# THEFT OF INFORMATION IN THE TAKE-GRANT PROTECTION MODEL

Ames
[illegible]

Matt Bishop

IN-32

293386

Technical Report PCS-TR88-137 (revised)

DATE
[illegible]

# Theft of Information in the Take-Grant Protection Model

*Matt Bishop*

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

*ABSTRACT*

Questions of information flow are in many ways more important than questions of access control, because the goal of many security policies is to thwart the unauthorized release of information, not merely the illicit obtaining of access rights to that information. The Take-Grant Protection Model is an excellent theoretical tool for examining such issues because conditions necessary and sufficient for information to flow between two objects, and for rights to objects to be obtained or stolen, are known. In this paper we extend these results by examining the question of information flow from an object the owner of which is unwilling to release that information. Necessary and sufficient conditions for such "theft of information" to occur are derived, and bounds on the number of subjects that must take action for the theft to occur are presented. To emphasize the usefulness of these results, the security policies of complete isolation, transfer of rights with the cooperation of an owner, and transfer of information (but not rights) with the cooperation of the owner are presented; the last is used to model a simple reference monitor guarding a resource.

Categories and Subject Descriptors: C.1.3 [**Processor Architectures**]: Other Architectural Types – *capability architectures*; C.2.0 [**Computer-Communication Networks**]: General – *security and protection*; D.2.0 [**Software Engineering**]: General – *protection mechanisms*; D.4.6 [**Operating Systems**]: Security and Protection – *access controls*; *information flow controls*; *security kernels*; H.1.0 [**Models and Principles**]: General; H.2.0 [**Database Management**]: General – *security, integrity, and protection*; K.6.m [**Management of Computing and Information Systems**]: Miscellaneous – *security*

General Terms: Design, Management, Security, Theory, Verification

Additional Key Words and Phrases: conspiracy, information flow, isolation, sharing, reference monitor, safety problem, security policy, take-grant protection model, theft

Portions of this work were supported by grant NAG2-480 from the National Aeronautics and Space Administration to Dartmouth College, and stem from the author's Ph.D. dissertation. A preliminary version of this paper was presented at the First Computer Security Foundations Workshop in Franconia, NH (1988).

## 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, or information, can be transferred. The term "secure" applies to a concrete system; it requires not only that the abstract model of the system be safe, but also that the concrete system correctly implement the abstract model. Because of the complexity of demonstrating correctness of implementation, analyses of safety are far more common than proofs of security.

Among the earliest models used to analyze the safety of models of computer systems was the *access control matrix* [6][9][12]; by representing commands as a set of tests for the presence of specific rights in specific locations which, if true, caused the execution of a sequence of primitive operations, Harrison, Ruzzo, and Ullman were able to prove that in general, it cannot be determined whether a system is safe [9]. The efforts to understand the limits of this *safety question* – that is, under what conditions can the safety of a system be determined – have spurred the study of several models.

Harrison and Ruzzo used the access control matrix model to study the relationship of the complexity of commands to the safety question. In [9], it had been shown that if each command consisted of exactly one primitive operation, the safety question was decidable; in [8], Harrison and Ruzzo showed that when the system is *monotonic* (that is, no right, no subject, and no object may be deleted or destroyed), the safety question is still undecidable, and restricting the set of tests at the beginning of each command to have as few as two elements (*biconditional*) does not make it decidable. But if that set has only one test, then the safety of monotonic systems is decidable.

The Schematic Protection Model [17][18] has been used to examine a limited class of systems (called *acyclic attenuating systems*), and the safety question for such schemes was shown to be decidable. These systems place restrictions on the creation of subjects and objects; specifically, at no time may a child have more rights than its parent (attenuation), and a subject of type $A$ cannot create a subject of type $B$ if a subject of type $B$ can (either directly or, through its descendents, indirectly) create a subject of type $A$ (acyclicness; this holds for all $A \neq B$). This model allows very general rules to be used to determine when one of three primitive operations (*copy*, *demand*, and *create*) may be performed; in terms of the access control matrix model, it essentially restricts the sequences of primitive operations that make up the body of commands to a set of three without restricting the set of tests in the commands.

The Take-Grant Protection Model [11] differs from the access control matrix model and the schematic protection model by specifying both the sequences of primitive operations making up the body of the commands and the set of tests upon which the execution of those sequences is conditioned. Further, and more surprisingly, it also prevents a subject from obtaining rights over itself; this makes representing it in terms of the other two models rather difficult. (If reflexivity were allowed, Take-Grant would fall into the class of acyclic attenuating systems characterizable using the schematic protection model, showing the safety question to be decidable for that system.) However, irreflexivity does not appear to be fundamental to the model [20]. We shall touch on this again in the conclusion.

General questions of safety aside, questions of access control often require specific answers in specific settings; for example, given some explicit set of operations, can one user access a file belonging to another user? This question has been discussed for many types of systems, for many sets of rules, and in contexts other than formal modelling (for example, see [5][15][16]). The advantage of not using formal models is that the discussion focuses on what access control policies are realistic, both in implementation and in enforcement; the disadvantage is that the effects of the rules implementing those policies depend on a host of factors that are often overlooked or not understood thoroughly.

The Take-Grant Protection Model presents features of both types of models, namely those used to analyze questions of safety and those used to analyze the effects of specific access control policies. It represents systems as graphs to be altered by specific operations and, although the model was developed to test the limits of the results in [9], in this model, safety is not merely 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. Further, the focus of most studies of this model have been on characterizing conditions necessary and sufficient for the transfer of rights, on the number of active entities that must cooperate for such transfers to take place, and on the complexity of testing for those conditions in a representation of a system. For this reason it is in some sense of more "practical" use than other formal systems, in that the safety question is decidable and the study of the complexity of conditions allowing compromise is emphasized.

Early work on the Take-Grant Protection Model [11][14] dealt with the transfer of rights assuming all active agents in the system would cooperate. Snyder extended these characterizations to include conditions under which rights could be stolen [21]; Bishop and Snyder introduced the notion of information flow and formulated necessary and sufficient conditions for information sharing [4]. Applications of the model to various systems have been explored [3][10][19][22].

This paper extends that work and combines it with the notion of theft. 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 briefly compare our results to similar ones for theft of rights. To demonstrate the usefulness of the concepts, we express the security policies of total isolation, owners being allowed to transfer rights, owners being allowed to transfer information but not rights, and then analyze the concept of a reference monitor. Finally, we suggest areas for future research.

## 2. Transfers of Authority

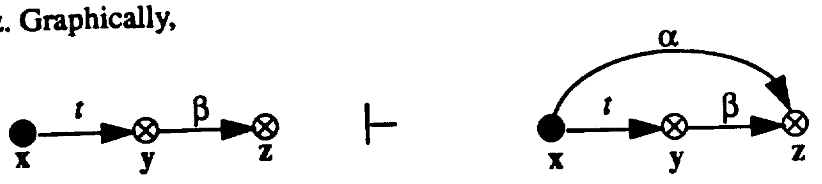
Let a finite, directed graph called a *protection graph* represent a system to be modelled. A protection graph has two distinct kinds of vertices, called *subjects* and *objects*. Subjects are active vertices, and (for example) can represent users; they can pass authority by invoking *graph rewriting 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 a graph rewriting rule on the graph preceding it. The symbol $\vdash^*$ represents several successive 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 $R = \{ r, w, t, g \}$, where $r$, $w$, $t$, and $g$ represent read, write, take, and grant rights, respectively. When written as labels on a graph, or when the set of rights under discussion contains only one element, the set braces are normally omitted.

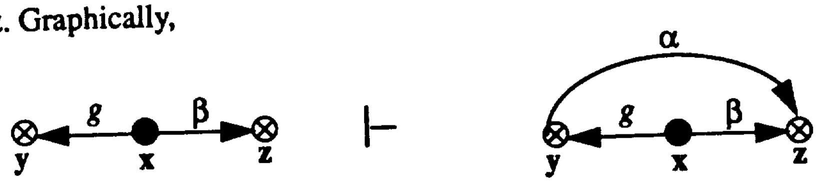
The Take-Grant Protection Model permits vertices with certain rights to transfer rights from one vertex to another. The rules governing the transfer of rights are called *de jure* rules and are as follows:

*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 $\gamma$ with $t \in \gamma$, an edge from y to z labelled $\beta$, 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 $\alpha$. Graphically,



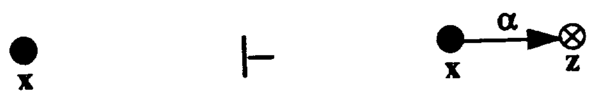
The rule is written "x takes ($\alpha$ to z) from y."

*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 $\gamma$ with $g \in \gamma$, an edge from x to z labelled $\beta$, and $\alpha \subseteq \beta$. Then the *grant* rule defines a new graph $G_1$ by adding an edge to the protection graph from x to z labelled $\alpha$. Graphically,



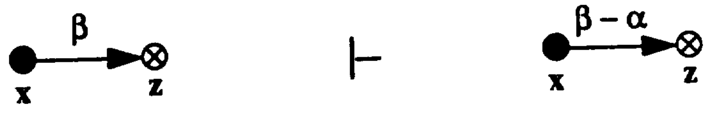
The rule is written "x grants ($\alpha$ to z) to y."

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



The rule is written "x creates ($\alpha$ 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 $\beta$, and let $\alpha \subseteq R$. Then *remove* defines a new graph $G_1$ by deleting the $\alpha$ labels from $\beta$. If $\beta$ becomes empty as a result, the edge itself is deleted. Graphically,

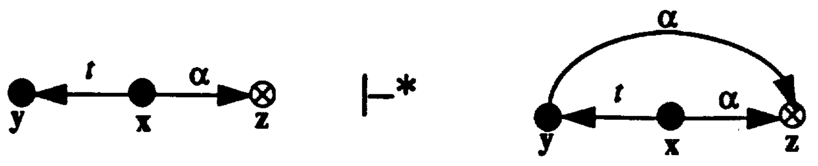


The rule is written "x removes ($\alpha$ to) y."

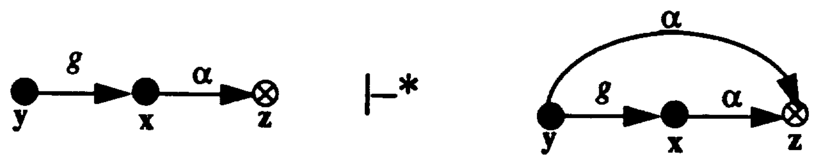
The edges which appear in the above graph are called *explicit* because they represent authority known to the protection system. Further, and more formally, a vertex is an *actor* if the application of a graph rewriting rule requires that vertex to be a subject for that rule to be applied.

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 lemmata [11] demonstrate this:

*Lemma 1.*



*Lemma 2.*



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

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 $v_0, \ldots, v_n$ of distinct vertices such that for all $i$, $0 \le i < n$, $v_i$ 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.

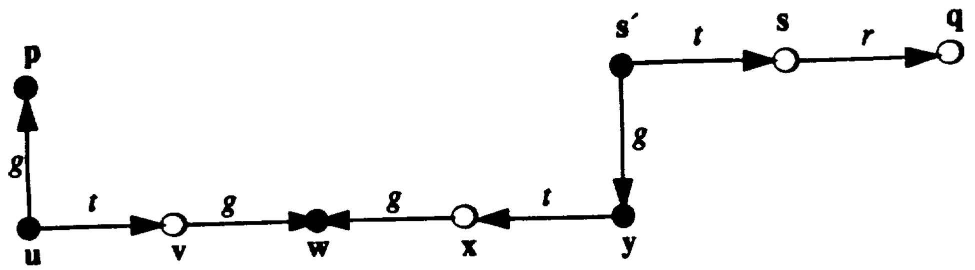
*Definition.* Vertices are *tg-connected* if there is a tg-path between them.

