence *can•know*($\mathbf{p}'$, $\mathbf{q}'$, G_0) immediately follows.

($\Leftarrow$) If condition (i) holds, *can•snoop*($\mathbf{p}$, $\mathbf{q}$, G_0) is true.

So, assume condition (ii) holds. Part (a) of the definition is the same as condition (ii)(a) of the theorem. By theorem 6, conditions (ii)(b), (ii)(c), and (ii)(d) establish part (b) of the definition of *can•snoop*. And as $\mathbf{q}' \neq \mathbf{q}$ when $\mathbf{q}$ is a subject, part (c) of the definition is also true. This completes the proof of theorem 9. □

As an example, let us return to the office described at the beginning of this section. Consider G_1. By theorem 4, *can•steal*(Bobby, data, G_1) is true, so by condition (i) of the above theorem, *can•snoop*(Bobby, data, G_1) is also true. This conforms to our intuition; as we noted earlier, it doesn't matter whether or not Alice co-operates with Bobby. However, in G_2, even though *can•know*(Robin, data, G_2) is true, *can•steal*(Robin, data, G_2) is false (specifically, in theorem 4, taking $\mathbf{p}'$=Robin, $\mathbf{s}$=Alice, and $\mathbf{q}$=data, condition (iv) is false, and in theorem 9, part (ii), taking $\mathbf{p}'=\mathbf{p}$=Robin, $\mathbf{q}'$=Alice, and $\mathbf{q}$=data, condition (c) fails.) So the predicate *can•snoop* captures the distinction between Alice's assisting in Robin's seeing the information, and Robin's seeing the information whether or not Alice co-operates.

6. Conspiracy in a Single-Path Graph

Given that we can determine whether knowing, the sharing of information, is possible in a take-grant graph, how many vertices must co-operate in the sharing? The answer to this question will give us the answer to a more interesting one involving snooping, namely how many actors are necessary to steal information. Before we tackle these questions in all their generality, let us restrict our attention to a specific type of graph.

Let G be a graph with vertices $\mathbf{p}$, $\mathbf{q}$, with *can•know*($\mathbf{p}$, $\mathbf{q}$, G) true, and containing only those vertices and edges needed to witness this predicate. Thus, G is composed of the path along which information is to be progagated. Let the set of vertices

$$V = \{\ \mathbf{x}_i \mid \mathbf{p} = \mathbf{x}_0, \mathbf{p}' = \mathbf{x}_1, \ldots, \mathbf{x}_n = \mathbf{q}', \mathbf{x}_{n+1} = \mathbf{q}\ \}$$

and by theorem 6, each edge $\{\ x_{i-1}, x_i\ \} \in V$ is either an edge with associated word in $B \cup C$, an rw-terminal span from $\mathbf{q}'$ to $\mathbf{q}$, or an rw-initial span from $\mathbf{p}'$ to $\mathbf{p}$.

We shall now capture the notion of the "reach" of a vertex:

Definition: An *access set* $A(\mathbf{y})$ is defined as

$$A(\mathbf{y}) = \{\mathbf{y}\} \cup \{\mathbf{x} \mid \mathbf{y} \text{ rw-initially or rw-terminally spans to } \mathbf{x}\} \cup \\ \{\mathbf{x} \mid \mathbf{y} \text{ initially or terminally spans to a subject } \mathbf{x}\}.$$

$\mathbf{y}$ is called the *focus* of the access set.

That is, $A(y)$ is the maximal set of vertices from which $\mathbf{y}$ can obtain information, or to which $\mathbf{y}$ can pass on information by itself.

Definition: A subject $\mathbf{v}$ is an *information sink* if any one of the following conditions holds:

(i) $\mathbf{v} = \mathbf{x}_0$, the only word associated with the edge $\mathbf{x}_0\mathbf{x}_1$ is $\overleftarrow{t}$, $\overleftarrow{g}$, or $\overrightarrow{r}$, and there are no other edges incident to $\mathbf{v}$;

(ii) $\mathbf{v} = \mathbf{x}_i$, there are exactly two edges incident to $\mathbf{v}$, and the word associated with the edge $\mathbf{x}_{i-1}\mathbf{x}_i\mathbf{x}_{i+1}$ is in the set

$$\{ \overrightarrow{t}\overleftarrow{r}, \overrightarrow{g}\overleftarrow{r}, \overrightarrow{r}\overleftarrow{t}, \overrightarrow{r}\overleftarrow{g}, \overrightarrow{t}\overleftarrow{w}, \overrightarrow{g}\overleftarrow{w}, \overrightarrow{w}\overleftarrow{t}, \overrightarrow{w}\overleftarrow{g}, \overrightarrow{t}^{*}\overleftarrow{t}, \overrightarrow{g}\overleftarrow{g} \};$$

(iii) $\mathbf{v} = \mathbf{x}_n$, the only word associated with the edge $\mathbf{x}_n\mathbf{x}_{n+1}$ is $\overleftarrow{t}$ or $\overleftarrow{g}$, and there are no other edges incident to $\mathbf{v}$.

