Extending The Take-Grant Protection System

Jeremy Frank and Matt Bishop
Department of Computer Science
University of California at Davis
Davis, CA. 95616-8562

December 5, 1996

1. Abstract.

The Take-Grant Model is used to analyze systems in order to determine how information and access rights can be passed from one subject to another. However, the Take-Grant model does not contain enough power to discriminate between two different flows. We propose a method of extending Take-Grant to generate information and rights flows in order of *costs*. Our method involves finding the most permissive Take-Grant graph achievable from the initial graph, and adding the notion of cost to the application of the *de jure* and *de facto* rules. We also discuss the complexity of finding this most permissive graph or *closure* and the complexity of generating witnesses from the closure graph. We discuss the use of this extended model both for finding witnesses and for finding paths of information and rights flow.

2. Introduction.

In 1976, Harrison, Ruzzo, and Ullman proved that the general question of security in an arbitrary system is undecidable [1]. To study under what conditions it was decidable, Jones, Lipton, and Snyder developed a model of protection called the Take-Grant protection Model [2] in which questions of security were not only decidable, but decidable in time linear with the number of objects and rights. The basis of the work was a graphical representation of a system in which the nodes are subjects or objects (active objects are called subjects) and the directed edges indicate the rights that one object has over another.

The Take-Grant Protection Model provides a set of graph rewriting rules that allow one to study the changes induced in the graph by the transfer of authority. The rules are summarized in [5]. The *take* and *grant* rules control the addition of rights; the *create* rule governs the creation of new nodes and rights. The upper diagram in figure 1 shows an example of the *grant* rule, in which **x** gives **y** the *read* right over **z**. A *remove* rule also exists, but it is rarely used in security modelling because once a right is acquired, only its owner can delete it, and such

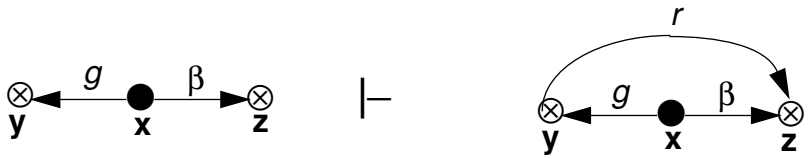

Figure 1. The *grant* rule: **x** *grants* **y** α rights to **z**.

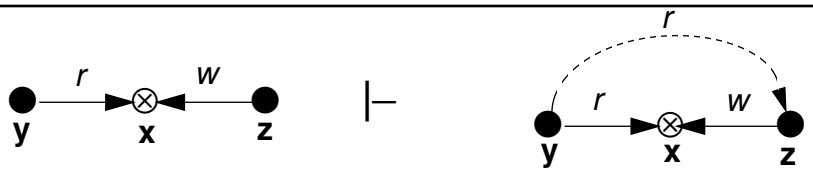

Figure 2. The *post* rule: **z** *posts* to **x** through **y**.

deletions are rare. Sequences of rule applications are called *witnesses*, and models and witnesses in which the remove rule is not used are said to be *monotonic*. Because these rules change the state of the graph, they are called *de jure* rules.

Accompanying these rules are a set of 4 other rules which do not change the state of the graph but capture the paths along which information may flow throughout the graph. Application of these rules does not alter the protection state of the graph, and so they are called *de facto* rules because they model what may happen in fact (as opposed to "by right," which is captured by the *de jure* rules). We may apply these rules to determine if information may flow from one entity to another, with or without the cooperation of other entities. The lower diagram in figure 2 shows an example of the *post* rule, which states that since **z** can *write* **x** and **y** can *read* what is written to **x**, then in fact **y** can *read* **z**.

