Quantitative Hook-Up Security for Covert Channel Analysis

D.G. Weber

Odyssey Research Associates, Inc.
301A Harris B. Dates Drive
Ithaca, NY 14850-1313
607 277 2020

Abstract

We construct a class of security policies for multi-level systems. Policies of this class allow some downgrading of information but limit its bandwidth. A policy that prevents downgrading completely is a special instance within this class. Security as defined by this class is quantitative *in the sense that deviations from perfect security can be compared by the degree to which they deviate. It is* analytical *or* "hook-up security" *in the sense that the degree of security or insecurity of a large system can be derived formally from the degree of security or insecurity of subsystems from which it is composed.*

1 Introduction

The task of building multi-level secure (MLS) computer systems has shown itself to be difficult. Part of the difficulty has come from definitions of "security" that are not precise enough, not flexible enough, or not general enough, to be used as specifications against which real systems can be measured. In this paper, we attempt to bring security specifications closer to the point at which they can be applied naturally and confidently to real systems.

To begin, we review some previous developments of MLS specifications. The Bell-LaPadula security policy[1] prevented the most direct, Trojan-Horse attacks against multi-level systems, and showed the danger of assuming that security was trivial. Covert channels could be handled within the Bell-LaPadula framework, but the methods used were *ad hoc*.

[0]This work was supported by the Army Communications and Electronics Command at Fort Monmouth under Contract No. DAAB07-87-C-A011. The views and conclusions contained in this paper are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Army or the U.S. Government.

The Goguen-Meseguer security policy[2] was an attempt to capture the intuitive idea of *non-interference*. Preventing requests by users at high security levels from "interfering" with the responses given to users at lower security levels handles all threats handled by Bell-LaPadula, and prevents covert storage channels as well. However, the class of systems covered by the Goguen-Meseguer policy was limited; whether the policy was also *ad hoc* was not clear.

The model of information flow proposed by Sutherland [7] treated flows of information as "deducibility". The model was very general, but could be instantiated to produce a multi-level security policy, called *deducibility security*, similar to Goguen-Meseguer.

Hook-up security was introduced by McCullough[4]. While it may seem obvious that "secure" interconnection of "secure" systems will produce a larger system that is also "secure", McCullough showed that this depends crucially on the definition of security used. Using the formal definitions of non-interference and of deducibility security, the interconnection of secure systems was at best ill-defined and was at worst insecure. McCullough strengthened deducibility security so that it became a *composable* property, i.e., the interconnection of two component systems for which the property holds yields a larger system for which the property also holds. This property was called "restrictiveness" or "restriction".

Hook-up security is useful for showing, formally, that large systems are secure. One need not apply Goguen-Meseguer or deducibility security directly to the entire system, but may *analyze* the system into components, and show that each component is restrictive. The theorem that restriction is composable then serves as the basic tool of security analysis. In addition, however, many other theorems about the hook-up of systems are possible and useful. Other composable properties, related to security, were found[8]. Other theorems were developed to show that components with properties other than restriction can be interconnected to produce restrictive systems[10],[6]; these theorems are useful for analyzing the security of systems in which individual parts are not restrictive.

Yet a major problem remains: real-world "secure" systems are often not intended to be completely secure! Complete and perfect security often interferes with the functionality desired in real systems, and practicality sometimes rules out perfectly secure designs. Therefore, designers often prefer to approximate perfect security. Security properties which allow some form of insecurity yet retain limits on what can be deduced by low-level users about the behavior of users at higher level have been proposed: by Goguen-Meseguer ("conditional-noninterference")[2], by this author in the context of fault tolerance[9], and by others[8],[3]. These proposed properties all suffer from two drawbacks:

1. The degree to which systems are allowed to be insecure is not quantified. Thus, there is no formal method for stating whether a system meeting one of these policies is slightly or grossly insecure.

2. Even if informal methods are used to show that various components are only slightly insecure, there is no proof that the interconnection of slightly insecure components won't yield a larger system that leaks massively. In other words, a theory of security analysis for these properties does not exist.