The idea here is that information can be passed into an information sink without the sink taking any action, but the only way information can be sent out is if the sink is active in a rule application. Notice the sink need *not* apply the rule; if it does not, it must then be a subject in a *de facto* rule, because unless the subjects shown in those rules act, information cannot flow along the implicit edge. This is a subtlety not evident when dealing with conspiracies in graphs using only *de jure* rulesets.

Definition: An *access set cover for G with foci* $\mathbf{y}_1, \ldots, \mathbf{y}_n$ is a family of sets $A(\mathbf{y}_1), \ldots, A(\mathbf{y}_u)$ where for all i, $\{ \mathbf{x}_{i-1}, \mathbf{x}_i \} \subseteq A(\mathbf{y}_j)$ for some $j \le u$. If the cover minimizes u over all possible access set covers, it is said to be a *minimal* cover.

Notice that the number of actors needed to implement *can•know* generates a cover for G. In fact,

Lemma 10: A minimal set of actors $\mathbf{y}_1, \ldots, \mathbf{y}_u$ which implement *can•know*($\mathbf{p}$, $\mathbf{q}$ G) generate an access set cover for G.

Proof: Let $\rho_1, \ldots, \rho_a$ be a minimal set of rules required for a minimal set of actors $\mathbf{y}_1, \ldots, \mathbf{y}_u$ to implement *can•know*($\mathbf{p}$, $\mathbf{q}$ G). Let the access sets $A(\mathbf{y}_1), \ldots, A(\mathbf{y}_u)$ with foci $\mathbf{y}_1, \ldots, \mathbf{y}_n$ be defined on G. Suppose $\mathbf{x} \notin A(\mathbf{y})$ for all i. By definition of "access set", no actor can receive information or rights from, or pass information or rights to, $\mathbf{x}$; hence, $\mathbf{x}$ and its incident edges may be deleted without affecting rules $\rho_1, \ldots, \rho_a$. But this violates condition (iii) of theorem 6, contradicting the minimality of $\rho_1, \ldots, \rho_a$. This proves the claim.

We next make formal our claim that information sinks must act for information or rights to be passed along their incident edges.

Lemma 11: If vertex $\mathbf{x}_i$ is an information sink, and information or rights are passed along the path it lies on in a witness to *can•know*($\mathbf{p}$, $\mathbf{q}$, G), then the vertex must be an actor.

Proof: We demonstrate this for the case of $\mathbf{x}_i$'s incident edges being $\overrightarrow{t}$ and $\overleftarrow{r}$; the proof for the other cases is similar. (The previous section contains some useful lemmas for these proofs.)

Note first that by condition (iii) of theorem 6, $\mathbf{x}_i$ must be a subject (for if not, *can•know*($\mathbf{p}$, $\mathbf{q}$, G) is false because the edges involved in an information sink are neither bridges nor connections.) So, assume $\mathbf{x}_i$ is not an actor, and consider the effects of this on a minimal set of rules $\rho_1, \ldots, \rho_a$ required for a minimal set of actors $\mathbf{y}_1, \ldots, \mathbf{y}_u$ to implement *can•know*($\mathbf{p}$, $\mathbf{q}$, G).

First, no rules ρ_j are of the form "$\mathbf{z}$ takes (α to $\mathbf{y}$) from $\mathbf{x}_i$" since $\mathbf{x}_i$ has no outgoing edges and by the nature of the *de jure* rules can never be assigned any. As the number a of rules is minimal, no rules of the form "$\mathbf{z}$ takes (t to $\mathbf{x}_i$) from $\mathbf{y}$" or "$\mathbf{x}_{i-1}$ grants (t to $\mathbf{x}_i$) to $\mathbf{z}$" are ever executed since the t right assigned could not be used. Hence no *de jure* rules involve $\mathbf{x}_i$.

Now consider the *de facto* rules. Clearly, only information passing through $\mathbf{x}_i$ is relevant; hence, information will never be written into $\mathbf{x}_i$ and not later read (because then that rule could be deleted, contradicting the minimality of a), or read before any information is written into it (which makes sense only if $\mathbf{x}_i = \mathbf{x}_{n+1}$, in which case there are two incident edges to $\mathbf{x}_{n+1}$ and it is not an information sink, contradiction.) The post, pass, and find rules could not be used as $\mathbf{x}_i$ has no incident write edges, and the spy rule could not be used because $\mathbf{x}_i$ would have to act, contradicting assumption. Hence no *de facto* rule involves $\mathbf{x}_i$.

Combining these, if $\mathbf{x}_i$ is not an actor, it and its incident edges can be deleted from G; but this contradicts the minimality of the rule set $\rho_1, \ldots, \rho_a$. Hence the lemma is proved.