*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 subjects which possess common rights.

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

*Definition.* A vertex $v_0$ *initially spans* to $v_n$ if $v_0$ is a subject and there is a tg-path between $v_0$ and $v_n$ with associated word in $\{\overrightarrow{t}^{*}\overrightarrow{g}\} \cup \{\nu\}$.



| | | | |
|---|---|---|---|
| islands | { p, u }, { w }, { y, s′ } | bridges: | u, v, w and w, x, y |
| initial span: | p (associated word v) | terminal span: | s′, s (associated word v) |

Figure 1. Some examples of Take-Grant Terms. The reader is encouraged to find others.

The span is called "initial" because it is the first part of a tg-path along which rights can be transferred.

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

The span is called "terminal" because it is the last part of a tg-path along which rights can be transferred.

*Definition.* A *bridge* is a tg-path with $v_0$ and $v_n$ 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 last vertex of 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 a path along which a right can be passed, possibly by using Lemma 1 and Lemma 2 as well as the *de jure* rules. Figure 1 illustrates these terms.

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

*Definition.* The predicate *can•share*(α, x, y, $G_0$) is true for a set of rights 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 over 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 [14]:

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

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

(3.2) there exists a subject vertex **x′** such that **x′** = **x** or **x′** initially spans to **x**;

(3.3) there exists a subject vertex **s′** such that **s′** = **s** or **s′** terminally spans to **s**; and

(3.4) there exist islands $I_1, \ldots, I_n$ such that **x′** is in $I_1$, **s′** is in $I_n$, and there is a bridge from $I_j$ to $I_{j+1}$ ($1 \le j < n$).

*Corollary 4*. There is an algorithm for testing *can•share* that operates in linear time in the size of the graph.

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*($\alpha$, x, y, $G_0$) is true if and only if there is no edge labelled $\alpha$ 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 $\alpha$, and if there is an edge labelled $\alpha$ from s to q in $G_0$, then no rule has the form "s grants ($\alpha$ to q) to z" for any $z \in G_j$ ($1 \le j < n$).

Essentially, this says that *can•steal* is true if *can•share* is true, the thief did not have the right initially and no owner of the right in the initial graph gave it away. Necessary and sufficient conditions for this to be true are [21]:

*Theorem 5*. The predicate *can•steal*($\alpha$, x, y, $G_0$) is true if and only if the following hold simultaneously:

(5.1) there is no edge labelled $\alpha$ from x to y in $G_0$;

(5.2) there exists a subject vertex **x′** such that **x′** = **x** or **x′** initially spans to **x**;

(5.3) there is a vertex s with an edge from s to y labelled $\alpha$ in $G_0$;

(5.4) *can•share*($t$, **x′**, s, $G_0$) is true.

*Corollary 6*. There is an algorithm for testing *can•steal* that operates in linear time in the size of the graph.

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

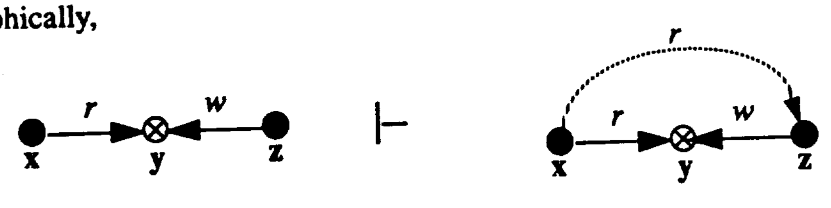
The *de jure* rules control the transfer of authority only; they say nothing about the transfer of information. The two are clearly different; for example, if a user is shown a document containing information which he does not have authority to read, the information has been transferred to the user. 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 *de jure* rules, since the *de jure* rules only affect 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.

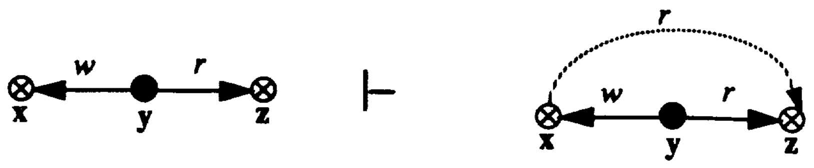
The following set of *de facto* rules was introduced in [4] to model transfers of information:

*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 $\alpha$ with $r \in \alpha$ and an edge from **z** to **y** labelled $\beta$, 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,



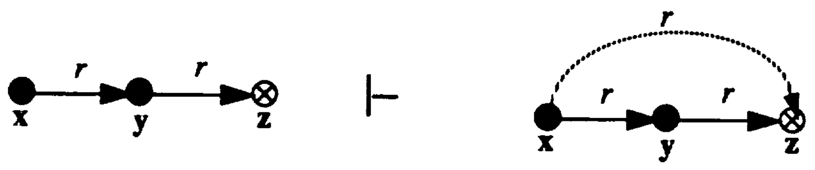
The rule is written "**z** posts to **x** through **y**," and is so named because it is reminiscent of **y** being a mailbox to which **z** posts a letter that **x** reads.

*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 $\alpha$ with $w \in \alpha$ and an edge from **y** to **z** labelled $\beta$, 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,



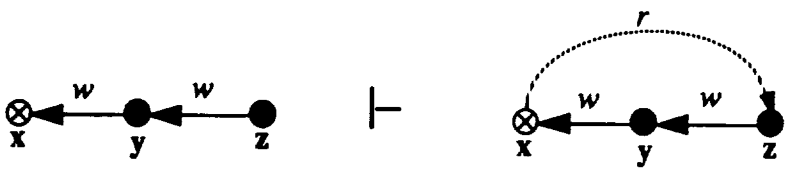
The rule is written "y passes from z to x," and is so named because y acquires the information from z and passes it on to x.

*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 $\alpha$ with $r \in \alpha$ and an edge from y to z labelled $\beta$, 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,



The rule is written "x spies on z using y," and is so named because x is "looking over the shoulder" of y to monitor z.

*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 $\alpha$ with $w \in \alpha$ and an edge from z to y labelled $\beta$, where $w \in \beta$. Then the *spy* rule defines a new graph $G_1$ with an implicit edge from x to z labelled $r$. Graphically,



The rule is written "x finds from z through y," and is named because x is completely passive; z and y give it information, which x then "finds."

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.

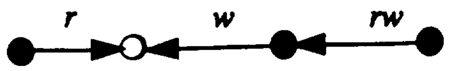


Figure 2. Protection graph with associated words $\vec{r}\overleftarrow{w}\overleftarrow{r}$ and $\vec{r}\overleftarrow{w}\overleftarrow{w}$.

### 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 protection graphs $G_0, \ldots, G_n$ such that $G_i \vdash^* G_{i+1}$ using only *de facto* rules, and in $G_n$ there is an edge from x to y labelled *r* or an edge from y to x labelled *w*, 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 from 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 can effectively 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 $v_0, \ldots, v_n$ of distinct vertices such that for all *i*, $0 \le i < n$, $v_i$ is connected to $v_{i+1}$ by an edge (in either direction) with a label containing *r* or *w*.

With each rw-path, associate one or more words over the alphabet $\{\vec{r}, \overleftarrow{r}, \vec{w}, \overleftarrow{w}\}$ in the obvious way (see figure 2 for an example). If the path has length 0, then the associated work is the null word $\nu$.

*Definition.* An rw-path $v_0, \ldots, v_n$, $n \ge 1$, is an *admissible rw-path* if and only if it has an associated word $a_1 a_2 \ldots a_n$ in the regular language $(\vec{r} \cup \overleftarrow{w})^*$, and if $a_i = \vec{r}$ then $v_{i-1}$ is a subject, and if $a_i = \overleftarrow{w}$ then $v_i$ is a subject.

Note that there cannot be two consecutive objects on an rw-admissible path. Given these definitions, it can be shown [4]:

*Theorem 7.* Let x and y be vertices in a protection graph. 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 p [illegible] te *can•know*(x, y, $G_0$) is true if and only if there exists a sequence of protection graphs $G_0$, [illegible] , $G_n$ such that $G_0 \vdash^* G_n$, and in $G_n$ there is an edge from **x** to **y** labelled *r* or an edge from **y** to **x** labelled *w*, 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 $v_0, \ldots, v_n$ of distinct vertices such that for all $i$, $0 \le i < n$, $v_i$ is connected to $v_{i+1}$ by an edge (in either direction) with a label containing *t*, *g*, *r* or *w*.

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

*Definition*. A vertex $v_0$ *rw-initially spans* to $v_n$ if $v_0$ is a subject and there is an rwtg-path between $v_0$ and $v_n$ with associated word in $\{\vec{t}^* \vec{w}\} \cup \{\nu\}$.

The span is called "rw-initial" because it is the first part of an rwtg-path along which information can be transferred; the "rw" emphasizes this is *information* transfer, not authority transfer.

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

The span is called "rw-terminal" because it is the last part of an rwtg-path along which information can be transferred.

*Definition*. A *bridge* is an rwtg-path with $v_0$ and $v_n$ both subjects and the path's associated word in the regular language $B = \{\vec{t}^*, \overleftarrow{t}^*, \vec{t}^* \vec{g} \overleftarrow{t}^*, \vec{t}^* \overleftarrow{g} \overleftarrow{t}^*\}$ (note this is the same as the definition given earlier).

*Definition*. A *connection* is an rwtg-path with $v_0$ and $v_n$ both subjects and the path's associated word in $C = \{\vec{t}^* \vec{r}, \overleftarrow{w} \overleftarrow{t}^*, \vec{t}^* \vec{r} \overleftarrow{w} \overleftarrow{t}^*\}$.

An rw-initial span is an rwtg-path along which the last vertex of the path can transmit information; an rw-terminal span is an rwtg-path along which the first vertex in the path can acquire information. A connection is a path along which information can be passed.

The next result [4] characterizes the set of graphs for which *can•know* is true:

*Theorem 8*. The predicate *can•know*(**x**, **y**, $G_0$) is true if and only if there exists a sequence of subjects $\mathbf{u}_1, \ldots, \mathbf{u}_n$ in $G_0$ ($n \geq 1$) such that the following conditions hold:

(8.1) $\mathbf{u}_1 = \mathbf{x}$ or $\mathbf{u}_1$ rw-initially spans to **x**;

(8.2) $\mathbf{u}_n = \mathbf{y}$ or $\mathbf{u}_n$ rw-terminally spans to **y**;

(8.3) for all $i$, $1 \leq i < n$, there is an rwtg-path between $\mathbf{u}_i$ and $\mathbf{u}_{i+1}$ with associated word in $B \cup C$.

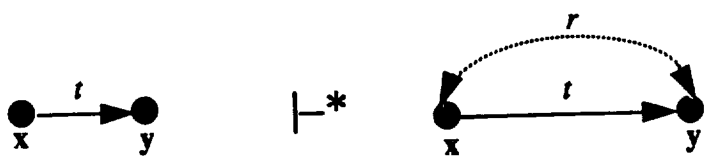
*Corollary 9*. There is an algorithm for testing *can•know* that operates in linear time in the size of the graph.

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.