This paper is a first attempt at addressing these problems. A class of security policies is introduced; policies of the class put quantitative limits on downgrading, and are also composable. The class is a generalization of a composable MLS property, *flow-security*[5], introduced by McCullough as a variant of restriction. Section 2 discusses flow-security. Section 3 gives a formal meaning to the interconnection of systems. Section 4 motivates the definition of *n-limited secure* state machines, which deviate from perfect security to a degree indicated by the positive integer n. Section 5 defines n-limited security precisely. In section 6, it is claimed and argued informally that n-limited security is a kind of composable property. Finally, section 7 notes the limitations of this result, and gives directions for future research.

2 Multi-level Security

The property of **flow-security** is defined for a particular class of state machines. A state machine of this class is a 6-tuple $(\Sigma, E, I, O, \Phi, lvl)$, where:

- Σ is a set of states;
- E is a set of events;
- I is a set of input events, with $I \subseteq E$;
- O is a set of output events, with $O \subseteq E$ and with O and I disjoint;
- Φ is a ternary relation on events and ordered pairs of states;
- lvl is a function from events into the (fixed) set of possible security levels.

At each moment of time, the state machine exists in exactly one state from the set Σ. It progresses through time by engaging sequentially in events from the set E. When the machine is in the state s_{old}, the event e may occur taking the machine into state s_{new} only if Φ holds for (e, s_{old}, s_{new}). This *transition* is written $s_{old} \xrightarrow{e} s_{new}$, and in general Φ is called the *transition relation*. When the machine can make transitions involving a sequence of events, γ, we write $s_{old} \xrightarrow{\gamma} s_{new}$.

Every state machine we consider will be *input-total*. This means that from every state and for every input event it is possible for the machine to make some transition. It can never refuse an input. Input-totality can be thought of either as a limitation on the class of state machines, or as a precondition for defining an event to be an input. Input-totality is needed to prevent the refusal of an input event from conveying information back to the source of that input.

We assume that a machine is started in some state, s_0, chosen in advance and not on the basis of classified information. The choice of an input event, e_{in}, can be made at any time but only by some external users or agents who have authorized access to information at security level $lvl(e_{in})$. Any output event, e_{out}, is assumed to be visible to external users or agents authorized at security level $lvl(e_{out})$ or greater. We will often consider what can be deduced by an observer at some particular level l; security levels less than or equal to l will be called "low" levels, while other levels will be called "high". If the machine is to be secure for the observer at l, high-level information, i.e., that received via inputs at high levels, must be prevented from affecting observations of outputs at low levels.

The effects of inputs from high-level users are propagated through the machine at each transition. Rather than constrain the effect of high-level inputs on low-level outputs directly, we constrain the effects on the state of the machine after each transition. Suppose that it is possible to distinguish a high-level "aspect" and a low-level "aspect" of each state, and to define an equivalence relation on states such that two states belong to the same equivalence class exactly when they share the same low-level aspects. Security can then be expressed as properties of the transition relation that prevent high-level aspects of the state from affecting low-level ones.

To illustrate the distinction between different "aspects" of states, consider an example. If the set of states is defined as the direct product of sets of values for different "state variables", then we might assign security levels to state variables. The high-level "aspect" of a state would be the set of values of its high-level variables, and similarly the low-level "aspect" of a state would be the set of values of its low-level variables. In this case, states will belong to the same equivalence class when low-level variables take equal values. Rather than assume that states are always combinations of state variables, as in this example, we will simply rely on the equivalence relation on states to separate the high- from the low-level aspects.

Given an equivalence relation on states for level l, a machine is flow-secure at l if the following conditions hold:

- High-level input transitions do not affect the low-level aspects of the state;
- Low-level input transitions may affect any aspect of the state, but high-level aspects of the state before the transition cannot affect low-level aspects of the state after the transition;
- During a sequence of output transitions, the high-level aspects of the state before the sequence may affect high-level outputs and high-level aspects of the state, but they cannot affect the sequence of low-level outputs, nor can they affect the low-level aspects of the state afterward.

A machine is flow-secure if it is flow-secure at every level.

The conditions that define flow security are stated informally above. Formality will be added in later sections, when the property of n-limited security is stated. n-limited security, for $n = 1$, is slightly stronger than flow-security at a given level.

Note that there may be events of the machine (set E) which are neither inputs nor outputs (sets I and O). These are *internal* events, and for simplicity we will treat these events as though they were as visible to external users as outputs. This simplification can only overestimate the insecurity of the machine.