With these two lemmas we are able to obtain a lower bound on the number of actors needed to share information:

Theorem 12: Let k be the number of access sets in a minimal cover of G, and let l be the number of information sinks. Then $k+l$ actors are necessary.

Proof: Let $\rho_1, \ldots, \rho_a$ be a minimal set of rules required for a minimal set of actors $\mathbf{y}_1, \ldots, \mathbf{y}_u$ to implement *can•know*(**p**, **q**, G). Let the access sets $A(\mathbf{y}_1), \ldots, A(\mathbf{y}_k)$ with foci $\mathbf{y}_1, \ldots, \mathbf{y}_k$ be defined over G. By lemma 10, $A(\mathbf{y}_1), \ldots, A(\mathbf{y}_k)$ at least cover G. By lemma 11, every vertex $\mathbf{x}_i$ which is an information sink must also be an actor. Then $A(\mathbf{x}_1), \ldots, A(\mathbf{x}_l)$ are all singleton sets, and each of these vertices is a member of its adjacent access sets. Thus these and the other access sets $A(\mathbf{y}_1), \ldots, A(\mathbf{y}_k)$ constitute an access set cover $A(\mathbf{z}_1), \ldots, A(\mathbf{z}_{k+l})$ for G (where $\mathbf{z}_1 = \mathbf{x}_1, \ldots, \mathbf{z}_l = \mathbf{x}_l, \mathbf{z}_{l+1} = \mathbf{y}_1, \ldots, \mathbf{z}_{k+l} = \mathbf{y}_k$.) This proves the theorem.

To derive an upper bound we shall find two lemmas useful:

Lemma 13: Let $A(\mathbf{y}_1), \ldots, A(\mathbf{y}_k)$ be a minimal access set cover for G_0 ordered by increasing indices of **x**. If *can•know*($\mathbf{y}_{i+1}$, **q**, G_0) is true, then for some index m, there exists a graph G_m such that *can•know*($\mathbf{y}_i$, **q**, G_0) is true and all rules in $G_0 \vdash^* G_m$ are initiated by $\mathbf{y}_i$, $\mathbf{y}_{i+1}$, and perhaps $\mathbf{z} = A(\mathbf{y}_i) \cap A(\mathbf{y}_{i+1})$.

Proof: First, recall that we are assuming throughout this section that *can•know*(**p**, **q**, G_0) is true. Consider the spans to **z** from $\mathbf{y}_i$ and $\mathbf{y}_{i+1}$. (See the previous section for a detailed analysis of the vertices which must act for information to be passed along these spans.) In all cases, the vertices acting in the rule applications are $\mathbf{y}_i$, $\mathbf{y}_{i+1}$, and (occasionally) **z**.

Corollary 14: For adjacent access sets $A(\mathbf{y}_i)$, $A(\mathbf{y}_{i+1}$, information can be transferred from $\mathbf{y}_i$) to $\mathbf{y}_{i+1}$ with no other actors unless there are consecutive edges with their only associated words in the set $\{ \overrightarrow{t}\overleftarrow{t}, \overrightarrow{g}\overleftarrow{g}, \overrightarrow{t}\overleftarrow{w}, \overrightarrow{g}\overleftarrow{w}, \overrightarrow{r}\overleftarrow{t}, \overrightarrow{r}\overleftarrow{g} \}$; in this case additional actions performed by $\mathbf{z} = A(\mathbf{y}_i) \cap A(\mathbf{y}_{i+1})$ are sufficient.

Proof: By inspection of the witnesses to the preceding lemma.

We can now use these two results to obtain an upper bound on the number of vertices which must act to share information:

Theorem 15: $k+l$ actors suffice to generate an (implicit or explicit) read edge from **p** to **q** in G.

Proof: Clearly $\mathbf{p} \in A(\mathbf{y}_1)$ and $\mathbf{q} \in A(\mathbf{y}_k)$. Consider first $\mathbf{y}_k$ and **q**. Five cases arise:

- $\mathbf{y}_k = \mathbf{q}$. As $\mathbf{y}_k$ can trivially pass on to **q** any information it receives, *can•know*($\mathbf{y}_k$, **q**, G) is true.

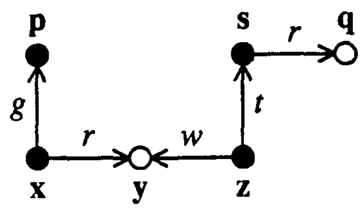
- $\mathbf{y}_k$ terminally spans to **q**. By condition (iii) of theorem 6, this means **q** is a subject, so apply lemma 7 to get the desired result. Note that **q** is an information sink in this case.
- $\mathbf{y}_k$ initially spans to **q**. By condition (iii) of theorem 6, this means **q** is a subject, so apply lemma 8 to get the desired result. Note again that **q** is an information sink in this case.
- $\mathbf{y}_k$ rw-terminally spans to **q**. Apply the take rule repeatedly to get an explicit read edge; this gives the desired result.
- $\mathbf{y}_k$ rw-initially spans to **q**. By conditions (ii) and (iii) of theorem 6, *can•know*($\mathbf{y}_k$, **q**, G) is false (if **q** is a subject, the word associated with the $\mathbf{y}_k$-to-**q** edge is not in $B \cup C$, and if **q** is an object, $\mathbf{y}_k$ does not rw-terminally span to it.) This contradicts the hypothesis, so $\mathbf{y}_k$ cannot rw-initially span to **q**.

