Secrecy in Multiagent Systems*

Joseph Halpern
Department of Computer Science
Cornell University
halpern@cs.cornell.edu

Kevin O'Neill
Department of Computer Science
Cornell University
oneill@cs.cornell.edu

Abstract

We introduce a general framework for reasoning about secrecy requirements in multiagent systems. Because secrecy requirements are closely connected with the knowledge of individual agents of a system, our framework employs the modal logic of knowledge within the context of the well-studied runs and systems framework. Put simply, "secrets" are facts about a system that low-level agents are never allowed to know. The framework presented here allows us to formalize this intuition precisely, in a way that is much in the spirit of Sutherland's notion of nondeducibility. Several well-known attempts to characterize the absence of information flow, including separability, generalized noninterference, and nondeducibility on strategies, turn out to be special cases of our definition of secrecy. However, our approach lets us go well beyond these definitions. It can handle probabilistic secrecy in a clean way, and it suggests generalizations of secrecy that may be useful for dealing with resource-bounded reasoning and with issues such as downgrading of information.

1 Introduction

There have been many attempts in the past two decades to define what it means for a system to be perfectly secure, in the sense that unclassified, or "low-level" agents are unable to deduce anything about the actions or state of classified, or "high-level" agents. These attempts include *nondeducibility* [22], *noninterference* [7, 13], *selective interleaving functions* [15], *separability* [15], *nondeducibility on strategies* [23], *probabilistic noninterference* [9, 8], as well as more recent approaches based on process algebras such as CCS [5] and CSP [17, 19]. Despite this abundance of definitions, there is no widespread agreement on a general model.

In this paper, we provide such a model, by taking a somewhat different approach to the problem. We define a general notion of secrecy in multiagent systems. More specifically, we define what it means for the actions and state of one agent to be kept secret from some other agent. Our definition is based on Sutherland's notion of nondeducibility. Sutherland's definition has been criticized by McLean [14] both for treating the high and low-level agents symmetrically and for not being general enough, and by Wittbold and Johnson [23] for ignoring the role of strategies. We argue that all these criticisms are unfounded.

Although our work builds on previous work on noninterference and information flow, we use the term "secrecy" partly because the term "noninterference" is overloaded and partly because we are interested in a broader class of "secrets" than those considered by traditional definitions of noninterference.

We present our notion of secrecy within the context of the runs and systems model [4], where a run (or execution sequence) of a system is represented by a sequence of global states, each of which consists of the local states of the individual agents in the system. This model can be viewed as a generalization of the standard trace-based model, which has often been used in security (see, for example, [15]). This added generality is important because it allows us to reason about an agent's initial information and how it changes over time. The trace-based approach has been concerned primarily with the input and output values exchanged as a user or observer interacts with the system; using this approach, it is possible to define secrecy only for systems that can be characterized by observable input and output events. As pointed out by Focardi and Gorrieri [6], it is difficult to deal with issues such as deadlock using a trace-based approach. It is also difficult to deal with the whole spectrum of behavior between synchronous systems and completely asynchronous systems (a problem that seems to be shared by process-algebra approaches). As we shall see, the added generality of local and global states lets us deal with these

*Authors supported in part by NSF under grant IRI-96-25901 and IIS-0090145, by ONR under grants N00014-00-1-03-41 and N00014-01-10-511, and by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grant N00014-01-1-0795.

issues in a straightforward way.

We show that a number of previous notions of security, such as noninterference, nondeducibility on strategies, and separability, are special cases of our notion of secrecy. In doing so, we are able to clarify a number of subtleties, especially regarding the relationship between noninterference and assumptions of timing and synchrony. However, our framework lets us go well beyond the earlier notions of security. In addition, the framework can be cleanly extended to handle probabilistic security, as we show. Perhaps more important, it lets us make precise the intuition that a secret is something that the agent (or some set of agents) should never be allowed to know. If a classified user of a system has a secret cryptographic key, for example, this key should be kept secret from unclassified users. In any secure, multiagent system, there are likely to be a wide variety of facts that must be kept secret from various agents. We show how such secrecy requirements can be captured in a precise way, both at a semantic level and syntactically, using a modal logic of knowledge. At a syntactic level, we say that a formula φ is kept secret from an agent a if a never knows that φ is true. The task of a "secrecy administrator" is then to determine which formulas must be kept secret from various agents. In this way, we get a fine-grained tool that can be used to characterize secrecy requirements.

The rest of the paper is organized as follows. Section 2 reviews the multiagent systems framework and the definition of knowledge in multiagent systems. In Section 3, we give our definition of secrecy and discuss its relationship to Sutherland's notion of nondeducibility. In Section 4, we consider syntactic definitions of secrecy, using a logic of knowledge. Section 5 considers probabilistic secrecy. In Section 6, we compare our notions to some of the other definitions of security in the literature. We conclude in Section 7. Due to lack of space, many proofs are deferred to the full paper, or only sketched here.

2 Knowledge and Multiagent Systems

A multiagent system consists of n agents, each of which is in some *local state* at a given point in time. We assume that an agent's local state encapsulates all the information to which the agent has access. In the security setting, the local state of an agent might include initial information regarding keys, the messages she has sent and received, and perhaps the reading of a clock. The basic framework makes no assumptions about the precise nature of the local state.

We can view the whole system as being in some *global state*, which is a tuple consisting of the local state of each agent and the state of the environment, where the environment consists of everything relevant to the system that is not contained in the state of the agents. Thus, a global state has the form $(s_e, s_1, \ldots, s_n)$, where s_e is the state of the environment and s_i is agent i's state, for $i = 1, \ldots, n$.

A *run* is a function from time to global states. Intuitively, a run is a complete description of what happens over time in one possible execution of the system. A *point* is a pair (r, m) consisting of a run r and a time m. For simplicity, we take time to range over the natural numbers. At a point (r, m), the system is in some global state $r(m)$. If $r(m) = (s_e, s_1, \ldots, s_n)$, then we take $r_i(m)$ to be s_i, agent i's local state at the point (r, m). Formally, a *system* consists of a set of runs (or executions). Let $\mathcal{P}(\mathcal{R})$ denote the points in a system $\mathcal{R}$.

Given a system $\mathcal{R}$, let $\mathcal{K}_i(r, m)$ be the set of points in $\mathcal{P}(\mathcal{R})$ that i thinks are possible at (r, m), i.e.,

$$\mathcal{K}_i(r, m) = \{(r', m') \in \mathcal{P}(\mathcal{R}) : r'_i(m') = r_i(m)\}.$$

Agent i knows a fact φ at a point (r, m) if φ is true at all points in $\mathcal{K}_i(r, m)$. To make this precise, we need to be able to assign truth values to basic formulas in a system. We assume that we have a set Φ of primitive propositions, which we can think of as describing basic facts about the system. In the context of security protocols, these might be such facts as "the key is n" or "agent A sent the message m to B". An interpreted system $\mathcal{I}$ consists of a pair $(\mathcal{R}, \pi)$, where $\mathcal{R}$ is a system and π is an interpretation for the primitive propositions in Φ that assigns truth values to the primitive propositions at the global states. Thus, for every $p \in \Phi$ and global state s that arises in $\mathcal{R}$, we have $(\pi(s))(p) \in \{\mathbf{true}, \mathbf{false}\}$. Of course, π also induces an interpretation over the points in $\mathcal{P}(\mathcal{R})$: simply take $\pi(r, m)$ to be $\pi(r(m))$.