3 Hook-Up

Two state machines, A and B, can be "hooked up" or interconnected to form a larger state machine, denoted by $A\|B$. Once a precise definition has been given for the interconnection operator, $\|$, properties of $A\|B$ can be inferred from the properties of A and B individually.

Interconnection of two machines means that events belonging to one machine but not to the other may be executed independently by each machine, while events that are shared by the machines must be executed simultaneously by both. Let $A = (\Sigma_A, E_A, I_A, O_A, \Phi_A, lvl_A)$ and $B = (\Sigma_B, E_B, I_B, O_B, \Phi_B, lvl_B)$. The two machines may have some events in common; let $E_{int} = E_A \cap E_B$. The interconnection $A\|B$ will be defined if each event, e, of E_{int} is an output of one machine and an input of the other, and if $lvl_A(e) = lvl_B(e)$. The machine $A\|B$ is the tuple $(\Sigma, E, I, O, \Phi, lvl)$, where

- $\Sigma = \Sigma_A \times \Sigma_B$, i.e., each state of Σ is an ordered pair of states from Σ_A and Σ_B;
- $E = E_A \cup E_B$;
- $I = (I_A \cup I_B) - E_{int}$;
- $O = (O_A \cup O_B) - E_{int}$;
- Let $e \in E$ be an event, and let $s_1 = (s_{A1}, s_{B1})$ and $s_2 = (s_{A2}, s_{B2})$ be states of $A\|B$ formed as pairs of states of its components. There are three cases:
    1. When $e \in E_A$ and $e \notin E_B$, $\Phi(e, s_1, s_2)$ holds iff $\Phi_A(e, s_{A1}, s_{A2})$ and $s_{B1} = s_{B2}$.
    2. When $e \in E_B$ and $e \notin E_A$, $\Phi(e, s_1, s_2)$ holds iff $\Phi_B(e, s_{B1}, s_{B2})$ and $s_{A1} = s_{A2}$.
    3. When $e \in E_{int}$, $\Phi(e, s_1, s_2)$ holds iff $\Phi_A(e, s_{A1}, s_{A2})$ and $\Phi_B(e, s_{B1}, s_{B2})$.
- lvl is the union of lvl_A with lvl_B.

A property is **composable** if, given that it holds both of A and of B, then it holds for $A\|B$. Flow-security, as defined in the previous section, is composable.

4 Covert Channels

When a system is not secure, it is because the design of that system allows a high-level user to encode information in the sequence of high-level inputs and outputs, and also allows lower-level users to decode the information later by interacting with the system at lower levels. The method of encoding information is called a **channel**. A channel allows high-level information to be **downgraded**, and therefore compromised.

Usually two types of channel are distinguished: **overt** and **covert**. Overt channels are distinguished from covert ones in that the method used to encode information is trivial. For example, if a high-level user can write into a file that can be read by a lower-level user, the channel is overt.

Covert channels are often more subtle. For example, suppose a file system implements concurrency control by locking a file when it is open for writing, and promptly rejecting other requests for access. A high-level user could choose whether to lock a particular file at a particular time depending on one bit of high-level information. A low-level user could try to access the same file at an agreed-upon time after the high-level choice was made; whether the request

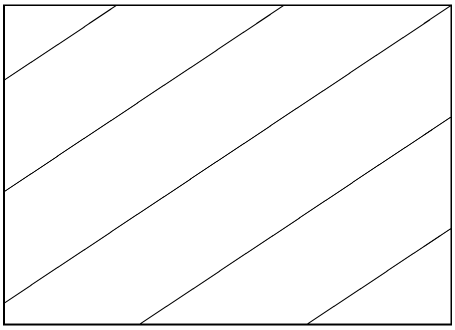

Figure 1. Coarse-grained partition of states.

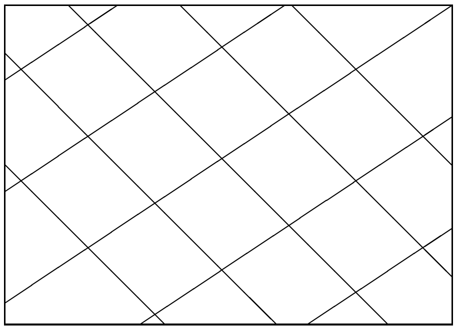

Figure 2. Fine-grained partition of states.

for access is accepted or rejected allows the low-level user to determine the single bit of high-level information. By agreeing on a schedule of times at which the lock can be reset, a sequence of bits can be downgraded.