In all cases where *can•know*($\mathbf{y}_k$, **q**, G) is true, the only actors are the focus of $A(\mathbf{y}_k)$ and, possibly, the vertex **q**; in addition, **q** only acts if it is an information sink. Applying corollary 14 inductively, we have that whenever *can•know*($\mathbf{y}_k$, **q**, G) is true, the only actors are the foci of the relevant access sets and the information sinks. So, we now consider how the information is transferred from $\mathbf{y}_1$ and **p**. Again, 5 cases arise.

- $\mathbf{y}_1 = \mathbf{p}$. We are done.
- $\mathbf{y}_1$ initially spans to **p**. By condition (iii) of theorem 6, this means **p** is a subject, so apply lemma 7 to get the desired result. Note that **p** is an information sink in this case.
- $\mathbf{y}_1$ terminally spans to **p**. By condition (iii) of theorem 6, this means **p** is a subject, so apply lemma 8 to get the desired result. Note again that **p** is an information sink in this case.
- $\mathbf{y}_1$ rw-initially spans to **p**. Apply the take rule repeatedly to get an explicit write edge; then $\mathbf{y}_1$ applies the post rule to obtain the desired result.
- $\mathbf{y}_1$ rw-terminally spans to **p**. By conditions (i) and (iii) of theorem 6, *can•know*($\mathbf{y}_1$, **q**, G) is false (if **p** is a subject, the word associated with the $\mathbf{y}_1$-to-**p** edge is not in $B \cup C$, and if **p** is an object, $\mathbf{y}_1$ does not rw-initially span to it.) This contradicts the hypothesis, so $\mathbf{y}_1$ cannot rw-terminally span to **q**.

Again, notice that the only actors are the foci of the access sets and (where present) the information sinks. This proves the claim. □

At this point let us take stock of what we have done by working a simple example. Consider the following protection graph:



Taking $\mathbf{u}_1 = \mathbf{p}$, $\mathbf{u}_2 = \mathbf{x}$, $\mathbf{u}_3 = \mathbf{z}$, and $\mathbf{u}_4 = \mathbf{s}$, we see *can•know*(**p**, **q**, G) is true. (Incidentally, so is *can•snoop*(**p**, **q**, G); take $\mathbf{p}' = \mathbf{p}$ and $\mathbf{q}' = \mathbf{z}$ in condition (ii) or theorem 9.)

This graph is a single-path graph of the variety we have been discussing, since information flows from **q** to **p** along the (sole) path between them. This is demonstrated by the following witness to *can•know*(**p**, **q**, *G*):

(1) **z** takes (*r* to **q**) from s;

(2) **x** grants (*r* to **y**) to **p**;

(3) **p** and **z** use the post rule to add an implicit edge labelled *r* from **p** to **z**;

(4) **p** and **z** use the spy rule to add an implicit edge labelled *r* from **p** to **q**.

Given this *G*, we may take $x_0=p$, $x_1=x$, $x_2=y$, $x_3=z$, $x_4=s$, and $x_5=q$. The only information sink here is **p** (by (i) of the definition.) The access sets of the four objects are:

$A(p)=\{ p \}$,
$A(x)=\{ p,x,y \}$,
$A(z)=\{ y,z,s,q \}$,
$A(s)=\{ s,q \}$

It is clear that these four access sets form a cover for *G*; it is equally clear that the middle two sets form a minimal access set cover for *G*. Applying theorem 12, $k=2$ and $l=1$, so a minimum of 3 actors are necessary for information to flow from **p** to **q**; similarly by theorem 15, 3 actors are sufficient. This agrees exactly with the witness presented above, which was in fact composed of a minimal number of rules and actors.

7. Conspiracy in a General Graph

In the previous section, we restricted our attention to graphs in which *can•know* was true, and the only edges in the graph were those along which either rights or information was transferred. We shall now ease the latter restriction, and consider any valid take-grant protection graph in which the predicate*can•know* involving the relevant vertices is true. Our goal is to derive a bound on the number of actors needed to witness the relevant *can•know*.

In order to do this, we need to develop an analogue to the protection graph called an *acting graph*. (This is the analogue of the conspiracy graph in[7] with allowances made for the *de facto* rules.) Essentially, this graph connects all actors with other subjects to which they can pass, or from which they can receive, information and rights.

Consider first a vertex **v** in an access set $A(x)$ with focus **x**. Basically, there are five reasons **v** may be in $A(x)$:

- $v=x$;
- **x** initially spans to **v**;
- **x** rw-initially spans to **v**;
- **x** terminally spans to **v**; or
- **x** rw-terminally spans to **v**.