The issue of *cooperation* is central to the question of theft. Each of the *de jure* and *de facto* rules require one or more subjects to apply the rule. For *de jure* rules, exactly one actor is required; for *de facto* rules, either one or two actors is required, depending on the rule. Snyder tightly bounded the number of subjects (called *conspirators* in this context) that had to participate in a transfer of rights; in [6], Bishop determined how to obtain a bound on the number of conspirators involved in a transfer of information . The obvious remaining problem is to identify those conspirators. In a monotonic system, this is somewhat straightforward: one can use known techniques [3][6] to determine the paths along which rights or information may flow, and then monitor key nodes and edges. In a small graph, this is easy. In a large graph, this may be very difficult. We can change the focus of the problem slightly: given a Take-Grant graph representing a large system, what entities (systems) and/or edges (network links) should be monitored to detect the illicit transfer of information with highest probability? In other words, given that it is infeasible to monitor all paths between two vertices, and given that we cannot monitor the endpoints, over which edges and through

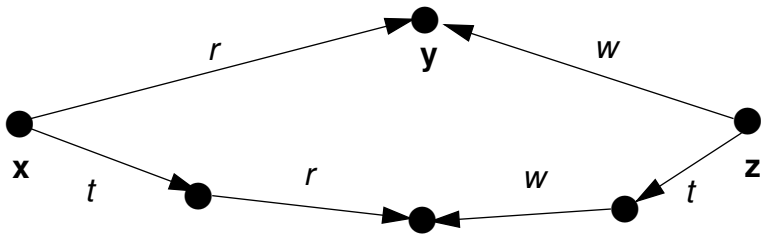

Figure 3. Scenarios where **z** is to send information to **x**; the path through **y** is intuitively the best one, but in various contexts this would be wrong.

which vertices does the rights or information have the greatest probability of flowing? The reformulation also applies to post-mortem analysis of a breach of policy or security. If information or rights illicitly flowed between two subjects, what is the most likely path? Take-Grant is not equipped to handle questions of this type; we can conclude that it is possible for information or rights to flow, but we must turn to intuition to determine which of two possible witnesses is most likely. Further, in many cases (and in most realistic ones), generating the witnesses in some order will focus the analysis on the most likely candidates.

Suppose the higher the number of edges the greater the cost of using the path. In figure 3, our intuition tells us that traffic between **x** and **z** would most likely use the path through **y**. However, if the information being transferred was information identifying members of a group that pirated software, and **y** were the Software Publishers' Association's site, then **x** and **z** would use the longer route to minimize the chance of detection.

We extend the Take-Grant model to include a notion of the *cost* of information or rights flows. With this notion we will then be able to generate witnesses in increasing order of cost. We assume low cost flows with high probability; as a result, a low cost witness for some illicit information or rights flow is a high-probability witness, and therefore should be investigated first or given preference in monitoring. We will see that even in Take-Grant models incorporating applications of the remove rule, we can still generate witnesses by first considering the most permissive Take-Grant graph derivable from the initial graph, called the *closure*. We will give results for the complexity of finding the closure and for finding the witnesses.

The organization of the paper is as follows. In the next section we discuss previous work on the Take-Grant Protection Model. Then we discuss the ordered generation of witnesses for information flow. In this section we will show how to find the closure and discuss the addition of costs to Take-Grant models. Following that, we will show how to generate witnesses for rights flow. In the penultimate section we will present some examples, and the final section concludes and presents some ideas for

future investigations.

3. Previous Work

In 1979, Bishop and Snyder [4] added the notion of information flow to the Take-Grant model, creating a new type of edge called *implicit* and a new set of rules called *de facto* rules. These rules represent how information may flow about the graph, and the implicit edges capture the resulting paths. Note that these rules do not alter the graph; they simply represent the flow of information.

The notion of *theft* of rights was introduced in 1981 [3], and described how one vertex acquires rights over another without cooperation of any owner of those rights. This was generalized to cover the theft of information [5], and was applied to the Take-Grant model to analyze a theft of information over a network [6]. In the model, each site (repository) is represented by a process or subject vertex. The ability to "take" a right over a file requires that the file be represented by a separate object, with parent the process of the site upon which the file resides.

These results were applied to determine the number of conspirators to networks, but did not touch upon which paths information is likely or able to flow over because of complexities in the costing of the transfer of rights and information. In other words, the generation of witnesses serves to identify possible conspirators without regard to likelihood or simplicity.

In this paper, we expand upon this work to identify likely paths along which rights and information can flow, and the cost of specific sets of witnesses to conduct the transfer.