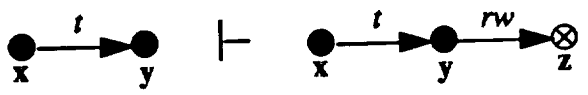
## 4. Some Examples of Combined *de jure* and *de facto* Rule Applications

In this section we present results which are not only 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 if one subject has take or grant rights over another subject, either can (with the cooperation of the other) read information from the other. More formally:

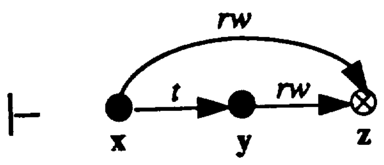
*Lemma 10.*



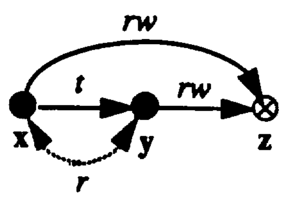
*Proof:* First, **y** creates (*rw* to new vertex) **z**:



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

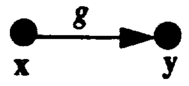
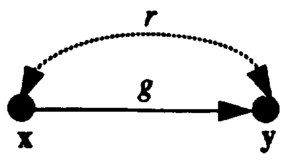


Finally, **x** and **y** use the post rule through **z**:

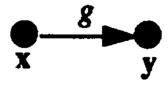
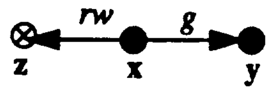
$\vdash$ 

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 goes from **y** to **x**. ■

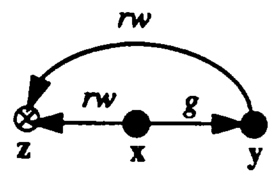
*Lemma 11.*

 $\vdash^*$ 

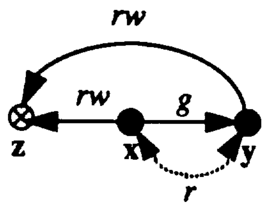
*Proof:* First **x** creates (*rw* to new vertex) **z**:

 $\vdash$ 

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

$\vdash$ 

Finally, **x** and **z** use the post rule through **z**:

$\vdash$ 

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 problem: suppose we have a graph $G_0$ with at least three vertices, namely 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.(Note that these paths may include vertices other than their endpoints. However, we are assuming that except for such internal vertices, there are no vertices other than **x**, **y**, and **z**.) 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, $G_0$) for those combinations of paths for which that predicate is true, and to prove that predicate is false for the others.

Without loss of generality, we can assume that initial, terminal, rw-initial, and rw-terminal spans are all of length 1, 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. So let us consider each combination of paths.

First, if the path from y to z is rw-terminal and z is a subject, then the word associated with the y to z path is not in the set $B \cup C$ and so *can•know*(x, y, $G_0$) is false by condition (8.3) of Theorem 8; similarly, if z is an object, the word associated with the x to y path will not be in the set $B \cup C$ and, again, *can•know*(x, y, $G_0$) is false. If the path from x to z is rw-initial, similar reasoning shows again that *can•know*(x, y, $G_0$) is false whether or not z is a subject.

Let us consider the remaining cases one by one. We shall not draw the pictures as we did for the previous two lemmata.

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

If z is an object, the word associated with the path between x and y is not in the set $B \cup C$ and so the predicate is false. 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) x and y use the post rule to obtain an implicit *r* edge from x to y through z.

This verifies that *can•know*(x, y, $G_0$) is true. Note that all three vertices x, y, and z must act.

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

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

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

This verifies that *can•know*(x, y, $G_0$) is true. 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 the set $B \cup C$

and so the predicate is false. 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** through **v**.

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

This verifies that *can•know*(**x**, **y**, $G_0$) is true. Here, **x**, **y** and **z** need to act.

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

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

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

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

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

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

This verifies that *can•know*(**x**, **y**, $G_0$) is true. Again, only **x** and **y** must act.

**x** *to* **z** *initial*, **y** *to* **z** *initial*

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

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

(2) **y** creates (*rw* to new vertex) **w**.

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

(3) **y** grants (*r* to **w**) to **z**.

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

(5) **x** and **z** use the spy rule to obtain an implicit *r* edge from **x** to **w** through **z**.

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

This verifies that *can•know*(**x**, **y**, $G_0$) is true. As in the first case, all of **x**, **y**, and **z** must act.

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

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

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

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

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

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

This verifies that *can•know*(**x**, **y**, $G_0$) is true. Again, **x**, **y** and **z** need to act.

|  | x-to-z path |  |  |  |
|---|---|---|---|---|
| y-to-z path | initial | terminal | rw-initial | rw-terminal |
| initial | maybe | true | false | maybe |
| terminal | true | maybe | false | maybe |
| rw-initial | maybe | maybe | false | true |
| rw-terminal | false | false | false | false |

Table 1. Summary of values of *can•know*(**x**, **y**, $G_0$) in a 3-vertex graph $G_0$. The cases marked "maybe" are true if and only if **z** is a subject (and an actor).

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

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

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

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

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

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

This verifies that *can•know*(**x**, **y**, $G_0$) is true. All of **x**, **y** and **z** need to act.

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

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

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

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

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

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

This verifies that *can•know*(**x**, **y**, $G_0$) is true. Once more, **x**, **y** and **z** need to act.

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

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

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

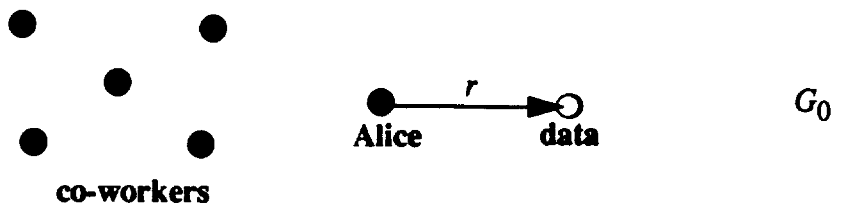
This verifies that *can•know*(**x**, **y**, $G_0$) is true. This time, only **x** and **y** need act.

Table 1 summarizes these results. Let us now consider the question of theft of information.

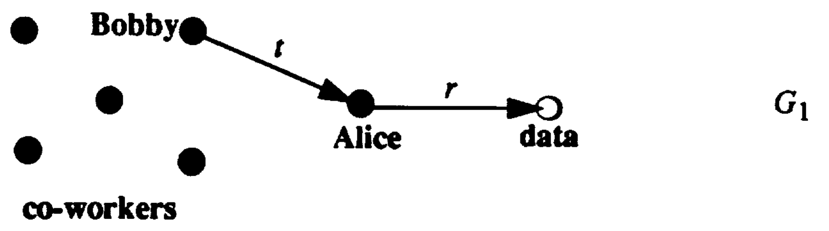
## 5. Snooping, of the Theft of Information

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 a right to read information from another vertex flatly refuses to pass this 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 and superiors. The situation, in Take-Grant terms, is:

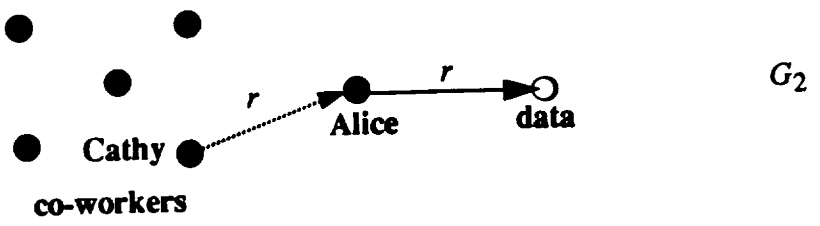


Any documents as sensitive as those which Alice consults must be kept under lock and key. Alice's company has several large vaults, and Alice has a key to one. To prevent employees with the keys from making copies, when an employee leaves the office, her key must be placed in her desk and the desk locked. To handle emergencies (such as Alice misplacing the key which unlocks her desk), a set of master desk keys is kept in the office of the chief of building security. The only person with a key to that office is the chief of building security, Bobby. Now, Bobby is not cleared to read any of Alice's documents, but he can "take" Alice's right to do so by opening her desk with his master key and copying her vault key. He can then pass the information on to someone else. This is an example of Alice's sharing (albeit unknowingly) her right to read the documents



(To make the analogy precise, the key to the vault is the "read right" and the key to the desk is the "take right," because with the latter key one can "take," or copy, the former key.)

Alice takes a sensitive document out of the vault, goes back to her desk, and begins to read the document. Normally, Alice is shielded from Cathy (who sits directly behind Alice in her office) by a partition which for this dayonly has been removed for repairs; hence Cathy can see what Alice is reading by looking over Alice's shoulder. In Take Grant terms, this situation is:



By the spy rule, Cathy can read the information Alice is reading (the Cathy-to-Alice edge, being unauthorized, is implicit); hence, *can•know*(**Cathy**, **data**, $G_2$) is true so long as Cathy can look over Alice's shoulder; if Alice read the document elsewhere, such as in the vault, Cathy would no longer be able to read the document over Alice's shoulder (so, in Take-Grant terms, the spy rule would not be applicable as there is no implicit or explicit Cathy-to-Alice read edge). Notice the difference between this case and the previous one: here, Alice must "cooperate" in some sense of the word by reading the data where Cathy could see it (such as at Alice's desk); were Alice to read it elsewhere, for example in the vault, Cathy could not see it. But so long as data is stored in the vault (a company requirement), Alice need not cooperate in any way with the building security chief in order for him to obtain the data, because by using his master key for the desks, not only does he have a copy of Alice's right to read, but also he can exercise that right whenever he wishes, regardless of Alice's presence. And the key to his office controls access to his copies of those other keys.

The *can•know* predicate fails to capture the distinction between these two situations. To embody the notion of "cooperation," 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*(**Cathy**, **data**, $G_2$) is false, since Alice must cooperate by passing the information to Cathy (in this case, by reading in such a way that Cathy can look over her shoulder). But *can•snoop*(**Bobby**, **data**, $G_1$) is true, since Bobby could see the documents whether or not Alice cooperated, once Bobby had "taken" the key to the vault.

*Definition*. The predicate *can•snoop*(x, y, $G_0$) is true if and only if *can•steal*(*r*, x, y, $G_0$) is true or there exists a sequence of graphs and rule applications $G_0 \vdash_{\rho 1} \ldots \vdash_{\rho n} G_n$ for which all of the following conditions hold:

(a) there is no explicit edge from x to y labelled *r* in $G_0$;

(b) there is an implicit edge from x to y labelled *r* in $G_n$; and

(c) neither y nor any vertex directly connected to y is an actor in a grant rule or a *de facto* rule resulting in an (explicit or implicit) read edge with y 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 to *can•steal*, just as *can•know* is the *de facto* analogue to *can•share*. If x can steal read rights to y, clearly no-one who owns those rights over y can prevent x from obtaining information from y. Similarly, if x has authority to read y, it would strain the meaning of what we are trying to define to say that the predicate *can•snoop*(x, y, $G_0$) is true. If *can•steal*(*r*, x, y, $G_0$) is false, note that any read edge from x to y in $G_n$ must be implicit. And for the purposes of this discussion, we will assume that y will not cooperate (either wittingly or unwittingly) with any snooping; it would be equally reasonable to assume that y would cooperate, in which case what follows must be modified somewhat. (We shall return to this point later.)