Define the set $\Delta(x,x')$ to be all vertices in $A(x)\cap A(x')$ *except* those **z** which are information sinks and the only reason **z** is in both $A(x)$ and $A(x')$ is because the words associated with the edges **x**, **z** and **x′**, **z** are those that make **z** an information sink. Δ therefore excludes vertices which must act for information to pass along the path through that vertex, thereby preventing two foci from being directly connected when their only connecting paths contain an information sink.

Given a protection graph G with subject vertices $\mathbf{x}_1, \ldots, \mathbf{x}_n$, we need to generate an acting graph G' with vertices $\mathbf{y}_1, \ldots, \mathbf{y}_n$. Each $\mathbf{y}_i$ has associated with it the access set $A(\mathbf{x}_i)$, and there is an undirected edge between $\mathbf{y}_i$ and $\mathbf{y}_j$ if $\Delta(\mathbf{x}_i, \mathbf{x}_j) \neq \varnothing$. We also define two special sets: let

$$\mathbf{y}_\mathbf{p} = \{ \mathbf{x}_i \mid \mathbf{x}_i = \mathbf{p} \text{ or } \mathbf{x}_i \text{ rw-initially spans to } \mathbf{p} \}$$

and

$$\mathbf{y}_\mathbf{q} = \{ \mathbf{x}_i \mid \mathbf{x}_i = \mathbf{q} \text{ or } \mathbf{x}_i \text{ rw-terminally spans to } \mathbf{q} \}$$

Since we intend to use the acting graph to derive a bound, we must first show it accurately preserves the notion of sharing information.

Theorem 16: *can•know*(**p**, **q**, G) is true if and only if some $\mathbf{y}_u \in \mathbf{y}_\mathbf{p}$ is connected to some $\mathbf{y}_v \in \mathbf{y}_\mathbf{q}$.

Proof: ($\Leftarrow$) Assume $\mathbf{y}_u$ is connected to $\mathbf{y}_v$ along the (undirected) path $\mathbf{y}_u = \mathbf{y}'_1, \ldots, \mathbf{y}'_k = \mathbf{y}_v$. By construction, $\mathbf{y}_{i+1}$ can pass information to $\mathbf{y}_i$, so by induction $\mathbf{y}_u$ can receive information from $\mathbf{y}_v$. Also, as $\mathbf{y}_v \in \mathbf{y}_\mathbf{q}$, $\mathbf{y}_v$ can obtain information from **q**, and as $\mathbf{y}_u \in \mathbf{y}_\mathbf{p}$, $\mathbf{y}_u$ can pass information to **p**. This means *can•know*(**p**, **q**, G) is true.

($\Rightarrow$) We must consider two cases involving any vertex **z** in the definition of Δ above.

First, we restrict **z** to being an object of $A(\mathbf{x}_i) \cap A(\mathbf{x}_j)$. Note that the subjects in G correspond to vertices in G', and the edges between the vertices in G' correspond to words with components in $B \cup C$ in G. So, applying theorem 6, as *can•know*(**p**, **q**, G) is true, some $\mathbf{y}_u \in \mathbf{y}_\mathbf{p}$ is connected to some $\mathbf{y}_v \in \mathbf{y}_\mathbf{q}$.

Next, assume **z** is a subject of $A(\mathbf{x}_i) \cap A(\mathbf{x}_j)$. Let **z** be associated with $\mathbf{y}_\mathbf{z}$. As is a focus (since it is an information sink, and therefore the focus of a singleton access set), it clearly has reason to be in $A(\mathbf{z})$; so $\{\mathbf{z}\} \subseteq \Delta(\mathbf{x}_i, \mathbf{z})$ and $\{\mathbf{z}\} \subseteq \Delta(\mathbf{x}_j, \mathbf{z})$. Hence, by construction of G', there are edges between $\mathbf{y}_i$ and **z**, and **z** and $\mathbf{y}_j$, so there is still a path between $\mathbf{y}_i$ and $\mathbf{y}_j$ (going through **z**). Hence $\mathbf{y}_i$ and $\mathbf{y}_j$ are still connected. This proves the theorem.

We may now state and prove the desired result:

Theorem 17: Let a be the number of vertices on the shortest path from an element $\mathbf{y}_u \in \mathbf{y}_\mathbf{p}$ to an element $\mathbf{y}_v \in \mathbf{y}_\mathbf{q}$ in an acting graph G'. Then to produce a witness to *can•know*(**p**, **q**, G), a actors are both necessary and sufficient.