4. Ordered Generation of Witnesses for Information Flow

The goal is to determine which witness(es) minimize cost of the transfer of information or rights. First, we compute the closure of the Take-Grant protection graph. From this, we can generate a subset of interesting witnesses with low cost. We select a witness with the minimum cost. If additional (non-cost) constraints apply, we can select the witness that meets the constraints and has the lowest cost.

First we examine the problem of generating all possible witnesses; then we augment the Take-Grant Protection Model with the notion of cost. Following that, we combine the two ideas.

4.1. Finding the Closure

We define the closure of a Take-Grant model by beginning with the notion of a **witness**:

Definition. Let $G = (V, E)$ be a Take-Grant graph where V is a set of vertices

and E is a set of edges. The set V is partitioned into *subjects* and *objects*, and the elements of E are labelled with members of the set of rights R. The elements t, g, r, and w of R are distinguished symbols for *take*, *grant*, *read*, and *write*, respectively. We write $G \vdash^* G'$. if the application of a *de facto* or *de jure* rule to G results in G'. If a sequence of rule applications to G produces G' then we say $G \vdash^* G'$, and the sequence of applications is called a *witness*.

We can think of the closure of a Take Grant graph similarly to the way we think of finding the closure of directed or undirected graphs: if there is a "path" between two nodes in a graph we add the edge connecting those two nodes.

We can now define the closure.

Definition. Let $G = (V, E)$ be a Take-Grant graph. G^* is the *closure* of G if there is a witness such that $G \vdash^* G^*$ and, for all *de facto* and *de jure* rules, $G^* \vdash^* G^*$.

To find the closure of a Take-Grant graph, we apply each *de facto* and *de jure* rule to the graph until we cannot apply them any more. There are two complicating factors, however: creates and deletes. The application of deletes removes rights from an edge. No rule in Take-Grant is predicated on the absence of rights, hence we should not use delete rules to form closures. Further:

Lemma. Delete rules can be re-ordered to be the last rule applications in any witness.

Proof: Suppose not. Then by re-ordering deletes to the end of a witness it would have to be the case that $G \vdash^* G$ is false. However, this is ridiculous, since moving the deletes until later does not make it impossible to apply any rule. (Re-ordering them to occur earlier, however, is a different matter!) So we have a contradiction. ■

Thus, if deletes are allowed, we can reorder them so they do not affect the generation of witnesses.

Creates are important in a Take-Grant model because of the *irreflexive* nature of the model. Unlike reflexive models of protection, a subject in an irreflexive system has no rights over itself. Thus, the ability to "take" a right over a file requires that the file be represented by a separate object, with parent the process of the site upon which the file resides. So, in the worst case, each subject would need to apply the create rule repeatedly in order to account for situations where reflexivity is mimicked by takes over files created by subjects. However, each subject needs to do only one create rule application, since now the subject has the power to grant itself all rights over this newly created object; no second object would add any more capability to the subject for the purposes of information or rights flow. This is clear from the *de jure* rules,

in which no target of an edge ever obtains rights not possessed by at least one of its sources.

As a final point, we can arbitrarily *take* and *grant* every right possible when these rules are applied, since we can always acquire these rights later.

Lemma. Let $G = (V, E)$ be a Take-Grant graph. The closure of G contains fewer than $2V$ nodes, $4V^2$ edges and $16V^2$ rights.

Proof: In the worst case, each node may perform a create, doubling the number of nodes. Also in the worst case, there could be a directed edge between every pair of nodes in the graph, and each edge can have 4 rights in the set of associated rights. ■

We now consider the complexity of finding the closure of G. We begin with the *de jure* rules. Let F be a list of all *take* and *grant* edges; clearly $O(E)$ operations produces this set. Let the set N be empty. For each edge e in F proceed as follows:

1. Add e's endpoints to N.
2. Check for a possible *de jure* rule application; this requires looking at the current edge and any other adjacent edges. Do this by looping through N, checking each node v in N and the endpoints of the current edge to see if any *de jure* rules involving the three vertices can be applied.
3. If any *de jure* rules can be applied, and the resulting edge and right has not yet been put in F, add the resulting edge and right to F. Note that an edge may already exist between the nodes; the key is the newly added right.
4. Delete the curent edge from F
5. Repeat steps 1, 2, and 3, stopping when F is empty.