*Theorem 12*. For distinct vertices x and y in a protection graph $G_0$ with explicit edges only, the predicate *can•snoop*(x, y, $G_0$) is true if and only if *can•steal*(*r*, x, y, $G_0$) is true or all of the following conditions hold:

(12.1) there is no edge from x to y labelled *r* in $G_0$;

(12.2) there is a subject vertex x′ such that x′ = x or x′ rw-initially spans to x in $G_0$;

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

(12.4) *can•know*(x′, y′, $G_0$) is true.

*Informal argument:* If *can•snoop* is true and *can•steal* false, we have to show all the conditions hold. Condition (12.1) follows from the definition. By part (b) of the definition, the predicate *can•know*(x, y, $G_0$) is true, giving condition (12.2). Also, by condition (8.2) of Theorem 8, we have y′. Combining this with the definition, it becomes clear that although y′ rw-terminally spans to y, y′ ≠ y, and there is no edge labelled *r* from y′ to y in $G_0$. The proof that *can•know*(x, y, $G_0$) is true involves proving that the first rule to add a read edge with target y is a take rule, and working backwards. (We should note that the latter technique is similar to the one used to prove the similar theorem for *can•steal* in [21].) Going from the conditions to *can•snoop* is trivial.

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

Part (a) of the definition gives condition (12.1) of the theorem.

By part (b) of the definition, there is an implicit read edge from **x** to **y** in $G_n$, whence by definition *can•know*(**x**, **y**, $G_0$) is true; so, condition (12.2) of this theorem results from condition (8.1) of Theorem 8.

By condition (8.2) of Theorem 8, there is a subject **y**′ such that **y**′ = **y** or **y**′ rw-terminally spans to **y**. If **y** is an object, we can take **y**′ to be the **y**′ in condition (12.3) of this theorem. If **y** 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 **y**′ ≠ **y**; choose **y**′ in condition (12.3) of the theorem to be this **y**′. Thus, in either case, the **y**′ in condition (12.3) of this theorem is the same as the **y**′ in condition (8.2) of Theorem 8.

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

It remains to be shown that *can•know*(**x**′, **y**′, $G_0$) is true. Let $G_0 \vdash_{\rho 1} \ldots \vdash_{\rho n} G_n$ 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 any vertex **z** to **y** in $G_{i-1}$ not in $G_0$, and there is an (explicit or implicit) read edge from **z** to **y** in $G_i$ that is not in $G_0$. That is, $G_i$ is the first graph in which an edge labelled $r$ with target **y** is added. Consider the rule $\rho_i$ which caused this edge to be added. It cannot be a grant rule since, by part (c) of the definition of *can•snoop*, the owner of $r$ rights to **y** will not grant them to anyone else. The rule $\rho_i$ cannot be a pass or spy rule, since by part (c) of the definition of *can•snoop*, the owner of $r$ rights to **y** will not pass information from **y** to anyone else. Again by part (c), **y** will not write information from itself to anyone else, so $\rho_i$ cannot be a post or find rule. As neither the create nor the remove rule adds edges to existing vertices, $\rho_i$ cannot be either. Hence, $\rho_i$ must be a take rule.

We therefore have $\rho_i$: **z** takes ($r$ to **y**) from **z**′, where **z**′ is a vertex in $G_{i-1}$. Recalling that *can•know*(**x**, **y**, $G_0$) is true, by Theorem 8 we see that *can•know*(**x**′, **y**, $G_0$) is true. Apply Theorem 8 again. By this theorem, there is a subject **y**′ such that **y**′ = **y** or **y**′ rw-terminally spans to **y**. Noting that there is no direct edge labelled $r$ from **y**′ to **y** in $G_0$, we take **y**′ = **z** in Theorem 8 and in this theorem, whence *can•know*(**x**′, **y**′, $G_0$) immediately follows.

(⇐) If *can•steal*($r$, **x**, **y**, $G_0$) holds, *can•snoop*(**x**, **y**, $G_0$) is true.

So, assume the other four conditions hold. Part (a) of the definition is the same as condition (12.1) of the theorem. By Theorem 8, conditions (12.2), (12.3), and (12.4) establish part (b) of the definition of *can•snoop*. And as **y**′ ≠ **y** when **y** is a subject, part (c) of the definition is also true. This completes the proof of Theorem 12. ■

The proof technique used above is very similar to the one used to prove theorem 6, the main difference being the consideration of *de facto* rule applications. This emphasizes the similarity of the two predicates.

As an example, let us return to the office described at the beginning of this section. Consider $G_1$. By Theorem 5, *can•steal*($r$, **Bobby**, **data**, $G_1$) is true, so *can•snoop*(**Bobby**, **data**, $G_1$) is also true by the above theorem. This conforms to our intuition; as noted earlier, it doesn't matter whether or not Alice cooperates with Bobby. However, in $G_2$, even though *can•know*(**Cathy**, **y**, $G_0$) is true, *can•steal*($r$, **Cathy**, **data**, $G_0$) is false (specifically, in Theorem 5, taking **x**′ = **Cathy**, **s** = **Alice**, and **y** = **data**, condition (5.4) fails; taking **x**′ = **x** = **Cathy**, **y**′ = **Alice**, and **y** = **data**, condition (12.3) in Theorem 12 fails). So the predicate *can•snoop* captures the distinction between Alice's assisting in Cathy's seeing the information, and Bobby's seeing the information whether or not Alice cooperates.

*Corollary 13*. There is an algorithm for testing *can•snoop* in time linear in the size of the graph.

## 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 cooperate in the sharing? The answer to this question will give us an answer to a more interesting one involving snooping, namely how many *actors* are necessary to steal information. Our study of these questions will generally follow the analysis of theft of rights in [21], but with modifications for the *de facto* rules throughout.

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 **x**, **y**, with *can•know*(**x**, **y**, $G_0$) 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 propagated. Let the set of vertices

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

and by Theorem 8, each edge $\mathbf{v}_i\mathbf{v}_{i+1}$, where $\{ \mathbf{v}_i, \mathbf{v}_{i+1} \} \subseteq V$, is either an edge with associated word in $B \cup C$, an rw-terminal span from **y**′ to **y**, or an rw-initial span from **x**′ to **x**.

We capture the notion of the "reach" of a vertex by generalizing a term used in [21]:

*Definition*. An *access set* $A$(**y**) is defined as the set containing **y** and all vertices to which **y** initially, terminally, rw-initially, or rw-terminally spans.

That is, $A$(**y**) is the maximal set of vertices from which **y** can obtain information, or to which **y** can pass on information.

*Definition*. A subject **x** is an *information gate* if any one of the following conditions holds:

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

(ii) $\mathbf{x} = \mathbf{v}_i$, there are exactly two edges incident to **x**, and the word associated with the edge $\mathbf{v}_{i-1}\mathbf{v}_i\mathbf{v}_{i+1}$ is in the set $\{\overrightarrow{t}\overleftarrow{r}, \overrightarrow{g}\overleftarrow{r}, \overrightarrow{r}\overleftarrow{t}, \overrightarrow{r}\overleftarrow{g}, \overrightarrow{t}\overleftarrow{w}, \overrightarrow{g}\overleftarrow{w}, \overrightarrow{w}\overleftarrow{t}, \overrightarrow{w}\overleftarrow{g}, \overrightarrow{t}\overleftarrow{t}, \overrightarrow{g}\overleftarrow{g}, \overrightarrow{r}\overrightarrow{r}, \overrightarrow{w}\overrightarrow{w}, \overleftarrow{r}\overleftarrow{r}, \overleftarrow{w}\overleftarrow{w}\}$ ; or

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

The idea is that information can be passed into an information gate, or out of an information gate, without the gate taking any action, but in order for information to be passed through a gate (that is, both in and out), the information gate must be active in a rule application. Note that the information gate 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* rule sets, and although the information gate is analogous to a *sink* in [21], the difference in definition is substantial and reflects the difference between information and rights transfer.

*Definition*. An *access set cover for* G *with foci* $\mathbf{v}_1, \ldots, \mathbf{v}_n$ is a family of sets $A(\mathbf{v}_1), \ldots, A(\mathbf{v}_n)$ where for all $i$, there exists a $j \leq n$ such that $\{ \mathbf{v}_{i-1}, \mathbf{v}_i \} \subseteq A(\mathbf{v}_j)$. If the cover minimizes $n$ over all possible access set covers, it is said to be a *minimal* cover.

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

*Lemma 14*. A minimal set of actors $v_1, \ldots, v_n$ in a sequence of rule applications producing a witness to *can•know*(**x**, **y**, $G$) generates an access set cover for $G$.

*Proof:* Let $\rho_1, \ldots, \rho_m$ be a minimal set of rules required for a minimal set of actors $v_1, \ldots, v_n$ to produce a witness to *can•know*(x, y, $G$). Let the access sets $A(v_1), \ldots, A(v_n)$ with foci $v_1, \ldots, v_n$ be defined on $G$. Suppose $x \notin A(v_i)$ for all $i$. By definition of access set, no actor can receive information or rights from, or pass information or rights to, x; hence x and its incident edges may be deleted without affecting rules $\rho_1, \ldots, \rho_m$. But this violates condition (8.3) of Theorem 8, contradicting the minimality of $\rho_1, \ldots, \rho_m$. This proves the claim. ■

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

*Lemma 15*. If vertex $v_i$ is an information gate, and in a witness to *can•know*(x, y, $G$) information or rights are passed along the path it lies on, then the vertex must be an actor.

*Proof:* We demonstrate this for the case of $v_i$'s incident edges being $\overset{\rightharpoondown}{t}$ and $\overset{\leftharpoonup}{r}$; the proof for the other cases is similar. (Section 4 contains some useful witnesses, and proof of inability to supply other witnesses, for these proofs.)

First, by condition (8.3) of Theorem 8, $v_i$ must be a subject, for if not, *can•know*(x, y, $G$) is false because the paths through that information gate are neither bridges nor connections. So, assume $v_i$ is not an actor, and consider the effects of this on a minimal set of rules $\rho_1, \ldots, \rho_m$ required for a minimal set of actors to produce a witness showing that *can•know*(x, y, $G$) holds.

No rule is of the form "z takes ($\alpha$ to y) from $v_i$" for any z in $G$, since $v_i$ has no outgoing edges and by the nature of the *de jure* rules can never be assigned any. As the number $m$ of rules applied is minimal, no rules of the form "z takes ($t$ to $v_i$) from y" or "$v_{i-1}$ grants ($t$ to $v_i$) to z" for any vertex z in $G$ are ever executed since the $t$ right so assigned could not be used. Hence no *de jure* rule involves $v_i$.

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

Combining these, if $v_i$ is not an actor, it and its incident edges can be deleted from $G$; but this contradicts the minimality of $m$. This proves the lemma. ■

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

*Theorem 16*. Let $k$ be the number of access sets in a minimal cover of $G$, and let $l$ be the number of information gates. Then $k+l$ actors are necessary to produce a witness to *can•know*.