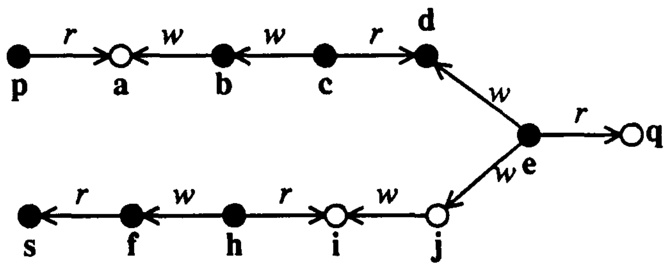
Proof:

Necessity: Let $\mathbf{y}_u = \mathbf{z}_1, \ldots, \mathbf{z}_w = \mathbf{y}_v$ be vertices along a shortest path from $\mathbf{y}_u$ to $\mathbf{y}_v$. If there exist only rwtg-connected paths in G from $\mathbf{z}_i$ to $\mathbf{z}_{i+1}$ $(1 \leq i < w)$, the $\mathbf{z}_i$ are foci of an access set cover for the path. By construction of G' there are no information sinks and if $\mathbf{y}_u$ is not associated with **p** then the subject associated with $\mathbf{y}_u$ rw-initially or initially spans to **p**. So, $\mathbf{y}_u$ need not act. A similar argument holds for $\mathbf{y}_v$ and **q**. By theorem 12, w actors are necessary.

Now suppose there is an (induced) path in G' that is not in G. Even though redundent rule applications may occur, clearly duplicated vertices along a span affect this theorem only if they reduce the number of required actors. We show this is not possible by contradiction. Suppose that actors $\mathbf{z}_1, \ldots, \mathbf{z}_{i-1}, \mathbf{z}_{i+1}, \ldots, \mathbf{z}_w$ could produce a witness. Then there is some vertex $v \in A(\mathbf{z}_{i-1}) \cap A(\mathbf{z}_{i+1})$. By choice of the $\mathbf{z}_i$ vertices being on the shortest path, there is no edge between $\mathbf{z}_{i-1}$ and $\mathbf{z}_{i+1}$, so $\mathbf{v} \neq \mathbf{z}_{i-1}$, $\mathbf{v} \neq \mathbf{z}_{i+1}$, and $\mathbf{v} \notin \Delta(\mathbf{z}_{i-1}, \mathbf{z}_{i+1})$. Hence, if $\mathbf{v}$ is an object, there is no word in $B \cup C$ between $\mathbf{z}_{i-1}$ and $\mathbf{z}_{i+1}$, so *can•know* is false by theorem 6, whence $\mathbf{z}_1, \ldots, \mathbf{z}_{i-1}, \mathbf{z}_{i+1}, \ldots, \mathbf{z}_w$ cannot produce a witness. Also, if $\mathbf{v}$ is a subject, it must be an information sink, in which case it must also be an actor. In either case, the vertices $\mathbf{z}_1, \ldots, \mathbf{z}_{i-1}, \mathbf{z}_{i+1}, \ldots, \mathbf{z}_w$ cannot produce a witness without another vertex being added.

Sufficiency: First, as $\mathbf{p}$ and $\mathbf{q}$ are distinct and all the $\mathbf{y}_i$ on the shortest path are distinct, All spans between these vertices allow the rules described in a previous section to be applied provided the foci of the access sets differ from their common elements. By inspecting the rules, whenever a focus and a common element do coincide the rule whose application is prevented either provides a right already possessed, a right used in the subsequent rule application to acquire a right already possessed, or provides an implicit edge where one already exists. In these cases the rule application is unnecessary. Noting this, we need only induct on the spans corresponding to the edge of the shortest path using lemma 13 to obtain the result. □

Let us apply these results to a simple protection graph:



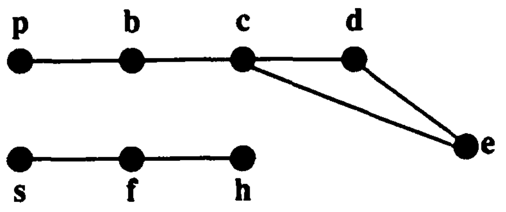
Clearly, $\mathbf{p}$ and $\mathbf{s}$ are the only information sinks, and the access sets of the subjects are:

$$\begin{array}{ll} A(\mathbf{p})=\{\mathbf{p},\mathbf{a}\} & A(\mathbf{e})=\{\mathbf{d},\mathbf{e},\mathbf{q},\mathbf{j}\} \\ A(\mathbf{b})=\{\mathbf{a},\mathbf{b}\} & A(\mathbf{h})=\{\mathbf{f},\mathbf{h},\mathbf{i}\} \\ A(\mathbf{c})=\{\mathbf{b},\mathbf{c},\mathbf{d}\} & A(\mathbf{f})=\{\mathbf{f}\} \\ A(\mathbf{d})=\{\mathbf{d}\} & A(\mathbf{s})=\{\mathbf{s},\mathbf{f}\} \end{array}$$