Steps 1 and 4 are constant time operations. Step 3 requires keeping a list of edges added to F; checking for redundency is at worst $O(V^2)$ time. Step 2 is more complex; before evaluating it, we show:

Theorem: The above algorithm is a correct implementation of the closure algorithm in that it does not add multiple edges between vertices and all possile edges and rights in the closure are added.

Proof: Step 3 eliminates redundent edges. Now suppose some edge is not added. Its endpoints could not appear in N, because if they did, step 2 would cause the rule application that would add the edge to F. But then no edge incident on either endpoint could appear in F, because if such an edge did appear in F, the endpoints would be added by step 2. Continuing on in this fashion, it becomes clear that the only graph meeting these conditions is one with no edges, in which case the theorem vacuously holds. ■

Theorem: Let $G = (V, E)$ be a Take-Grant graph such that all creates have

been performed. The complexity of finding the closure G^* with respect to *de jure* rules is $O(V^4)$.

Proof: As the algorithm does not add multiple edges between vertices, at most $O(V^2)$ edges will be added. The loop in the above algorithm takes $O(V^2)$ time as each node match could take V time, and there are V nodes, with 4 matches possible for each node. Hence the worst case complexity is $O(V^4)$. ■

Adding *de facto* rules is straightforward once one observes that *de facto* rules add only (implicit) *read* edges. Let E be the initial set of edges. All *de facto* rules require two adjacent edges, so at worst, every 2 edges could be paired to form a new, third edge. The complexity of this is $O(V^2)$ for the set of possible edges and $O(V^2)$ for each possible pairing; hence the complexity of this is $O(V^4)$. Combining this with the result above, this shows:

Theorem: Let $G = (V, E)$ be a Take-Grant graph such that all creates have been performed. The complexity of finding the closure G^* is $O(V^4)$.

Once we have found the closure we can now determine whether or not information or rights have flowed between any two nodes in the graph. The reason is that the closure propagates all information and rights to every subject in the graph that could ever achieve them through any witness. If **a** initially had read permission over **x** and there is a witness that gives **b** read permission over **x**, then **b** will have read permission over **x** in the closure. However, both **a** and **b** having read permission over **x** does not imply **b** obtained its permission from **a**; in fact, it is possible that **b**'s read permission was derived from another subject **c** and not **a** at all.

4.2. Adding Costs to Take-Grant

Simply determining the closure does not give us the tools we need to generate witnesses in a particular order. We must augment Take-Grant with a notion of "cost" in order to determine the cost of any witness. Our method is to associate a cost with each rule application, and to assume that the cost of a witness is a function of the costs of each rule application. For example, associate a cost with each edge; then the cost of a rule application (and resulting edge) might be the sum of the costs of the edges involved.

Since our goal is to generate witnesses in increasing order of cost, we must first assign a semantics to the cost. There are several meanings we could assign costs, but the two most appealing are *probability* and *difficulty*. If we view cost as probability, then we would like to generate the most probable witness; *i.e.* that witness which reflects the most probable flow of information. There are some difficulties with assigning costs based on probabilities. If each rule application had a probability

associated with it, then the probability of a witness would then be the product of the probabilities of each rule application, assuming each rule application was independent of the others. Our task would then be to generate the witness with the highest probability. Maximizing the cost function would require determining the longest path in the Take-Grant protection graph, as in this case the longest path is that path with the highest probability. The problem of finding the longest path in a graph is known to be *NP-Hard*.

The other alternative is to have our cost reflect the difficulty of the rule application as seen by the agent applying the rules. Using this meaning of cost, we would like to generate the lowest cost witness, and the cost of a witness is the sum of the costs of each rule application.

We now foreshadow a moment: if each rule application results in a "cost" of reading in a Take-Grant graph, it is possible to cast our problem as one of finding the shortest path of information flow from the source to the destination, which has a well-known solution. While the notion of difficulty is not as appealing as that of probability, the prospect of a well-behaved solution is enough reason to pursue it for now.