We can now define what it means for a formula φ to be true at a point (r, m) in an interpreted system $\mathcal{I}$, written $(\mathcal{I}, r, m) \models \varphi$, by induction on the structure of formulas:

- $(\mathcal{I}, r, m) \models p$ iff $(\pi(r, m))(p) = \mathbf{true}$;
- conjunction and negation are defined in the obvious way;
- $(\mathcal{I}, r, m) \models K_i\varphi$ iff $(\mathcal{I}, r', m') \models \varphi$ for all $(r', m') \in \mathcal{K}_i(r, m)$.

As usual, we write $\mathcal{I} \models \varphi$ if $(\mathcal{I}, r, m) \models \varphi$ for all points (r, m) in $\mathcal{I}$.

The systems framework lets us express in a natural way some standard assumptions about systems. For example, the system is *synchronous* if all agents know the time. Formally, $\mathcal{R}$ is synchronous if, for all agents i and points (r, m) and (r', m') if $r_i(m) = r'_i(m')$, then $m = m'$.

Another standard assumption (implicitly made in almost all systems models considered in the security literature) is that agents have *perfect recall*. Roughly speaking, an agent with perfect recall can reconstruct his complete local history. In synchronous systems, an agent's local state changes with every tick of the external clock, so agent i's having perfect recall implies that the sequence $\langle r_i(0), \ldots, r_i(m) \rangle$ must be encoded in $r_i(m+1)$. Our general definition of perfect recall is based on the intuition that an agent can sense that something has happened only when there is a change in his local state. Let *agent i's local-state sequence at the point* (r, m) be the sequence of local states she has gone through in run r up to time m, without consecutive repetitions. Thus, if from time 0 through time 4 in run r agent i has gone through the sequence $\langle s_i, s_i, s_i', s_i, s_i \rangle$ of local states, where $s_i \neq s_i'$, then her local-state sequence at $(r, 4)$ is $\langle s_i, s_i', s_i \rangle$. Agent i's local-state sequence at a point (r, m) essentially describes what has happened in the run up to time m, from i's point of view. Intuitively, an agent has perfect recall if her current local state encodes her whole local-state sequence. More formally, we say that *agent i has perfect recall in system $\mathcal{R}$* if, at all points (r, m) and (r', m') in $\mathcal{P}(\mathcal{R})$, if $(r', m') \in \mathcal{K}_i(r, m)$, then agent i has the same local-state sequence at both (r, m) and (r', m'). Thus, agent i has perfect recall if she "remembers" her local-state sequence at all times. It is easy to check that perfect recall has the following key property: if $(r', m_1') \in \mathcal{K}_i(r, m_1)$, then for all $m_2 \leq m_1$, there exists $m_2' \leq m_1'$ such that $(r', m_2') \in \mathcal{K}_i(r, m_2)$. (See [4] for more intuition regarding this definition.)

3 Defining Secrecy

Many previous attempts to define secrecy, particularly definitions of noninterference, have not motivated the definitions in a clear way. In this section, we give abstract definitions of secrecy and motivate these definitions within the framework of the runs and systems model. In particular, we reason informally about the knowledge of the low-level and high-level agents as they interact with the system, and define secrecy in such a way as to ensure that low-level agents do not know anything about the state of high-level agents. In Section 4, we formalize these intuitions using an epistemic logic.

The strongest notion of secrecy that we consider is the requirement that a low-level agent, based on her local state, should never be able to determine anything about the local state of the high-level agent. More specifically, the low-level agent should never be able to rule out any possible high-level state. In terms of knowledge, this means that the low-level agent must never know that some high-level state is incompatible with her current low-level state. To ensure that the low-level agent L is not able to rule out any possible high-level states, we insist that every low possible low-level state is compatible with every possible high-level state.

Definition 3.1: Agent j maintains *total secrecy* with respect to i in system $\mathcal{R}$ if for all points (r, m) and (r', m') in $\mathcal{P}(\mathcal{R})$, we have that $\mathcal{K}_i(r, m) \cap \mathcal{K}_j(r', m') \neq \emptyset$. ∎

Note that if we take i to be the low-agent L and j to be the high-level agent H, then this definition just formalizes the informal one given above. At the point (r, m), L cannot rule out any possible local state of H.

Total secrecy is a very strong property. For many interesting systems, it is much too strong. There are two important respects in which it is too strong. The first is already implicit in Sutherland's definition of nondeducibility. In some systems, we might want only some part of the high-level agent's state to be kept secret from the low-level agent. For example, we might want the high-level agent to be able to see the state of the low-level agent, in which case our definitions are too strong because they rule out any interaction between the states of the high-level and low-level agents. We can correct this situation by extracting from H's state the information that is relevant and ensuring that the relevant part of H's state is kept secret. The "extraction" is by means of an arbitrary "information function" (in the terminology of Sutherland [22]).

Definition 3.2: A *j-information function* on $\mathcal{R}$ is a function f from $\mathcal{P}(\mathcal{R})$ to some range that depends only on j's local state; that is $f(r, m) = f(r', m')$ if $r_j(m) = r_j'(m')$. ∎

Thus, for example, if j's local state at any point (r, m) includes both its input and output, $f(r, m)$ could be just the output component of j's local state. As far as our definitions of secrecy are concerned, it is possible to view a j-information function f as representing another agent in the system, one whose local state always contains less information than the local state of the agent j.

Definition 3.3: If f is a j-information function, agent j maintains *total f-secrecy* with respect to i in system $\mathcal{R}$ if for all points (r, m) and values v in the range of f, $\mathcal{K}_i(r, m) \cap f^{-1}(v) \neq \emptyset$ (where $f^{-1}(v)$ is simply the preimage of v, that is, all points (r, m) such that $f(r, m) = v$). ∎

Of course, if $f(r, m) = r_j(m)$, then $f^{-1}(r_j'(m')) = \mathcal{K}_j(r', m')$, so total secrecy is a special case of total f-secrecy.

Total f-secrecy is a special case of *nondeducibility*, introduced by Sutherland [22]. Sutherland considers "abstract" systems that are characterized by a set W of worlds. He focuses on two agents, whose views are represented by information functions g and h on W. Sutherland says that *no information flows from g to h* if for all worlds $w, w' \in W$, there exists some world $w'' \in W$ such that $g(w'') = g(w)$ and $h(w'') = h(w')$. This notion is often called *nondeducibility (with respect to g and h)* in the literature. To see how total f-secrecy is a special case of nondeducibility, let $W = \mathcal{P}(\mathcal{R})$, the set of all points of the system. Given a point (r, m), let $g(r, m) = r_i(m)$, and let

$h = f$. Then total secrecy is equivalent to nondeducibility with respect to g and h.

Note that the nondeducibility is symmetric: no information flows from g to h iff no information flows from h to g. Since most standard noninterference properties focus only on protecting the state of some high-level agent, symmetry appears problematic, because it implies that if the actions of some high-level agent are kept secret from some low-level agent, then the reverse must also hold. Indeed, McLean [14] makes this argument. We claim that McLean's criticism is unfounded. As we now show, whether symmetry is a problem depends crucially on the choice of information functions g and h.