We will not try to make precise the distinction between covert and overt, but rather will look for security properties that can apply to both cases as appropriate.

In the definition of flow-security in section 2, we appealed to an equivalence relation on states to distinguish low-level aspects of a state from high-level aspects. When some downgrading is to be allowed, however, we may distinguish three aspects: low-level, high-level, and a shared level through which information can be downgraded from high to low. If the high-, low-level, and shared aspects are thought of as sets of state variables, then it is the shared variables which will act as storage for transferring information from high to low. Using state variables as an example, though, should not be mistaken for the general case. In general, a set of states might not be separable into a direct product of sets of values of state variables, and even if it is, the shared aspects of each state need not correspond to a particular variable.

The three aspects of each state can be distinguished by defining two equivalence relations: one which partitions states into relatively coarse equivalence classes, separating the low-level aspects from the shared and high-level ones (see Figure 1), and one which partitions states into relatively fine equivalence classes, separating the low-level and shared aspects from the high-level ones (see Figure 2). The first relation takes states as equivalent if they have the same low-level aspects; the second takes states as equivalent if they have the same low-level and shared aspects.

To limit the rate at which information can be downgraded, we prevent direct downgrading from high to low, and constraint the amount of information that can be stored at one time in the shared aspects.

How dangerous is a particular channel? The answer depends on a variety of factors:

- How many bits can be communicated at once? In the covert-channel example above, one bit at a time was communicated, while in the overt-channel example, the contents of an entire file could be downgraded in one operation.
- What information can be communicated through the channel? If no interesting high-level information can be fed into the channel, it is hardly dangerous.
- How quickly can the encoding mechanism be set up, reset, and re-used, if at all? If an improbable scenario must happen before the channel can be set up, if it can never be reset, or if resetting takes a long time, then the channel is generally less dangerous than if it can be set up quickly.

In what follows we will be concerned primarily with the first factor. Channels will be treated as useful for communicating any kind of information; only the rate of communication will be important. To take account of probability and other specialized facts about the set-up of the channel, one would need to consider the possible state-machine transitions in greater detail than we do here. These simplifications will tend to overestimate the danger posed by a given channel.

If the shared aspects of states permit the recording of at most a choice among n alternatives, then at most $\log_2 n$ bits can be downgraded during any one transition. If a sequence of k transitions can be accomplished during t seconds, then the **bandwidth** of the channel has an upper bound of $(k \log_2 n)/t$ bits per second.

Note that it is not sufficient to state bandwidths in terms of "bits per transition", and then give a typical time per transition. In our model of process interconnection, transitions of hooked-up machines may proceed concurrently. Therefore, the average time taken by transitions will generally be less than the typical time taken sequentially by each.

Note also that timing considerations do not occur in the model directly, but only enter through an interpretation in which channel bandwidths are derived. Therefore, channels that use the times at which events occur to encode information (**timing channels**) are not accounted for. Only **covert**

storage channels are quantified by this model.

In the following section, we give a precise definition of a security property to limit downgrading via storage channels.

5 Limited Downgrading

Definition 1 *A state machine is* **n-limited secure at level** l*, where* n *is a positive integer, if there exist two equivalence relations on states,* $\equiv_H$ *and* $\equiv_L$*, such that the following conditions hold:*

1. Equivalence relation $\equiv_H$ partitions the set of states into a finer, more detailed, set of classes than does $\equiv_L$. In other words, for any states p and q, $p \equiv_H q \rightarrow p \equiv_L q$. The differences between classes of $\equiv_L$ correspond to information known at level l, while differences between classes of $\equiv_H$ include additional, shared information that may be available for downgrading.

2. Equivalence relation $\equiv_H$ partitions each equivalence class of relation $\equiv_L$ into at most n sub-classes. Thus, no set of $n+1$ states, $s_1, s_2, \ldots, s_n, s_{n+1}$, can be found such that $s_i \equiv_L s_j$ while $s_i \not\equiv_H s_j$ for $i \neq j$.