We can construct $\Delta(\mathbf{x},\mathbf{y})$ from these for each pair of subjects $\mathbf{x}$ and $\mathbf{y}$; the non-null ones are:

$$\begin{array}{ll} \Delta(\mathbf{p},\mathbf{b})=\{\mathbf{a}\} & \Delta(\mathbf{s},\mathbf{f})=\{\mathbf{j}\} \\ \Delta(\mathbf{b},\mathbf{c})=\{\mathbf{b}\} & \Delta(\mathbf{f},\mathbf{h})=\{\mathbf{h}\} \\ \Delta(\mathbf{c},\mathbf{d})=\{\mathbf{d}\} & \Delta(\mathbf{c},\mathbf{e})=\{\mathbf{d}\} \\ \multicolumn{2}{c}{\Delta(\mathbf{d},\mathbf{e})=\{\mathbf{d}\}} \end{array}$$

this means that the acting graph G' corresponding to the above G is



Now, by theorem 6, *can•know*(**p**, **q**, G) is true (take $n=5$, $\mathbf{x}=\mathbf{u}_1=\mathbf{p}$, $\mathbf{u}_2=\mathbf{b}$, $\mathbf{u}_3=\mathbf{c}$, $\mathbf{u}_4=\mathbf{d}$, and $\mathbf{u}_5=\mathbf{e}$. Also, in G', $\mathbf{e} \in y_\mathbf{q}$ and $\mathbf{p} \in y_\mathbf{p}$, so some element of $y_\mathbf{p}$ is connected to some element of $y_\mathbf{q}$. This illustrates theorem 16. A witness to *can•know*(**p**, **q**, G) is:

(1) **e** and **c** use the post rule to add an implicit read edge from **e** to **c**;

(2) **e** and **c** use the spy rule to add an implicit read edge from **c** to **q**;

(3) **c** uses the pass rule to add an implicit read edge from **b** to **q**;

(4) **b** and **p** use the post rule to add an implicit read edge from **p** to **b**;

(5) **p** and **b** use the spy rule to add an implicit read edge from **p** to **q**.

Four vertices act in this witness, and indeed the shortest path from **e** to **p** in G' contains four vertices, just as theorem 17 states.

Consider now **s** and **q**. According to theorem 16, as they are not connected, *can•know*(**s**, **q**, G) should be false; and indeed as there is no rwtg-path from **h** to **e** with associated word in $B \cup C$, condition (iii) of theorem 6 fails, so *can•know*(**s**, **q**, G) is false, just as we expected.

8. Applications of the Theory

Before we discuss security breaches, we must describe security in terms of the Take Grant model and its extensions. We should say first a few words about the notion of a security policy.

Breaches of security are defined in terms of a set of rules governing the use (and abuse) of a computer system. These rules, called the *security policy*, define what is allowed and what is not allowed. For example, a security policy may require that rights to a file be given only by the owner; in this case, if the owner mailed a copy of a protected file to another user, no breach of security has occurred even if the recipient had no right to see the file. The test of a system's security is how well it implements the policy.

In this section we examine possible security policies and in particular the relationship of an enforcement mechanism to a statement of those policies.

8.1. Security in Terms of the Predicates

The first aspect of breaches of security is the illicit transfer of authority. Initially, the protection graph has a specific state. Sharing of rights should be permitted whenever authorized by the initial state; however, theft of rights should be barred. This way, if a process does not want to share its rights with another, the second cannot take those rights. In other words, a right may only be obtained with its owner's consent.

This leads to the condition:

$$\neg \textit{can•steal}(\alpha, \mathbf{x}, \mathbf{y}, G_0) \quad \text{for all } \alpha \subseteq R \text{ and all distinct } \mathbf{x,y} \in V$$

Dealing with information flow is a bit more complex, as a result of the *de facto* rules we use. These rules require the active cooperation of all subjects involved, and with one exception (the post rule) each requires two subjects at least one of which has access to the information being transferred.

In this case, the question of what constitutes a ''breach of security'' depends on the intent of the subject with the rights to the information. To take an example, suppose one process is allowed to read another process' memory. There are two processes, *A* and *B*. *B* has access to a file to which *A* has no rights. If *B* reads the confidential file, and *A* reads the memory of *B* at the same time (thereby seeing the contents of the confidential file), has there been a breach of security?

If *B*'s intent in reading the file was to make it available to *A*, then there has been no breach of security. All *B* has done is (indirectly) copied the confidential file into another file and changed the protections so *A* could read it – certainly allowed as *B* can read the confidential file. But if *B*'s intent in reading the file was simply to see what it contained without passing any of its contents on to *A*, a breach of security has certainly occurred, since *A* has read a file which it had no authority to read. But note that *A* has not violated any restrictions imposed upon it by the system!

Whether or not this is a security breach depends on the wording of the security policy. Either case may be expressed in terms of the predicates described above. We may use the restriction

$$\neg \textit{can•snoop}(\mathbf{x}, \mathbf{y}, G_0) \quad \text{for all distinct } \mathbf{x,y} \in V$$

to indicate that in light of the security policy of the system, *A* breached security, or

$$\textit{can•know}(\mathbf{x}, \mathbf{y}, G_0) \Rightarrow \textit{can•share}(r, \mathbf{x}, \mathbf{y}, G_0) \quad \text{for all distinct } \mathbf{x,y} \in V$$

to indicate that no security breach occurred.