Having settled on the notion of cost, we now must examine the rules to determine what costs to add to the graph with each rule application. There are a few preliminaries. First, any *read* edge in the graph must be assigned a cost, since they may play a part in the generation of new edge costs and in which witness is chosen.

Second is the question of what to apply the cost to: the edges or the rule applications. If the former, each added (implicit or explicit) edge is assigned a cost based upon the rights associated with the edge. The cost of a set of witnesses is simply the total of the costs of the final path along which the transfer occurs. In the second case, applying a rule increments the cost of the witness. The cost of each rule application may be constant, or it may vary based on the specific rule applied, the number of actors required, the co-operation required, and so forth. However, for the most part, the relative magnitude of costs must be motivated by the domain which the Take-Grant system is applied to.

Third is the issue of *de facto* rules and implicit *read* edges. Their cost in the transfer of rights is irrelevant, since only *de jure* rules and explicit edges are involved. In the transfer of information, both implicit and explicit edges are used, but the *de jure* rules can only manipulate explicit edges. Without loss of generality, we can reorder the rule applications so that all *de jure* rule applications precede all *de facto* rule applications. Hence, unless the metric is temporal (in some way), one can minimize over the set of (reordered) witnesses to obtain the best set of witnesses, and then derive the path.

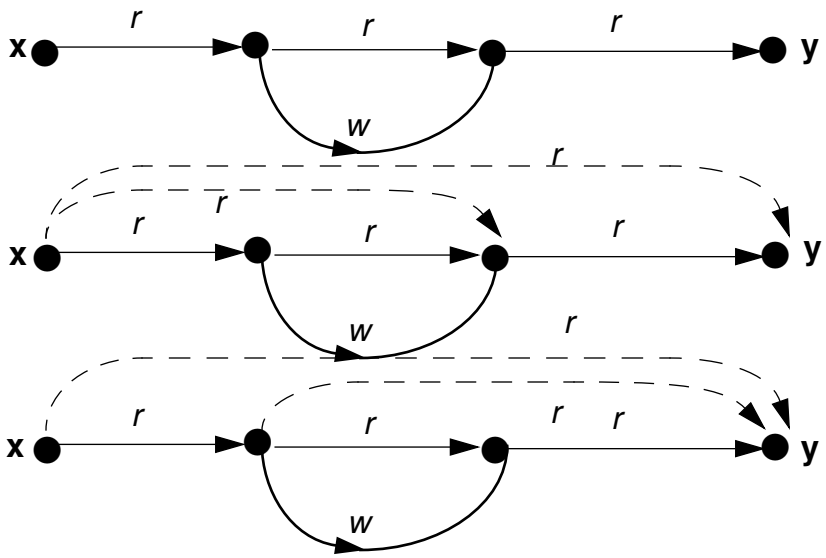

Figure 4. When cost is a function of rule applications, the (implicit) path from **x** to **y** has multiple cost. Suppose the *spy* rule costs 1 and the *post* rule costs 2. If the path is generated by to applications of the *spy* rule in the obvious way (see second graph), the path from **x** to **y** costs 2. But if the *post* rule and the *spy* rule are used to generate the graph (see third graph), the cost is 3.

4.3. Augmenting the Closure

This technique can both find witnesses and analyze the cost of information flow along paths in the graph. Each requires a similar augmentation of the closure of the graph. What the cost model applies to differentiates the interpretations. If the cost model applies to rule applications then we will be able to find the witness, but not the cost of information flow through the resulting graph. If the cost is the cost of the path through which the information flows, then we may have lost the information about the witness. We can use the model to generate either one depending on the requirements.

If we want to analyze the cost of information flow paths, then the closure need only assign the cost of each read edge added to the graph. However, edges with multiple cost can be added to the graph (see Figure 4).The closure graph G^* is now a weighted graph, and we can simply modify Dijkstra's algorithm to return the shortest path in $O(V^2)$ time. An extension of Breadth-first search can be used to return the k-shortest paths in order.