Our definition of total secrecy is indeed symmetric: j maintains total secrecy with respect to i iff i maintains total secrecy with respect to j. However, the definition of total f-secrecy is not symmetric. If j maintains total f-secrecy with respect to i, it may not even make sense to talk about i maintaining total f-secrecy with respect to j, because f is not an i-information function. But note that f-secrecy is just an instantiation of Sutherland's definition, for the appropriate choice of g and h. Roughly speaking, the symmetry at the level of g and h does not translate to symmetry at the level of f-secrecy, which is where it matters. The information functions g and h play different roles in the definition of secrecy. The function g focuses on i's information (in our terminology, i's local state); the function h focuses on that part of j's local state that is of interest to the designer.

The second respect in which total secrecy is too strong is not captured by Sutherland's definition in any obvious way. To understand the issue, consider synchronous systems (as defined in Section 2). In such systems, the low-level agent can always be certain that the high-level agent knows what time it is, and can thus rule out *all* high-level states except those that occur at the current time. Even in semisynchronous systems, where agents know the time to within some tolerance ϵ, total secrecy is impossible, because low-level agents can rule out high-level states that occur only in the distant past or future.

To solve this problem, we must relax the requirement that the low-level agent cannot rule out any possible high-level states. We make this formal as follows.

Definition 3.4: An *i-allowability function* on $\mathcal{R}$ is a function C from $\mathcal{P}(\mathcal{R})$ to subsets of $\mathcal{P}(\mathcal{R})$ such that $\mathcal{K}_i(r,m) \subseteq C(r,m)$. ∎

Intuitively, $\mathcal{P}(\mathcal{R}) - C(r,m)$ is the set of points that i is allowed to "rule out" at the point (r,m). It seems reasonable to insist that all the points that i considers possible at (r,m) (i.e., points $(r',m') \in \mathcal{K}_i(r,m)$) should not be ruled out.

Definition 3.5: If C is an i-allowability function and f is a j-information function, then j *maintains f-C-secrecy with respect to i* if for all points $(r,m) \in \mathcal{P}(\mathcal{R})$ and values $v \in f(C(r,m))$ (where $f(A) = \{f(w) : w \in A\}$), we have that $\mathcal{K}_i(r,m) \cap f^{-1}(v) \neq \emptyset$. ∎

If $C(r,m) = \mathcal{P}(\mathcal{R})$ for all points $(r,m) \in \mathcal{P}(\mathcal{R})$, then f-C-secrecy reduces to f-secrecy. We also refer to C-secrecy when there is no f function (or, equivalently, when $f(r,m) = r_j(m)$). In general, allowability functions give a generalization of secrecy that is orthogonal to information functions.

Synchrony can be captured by the allowability function $S(r,m) = \{(r',m) : r' \in \mathcal{R}\}$. Informally, this says that agent i is allowed to know what time it is. In synchronous systems, S-secrecy has a simple characterization.

Proposition 3.6: *In a synchronous system, j maintains S-secrecy with respect to i if for all r and r' in $\mathcal{R}$ and times m, we have that $\mathcal{K}_i(r,m) \cap \mathcal{K}_j(r',m) \neq \emptyset$.*

We sometimes call S-secrecy *synchronous secrecy*. We remark that synchronous secrecy essentially amounts to the standard notion of *separability* in synchronous systems [15]. Total secrecy (Definition 3.1) is separability in completely asynchronous systems. (See Sections 6.1 and 6.4 for further discussion of this issue.) The security literature has typically focused on either synchronous systems or completely asynchronous systems. One advantage of our framework is that we can easily model both of these extreme cases, as well as being able to handle in-between cases (which do not seem to have been considered at all in the literature).

For example, consider a semisynchronous system where agents know the time to within a tolerance of ϵ. Thus, at time 5, the agent will know that the true time is in the interval $[5-\epsilon, 5+\epsilon]$. This corresponds to the allowability function $SS(r,m) = \{(r',m') : |m-m'| \leq \epsilon\}$, for the appropriate ϵ. We believe that any attempt to define security for semisynchronous systems will require something like our allowability functions (which is perhaps why this issue has not been considered up to now in the literature).

We next consider two orthogonal ways of extending f-C-secrecy. The first is a fairly obvious extension to multiple agents. So far we have considered only pairwise security. It is natural to ask what it means for j to maintain secrecy with respect to some set of agents. There are several possible definitions. One is simply that j maintains secrecy with respect to each agent in the set (according to one of the definitions above). A second captures the intuition that j maintains secrecy even if all the agents in the set could put their information together. (This corresponds to the notion of distributed knowledge [4].) For example we can define total secrecy with respect to a group of agents as follows.

Definition 3.7: Agent j maintains *total secrecy* with respect to the set A of agents in system $\mathcal{R}$ if $\cap_{i \in A \cup \{j\}} \mathcal{K}_i(r,m) \neq \emptyset$ for every point $(r,m) \in \mathcal{P}(\mathcal{R})$. ∎

There are similar variants for the other definitions considered in this section.

The second extension is a strengthening of f-C-secrecy. To motivate it, consider our definition of synchronous secrecy. While it seems like a reasonable notion, it may seem that in some sense it goes too far in weakening total secrecy. Informally, j maintains *total* secrecy with respect to i if i never learns anything not only about j's current state, but also his possible future and future states. Synchronous secrecy seems to say only that i never learns anything about j's state *at the current time*. More generally, this is true of the definition of f-C-secrecy. The following strengthening of f-C secrecy addresses this concern.

Definition 3.8: If C is an i-allowability function, and f is a j-information function, then j *maintains strong f-C-secrecy with respect to i* if j maintains f-C-secrecy with respect to i, and if for all points (r, m) and values $v \in f(\mathcal{P}(\mathcal{R}) - C(r, m))$, there exists a run r'' and times m_1 and m_2 such that $(r'', m_1) \in \mathcal{K}_i(r, m)$ and $(r'', m_2) \in f^{-1}(v)$. ∎

Informally, strong f-C-secrecy says that i will never learn anything regarding f's behavior in the past, present, or future. Although strong f-C-secrecy may seem to be a significant strengthening of f-C-secrecy, in many cases of interest, the two notions are equivalent.

To make this precise, we need two definitions.

Definition 3.9: An allowability function C *depends only on timing* if (a) for all points (r, m) and runs r' there exists a time m' such that $(r, m) \in C(r', m')$, (b) for all points (r, m) and runs r' there exists a time m' such that $(r', m') \in C(r, m)$, and (c) if $(r'_1, m'_1) \in C(r_1, m_1)$, $(r'_1, m'_2) \in C(r_1, m_2)$, $m_1 \leq m_2$ and $m'_2 \leq m'_1$, then $C(r_1, m_1) = C(r_1, m_2)$. ∎

Clause (c) in the definition says that allowability functions cannot "cross". Clearly, both the synchronous allowability function S and the semisynchronous allowability function *SS* depend only on timing.