*Proof:* Let $\rho_1, \ldots, \rho_m$ be a minimal set of rules required for a minimal set of actors $v_1, \ldots, v_n$ to produce a witness to *can•know*. Let the access sets $A(v_1), \ldots, A(v_n)$ with foci $v_1, \ldots, v_n$ be defined on $G$. By Lemma 14, $A(v_1), \ldots, A(v_n)$ at least cover $G$. Without loss of generality, take the vertices $v_1, \ldots, v_l$ to be the information gates. By Lemma 15, every one of these must be an actor. Then each of $A(v_1), \ldots, A(v_l)$ is a singleton set, and its focus is a member of its adjacent access sets. Thus the other access sets $A(v_{l+1}), \ldots, A(v_{l+k})$ (where $k + l = n$) constitute an access set cover for $G$, and their foci must also be actors. This proves the theorem. ■

To derive an upper bound we shall find two more results useful:

*Lemma 17*. Let $A(v_1), \ldots, A(v_n)$ be a minimal set access cover for $G_0$ ordered by increasing indices of v. If *can•know*($v_{i+1}$, y, $G$) is true, then for some index $m$ there exists a graph $G_m$ such that *can•know*($v_i$, y, $G$) is true and all rules in the derivation sequence $G_0 \vdash^* G_m$ are initiated by $v_i$, $v_{i+1}$, and perhaps $z = A(v_i) \cap A(v_{i+1})$.

*Proof:* First, recall that we are assuming throughout this section that *can•know*(x, y, $G$) is true. Consider the spans to z from $v_i$ and $v_{i+1}$. By the series of witnesses presented in section 4, in all cases the vertices acting in the rule applications witnessing *can•know*(x, y, $G$) are $v_i$, $v_{i+1}$, and occasionally z. ■

*Corollary 18*. For adjacent access sets $A(v_i)$ and $A(v_{i+1})$, information can be transferred from $v_i$ to $v_{i+1}$ with no other actors unless there are consecutive edges with their only associated word in the set $\{\overleftarrow{r}\overline{r}, \overrightarrow{g}\overline{g}, \overleftarrow{r}\overline{w}, \overrightarrow{g}\overline{w}, \overrightarrow{r}\overline{r}, \overrightarrow{r}\overline{g}\}$ ; in this case additional actions performed by $z = A(v_i) \cap A(v_{i+1})$ 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 19.* $k+l$ actors suffice to generate an (implicit or explicit) read edge from x to y in $G$.

*Proof:* Let $A(\mathrm{v}_1), \ldots, A(\mathrm{v}_k)$ be a minimal set access cover for $G_0$ with $\mathrm{x} \in A(\mathrm{v}_1)$ and $\mathrm{y} \in A(\mathrm{v}_k)$. Consider first y and $\mathrm{v}_k$. Five cases arise:

- $\mathrm{v}_k = \mathrm{y}$. Then *can•know*($\mathrm{v}_k$, y, $G$) is trivially true.
- $\mathrm{v}_k$ terminally spans to y. By condition (8.3) of Theorem 8, this means y is a subject, so apply lemma 7 to get the desired result. Note that y is an information gate in this case.
- $\mathrm{v}_k$ initially spans to y. By condition (8.3) of Theorem 8, this means y is a subject, so apply lemma 8 to get the desired result. Again, y is an information gate in this case.
- $\mathrm{v}_k$ rw-terminally spans to y. Apply the take rule repeatedly to get an explicit edge; this gives the desired result.
- $\mathrm{v}_k$ rw-initially spans to y. By conditions (8.2) and (8.3) of Theorem 8, *can•know*($\mathrm{v}_k$, y, $G$) is false (for if y is a subject, the word associated with the $\mathrm{v}_k$ to y edge is not in $B \cup C$, and if y is an object, $\mathrm{v}_k$ does not rw-terminally span to it). This contradicts the hypothesis, so $\mathrm{v}_k$ cannot rw-initially span to y.

In all cases where *can•know*($\mathrm{v}_k$, y, $G$) is true, the only actors are the focus of $A(\mathrm{v}_k)$ and, possibly, the vertex y; in addition, y acts only if it is an information gate. Applying Corollary 18 inductively, we have that whenever *can•know*($\mathrm{v}_i$, y, $G$) is true for $i = 1, \ldots, k$, the only actors are the foci of the relevant access sets and the information gates. So, we now consider how information is transferred from $\mathrm{v}_1$ to x. Again, five cases arise:

- $\mathrm{v}_1 = \mathrm{x}$. We are done.
- $\mathrm{v}_1$ terminally spans to x. By condition (8.3) of Theorem 8, this means x is a subject, so apply lemma 7 to get the desired result. Note that x is an information gate in this case.
- $\mathrm{v}_1$ initially spans to x. By condition (8.3) of Theorem 8, this means x is a subject, so apply lemma 8 to get the desired result. Again, x is an information gate in this case.
- $\mathrm{v}_1$ rw-terminally spans to x. By conditions (8.2) and (8.3) of Theorem 8, the predicate *can•know*($\mathrm{v}_1$, x, $G$) is false (for if x is a subject, the word associated with the $\mathrm{v}_1$ to x edge is not in $B \cup C$, and if x is an object, $\mathrm{v}_1$ does not rw-initially span to it). This contradicts the hypothesis, so $\mathrm{v}_1$ cannot rw-terminally span to x.

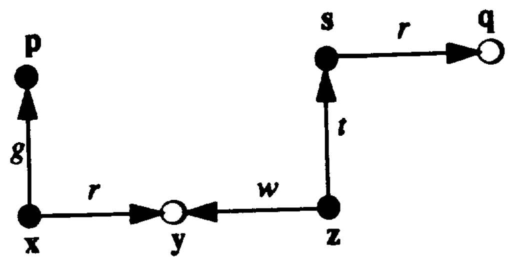


Figure 3. A sample Take-Grant protection graph to demonstrate conspiracy in a single path graph.

- $v_1$ rw-initially spans to **x**. Apply the take rule repeatedly to get an explicit write edge; then $v_1$ applies the post rule to obtain the desired result.

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

As one would expect, these bounds are similar to the ones on the number of conspirators necessary and sufficient to steal rights. The difference lies in the definitions of "access set" and "information gate;" these include at least as many vertices in the *can•snoop* case as in the *can•steal* case. However, given a specific protection graph, computing the numbers $k$ and $l$ is of complexity comparable to the complexity of computing them in the *can•steal* case, since only a small number of new conditions must be tested.

At this point, let us take stock of what we have done by working a simple example. Consider the protection graph $G$ in figure 3. Taking $u_1$ = **p**, $u_2$ = **x**, $u_3$ = **z**, and $u_4$ = **s**, we see that the predicate *can•know*(**p**, **q**, $G$) is true by Theorem 8. (Incidentally, so is *can•snoop*(**p**, **q**, $G$); in the conditions to Theorem 12, take **x** = **x′** = **p**, **y′** = **z**, and **y** = **q**.) The graph is a single path graph of the variety we have been discussing, since information flows from **p** to **q** along the (sole) path between them. The following witness to *can•know*(**p**, **q**, $G$) demonstrates this:

(1) **z** takes ($r$ to **q**) from **s**.

(2) **x** grants ($r$ to **y**) to **p**.

(3) **p** and **z** use the post rule through **y** to add an implicit edge from **p** to **z**.

(4) **p** and **z** use the spy rule to obtain an implicit $r$ edge from **p** to **q** through **z**.

In this graph, the only information gate is **p** (by condition (i) of the definition of information gate). The access sets of the four subjects are:

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

It is clear that these four access sets form a cover for $G$; it is equally clear that the sets $A(\mathbf{x})$ and $A(\mathbf{z})$ form a minimal access set cover for $G$. Applying Theorem 16, $k = 2$ and $l = 1$, so a minimum of 3 actors are necessary for information to flow from **p** to **q**; similarly, by Theorem 19, 3 actors are sufficient. This agrees exactly with the witness presented above, which in fact used a minimal number of actors (**p**, **x**, and **z**).

## 7. Conspiracy in a General Graph

In the previous section, we restricted our attention to graphs in which *can•know* is true, and the only edges in the graph were those along which either rights or information were transferred. We shall now ease the latter restriction, and consider any valid Take-Grant protection graph in which the predicate *can•know* is true. Our goal is to derive a bound on the number of actors needed to produce a witness to *can•know*. We shall take the approach suggested by [21], again with suitable modifications.

In order to do this, we shall develop an analogue to the protection graph called an *acting graph*. Basically, this graph will consist of vertices corresponding to access sets in the original graph with edges corresponding to paths along which the focus of each access set can pass information by acting alone (that is, no other subject will have to act in a rule application to help the first transmit the information). In other words, this graph connects all actors with other subjects to which they can pass, or from which they can receive, information.

Given a protection graph $G$ with subject vertices $\mathbf{v}_1, \ldots, \mathbf{v}_n$, we need to generate an acting graph $G'$ with vertices $\mathbf{u}_1, \ldots, \mathbf{u}_n$. Each $\mathbf{u}_i$ has associated with it the access set $A(\mathbf{v}_i)$. Consider now under what circumstances information can be passed from a member of one access set to a member of another.

Let **y** be a vertex in an access set $A(\mathbf{x})$ with focus **x**. There are five reasons **y** may be in $A(\mathbf{x})$:

- $\mathbf{y} = \mathbf{x}$;
- **x** initially spans to **y**;
- **x** terminally spans to **y**;
- **x** rw-initially spans to **y**; or
- **x** rw-terminally spans to **y**.

Define the set $\Delta(\mathbf{x}, \mathbf{x}')$ to be all vertices in $A(\mathbf{x}) \cap A(\mathbf{x}')$ *except* those vertices **y** which are information gates and the only reason **y** is in both $A(\mathbf{x})$ and $A(\mathbf{x}')$ is that the words associated with the paths **xy** and **x′y** are those that make **y** an information gate. This means the set $\Delta$ includes only those vertices to which the foci can pass information (or from which they can receive information) with the foci being the only actors.

To complete the construction of the acting graph $G'$, we add an undirected edge between $\mathbf{u}_i$ and $\mathbf{u}_j$ when $\Delta(\mathbf{v}_i, \mathbf{v}_j) \neq \varnothing$. (This corresponds to a bridge or connection existing between $\mathbf{v}_i$ and $\mathbf{v}_j$ in $G$.) We also define two special sets; let

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

and

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

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

*Theorem 20. can•know*(x, y, $G$) is true if and only if some vertex $\mathbf{u}_a \in \mathbf{u}_\mathbf{x}$ is connected to some vertex $\mathbf{u}_b \in \mathbf{u}_\mathbf{y}$.

*Proof:* (⟹) Let $\mathbf{v}_i$ be the vertex in $G$ corresponding to the vertex $\mathbf{u}_i$ in $G'$ (for $i = 1, \ldots, n$). We must consider two cases involving any vertex z in the definition of $\Delta$ above.

First, we restrict z to being an object in $A(\mathbf{v}_i) \cap A(\mathbf{v}_j)$. 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 8, as *can•know*(x, y, $G$) is true, some $\mathbf{u}_a \in \mathbf{u}_\mathbf{x}$ is connected to some $\mathbf{u}_b \in \mathbf{u}_\mathbf{y}$.

Next, assume z is a subject in $A(\mathbf{v}_i) \cap A(\mathbf{v}_j)$. Let z be associated with $\mathbf{u}_a$. As z is a focus (since it is an information gate, and therefore the focus of an access set), it clearly has reason to be in $A(\mathbf{z})$; so $\{\mathbf{z}\} \subseteq \Delta(\mathbf{v}_i, \mathbf{z})$ and $\{\mathbf{z}\} \subseteq \Delta(\mathbf{z}, \mathbf{v}_j)$. Hence, by construction of $G'$, there are paths between $\mathbf{u}_i$ and $\mathbf{u}_a$, and $\mathbf{u}_a$ and $\mathbf{u}_j$, so there is still a path between $\mathbf{u}_i$ and $\mathbf{u}_j$ (going through $\mathbf{u}_a$). Hence $\mathbf{u}_i$ and $\mathbf{u}_j$ are connected.