As it happens, we can do exactly the same thing if we want to generate the cost of witnesses. However, we only want costs of edges in this graph to reflect the application of rules, where in the previous case read edges cost whether they were added by rules or present in the original graph. So initially, all read edges in the graph cost nothing. As new

read edges are created during closure, they are annotated with the cost of the rules used to create them. In some cases we may need an auxiliary copy of the graph to keep track of information used in computing the cost of rule applications; for instance, if the cost of rule applications is a function of some other costs of edges between the nodes. As before, we can apply Dijkstra's algorithm or breadth-first search to find the shortest paths.

As an aside, we note that if computing the cost of each rule application is constant, then the cost of computing the closure is polynomial in the size of the graph.

5. Ordered Generation of Witnesses for Rights Flow

An equally important question in Take-Grant graphs is that of rights flow. Often we would like to determine whether or not rights could flow between two subjects. We now show how to modify the closure so that we can determine the minimum cost path for the flow of rights. In this case, we must build an auxiliary graph such that a directed edge between two nodes in the auxiliary graph is labeled with the cost of passing the right from one node to another according to the cost model. Fortunately, in this case, we can dispense with the *de facto* rules, since these rules do not alter the state of rights in the graph. As we showed earlier, we can also dispense with deletes. As before, we can interpret these costs as either the cost of generating the appropriate witness or as the cost of excercising the rule to accomplish the rights flow. We can run Dijkstra's algorithm on the resulting graph as usual.

6. Examples

We first present an example based on a model of network transfer [6] in which information is transferred from **f'** to **f**. Given the protection graph in Figure 5, the following is the set of edges in the initial graph; the notation is (*source*, *rights*, *target*):

(**p**, *r*, **f**) (**p**, *r*, **q**) (**s**, *r*, **p**) (**s**, *r*, **q**)
(**q**, *r*, **s**) (**v**, *r*, **s**) (**v**, *w*, **f'**)

The *spy* rule adds read edges by combining 2 adjacent read edges:

(**v**, *r*, **p**) (**v**, *r*, **q**) (**p**, *r*, **s**) (**q**, *r*, **p**)
(**s**, *r*, **f**) (**q**, *r*, **f**) (**v**, *r*, **f**)

The post rule adds one more read edge based on a read and a write edge:

(**f'**, *r*, **f**)

Before turning to witnesses, let us establish two cost metrics. Metric A is rule-based; each application of the *spy* rule costs 1, and each application of the *post* rule costs 2. Metric B is edge-based; each edge

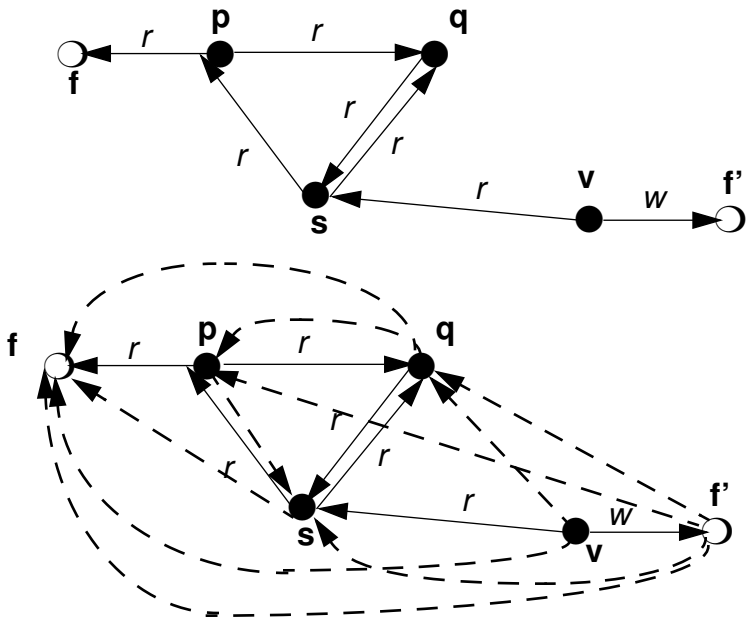

Figure 5. An example Take-Grant Protection Graph and its closure. The dashed lines are implicit edges, all *read*. As only *de facto* rules are used, no creations occur.

initially has cost 2, except for the edge (**s**, *r*, **p**), which has cost 1000; for this metric, the cost of a new edge is the sum of the costs of the edges used in the rule application that added the new edge.