Definition 3.10: A j-information function f *preserves perfect recall* if $f(r, m_1) = f(r', m'_1)$, then for all $m_2 \leq m_1$, there exists $m'_2 \leq m'_1$ such that $f(r, m_2) = f(r', m'_2)$. ∎

Proposition 3.11: *If $\mathcal{R}$ is a system where i has perfect recall, the j-information function f preserves perfect recall, and C depends only on timing, then j maintains f-C-secrecy with respect to i iff j maintains strong f-C-secrecy with respect to i.*

The fact that strong f-C- and f-C-secrecy coincide, at least in the case of allowability functions that depend on timing and for "reasonable" choices of f, lends support to the naturalness of the notion of f-C-secrecy. We do not have good examples of natural classes of allowability functions that do not depend on timing, so we cannot judge whether, in general, strong f-C-secrecy is the more appropriate notion.

We conclude this section by examining the extent to which f-C-secrecy can be captured in Sutherland's original framework. The notion of total f-secrecy is clearly a special case of Sutherland's notion of nondeducibility. We suggested earlier that f-C-secrecy is a generalization that cannot be captured in Sutherland's framework in any obvious way. However, as we now show, it can in a precise sense be captured in his framework, albeit in an awkward way.

Given a system $\mathcal{R}$, an allowability function C, and a j-information function f, we can show that there exists a set of worlds $W_{\mathcal{R},C}$ (depending only on $\mathcal{R}$ and C) and information functions g and h on $W_{\mathcal{R},C}$ such that j maintains f-C-secrecy in $\mathcal{R}$ if and only if no information flows from g to h in $W_{\mathcal{R},C}$. Actually, this result is almost trivially true. We can take a fixed set of worlds W' and two pairs of information functions (g, h) and (g', h') on W', such that no information flows from g to h and information does flow from g' to h'. We then map $(\mathcal{R}, C, f)$ to (W', g, h) if j preserves f-C-secrecy in R; otherwise we map it to (W', g', h'). To make this statement nontrivial, we require that the mappings satisfy a uniformity condition. As the following result shows, this too can be done.

Proposition 3.12: *Given a system $\mathcal{R}$, an i-allowability function C, and a j-information function f, there exists a set $W_{\mathcal{R},C}$ of worlds and information functions g and h on $W_{\mathcal{R},C}$ such that j maintains f-C-secrecy with respect to i in $\mathcal{R}$ iff information does not flow from g to h in $W_{\mathcal{R},C}$. Moreover, the mapping from $(\mathcal{R}, C, f)$ to $(W_{\mathcal{R},C}, g, h)$ is uniform in the sense that if $\mathcal{R}' \subseteq \mathcal{R}$, and f' and C' are the restrictions of f and C to $\mathcal{R}'$, then $W_{\mathcal{R}',C'} \subseteq W_{\mathcal{R},C}$ and j maintains f'-C' secrecy with respect to i in $\mathcal{R}'$ iff no information flows from g' to h' in $W_{\mathcal{R}',C'}$, where g' and h' are the restrictions of g and h to $W_{\mathcal{R}',C'}$.*

Proof: Let

$$W_{\mathcal{R},C} = \{((r', m'), C(r, m)) : (r', m') \in C(r, m)\},$$

let $g((r', m'), C(r, m)) = (r'_i(m), C(r, m))$ and let $h((r', m'), C(r, m)) = (f(r, m), C(r, m))$. It is straightforward to check that j maintains f-C-secrecy with respect to i in $\mathcal{R}$ iff information does not flow from g to h in $W_{\mathcal{R},C}$, and that the map is uniform as desired. We leave details to the reader. ∎

4 Knowledge and Secrecy

Our definition of secrecy is semantic; it is given in terms of the local states of the agents. As we shall see, it is helpful to reason syntactically about secrecy, using the logic of knowledge discussed in Section 2. Our goal in this section is to characterize secrecy in terms of the knowledge of the agent. To this end, we show that the state of an agent j is kept secret from an agent i exactly if i does not know any formulas that depend only on the state of j.

For this characterization, we use the modal logic of knowledge described in Section 2. But first, we need two definitions. The first one tells us when a formula depends on the local state of a particular agent. Given an agent j, a formula φ is *j-local* in an interpreted system $\mathcal{I}$ if, for all points (r,m) and (r',m') such that $r_j(m) = r'_j(m')$, $(\mathcal{I},r,m) \models \varphi$ iff $(\mathcal{I},r',m') \models \varphi$. It is easy to check that φ is j-local in $\mathcal{I}$ iff $\mathcal{I} \models K_j\varphi \vee K_j\neg\varphi$. Analogously, given a j-information function f, a formula φ is *f-local* if for all points (r,m) and (r',m') such that $f(r,m) = f(r',m')$, we have that $(\mathcal{I},r,m) \models \varphi$ iff $(\mathcal{I},r',m') \models \varphi$. (See [3] for an introduction to the logic of local propositions.)

The second definition describes when a formula is trivial, in the sense that it would be impossible to forbid an agent from knowing it. A formula φ is *nontrivial* in $\mathcal{I}$ if there exists a point (r,m) such that $(\mathcal{I},r,m) \not\models \varphi$. Clearly if a formula is trivial (i.e., not nontrivial), then every agent will always know it. We cannot expect to prevent agents from knowing trivial facts. More generally, if A is a set of points, φ is *A-nontrivial in $\mathcal{I}$* if there is some point $(r,m) \in A$ such that $(\mathcal{I},r,m) \models \neg\varphi$. We are interested in formulas that are $C(r,m)$-nontrivial. To see why, note that in a synchronous system, a formula such as "the current time is 5" is known by every agent at time 5. Thus if S is the synchronous allowability function defined in Section 3, such a formula is not $S(r,5)$-nontrivial for any run r. However, as the following theorem shows, if φ is $C(r,m)$-nontrivial and f-local, then i will not know φ at the point (r,m) if j maintains f-C-secrecy with respect to i.

Theorem 4.1: *Suppose $\mathcal{R}$ is a system, C is an i-allowability function, and f is a j-information function. Agent j maintains f-C-secrecy with respect to agent i in $\mathcal{R}$ iff for every interpretation π, if φ is f-local and $C(r,m)$-nontrivial in $\mathcal{I} = (\mathcal{R},\pi)$, then $(\mathcal{I},r,m) \models \neg K_i\varphi$.*

Proof: Suppose that j maintains f-C-secrecy with respect to i in $\mathcal{R}$. Let π be an interpretation, let (r,m) be a point in $\mathcal{P}(\mathcal{R})$, and let φ be a formula that is f-local and $C(r,m)$-nontrivial in $\mathcal{I} = (\mathcal{R},\pi)$. Because φ is $C(r,m)$-nontrivial, there exists a point $(r',m') \in C(r,m)$ such that $(\mathcal{I},r',m') \not\models \varphi$. By f-C-secrecy, there exists a point $(r'',m'') \in C(r,m)$ such that $(r'',m'') \in \mathcal{K}_i(r,m) \cap f^{-1}(f(r',m'))$. Because φ is f-local, $(\mathcal{I},r'',m'') \not\models \varphi$. Thus $(\mathcal{I},r,m) \models \neg K_i\varphi$, as required.