(⟸) Assume $\mathbf{u}_a$ is connected to $\mathbf{u}_b$ along the (undirected) path $\mathbf{u}_a = \mathbf{u}'_1, \ldots, \mathbf{u}'_n = \mathbf{u}_b$. By construction, $\mathbf{u}_{i+1}$ can pass information to $\mathbf{u}_i$, so by induction $\mathbf{u}_a$ can receive information from $\mathbf{u}_b$. Also, as $\mathbf{u}_b \in \mathbf{u}_\mathbf{y}$, $\mathbf{u}_b$ can obtain information from y, and as $\mathbf{u}_a \in \mathbf{u}_\mathbf{x}$, $\mathbf{u}_a$ can pass information to x. This means that *can•know*(x, y, $G$) is true. ■

We may now state and prove the desired result.

*Theorem 21*. Let $n$ be the number of vertices on the shortest path from an element $\mathbf{u}_a \in \mathbf{u}_\mathbf{x}$ to an element $\mathbf{u}_b \in \mathbf{u}_\mathbf{y}$ in an acting graph $G'$. Then $n$ actors are both necessary and sufficient to produce a witness to *can•know*(**x**, **y**, $G$).

*Proof:* (*Necessity*) Let $\mathbf{u}_a = \mathbf{u}'_1, \ldots, \mathbf{u}'_n = \mathbf{u}_b$ be vertices along a shortest path from $\mathbf{u}_a$ to $\mathbf{u}_b$, and let $\mathbf{v}'_i$ be the vertex in $G$ corresponding to the vertex $\mathbf{u}'_i$ in $G'$ (for $i = 1, \ldots, n$). If there exist only rwtg-paths in $G$ from $\mathbf{v}'_i$ to $\mathbf{v}'_{i+1}$ ($1 \le i < n$), the $\mathbf{v}'_i$ are foci of an access set cover for the path. By construction of $G'$ there are no information gates and if $\mathbf{u}_a$ is not associated with **x** then the subject associated with $\mathbf{u}_a$ rw-initially or initially spans to **x**. A similar argument holds for $\mathbf{u}_b$ and **y**. By Theorem 16, $n$ actors are necessary.

Now suppose there is an (induced) path in $G'$ that is not in $G$. Even though redundant rule applications m... occur, clearly duplicated vertices along a span affect the claim only if they reduce the number of required actors. We show this is not possible by contradiction. Suppose that actors $\mathbf{u}'_1, \ldots, \mathbf{u}'_{i-1}, \mathbf{u}'_{i+1}, \ldots, \mathbf{u}'_n$ could produce a witness. Then there is a vertex $\mathbf{z} \in A(\mathbf{v}_{i-1}) \cap A(\mathbf{v}_{i+1})$. As the $\mathbf{u}'_i$ are on the shortest path, there is no path between $\mathbf{u}'_{i-1}$ and $\mathbf{u}'_{i+1}$, so **z** is neither $\mathbf{v}'_{i-1}$ nor $\mathbf{v}'_{i+1}$, and further $\mathbf{z} \notin \Delta(\mathbf{v}_{i-1}, \mathbf{v}_{i+1})$. Hence if **z** is an object, there is no word in $B \cup C$ between the vertices $\mathbf{v}_{i-1}$ and $\mathbf{v}_{i+1}$, so *can•know* is false by Theorem 8, whence $\mathbf{u}'_1, \ldots, \mathbf{u}'_{i-1}, \mathbf{u}'_{i+1}, \ldots, \mathbf{u}'_n$ cannot produce a witness. On the other hand, if **z** is a subject, it must be an information gate, in which case it must be an actor. In either case, the vertices $\mathbf{u}'_1, \ldots, \mathbf{u}'_{i-1}, \mathbf{u}'_{i+1}, \ldots, \mathbf{u}'_n$ cannot produce a witness without another vertex being added.

(*Sufficiency*) First, as **x** and **y** are distinct, and all the $\mathbf{v}'_i$ corresponding to the $\mathbf{u}'_i$ on the shortest path distinct, all spans between these vertices allow the appropriate sequence of rule applications exhibited in section 4 to be applied, provided the foci of the access sets differ from their common elements. By inspecting the sequences, 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 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 17 to obtain the result. ■

In this section and the previous section, we very deliberately defined terms to capture the ability of a single node to pass information, or to prevent it from being passed; we then abstracted the instantiation of these terms to an acting graph. This is a generalization of Snyder's *conspiracy graph*, the derivation of which is similar but does not reflect information flows [21].

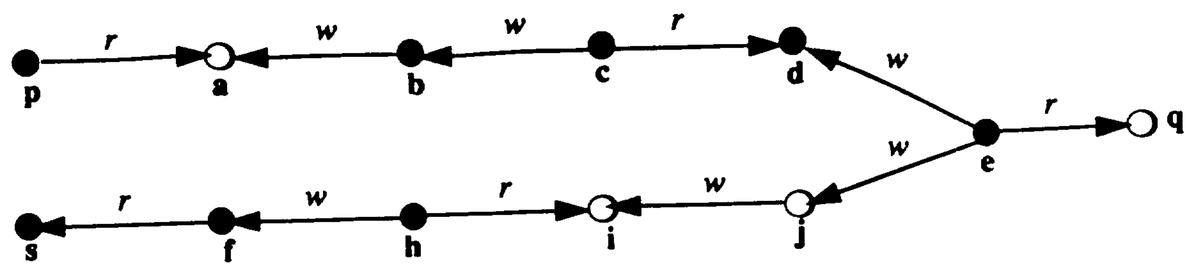


Figure 4a. A sample Take-Grant protection graph to demonstrate conspiracy in a general graph.

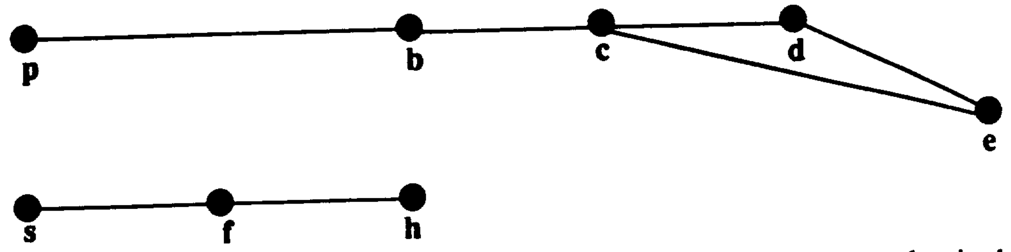


Figure 4b. The associated acting graph. For simplicity vertices are named as in the regular graph.

Let us apply these results to a simple protection graph. In figure 4a, **b** is the only information gate, and the access sets of the objects are:

| | |
|---|---|
| $A(\mathbf{p}) = \{\ \mathbf{p}, \mathbf{a}\ \}$ | $A(\mathbf{e}) = \{\ \mathbf{d}, \mathbf{e}, \mathbf{j}, \mathbf{q}\ \}$ |
| $A(\mathbf{b}) = \{\ \mathbf{a}, \mathbf{b}\ \}$ | $A(\mathbf{h}) = \{\ \mathbf{f}, \mathbf{h}, \mathbf{i}\ \}$ |
| $A(\mathbf{c}) = \{\ \mathbf{b}, \mathbf{c}, \mathbf{d}\ \}$ | $A(\mathbf{f}) = \{\ \mathbf{f}, \mathbf{s}\ \}$ |
| $A(\mathbf{d}) = \{\ \mathbf{d}\ \}$ | $A(\mathbf{s}) = \{\ \mathbf{s}\ \}$ |

From these, we can construct $\Delta(\mathbf{x}, \mathbf{y})$ for each pair of subjects **x** and **y**; the nonempty ones are:

| | |
|---|---|
| $\Delta(\mathbf{p}, \mathbf{b}) = \{\ \mathbf{a}\ \}$ | $\Delta(\mathbf{d}, \mathbf{e}) = \{\ \mathbf{d}\ \}$ |
| $\Delta(\mathbf{b}, \mathbf{c}) = \{\ \mathbf{b}\ \}$ | $\Delta(\mathbf{h}, \mathbf{f}) = \{\ \mathbf{f}\ \}$ |
| $\Delta(\mathbf{c}, \mathbf{d}) = \{\ \mathbf{d}\ \}$ | $\Delta(\mathbf{f}, \mathbf{s}) = \{\ \mathbf{s}\ \}$ |
| $\Delta(\mathbf{c}, \mathbf{e}) = \{\ \mathbf{d}\ \}$ | |

The resulting acting graph is shown in figure 4b. By Theorem 8, *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 $G'$, $\mathbf{e} \in \mathbf{y}_\mathbf{q}$ and $\mathbf{p} \in \mathbf{y}_\mathbf{p}$, so some element of $\mathbf{y}_\mathbf{p}$ is connected to some element of $\mathbf{y}_\mathbf{q}$. This illustrates Theorem 20.

The following sequence of rule applications is a witness to *can•know*(**p**, **q**, $G$):

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

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

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

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

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

Four vertices (**b**, **c**, **e**, and **p**) act in this witness, and indeed the shortest path in $G'$ between **p** and **e** contains four vertices. This illustrates Theorem 21.

Consider now **s** and **q**. According to Theorem 20, as they are not connected in $G'$, *can•know*(**s**, **q**, *G*) should be false. As there is no rwtg-path from **h** to **e** with associated word in $B \cup C$, condition (8.3) of Theorem 8 fails, so *can•know*(**s**, **q**, *G*) is indeed false.

Finally, let us consider just the top part of this graph (from **p** to **q**), which is a single-path graph of the sort discussed in the previous section. The only information gate is **b**, and the access sets with foci **p**, **c**, and **e** provide a complete cover for the subgraph. Hence by Theorem 16 and Theorem 19, four actors are necessary and sufficient to witness *can•know*(**p**, **q**, *G*), and our witness confirms this.

## 8. Comparison with Results for Theft of Rights

The similarity of the definitions of *can•steal* and *can•snoop* lead to the question of the relationship of these *de jure* and *de facto* results with the analogous *de jure* results in [21]; specifically, how different are the definitions, theorems and proofs, and how much more (or less) complex is it to determine bounds on the number of actors needed to steal information as opposed to steal rights?

The fundamental difference in the results presented here is the addition of extra conditions presenting more ways in which conspiracy can occur; for example, the *de jure* analogue to *access set* requires only that the focus initially or terminally span to every vertex in the set whereas here, we add those vertices to which the focus also rw-initially or rw-terminally spans. Most of the definitions in this work follow directly from their analogues; however, the changes add complexity to both the statements of the theorems and to the proofs. For example, the key theorem in [21] (theorem 6 in this paper), which states necessary and sufficient conditions for rights to be stolen, requires checking for only three (simple) conditions; the analogue of that theorem in this paper, theorem 9, requires four (more complex) conditions to hold.

Consider a Take-Grant protection graph *G* in which the predicates *can•steal*(*r*, **x**, **y**, *G*) and *can•snoop*(**x**, **y**, *G*) are true. Let $A^R$(**y**) be the set of nodes containing **y** and those nodes to which **y** initially or terminally spans, and let a *tg-sink* be a vertex with exactly two incident edges, both incoming and both labelled *t* or both labelled *g*. In [21], Snyder shows that a *conspiracy graph* can be constructed in the same way that an acting graph was constructed in section 7, above. Note that $A^R(y) \subseteq A(y)$, and that a tg-sink is also an information gate (by part (ii) of the definition). Hence, the conspiracy graph associated with $G$ will be a (possibly improper) subgraph of the acting graph of $G$. So, in no case will stealing information require more conspirators than stealing rights; and if the acting graph contains a shorter path between the vertices associated with **x** and **y** than does the conspiracy graph, stealing information will require fewer conspirators.