3. The machine is input-total.

4. High-level input transitions don't affect low-level aspects of the state. So, for any input, e, with $lvl(e) \not\leq l$, and for any transition, $p \stackrel{e}{\rightarrow} q$, we have $p \equiv_L q$.

5. During a low-level input transition, high-level aspects of the state before the transition can't affect low-level aspects afterward. In other words, for any input, e, with $lvl(e) \leq l$, and for any states p, p', and q, such that $p \stackrel{e}{\rightarrow} q$ and $p \equiv_H p'$, there exists a state, q', such that $p' \stackrel{e}{\rightarrow} q'$ and $q \equiv_L q'$.

6. During an output (or internal) transition, high-level aspects of the state before the transition can't cause low-level outputs or affect low-level aspects of the state afterward. In other words, suppose we are given an output, e, and states p, p', and q, such that $p \stackrel{e}{\rightarrow} q$ and $p \equiv_H p'$. There are two cases:

   (a) If e is a high-level output, with $lvl(e) \not\leq l$, then there exists a state, q', and a sequence of high-level outputs, γ, such that $p' \stackrel{\gamma}{\rightarrow} q'$, and $q \equiv_L q'$.

   (b) If e is a low-level output, with $lvl(e) \leq l$, then there exists a state, q', and a sequence of high-level outputs, γ, such that $p' \stackrel{e\gamma}{\rightarrow} q'$, and $q \equiv_L q'$.

A state machine that is 1-limited secure at every level is flow-secure. When $n = 1$, relations $\equiv_H$ and $\equiv_L$ must be the same, and the constraints on transitions become slightly stronger than those for flow-security. (The only difference occurs in rule 6b, in which the more general transition, $p' \stackrel{\gamma e \gamma'}{\rightarrow} q'$, where γ and γ' are sequences of high-level outputs, may exist to prevent deducibility of high-level events and aspects of the state.)

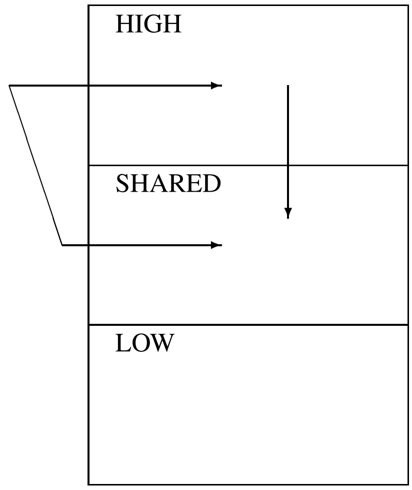

Figure 3. Flows possible for high-level input.

Why do the above rules limit downgrading? First, one is prevented from using high-level inputs to write directly into the low-level state, and one is prevented from using low-level outputs to read directly from the high-level state. Second, during a transition one may move information from high-level to shared aspects and from shared to low-level ones, but influence from high directly to low is ruled out. Therefore, high-level choices to be downgraded must pass through the shared aspects of the state, and this acts as a bottleneck.

The rules of the definition are constraints that limit cause-and-effect during transitions, and so limit possible information flow. However, it is helpful to visualize how information *can* flow during transitions. Figures 3, 4, 5, and 6 show schematically, for the four possible kinds of event, the information flows that could be involved in downgrading. (Of course, upgrading flows are always possible, but these do not change the rate of downgrading.) Each diagram is a pictorial division of the state into "HIGH", "LOW" and "SHARED" aspects. Information can flow from high to low, but only by passing through the shared region.

6 Analysis

The property defined in the previous section is composable, in the following sense:

Theorem 1 *If machine* A *is* n*-limited secure at level* l*, and machine* B *is* m*-limited secure at level* l*, then machine* $A \| B$ *is* nm*-limited secure at level* l*.*

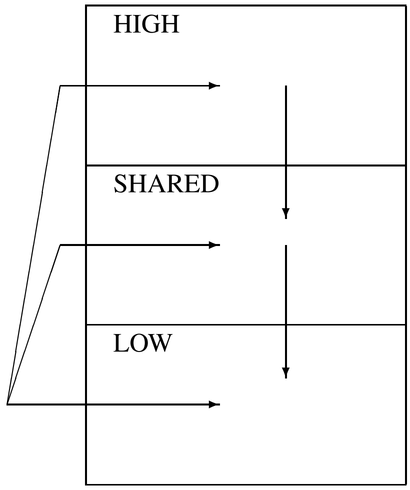

Figure 4. Flows possible for low-level input.