For the converse, given $(r,m) \in \mathcal{P}(\mathcal{R})$ and $(r',m') \in C(r,m)$, let

$$E = \{(r'',m'') \in \mathcal{P}(\mathcal{R}) : f(r''_j(m'')) \neq f(r'_j(m'))\}.$$

Let π be such that $\pi(r'',m'')(p) = \mathbf{true}$ iff $(r'',m'') \in E$. (That is, according to π, the primitive proposition p is true at precisely the points in E.) Let $\mathcal{I} = (\mathcal{R},\pi)$. Because $(\mathcal{I},r',m') \models \neg p$, it follows that p is $C(r,m)$-nontrivial. Moreover, it is clear from the definition that p is f-local. By assumption, $(\mathcal{I},r,m) \models \neg K_i p$. Thus, there exists some point $(r'',m'') \notin E$ such that $(r'',m'') \in \mathcal{K}_i(r,m)$. Because $(r'',m'') \in \mathcal{K}_i(r,m)$, it follows that $(r'',m'') \in C(r,m)$, and since $(r'',m'') \notin E$, it must be the case that $f(r'',m'') = f(r'_j,m')$. Thus, j maintains f-C-secrecy with respect to i in $\mathcal{R}$. ∎

It follows from Theorem 4.1 that total secrecy and synchronous secrecy have particularly elegant syntactic characterizations, using the logic of knowledge. The following corollary deals with total secrecy.

Corollary 4.2: *In a system $\mathcal{R}$, the following are equivalent:*

(a) agent j maintains total f-secrecy with respect to agent i;

(b) for every interpretation π, if the formula φ is f-local and nontrivial in $\mathcal{I} = (\mathcal{R},\pi)$, then $\mathcal{I} \models \neg K_i\varphi$.

Proof: This follows immediately from Theorem 4.1, if we let $C(r,m) = \mathcal{P}(\mathcal{R})$ for all points $(r,m) \in \mathcal{P}(\mathcal{R})$. ∎

A similar result holds for synchronous secrecy. In synchronous systems, however, we must exclude formulas that are trivial at any time step, since by synchrony i would know any such formula. Formally, we say that a formula φ is *nontrivial at every time step* in an interpreted system $\mathcal{I}$ if for every time m there exists a run r such that $(\mathcal{I},r,m) \not\models \varphi$.

Corollary 4.3: *In any system $\mathcal{R}$, the following are equivalent:*

(a) Agent j maintains f-S-secrecy with respect to agent i;

(b) for every interpretation π, if the formula φ is f-local and nontrivial at each time step in $\mathcal{I} = (\mathcal{R},\pi)$, then $\mathcal{I} \models \neg K_i\varphi$.

Proof: Consider the synchronous allowability function S, defined in Section 3. A formula φ is nontrivial at every time step if and only if it is $S(r,m)$-nontrivial at every point (r,m). The corollary then follows from Theorem 4.1. ∎

These results suggest that rather than defining secrecy in semantic terms—a requirement on all points that an agent considers possible—it may be better to define it in syntactic terms, by specifying what formulas an agent is allowed to know at any point. Theorem 4.1 shows that we can recover the semantic definition syntactically. Moreover, the syntactic definition has a number of advantages. For one thing, it allows model checking for secrecy (at least, for finite-state systems). Second, by specifying a noninterference property as a set of formulas that must be kept secret, it becomes easier to specify exactly what matters. For example, a system designer may not care that i not know all the formulas that f-C-secrecy requires i not to know. It may be enough that i does not know a small subset of them. This is particularly relevant for declassification or downgrading. If a noninterference property corresponds to a set of formulas that must be kept secret from the low-level agent, formulas can be declassified by removing them the set of secret formulas. Third, recall that the definition of knowledge described in Section 2 suffers from the *logical omniscience* problem: agents know all tautologies and know all logical consequences of their knowledge [4]. In the context of security, we are more interested in what *resource-bounded* agents know. It does not matter that an agent with unbounded computational resources can factor and decrypt a message. Rather, what matters is that a resource-bounded agent cannot. Representing secrecy in terms of knowledge allows us to take advantage of work on "algorithmic knowledge" to reason about noninterference where agents are assumed to have limited computational power [4, 10].

5 Probabilistic Secrecy

The definitions of secrecy that we have considered up to now have a major theoretical flaw: they are concerned only with *possibilistic*, rather than *probabilistic* events. Suppose that agent j maintains total secrecy with respect to agent i. That means that $\mathcal{K}_i(r,m) \cap \mathcal{K}_j(r',m') \neq \emptyset$ for all points (r,m) and (r,m'). However, it is completely consistent with total secrecy that, according to agent i, the probability of $\mathcal{K}_j(r,0)$ is .1 at the point $(r,0)$, and 0.99 at the point $(r,1)$. Certainly in this case there has been some loss of secrecy. We want a definition that captures this. As we now show, there are straightforward modifications of our earlier definitions that capture probabilistic secrecy.

To start with, given a system $\mathcal{R}$, suppose that there is a probability measure μ on $\mathcal{R}$. We call the pair $(\mathcal{R},\mu)$ a *probabilistic system*. For simplicity, assume that every subset of $\mathcal{R}$ is measurable (i.e., can be assigned a probability).We are interested in defining the probability that agent i assigns to an event at a point (r,m). We do this by essentially conditioning the probability μ on runs on $\mathcal{K}_i(r,m)$, the information that the agent has at the point (r,m). The problem is that $\mathcal{K}_i(r,m)$ is a set of *points*, not a set of *runs*, so we cannot condition μ on $\mathcal{K}_i(r,m)$. To capture this intuition, we follow the approach of Halpern and Tuttle [11].

Given a set U of points, let $\mathcal{R}(U)$ consist of the runs in $\mathcal{R}$ going through a point in U. That is,

$$\mathcal{R}(U) = \{r \in \mathcal{R} : (r,m) \in U \text{ for some } m\}.$$

The idea will be to condition μ on $\mathcal{R}(\mathcal{K}_i(r,m))$ rather than on $\mathcal{K}_i(r,m)$. To make sure that conditioning is well defined, we assume that $\mu(\mathcal{R}(\mathcal{K}_i(r,m))) > 0$ for each agent i, run r, and time m. That is, μ assigns positive probability to the set of runs in $\mathcal{R}$ compatible with what happens in run r up to time m, as far as agent i is concerned.

With this assumption, we can define a measure $\mu_{r,m,i}$ on the points in $\mathcal{K}_i(r,m)$ as follows. If $\mathcal{S} \subseteq \mathcal{R}$, define $\mathcal{K}_i(r,m)(\mathcal{S})$ to be the set of points in $\mathcal{K}_i(r,m)$ that lie on runs in $\mathcal{S}$; that is,

$$\mathcal{K}_i(r,m)(\mathcal{S}) = \{(r',m') \in \mathcal{K}_i(r,m) : r' \in \mathcal{S}\}.$$

Let $\mathcal{F}_{r,m,i}$, the measurable subsets of $\mathcal{K}_i(r,m)$ (that is, the sets to which $\mu_{r,m,i}$ assigns a probability), consist of all sets of the form $\mathcal{K}_i(r,m)(\mathcal{S})$, where $\mathcal{S} \subseteq \mathcal{R}$. Then define $\mu_{r,m,i}(\mathcal{K}_i(r,m)(\mathcal{S})) = \mu(\mathcal{S} \mid \mathcal{R}(\mathcal{K}_i(r,m)))$. It is easy to check that $\mu_{r,m,i}$ is a probability measure, essentially defined by conditioning.