## 9. Applications of the Theory

Before we discuss security breaches, we should say 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 consider three different security policies. The first is that of complete isolation of subjects; this technique is quite effective for solving the problem of covert channels, but in most cases is far too restrictive. The second is that the owner of an object controls the dissemination of rights over that object; this is the policy exhibited by some database systems such as System R. The third is that no subject other than the owner may have any rights to an object, and any information flowing to or from that object may do so only with the cooperation of the owner; this policy is one exhibited by reference monitors.

We consider first the statement of each policy in terms of the four predicates, then describe the set of graphs that satisfy that policy; finally, we construct protocols which preserve membership of the graphs in the set when the protocols are used.

### 9.1. Complete Isolation of Each Process

The policy of *total isolation* of each process prevents breaches of security and solves the confinement problem by preventing *any* transfer of information or rights among subjects [13].

To prevent any information or rights transfer (illicit or otherwise) it suffices to make all four predicates always be false.The set of graphs satisfying this policy contains exactly those graphs $h$ satisfying

$$\neg can\bullet share(\alpha,x,y,h) \wedge \neg can\bullet steal(\alpha,x,y,h) \wedge \neg can\bullet know(x,y,h) \wedge \neg can\bullet snoop(x,y,h)$$

for all pairs of vertices x and y in $h$, and all subsets of rights $\alpha \subseteq R$.

To characterize this requirement in terms of the rules, note that by Theorem 3 and Theorem 8, it suffices to prevent any bridges or connections between any two subjects to enforce this condition; in that case, both conditions (3.4) and (8.3) can never hold. Because the x and y referred to in the theorems may be subjects, it is not possible to prevent conditions (3.2), (3.3), (8.1), and (8.2) *a priori* from occurring; hence the above constraint is also necessary. This may be formalized to prove the following Take-Grant characterization of the set of graphs satisfying the policy:

*Theorem 22*. To enforce complete isolation of subjects, no bridges or connections may exist between any two subjects in the protection graph.

Determining whether or not a protection graph meets this requirement is easy: just look for bridges or connections between subjects. Also, when a new *de jure* rule is applied, we can test for violation of the restriction just by looking at the paths affected by the rule application; in the worst case, this requires checking every edge in the graph. So:

*Corollary 23*. Testing a graph for violation of the restriction may be done in time linear in the number of edges of the graph. Determining whether or not an application of a *de jure* rule violates the restriction may be done in time linear to the number of edges of the graph.

This does not require any changes to the take or grant rules, since in a graph in which creation is disallowed, bridges and connections may not be constructed unless they already exist. However, it does require changing the create rule, since if one subject creates another, the parent may give itself any rights in $R$ to the child. Should one of these rights be take, grant, read, or write, there will be a bridge, connection, or both between the parent and the child. We do so in the manner of Snyder [19] by requiring all subjects to create other subjects using the following *subject creation protocol*:

*subject creation protocol*: A subject x desiring to create a subject y over which x has the set of rights $\alpha$, x must execute the following rules atomically:

x creates ($\alpha$ to new subject y)

x removes ({ $t, g, r, w$ } to) y

By enforcing this protocol, and preventing any node from applying the create rule directly to create a subject, complete isolation can be enforced.

We should note though that in practise such systems would be useless, since no process could communicate with another in any fashion. Even in systems where isolation is desired, it is typically used to prevent specific sets of processes from communicating; other processes may communicate freely. (For example, a system enforcing security levels would allow two processes at the same level to communicate freely, whereas one at a higher level could not transmit information to a process at a lower level.) So, most likely the "complete isolation" would be enforced between (for example) children of different subjects, and a different statement of the security policy would be needed. Hence, we turn from this somewhat uninteresting policy to consider one in which the holder of a right over an object determines to what other subjects it shall pass that right.

## 9.2. Transfer on Possession of a Right

This policy, used for example in System R [7], creates a system in which mere possession of a right enables a subject to propagate that right. Hence the set of graphs meeting this policy is the set of graphs with elements satisfying

$$can\bullet share(\alpha,\mathbf{x},\mathbf{y},h) \wedge \neg can\bullet steal(\alpha,\mathbf{x},\mathbf{y},h) \wedge can\bullet know(\mathbf{x},\mathbf{y},h) \wedge \neg can\bullet snoop(\mathbf{x},\mathbf{y},h)$$

for all protection graphs $h$, all pairs of vertices $\mathbf{x}$ and $\mathbf{y}$ in $h$, and all subsets of rights $\alpha \subseteq R$. The reasoning is that any two subjects must be able to share rights or information, but no set of subjects can steal rights or information without the consent of the owner. To satisfy the first two predicates, the owner must cooperate in any transfer of rights; to satisfy the latter two, the owner must cooperate in any sharing of information. We also note that any pair of subjects can share rights or information by the construction of this condition.

For $can\bullet share(\alpha,\mathbf{x},\mathbf{y},h)$ to be true but $can\bullet steal(\alpha,\mathbf{x},\mathbf{y},h)$ to be false, condition (5.4) must be false since negating any of the other conditions of Theorem 5 negates one of of conditions (3.1), (3.2) or (3.3), making $can\bullet share(\alpha,\mathbf{x},\mathbf{y},h)$ false. For (5.4) to be false, either there must be no bridges between any two subjects (by the definition of "island") or no take edges incident upon any subject. We note that the former renders $can\bullet share(\alpha,\mathbf{x},\mathbf{y},h)$ false. But the latter renders $can\bullet snoop(\mathbf{x},\mathbf{y},h)$ false (as all rw-terminal spans will have length 1, so condition (12.3) fails) without necessarily negating $can\bullet know(\mathbf{x},\mathbf{y},h)$ or $can\bullet share(\alpha,\mathbf{x},\mathbf{y},h)$. Clearly, this characterization is best, and from the above it can be shown:

*Theorem 24*. If no subject has any incident take edges, then the security policy allowing the possessor of a right to transfer that right to another is implemented.

Testing for this condition is simple:

*Corollary 25.* Testing a graph for a violation of the restriction may be done in time linear in the number of edges in the graph. Further, determining whether or not an application of a *de jure* rule violates the restriction may be done in constant time.

The obvious way to prevent creation of take edges is to make the appropriate modification to the subject creation protocol above. However, the lack of take edges also inhibits sharing; to see why, notice that the grant rule does not add any edges to the source of the edge labelled grant. Hence, Lemma 2 cannot hold (in fact, the proof that it is true requires the use of a take rule). Now consider a graph *h* meeting this security policy. As there are no take edges, the only take edges that could be added using the rules would be to new vertices. If those vertices are subjects, any subsequent manipulation of those rights would require an application of Lemma 2, which is false given the replacement of the create rule for subjects by the modified subject creation protocol.

To overcome this problem, we can require that in the initial graph, all subjects have grant rights to one another, and modify the subject creation protocol to be:

*granting subject creation protocol*: A subject x desiring to create a subject y over which x has the set of rights $\alpha$, x must execute the following rules atomically:

x creates ($\alpha$ to new subject y)

x removes ({ $t$ } to) y

x grants ({ $g$ } to all other subjects) to y

x grants ({ $g$ } to y) to all other subjects

If this protocol is used to create subjects, creating a new subject would add an incoming grant edge to the parent, in which case the proof of Lemma 2 is straightforward (see figure 5). Then as Lemma 2 is true, the transfer of any rights from a newly created subject to the parent using take can be emulated by the child granting the parent the rights. If the child is an object, of course, the issue never arises since the only way a right can be added to the child is by the parent granting the right, in which case application of the take rule is redundant and can be eliminated. Hence the security policy will hold.

## 9.3. Owner Propagates Information but Not Rights

This policy creates a system in which only one subject is to have rights over a designated object. Those rights may not be transferred (with or without the consent of the subject). However, the subject may allow information to flow out of the object. An example of this policy would be a network server which responds to queries by giving information about the users active on a com-

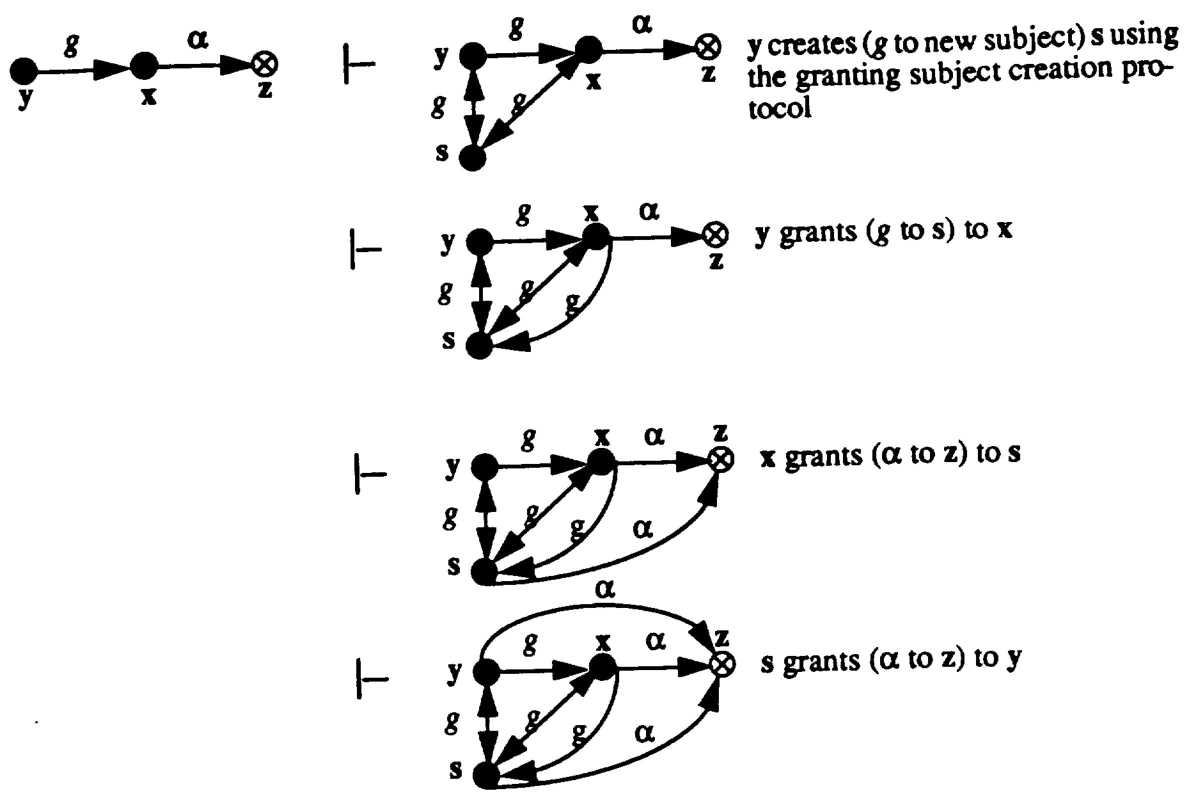

y creates (g to new subject) s using the granting subject creation protocol
y grants (g to s) to x
x grants (α to z) to s
s grants (α to z) to y

Figure 5. Proof of Lemma 2 using the granting subject creation protocol

puter; the server has the right to obtain that information from the system and pass that data to another process, but it cannot give any other process the right to obtain that information directly. The set of graphs meeting this policy is the set of graphs with elements satisfying

$$\neg can\bullet share(\alpha,x,r,h) \wedge \neg can\bullet steal(\alpha,x,r,h) \wedge can\bullet know(x,r,h) \wedge \neg can\bullet snoop(x,r,h)$$

for all protection graphs *h*, any element **r** of the set of distinguished objects (called "resources"), any vertex **x** *h*, and all subsets of rights $\alpha \subseteq R$. The reasoning is that no rights to the resource may be transferred (hence the first two predicates must be false). To satisfy the latter two predicates, the owner must cooperate in any sharing of information.