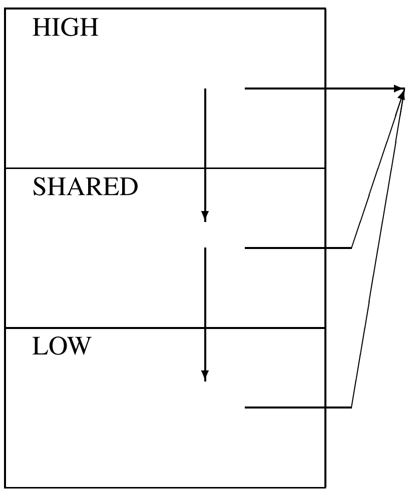

Figure 5. Flows possible for high-level output.

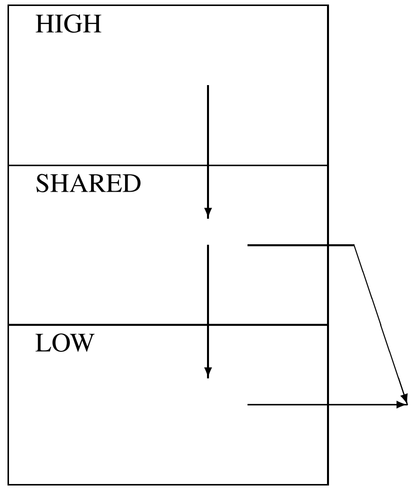

Figure 6. Flows possible for low-level output.

The proof is straightforward, and we sketch it here. Since A is n-limited secure, there are equivalence relations $\equiv_{AH}$ and $\equiv_{AL}$ on the states of A, such that properties 1 through 6 of the previous section hold. Similarly, there are equivalence relations $\equiv_{BH}$ and $\equiv_{BL}$ on the states of B. Let s_{A1} and s_{A2} be states of A, let s_{B1} and s_{B2} be states of B, and let $s_1 = (s_{A1}, s_{B1})$ and $s_2 = (s_{A2}, s_{B2})$ be states of $A \| B$ formed as pairs of states of its components. We define equivalence relations on the states of $A \| B$ as follows:

- $s_1 \equiv_H s_2$ iff $s_{A1} \equiv_{AH} s_{A2}$ and $s_{B1} \equiv_{BH} s_{B2}$
- $s_1 \equiv_L s_2$ iff $s_{A1} \equiv_{AL} s_{A2}$ and $s_{B1} \equiv_{BL} s_{B2}$

The six properties now hold for the interconnected machine, and we enumerate the cases of the proof.

1. $\equiv_H$ defines classes of finer granularity than $\equiv_L$ because its components do.
2. An arbitrary class of $\equiv_L$ is partitioned at most n times by $\equiv_{AH}$ and at most m times by $\equiv_{BH}$, and therefore, at most nm times by $\equiv_H$.
3. An input to $A \| B$ is an input to exactly one of its components, so the interconnected machine is input-total when its components are.
4. A high-level input affects exactly one of the component machines, and does not change its low-level state. Therefore, the low-level state of the entire machine is unchanged.
5. A low-level input affects exactly one of the component machines, and changing high-level or shared aspects of a state before a transition does not affect the low-level state after. Therefore, the low-level state of the entire machine is unaffected.

6. The two cases corresponding to output (and internal) events are similar. The case of low-level events is more difficult, and will be discussed as an example. Given that a low-level non-input event is possible for some state of the interconnected machine, the event will either be an output (or internal event) of one machine or the output of one and the input of the other. Without loss of generality, let A be the machine which performs the output. A individually obeys rule 6 of the previous section; therefore, a state of A before the transition, with different high-level aspects, may require that additional high-level outputs be performed, but the low-level state resulting after this extended sequence of transitions will be unaffected. Even if some of the additional high-level outputs from A are also inputs to B they will not affect B's low-level state. If B must accept the low-level output from A, differences in B's state before the transition will not affect B's low-level state after. Thus, the low-level state of the interconnected machine will also be unaffected.