The (possibilistic) definition of total secrecy requires that $\mathcal{K}_i(r,m) \cap \mathcal{K}_j(r',m') \neq \emptyset$ for all points (r,m) and (r,m'). If we assume that all runs get positive probability, then this amounts to saying that

$$\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \mid \mathcal{R}(K_j(r',m'))) > 0$$

or, equivalently, that

$$\mu(\mathcal{R}(\mathcal{K}_j(r',m')) \mid \mathcal{R}(K_i(r,m))) > 0$$

for all points (r,m) and (r',m'). Our definition of probabilistic secrecy requires that all these probabilities are the same, not just positive. Intuitively, it says that i never learns anything about the probability of j being in a particular local state. If i is the low-level agent and j is the high-level agent, this just says that the low-level agent's view of the system should not change her probability of some high-level event occurring.

Definition 5.1: Agent j maintains *total probabilistic secrecy* with respect to i in system $(\mathcal{R},\mu)$ if for any three points (r,m), (r',m'), and (r'',m''), we have

$$\mu_{r,m,i}(\mathcal{K}_j(r'',m'')) = \mu_{r',m',i}(\mathcal{K}_j(r'',m'')).$$

∎

It is immediate from Definition 3.1 that total (possibilistic) secrecy is symmetric. As the following proposition shows, total probabilistic secrecy is also symmetric. Moreover, it shows that total probabilistic secrecy amounts to i's information being (probabilistically) independent of j's information.

Proposition 5.2: *Given a probabilistic system $(\mathcal{R},\mu)$, the following are equivalent:*

(a) *agent j maintains total probabilistic secrecy with respect to i;*

(b) *agent i maintains total probabilistic secrecy with respect to j;*

(c) *for all points (r, m) and (r', m') the events $\mathcal{R}(\mathcal{K}_i(r, m))$ and $\mathcal{R}(\mathcal{K}_j(r', m'))$ are independent.*

Our definition of probabilistic secrecy can easily be extended to yield probabilistic f-secrecy.

Definition 5.3: If f is a j-information function, then agent j maintains *probabilistic f-secrecy with respect to i* in system $(\mathcal{R}, \mu)$ if for any points (r, m), (r', m'), and values v of f, we have $\mu_{r,m,i}(f^{-1}(v)) = \mu_{r',m',i}(f^{-1}(v))$. ∎

Probabilistic f-secrecy is also equivalent to probabilistic independence.

Proposition 5.4: *Given a probabilistic system $(\mathcal{R}, \mu)$, the following are equivalent:*

(a) *agent j maintains probabilistic f-secrecy with respect to i in $(\mathcal{R}, \mu)$;*

(b) *for all values $v, v' \in f(\mathcal{P}(\mathcal{R}))$ and points (r, m), we have $\mu(\mathcal{R}(\mathcal{K}_i(r, m)) \mid \mathcal{R}(f^{-1}(v)) = \mu(\mathcal{R}(\mathcal{K}_i(r, m)) \mid \mathcal{R}(f^{-1}(v'))$;*

(c) *for all points (r, m) and values v of f, the events $\mathcal{R}(\mathcal{K}_i(r, m))$ and $\mathcal{R}(f^{-1}(v))$ are independent.*

The extension to probabilistic f-C-secrecy is also immediate.

Definition 5.5: If C is an i-allowability function and f is a j-information function, then *j maintains probabilistic f-C-secrecy with respect to i* in system $(\mathcal{R}, \mu)$ if, for all points (r, m) and (r', m') and values $v \in f(C(r, m))$, we have $\mu_{r,m,i}(f^{-1}(v)) = \mu_{r',m',i}(f^{-1}(v))$. ∎

Up to now, we have assumed that in a probabilistic system, there is a single probability measure on all the runs. As argued by Halpern and Tuttle [11], in many systems of interest, it is not reasonable to assume a probability distribution on the set of all runs. In practice, this is because some choices in the system are best viewed as being made nondeterministically. The standard approach to dealing with this (implicitly taken, for example, when proving properties of randomized algorithms) is to "factor out" the nondeterministic choice. The effect of this is to partition the runs in $\mathcal{R}$ into cells, with one cell corresponding to each nondeterministic choice. There is then a separate probability measure on each cell. For example, suppose that agent 1 is using one of a set $\{P_1, \ldots, P_m\}$ of (possibly probabilistic) protocols, and agent 2 is using one of a set $\{Q_1, \ldots, Q_n\}$ of protocols. Then $\mathcal{R}$ consists of all the runs generated by running (P_k, Q_l) for $k = 1, \ldots, m$ and $l = 1, \ldots, n$. The partition in this case consists of mn cells, one for each joint protocol (P_k, Q_l). We do not want to assume a probability on how likely agent 1 is to use protocol P_k or agent 2 is to use protocol Q_l. However, for each fixed k and l, there is a natural probability on the runs generated by (P_k, Q_l). (We remark that this is actually precisely the situation considered by Gray and Syverson [8] in their definition of probabilistic noninterference; we discuss their work in more detail in Section 6.3.)

Formally, assume that we are given a partition $\mathcal{D}$ of $\mathcal{R}$, and a set $\Delta = \{\mu_D : D \in \mathcal{D}\}$ of probability measures, where μ_D is a probability measure on the runs in $D \in \mathcal{D}$. Let $D = D(r)$ be the unique cell in $\mathcal{D}$ that contains the run r. There are now two ways to proceed. The first approach essentially views the measures in Δ as constraints on a single measure on $\mathcal{R}$. That is, it considers all the probability measures μ on $\mathcal{R}$ that, when conditioned on $D \in \mathcal{D}$, give μ_D, for all $D \in \mathcal{D}$. Then probabilistic security with respect to $(\mathcal{R}, \Delta)$ is reduced to probabilistic security with respect to $(\mathcal{R}, \mu)$. The second approach just considers the measures in μ_D separately, without trying to combine them into a single measure on $\mathcal{R}$.

For the first approach, say that a probability measure μ on $\mathcal{R}$ is *compatible with* Δ if for each cell $D \in \mathcal{D}$, we have that $\mu|D = \mu_D$ (that is, μ_D is the result of conditioning μ on D) if $\mu(D) \neq 0$. If $\mathcal{D}$ is countable (which is the only case in which we are interested), then it is easy to see that μ is compatible with Δ iff there is a probability ν on $\mathcal{D}$ such that, for all $D \in \mathcal{D}$, if $\mathcal{S} \subseteq D$, then $\mu(\mathcal{S}) = \nu(D)\mu_D(\mathcal{S})$.

Definition 5.6 : Agent j maintains *probabilistic f-C-secrecy* with respect to i in system $(\mathcal{R}, \Delta)$ if, for every μ compatible with Δ, j maintains total probabilistic f-C-secrecy with respect to i in $(\mathcal{R}, \mu)$. ∎

The second approach is essentially that taken by Halpern and Tuttle [11]. Rather than taking $\mu_{r,m,i}$ to be a probability on all the points in $\mathcal{K}_i(r, m)$, we now take it to be a probability on the points in $D_{r,m,i} = \mathcal{K}_i(r, m)(D(r))$. (Thus, $D_{r,m,i}$ is the set of points in $\mathcal{K}_i(r, m)$ that lie on runs in $D(r)$.) We now define $\mu_{r,m,i}$ by conditioning μ_D on $\mathcal{R}(\mathcal{K}_i(r, m))$. Thus, we must assume that $\mu_{D(r)}(\mathcal{R}(\mathcal{K}_i(r, m))) > 0$ for all points (r, m). Formally, $\mu_{r,m,i}$ is defined so that

$$\mu_{r,m,i}(D_{r,m,i}(\mathcal{S})) = \mu_D(\mathcal{S} \mid \mathcal{R}(\mathcal{K}_i(r, m))).$$

With this modification in the definition of $\mu_{r,m,i}$, all of our earlier definitions remain unchanged. The only difference is that now security is taken with respect to $(\mathcal{R}, \Delta)$, so $\mu_{r,m,i}$ is now computed by conditioning on $D(r)$ rather than on $\mathcal{R}(\mathcal{K}_i(r, m))$.