If *can•share*(α,x,r,*h*) is false, so is *can•steal*(α,x,r,*h*), and without loss of generality we can consider only the former. Given that **s** in Theorem 3 is the unique subject (or monitor) **m** which possesses rights to **r**, (3.1) and (3.3) both hold; as **x** may be a subject, (3.2) holds. Hence (3.4) must be made to fail; the obvious way is to prevent there being bridges between **m** and any other subject. This means that no take or grant rights may be incident on **m** (unless they are between **m** and **r**).

The ability to transfer information is required; however, such transfers require **m** be an actor. So we must prevent the predicate *can•snoop*(**x**,**r**,*h*) from holding. As *can•steal*(α,**x**,**r**,*h*) is false for all $\alpha \subseteq R$, we need only consider the conditions in Theorem 12. As only **m** has *r* rights over **r**, condition (12.1) holds; as **x** may be a subject, (12.2) may hold; but as no take edges may be incident on **m**, condition (12.3) always fails. So *can•snoop*(**x**,**r**,*h*) also fails. But all of the conditions in Theorem 8 hold, so *can•know*(**x**,**r**,*h*) holds, as required.

Finally, we note that as **m** rw-terminally spans to **r**, **r** ∈ *A*(**m**). Further, as no other vertex terminally, initially, rw-terminally, or rw-initially spans to **r**, the only way for information to flow out of **r** to some vertex **x** is through **m**. By the nature of the *de facto* rules, **m** must be an actor, as required. So the above formula accurately captures the security policy, and in Take-Grant terms, this discussion may be formalized to show:

*Theorem 26*. Let a subject **m** possess exclusive rights to a resource **r**. If there are no take or grant edges incident upon **m** (except possibly between **m** and **r** only), then no other vertex can acquire rights over **r**. Further, information may flow out of **r** only with **m**'s being an actor in a *de facto* rule.

Finally, we note that the same problems for complete isolation arise if **m** can create subjects; but if the following creation protocol is always used by **m**, the security policy will be enforced:

*monitoring creation protocol*: A monitor **m** desiring to create a vertex **y** over which **m** has the set of rights α, **m** must execute the following rules atomically:

**m** creates (α to new vertex **y**)
**m** removes ({ *t*, *g* } to) **y**

This preserves the condition that no take or grant edges be incident upon **m**.

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

We now restate these three conditions in terms of the Take-Grant Protection Model. Let the reference monitor be **m** and the resource it protects be **r**. Isolation means that no-one can write over the monitor; this may be stated as $\neg can\bullet share(\{w\},\mathbf{x},\mathbf{m},G_0)$ for all $\mathbf{x} \in G_0$. Completeness means that whenever any $\mathbf{x} \in G_0$ obtains access to the resource, then **m** is an actor in the witness to the access. 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 or rights be given away when the monitor is implemented correctly, is merely the condition in the preceding section.

Using that analysis, we may characterize what protection graphs containing reference monitors look like. Let $G_0$ be a protection graph with a single reference monitor **m** protecting a single resource **r**. Then no edge with a right in $\{ w \}$ may have **m** as its target and no edge with a right in $\{ t, g \}$ may be incident on **m** (with the possible exception of such an edge between **m** and **r**). As demonstrated in the previous section, the latter ensures information or rights may not be stolen from **m**, that **m** may not transfer rights, and that **m** must be an actor in any witness to a transfer of information. The former simply ensures no subject will ever be able to write to **m**, since no rule adds an edge labelled $\alpha$ unless there is already an edge labelled $\alpha$ already incident upon that vertex.

One general observation about modelling a reference monitor should be made. In some sense this is a circular exercise, since a reference monitor would be used to enforce rights indicated by the model; so the proof that the monitor is "secure" requires the assumption that the model is correctly implemented by the thing being modelled. However, this tautology holds for *any* method used to model a system; if there is a mechanism that enforces rights within the system based upon the model, the model cannot be used to examine that mechanism for security. But reference monitors do more than simply control access rights because they also control access; so given a mechanism to enforce those rights, the above applies to reference monitors guarding resources other than the access control graph (such as printers, the CPU, and so forth).

## 10. 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 put on the number of vertices necessary and sufficient to steal information. 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 protection 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 paralleling the work done earlier by Bell and LaPadula [2]; now, the model can be used to capture the key notions of reference monitors and of several security policies. In short, the model can no longer be held to be a purely theoretical tool.

Some aspects of the model are particularly enticing in terms of applicability to real systems. For example, many models allow reflexive rights. The Take-Grant protection model does not. Correctly representing a real system in a theoretical model requires that all parts of the entity being modelled be represented as dictated by their attributes. Thus, a process would not be represented by a single subject; it would be a set of objects, some of which represent those parts of the process instructions and data resident in memory, others of which represent the resources held by the process, and so forth. The subject would represent the execution of the process. As the execution controls all the objects representing the process, it would have various rights over each. So if a process needed to access a part of its memory (say, to alter a variable which is shared with other processes), the subject would not need write permission on itself but rather write permission on the object representing that variable. Similarly, if certain portions of the process' memory were not writable (for example, the instructions making up the program being executed), the subject representing the process execution might have read rights, but not write rights, over that object. Hence the transfer of information does not appear to require reflexivity when a system is appropriately represented in a theoretical model. Similarly, as no subject should be able to confer upon itself rights which it does not already have, reflexivity adds nothing to the modelling of transfer of authority. This intuition indicates the model does not suffer from being irreflexive; indeed, it forces the modeler to break the system being modelled into all component parts. Perhaps this would reveal unrealized assumptions or dependencies.

As an aside, we note that formulating a reflexive version of the Take-Grant Protection Model would alter its nature radically. Even though this reflexive version would be biconditional (and hence in a class in which, in the general case, safety is undecidable [8]), questions of safety would be decidable, as we pointed out in the introduction. In fact, if a subject had *take* rights over any other node, that subject would have all defined rights over that node. Further, if one subject had *grant* rights over another, that subject could give the target of the *grant* rights any authority over itself. While this would most likely not change the statement of the theorems giving necessary and sufficient conditions for sharing and theft, it would certainly change their interpretation (for example, the conditions in the theorems requiring the existence of a vertex s with an edge from s to another vertex y would be trivially true; just take s to be y). The precise effects of reflexivity, as well as its necessity, is an area in which further work is needed.

In the definition of *can•snoop*, the target of the snooping was not to cooperate with the snooping. An alternate definition would be to allow the target to cooperate, but not the owners of the target; in this case, one could treat the target as a Trojan horse designed to "leak" information. Under this assumption, the proof presented for Theorem 12 does not apply (specifically, the rule $\rho_i$ could be a *post* or *find* rule); this is not surprising, since condition (12.2) of that theorem is overly restrictive if q is a subject and allowed to act in a rule application. Another area for future work lies in the precise reformulation of the necessary and sufficient conditions for *can•snoop* to be true if the target of the snooping is allowed to act.

A 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 affect 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 these rules might be modified to take these situations into account is another open area.

This leads 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 "metatheorems" instead, so that given a set of rules, one could use the metatheorems to state necessary and sufficient conditions for each of the predicates instead of having to rederive those results. This is yet another area for research.

***Acknowledgments***: 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 her, and to the anonymous referees, whose diligence greatly improved the quality of this paper.

## References

[1] Anderson, J. Computer security technology planning study. Technical Report ESD-TR-73-51, USAF Electronics Systems Division, Bedford, MA (Oct. 1972).

[2] Bell, D. and LaPadula, L. Secure computer systems: mathematical foundations and model. Technical Report M74-244, The MITRE Corp., Bedford, MA (May 1973).

[3] Bishop, M. Hierarchical take-grant protection systems. *Proc. 8th Symp. on Operating Systems Principles* (Dec. 1981), 107-123

[4] Bishop, M. and Snyder, L. The transfer of information and authority in a protection system. *Proc. 7th Symp. on Operating Systems Principles* (Dec. 1979), 45-54.

[5] Feiertag, R. and Neumann, P. The foundations of a provably secure operating system (PSOS). *Proc. of the National Computer Conference 48* (1979), 329-334.

[6] Graham, G. and Denning, P. Protection: principles and practices. *Proc. AFIPS Spring Joint Computer Conf. 40* (1972), 417-429

[7] Griffiths, P. and Wade, B. An authorization mechanism for a relational database system. *Trans. on Database Systems 1*,3 (Sep. 1976), 242-255.

[8] Harrison, M. and Ruzzo, W. Monotonic protection systems. In *Foundations of Secure Computing*, Academic Press, New York City, NY (1978), 337-366.

[9] Harrison, M., Ruzzo, W., and Ullman, J. Protection in operating systems. *CACM 19*, 8 (Aug. 1976), 461-471

[10] Jones, A. Protection mechanism models: their usefulness. In *Foundations of Secure Computing*, Academic Press, New York City, NY (1978), 237-254.

[11] Jones, A., Lipton, R., and Snyder, L. A linear time algorithm for deciding security. *Proc. 17th Annual Symp. on the Foundations of Computer Science* (Oct. 1976), 33-41.

[12] Lampson, B. Protection. *Fifth Princeton Conf. on Information and Systems Sciences* (Mar. 1971), 437-443.

[13] Lampson, B. A note on the confinement problem," *CACM 16*, 10 (Oct. 1973), 613-615.

[14] Lipton, R. and Snyder, L. A linear time algorithm for deciding subject security. *J. ACM. 24*, 3 (Jul. 1977), 455-464.

[15] McCauley, E. and Drongowski, P. KSOS – the design of a secure operating system. *Proc. of the National Computer Conference 48* (1979), 345-353.

[16] Popek, G., Kampe, M., Kline, C., Stoughton, S., Urban, M., and Walton, E. UCLA secure UNIX. *Proc. of the National Computer Conference 48* (1979), 355-364.

[17] Sandhu, R. Expressive power of the schematic protection model (extended abstract). *Proc. of the Computer Security Foundations Workshop*, Technical Report M88-37, The MITRE Corp., Bedford, MA (Jun. 1988), 188-193.

[18] Sandhu, R. The schematic protection model: its definition and analysis for acyclic attenuating schemes. *J. ACM 35*, 2 (Apr. 1988), 404-432.

[19] Snyder, L. On the synthesis and analysis of protection systems. *Proc. Sixth Symp. on Operating Systems Principles* (Nov. 1977), 141-150.

[20] Snyder, L. Formal models of capability-based protection systems. *IEEE Transactions on Computers C-30*,3 (Mar. 1981), 172-181.

[21] Snyder, L. Theft and conspiracy in the take-grant protection model. *JCSS 23*, 3 (Dec. 1981), 333-347.

[22] Wu, M. Hierarchical protection systems. *Proc. 1981 Symp. on Security and Privacy* (Apr. 1981), 113-123.