8.2. Reference Monitors

The concept of a *reference monitor* was first described in[1] as a subject or process meeting three requirements:

1 the process must be *isolated* (that is, tamper-proof);

2 the process must be *complete* (that is, always invoked when the resource it controls is accessed);

3 the process must be *verifiable* (that is, small enough to be subject to analyses and tests the completeness of which can be ensured).

In this section, we shall examine reference monitors from the point of view of the Take–Grant model and its extensions, in order to demonstrate that the theory done in the preceding two chapters may be applied to very general practical aspects of computer security. Restating these three conditions in terms of our model is quite straightforward. Let the reference monitor be called **m** and let the resource it protects be called **r**. The issue of isolation simply means that no-one can write over the monitor; this may be stated as $\neg \textit{can•share}(\{w\}, \mathbf{x}, \mathbf{m}, G_0)$ for all $\mathbf{x} \subseteq V$. The issue of completeness may be restated as requiring that **m** always be invoked to give a subject some rights α over $\mathbf{r}$, which becomes the requirement that for all vertices $\mathbf{x} \in V$, only $\mathbf{m}$ have an edge labelled α to $\mathbf{r}$, and for all vertices $\mathbf{x} \in V$ such that $\mathbf{x} \neq \mathbf{m}$, $\neg$*can•steal*(α, $\mathbf{x}$, $\mathbf{r}$, G_0). The issue of verifiability is a bit more tricky, since it consists of two parts. The first, an implementation-level task of verifying that the monitor is indeed implemented correctly, is beyond the scope of this paper; however, the second, which is a verification that no information will leak when the monitor is implemented correctly, is simply the security property described above.

9. Conclusion

This paper has explored several aspects of information transfer in the Take-Grant protection model. The notion of information theft was developed, and then a bound on the number of vertices necessary and sufficient to effect an information transfer was developed. Finally, as an application of this theory, the theoretical characteristics of reference monitors were expressed in terms of those predicates.

This has several ramifications. The first is that the Take-Grant model, while primarily a theoretical tool, can be used to model very practical concepts. In[3] , the protection model was used to represent hierarchies and derive results parallelling the work done earlier by Bell and LaPadula; now, the model has been used to capture the key notions of reference monitors. In short, the model can no longer be held to be a purely theoretical tool. Clearly, though, much more work needs to be done before its practical aspects will be very usable; this is one area for future research.

A second, related area is to incorporate a notion of "group" into the model. Changes of protection state in most computers do not affect a single process (subject) or resource (object); they effect several. However, within the Take-Grant Protection Model, each rule affects only one subject and one object (the source and target of the added implicit or explicit edge.) How the rules might be modified to take these situations into account is another open area.

This leads us to the following question: when the rules are changed to these "group rules," new theorems stating necessary and sufficient conditions for the predicates *can•share*, *can•steal*, *can•know*, and *can•snoop* to be true will have to be derived. It would be most useful if one could derive "meta-theorems" instead, so that given a set of rules, one could use the theorem to state necessary and sufficient conditions for each of those four predicates to be true. This is yet another area for research.

Acknowledgements: This work owes much to Larry Snyder, who first interested me in the Take-Grant Protection Model and whose work on it has been the basis for many of the results presented here. Thanks also to Dorothy Denning, who suggested a line of research that caused me to consider the particular problem leading to the results presented here.

References

1. J. Anderson, "Computer Security Technology Planning Study", ESD-TR-73-51, Air Force Electronic Systems Division, Hanscom Air Force Base, MA, 1972.
2. M. Bishop and L. Snyder, "The Transfer of Information and Authority in a Protection System", *Proceedings of the Seventh Symposium on Operating System*

Principles, , 45-54 (December 1979).

3. M. Bishop, ''Hierarchical Take-Grant Protection Systems'', *Proceedings of the Eighth Symposium on Operating System Principles*, , 107-123 (December 1981).
4. M. Harrison, W. Ruzzo and J. Ullman, ''Protection in Operating Systems'', *Communications of the ACM, 19*, (August 1976) 461-471.
5. A. Jones, R. Lipton and L. Snyder, ''A Linear Time Algorithm for Deciding Security'', *Proceedings of the Seventeenth Annual Symposium on Foundations of Computer Science*, (1976).
6. R. Lipton and L. Snyder, ''A Linear Time Algorithm for Deciding Subject Security'', *Journal of the ACM*, *24*, 3 (July 1977) 455-464.
7. L. Snyder, ''Theft and Conspiracy in the Take-Grant Protection Model'', *Journal of Computer and System Sciences*, *23*, 3 (December 1981) 333-347.