Definition 5.7 : Agent j maintains *probabilistic f-C-secrecy$'$* with respect to i in system $(\mathcal{R}, \Delta)$ if, for all points (r, m) and (r', m') and values $v \in f(C(r, m))$, we have $\mu_{r,m,i}(f^{-1}(v)) = \mu_{r',m',i}(f^{-1}(v))$. ∎

For the situations considered by Halpern and Tuttle [11], it sufficed to consider the probability measures in Δ separately, rather than trying to combine them. Since the first approach has the extra overhead of needing to extend the probability measures in Δ to a single measure, they focused on the second approach. However, here the choice of approach turns out to make a significant difference; as the notation suggests, we view secrecy (the first approach) as the better definition. The following result supports this intuition.

Proposition 5.8: *If j maintains probabilistic f-C-secrecy$'$ with respect to i in system $(\mathcal{R}, \Delta)$, then j maintains probabilistic f-C-secrecy with respect to i in $(\mathcal{R}, \Delta)$.*

The converse of Proposition 5.8 does not hold in general. In fact, as we show in Section 6.3, Gray and Syverson's probabilistic noninterference is an instance of f-C-secrecy but is not an instance of f-C-secrecy$'$. Of course, f-C-secrecy and f-C-secrecy$'$ coincide in the special case where there is a single probability measure on all of $\mathcal{R}$.

6 Related Work

In this section, we consider how our definition relates to other attempts to define security. We show in particular how it can capture work that has been done in the synchronous setting, the asynchronous setting, and the probabilistic setting. We also discuss its relationship to the process-algebra approach to security.

6.1 Noninterference in Synchronous Trace Systems

Many papers in computer security define noninterference properties using trace-based models. Traces are usually defined as sequences of input and output events, where each event corresponds to some agent. As such, they are special cases of runs. However, there have been some subtle differences among the trace-based models. In some cases, infinite traces are used; in others, only finite traces. In addition, some models assume that the underlying systems are synchronous, while others implicitly assume asynchrony. In this subsection, we consider synchronous trace-based systems. Both McLean [15] and Wittbold and Johnson [23] present their definitions of security within a framework of synchronous input/output traces. These traces are essentially restricted versions of the runs introduced in this paper. Here we introduce a simplified version of the traces presented by McLean [15] and describe two well-known noninterference properties within the framework.

Let I be a finite set of input variables and O be a finite set of output variables. A tuple $t = \langle in_L, in_H, out_L, out_H \rangle$ (with $in_L, in_H \in I$ and $out_L, out_H \in O$) represents a snapshot of a system at a given a point in time, that is, the input provided to the system by a low-level agent L and a high-level agent H, as well as the output produced by the system that is viewable to L and H. A trace τ is a sequence of tuples $\tau = \langle t_1, t_2, \ldots \rangle$. It represents an infinite execution sequence of the entire system by describing the input/output behavior of the system at any given point in time. A *synchronous trace system* is a set Σ of traces, representing the possible execution sequences of the system.[1]

To talk about the local state of agents in such trace-based system, we use *trace projection functions*, which take a trace τ and project each state tuple t of τ onto some low-dimensional tuple. For example, if

$$\begin{aligned} \tau \;=\; & \langle \langle (in_L)_1, (in_H)_1, (out_L)_1, (out_H)_1 \rangle, \\ & \langle (in_L)_2, (in_H)_2, (out_L)_2, (out_H)_2 \rangle, \ldots \rangle \end{aligned}$$

and $\mathcal{L}$ is a function projecting τ on to the low-level input/output behavior of τ, then

$$\mathcal{L}(\tau) = \langle \langle (in_L)_1, (out_L)_1 \rangle, \langle (in_L)_2, (out_L)_2 \rangle, \ldots \rangle.$$

Similarly, we can extract high-level input/output traces with a function $\mathcal{H}$, or even just high-level *input* traces with a function $\mathcal{HI}$.

Given a trace $\tau = \langle t_1, t_2, \ldots \rangle$, a *trace prefix* of τ up to some time k is given by $\tau_k = \langle t_1, t_2, \ldots, t_k \rangle$, i.e., the finite sequence containing the first k state tuples of the trace τ. Trace projection functions apply to trace prefixes in the obvious way.

It is easy to see that synchronous trace systems can be viewed as systems in the multiagent systems framework. Given a trace τ, we can define the run r^τ such that $r^\tau(m) = \tau_m$. Thus, we view τ_m as a global state. We take $r^\tau_L(m) = \mathcal{L}(\tau_m)$ and $r^\tau_H(m) = \mathcal{H}(\tau_m)$. Note that there is no environment state here. (We can take it to be empty.) Given a synchronous trace system Σ, let $\mathcal{R}(\Sigma) = \{r^\tau : \tau \in \Sigma\}$. It is easy to check that $R(\Sigma)$ is synchronous, and that both agents L and H have perfect recall.

McLean describes how several definitions of noninterference can be described using his framework. We consider two of the best known here: *separability* [15] and *generalized noninterference* [13]. Separability, as its name suggests, ensures secrecy between the low-level and high-level agents, whereas generalized noninterference ensures

[1] The traces are said to be synchronous because the input and output values are specified for each agent at each time step, and both agents can infer the time simply by looking at the number of system outputs they have seen.

that the low-level agent is unable to know anything about high-level input behavior.

Definition 6.1: A synchronous trace system Σ satisfies *separability* if for every pair of traces $\tau, \tau' \in \Sigma$, there exists a trace $\tau'' \in \Sigma$ such that $\mathcal{L}(\tau'') = \mathcal{L}(\tau)$ and $\mathcal{H}(\tau'') = \mathcal{H}(\tau')$. ∎

Definition 6.2: A synchronous trace system Σ satisfies *generalized noninterference* if for every pair of traces $\tau, \tau' \in \Sigma$, there exists a trace $\tau'' \in \Sigma$ such that $\mathcal{L}(\tau'') = \mathcal{L}(\tau)$ and $\mathcal{HI}(\tau'') = \mathcal{HI}(\tau')$. ∎

These definitions are both special cases of nondeducibility, as discussed in Section 3. Take the set of worlds W to be Σ, the information function g to be $\mathcal{L}$, and the information function h to be $\mathcal{H}$ (for separability) and $\mathcal{HI}$ (for generalized noninterference).[2]

So how do these definitions relate to our notions? Separability essentially corresponds to synchronous secrecy, whereas generalized noninterference corresponds to f_{hi}-S secrecy, where f_{hi} maps a trace to the high input. We start with separability. Showing that separability in synchronous trace systems implies synchronous secrecy in the corresponding system is easy.