What does this theorem mean? It tells us that the severity of a channel for downgrading by a composite machine is limited by the maximum severity of channels on its component machines. We can hook up two leaky machines to get a machine with a (possibly) larger leak, but the combined leak won't be arbitrarily bad.

Superficially, the result appears quite natural, since an n-way choice for machine A and an m-way choice for machine B represent $\log_2 n$ and $\log_2 m$ bits respectively, and $\log_2 nm$ is their sum. However, in the final analysis it is the bandwidths of channels which should be compared.

Suppose the average time per transition is t and is equal for A and B. If A and B have no events in common, then running them concurrently will on the average execute one transition on A and one on B in time t, and will downgrade at most $\log_2 nm$ bits. This rate is $(\log_2 nm)/t$ bits per second. Could information be downgraded faster if the machines were hooked up? The theorem merely limits $A \| B$ to downgrading at most $\log_2 nm$ bits per transition. If most transitions executed by $A \| B$ are concurrent, then $A \| B$ might approach the rate of 2 transitions per time t. The limiting rate would then be $(2 \log_2 nm)/t$ bits per second. This is double the rate that is possible for the machines acting independently.

This speedup, while allowed by the theorem, is unlikely since it depends on most transitions happening concurrently on one or the other machine. Yet these concurrent transitions are precisely the ones which should be limited to downgrading either $\log_2 n$ or $\log_2 m$ bits. It is the events shared between A and B which may compromise $\log_2 nm$ bits, but these events take twice as long on average. A more detailed analysis of the relation between equivalence classes of states and bandwidth is needed before stricter limits on bandwidth can be set.

7 Conclusion

The property of n-limited security comes closer than its predecessors to being adequate as a security policy for real MLS systems. Yet it too has many inadequacies.

- The fact that different transitions may take different times, and hence affect the bandwidth of channels, is not built into the model.
- The rules limiting transitions of the state machine appear *ad hoc*, and are not formally derived from a general theory of information flow or deducibility.
- The degree to which a system is insecure is not measured, but merely bounded. An n-limited secure system is guaranteed not to compromise information faster than some rate, yet it may be that the design details of a specific n-limited secure machine limit its downgrading bandwidth to a much smaller rate instead. It would be far better to use a framework in which the security policy is characterized by the worst rate that can actually be realized by that machine.
- The bandwidth of a channel at level l may or may not be related to the bandwidth of a channel at level l', and n-limited security does not necessarily make this distinction. For example, a machine that downgrades simultaneously from TOP SECRET to SECRET and from SECRET to CONFIDENTIAL is leakier than a machine that can only do one or the other. Yet either machine may be, say, 2-limited secure at both SECRET and CONFIDENTIAL.
- Probabilistic considerations are not included. For example, a finite buffer which may become blocked can serve as a covert channel to downgrade information. Yet, if the buffer is made so large that it is unlikely even to be full, the channel is not useful.

These problems represent areas of future research.

References

[1] D. Bell and L. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical Report MTR-2997, Revision 2, MITRE Corp., Bedford, MA, Mar. 1976.

[2] J. Goguen and J. Meseguer. Security policy and security models. In *IEEE Symp. Security and Privacy*, 1982.

[3] D. Johnson and F. Thayer. Stating security requirements with tolerable sets. Technical report, MITRE Corp., July 1987.

[4] D. McCullough. Specifications for multi-level security and a hook-up property. In *IEEE Symp. Security and Privacy*, 1987.
[5] D. McCullough. Noninterference and the composability of security properties. In *IEEE Symp. Security and Privacy*, 1988.
[6] D. Rosenthal. An approach to increasing the automation of the verification of security. In *Comp. Security Foundations Workshop*, 1988.
[7] D. Sutherland. A model of information. In *Nat'l Comp. Security Conf.*, Sept. 1986.
[8] S. T. Vinter, D. Weber, et al. The secure distributed operating system project. Technical Report 6144, BBN Labs and Odyssey Research Associates, Feb. 1988.
[9] D. Weber. Specifications for fault tolerance. Technical Report 19-3, ORA Corp., Jan. 1988.
[10] D. Weber and B. Lubarsky. The SDOS project – verifying hook-up security. In *Comp. Security Applications Conf.*, pages 7–15, 1987.