Consider now the following two witnesses:

W1: spy (**s**, *r*, **f**); spy (**v**, *r*, **f**); post (**f'**, *r*, **f**)

W2: spy (**v**, *r*, **q**); spy (**q**, *r*, **f**); spy (**v**, *r*, **f**); post(**f'**, *r*, **f**)

W1 involves two applications of the *spy* rule (add the implicit edges from **s** to **f'** and from **v** to **f'**) and one application of the *post* rule (add the implicit edge from **f** to **f'**), and so under metric A costs 1 + 1 + 2 = 4. W2 involves three applications of the *spy* rule (add the implicit edges from **v** to **q**, from **q** to **f'**, and from **v** to **f'**) and one of the *post* rule (add the implicit edge from **f** to **f'**); under metric A, this costs 1 + 1 + 1 + 2 = 5. So by metric A, W1 costs less (and hence is more likely to be used) than W2, so given the choice between the two, under metric A, the path (**f'**, **p**), (**p**, **s**), (**s**, **v**), (**v**, **f**) should be monitored. Under metric B, the cost of W1 is (1000 + 2) + (1002 + 2) + (1004 + 2) = 3012, and the cost of W2 is (2 + 2) + (2 + 2) + (4 + 4) + (8 + 2) = 26; so given the choice between the two, under metric B, the path (**f'**, **p**), (**p**, **q**), (**q**, **s**), (**s**, **v**), (**v**, **f**) should be monitored.

Now consider figure 6. The second figure shows the closure; again, any of a number of information may flow using a number of witnesses. Here, the initial edges are:

(**x**, *t*, **y**) (**y**, *w*, **z**)

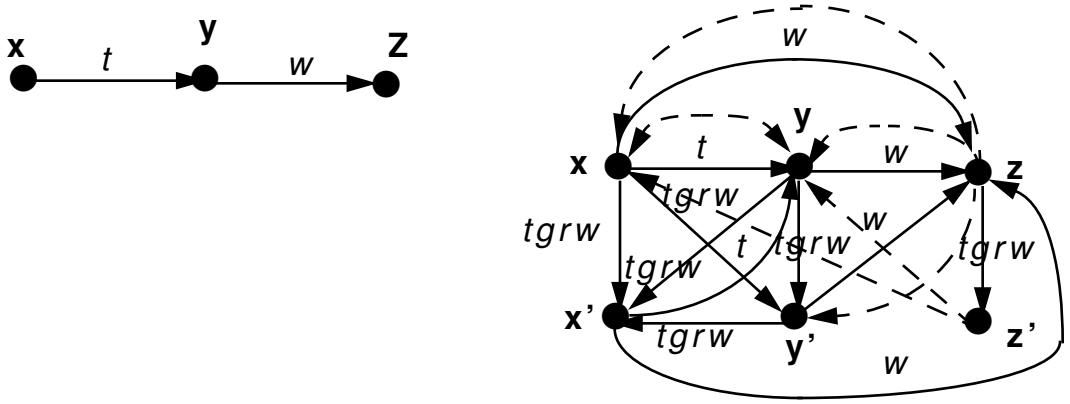

Figure 5. A second example Take-Grant Protection Graph and its closure. The dashed lines are implicit edges, all *read*. Note that the *create*s helped add implicit edges, but not explicit ones..

As *de jure* rules are needed to compute the closure, we first do *create*s:

(**x**, *tgrw*, **x'**) (**y**, *tgrw*, **y'**) (**z**, *tgrw*, **z'**)

Then we apply the *take* and *grant* rules to compute the *de jure* closure. The following edges are added by the *take* rule:

(**x**, *tgrw*, **y'**) (**x**, *w*, **z**) (**y**, *tgrw*, **x'**)

The following edges are added by the *grant* rule:

(**x'**, *t*, **y**) (**y'**, *tgrw*, **x'**) (**y'**, *w*, **z**) (**x'**, *w*, **z**)

The *post* rule adds two implicit edges:

(**x**, *r*, **y**) (**y**, *r*, **x**)

The *find* rule adds two more:

(**z'**, *r*, **x**) (**z'**, *r*, **y**)

So does the *spy* rule:

(**z**, *r*, **y**) (**z**, *r*, **x**)

However, the *pass* rule adds only one implicit edge:

(**z**, *r*, **y'**)

Cost metric C is rule-based; each application of a *de jure* rule or of the *pass* rule costs 1, and each application of other *de facto* rules costs 2 (the cost is the number of subjects that must act in the rule application). Consider now the following two witnesses showing how information can be transferred from **x** to **z**:

W3: create (**z**, *tgrw*, **z'**); take (**x**, *w*, **z**); find (**z'**, *r*, **x**); spy (**z**, *r*, **x**)

W4: create (**y**, *tgrw*, **y'**); take (**x**, *r*, **y'**); pass (**z**, *r*, **y'**); spy (**z**, *r*, **x**)

One uses the *find* rule and the other the *pass* rule. If cost is based solely on the rule application (metric C), then W3 costs 6 and W4 costs 5; hence W4 is more likely than W3, and the path it uses should be monitored.

7. Conclusions and Future Work

We presented the notion of "cost" to generate the most likely witnesses, and hence paths, for information and rights transfer. We carefully avoided prescribing a metric for cost. First, many models may be appropriate; we presented two, one based on the number and type of rule applications, and the other based on assigned edge cost. As we saw, the lowest cost path (or witnesses) is very sensitive to the choice of model, and a model may assign multiple weights to a single edge even when only one metric is used.

The choice of cost model is domain dependent, but we can identify two fundamental types: the fixed cost per rule models and the variable cost per rule models. The latter offers a rich set of possibilities; for example, one could partition nodes and edges into sets and base simple cost functions upon membership in those sets. Appropriate domains for this type of model include operating systems (the cost of paging protection information suggests one simple division of subjects and objects) and conspiracy classes (in which transfer among members of a conspiracy is of minimal cost, but transfer through a domain where a non-member might detect the transfer is of high cost).

The latter suggests that the most desirable path may not controlled by cost but by other factors such as probability of detection, certainty of completion of the transfer, and a bartering of favors in the non-technical realm. In these cases,a better metric for path selection is a probability taken over all possible paths (or sub-paths). Such an approach requires a different algorithm, because we try to maximize probability rather than minimize it, as we do with cost, and the algorithms we use to minimize cost are no applicable to maximizing probability.

Generalizing this approach to cover reflexive Take-Grant would be an interesting exercise. While the algorithms would be straightforward (nothing in this work depends upon the irreflexivity of the model) the derivation of the complexity would differ considerably, because the analysis would be correspondingly more complex. Whether the complexity of the reflexive model is greater than or equal to that of the irreflexive model may shed some light on the usefulness of the two.

Our goal has been to determine the path along which rights and information can be transferred with lowest cost. In practise, psychology plays a role in the cost function because an omniscient adversary may use knowledge of our cost function to plan the transfer. Under this scenario, one might add as part of the cost function a measure of how many paths

the edge (or node) lies on. This approach is somewhat speculative, in the sense that intrusions and countermeasures follows gaming theory rather than algorithm analysis. Still, the analysis in this paper provides a basis for handling the problem. The key is the cost function.

8. References

[1] M. Harrison, W. Ruzzo, and J. Ullman, “Protection in Operating Systems,” *Communications of the ACM* **19** (8) (Aug. 1976), 461-471.

[2] A. Jones, R. Lipton, and L. Snyder, “A Linear Time Algorithm for Deciding Security,” *Proc. 17th Annual Symp. on the Foundations of Computer Science* (Oct. 1976), 33-41.

[3] L. Snyder, “Theft and Conspiracy in the Take-Grant Protection Model,” *JCSS 23*, 3 (Dec. 1981), 333-347.

[4] M. Bishop and L. Snyder, “The Transfer of Information and Authority in a Protection System,” *Proc. 7th Symp. on Operating Systems Principles* (Dec. 1979), 45-54.

[5] M. Bishop, "Theft of Information in the Take-Grant Protection Model," *Journal of Computer Security* **3**(4) (1994/1995), 283-308.

[6] M. Bishop, "Conspiracy and Information Flow in the Take-Grant Protection Model," to appear in the *Journal of Computer Security.*