Proposition 6.3: *If a synchronous trace system Σ satisfies separability, then H maintains synchronous secrecy with respect to L in $\mathcal{R}(\Sigma)$.*

Proof: Suppose that Σ satisfies separability. Let r^τ and $r^{\tau'}$ be runs in $\mathcal{R}(\Sigma)$. We want to show that, for all time m, we have that $\mathcal{K}_L(r^\tau, m) \cap \mathcal{K}_H(r^{\tau'}, m) \neq \emptyset$. By separability there exists a trace $\tau'' \in \Sigma$ such that $\mathcal{L}(\tau'') = \mathcal{L}(\tau)$ and $\mathcal{H}(\tau'') = \mathcal{H}(\tau')$. It follows immediately that $\mathcal{L}(\tau''_m) = \mathcal{L}(\tau_m)$ and $\mathcal{H}(\tau''_m) = \mathcal{H}(\tau'_m)$. Thus, $(r^{\tau''}, m) \in \mathcal{K}_L(r^\tau, m) \cap \mathcal{K}_H(r^{\tau'}, m)$. ∎

The converse to Proposition 6.3 is not quite true. There is a subtle but significant difference between McLean's framework and ours. McLean works with *infinite* traces; separability and generalized noninterference are defined as closure properties on sets of *traces* rather than sets of *points* (or trace prefixes). To see the impact of this, consider a system Σ where the high-level agent inputs either infinitely many 0's or infinitely many 1's. The output to the low-level agent is always finitely many 0's followed by infinitely 1's, except for a single trace where the high-level agent inputs infinitely many 0's and the low-level agent inputs infinitely many 0's. (Assume that the high-level output and the low-level input are constant; they are unimportant.) The following table (quantified over all possible vales of k) gives a simplified representation of this system:

L	H
$0^k 1^\infty$	0^∞
$0^k 1^\infty$	1^∞
0^∞	0^∞

This system satisfies synchronous separability (and also synchronous generalized noninterference) because by looking at any finite trace prefix, the low-level agent cannot tell whether the high-level inputs have been 0's or 1's. However, if the low-level agent "sees" infinitely many 0's, he immediately knows that the high-level inputs have been 0's. For this reason, the system does not satisfy separability or generalized noninterference. The problem, of course, is that the agent will never see infinitely many 0's. After all, the agent only makes observations at finite times! The notion of separability with infinite traces seems to be placing too strong a demand on secrecy.

Note that if τ is a trace where the low-level outputs are all 0's and the high-level inputs are all 1's, each finite prefix of the trace τ is a prefix of a trace in Σ, even though τ is not in Σ. This turns out to be the key reason that the system satisfies separability but not synchronous satisfiability.

Definition 6.4: A synchronous trace system Σ is *limit closed* [2] if, for all traces τ, $\tau \in \Sigma$ iff for every time k there exists a trace $\tau' \in \Sigma$ such that $\tau'_k = \tau_k$. ∎

Under the assumption of limit closure, we do get the converse to Proposition 6.3.

Proposition 6.5: *A limit-closed synchronous trace system Σ satisfies separability iff H maintains synchronous secrecy with respect to L in $\mathcal{R}(\Sigma)$.*

The following results are the analogue to Propositions 6.3 and 6.5 for generalized noninterference.

Proposition 6.6: *If a synchronous trace system Σ satisfies generalized noninterference, then H maintains f_{hi}-S-secrecy with respect to L in $\mathcal{R}(\Sigma)$.*

Proposition 6.7: *A limit-closed synchronous trace system Σ satisfies generalized noninterference iff H maintains f_{hi}-S-secrecy with respect to L in $\mathcal{R}(\Sigma)$.*

These results demonstrate the close connection between our definition of synchronous f-secrecy and security properties such as separability and generalized noninterference that can be specified as closure properties on infinite input/output traces. By the results of Section 4, this shows that such properties are also closely related to knowledge. A system satisfies separability with respect to two agents i and j, for example, if i never knows any facts about the system that depend only on the input/output behavior of j, and vice-versa.

[2] Actually, it is not difficult to see that if the information functions g and h are restricted to trace projection functions, then nondeducibility is essentially equivalent in expressive power to *selective interleaving functions*, the mechanism for defining security properties introduced by McLean [15].

6.2 Nondeducibility on Strategies

Generalized noninterference, within the context of McLean's synchronous input/output traces, captures the intuition that the low-level agent cannot rule out any high-level input traces. But is protecting the input of the high-level agent enough to guarantee secrecy? There is a sense in which it is not. Consider the following system, a simplified version of one described by Wittbold and Johnson [23] and Gray and Syverson [8].

- All input/output values are restricted to be either 0 or 1.
- At each time step i, the high-level output value $(out_H)_i$ is nondeterministically chosen to be either 0 or 1.
- At each time step i, the low-level output value $(out_L)_i$ is $(out_H)_{i-1} \oplus (in_H)_i$, where $\oplus$ is the exclusive-or operator and $(in_H)_i$ is the high-level input at time i.
- For completeness, suppose that the low-level output at time 1 is 0, since $(out_H)_{i-1}$ is not defined at the first time step. (What happens at the first time step is unimportant.)

The set of traces that represents this system satisfies noninterference. At any time i, $(out_L)_i$ depends only on $(in_H)_i$ and $(out_H)_{i-1}$. But as the following table shows, any value of $(out_L)_i$ is consistent with any value of $(in_H)_i$:

$(in_H)_i$	$(out_H)_{i-1}$	$(out_L)_i$
0	0	0
0	1	1
1	0	1
1	1	0

So for any two traces τ, τ', we can construct a new trace τ'' by taking the low-level input/output of τ and the high-level input of τ'. For the high-level output of the resulting trace, we take $(out_H)_i = (out_L)_{i+1} \oplus (in_L)_{i+1}$. It should be clear that τ'' is a valid trace of the system; thus, the system satisfies generalized noninterference.

The problem with this system is that a malicious high-level agent (for example, a "Trojan Horse" program) who knows how the system works can transmit arbitrary strings of data directly to the low-level agent. That is, if the high-level agent wants to transmit the bit x at time k, and sees the high-level output bit y at time $k-1$, she can ensure that the *low-level* output is x at time k by inputting the bit $x \oplus y$ at time $k+1$.

Wittbold and Johnson [23] correctly point out that generalized noninterference is inherently insecure. We claim that the fundamental problem is not with generalized noninterference *per se*, but rather with an underlying system model that assumes that everything relevant to the state of the high-level agent can be encapsulated using input/output traces. If part of the high-level agent's state includes some string, that string should be part of the state of the agent. Similarly, if the agent has a protocol that involves sending a string, the protocol should be represented in the agent's local state. When a multiagent system is represented using only input/output traces, there is no way to do this. It is easy to show that if the input string or protocol is represented as part of H's local state, then the desired secrecy property is easy to model using our definition.

Wittbold and Johnson introduce *nondeducibility on strategies* (NOS) to capture this intuition. (Gray and Syverson's *probabilistic noninterference* is a probabilistic generalization of nondeducibility on strategies; see Section 6.3.) We modify the definition of nondeducibility on strategies slightly so that it fits into McLean's framework. A system satisfies nondeducibility on strategies if for all traces $\tau \in \Sigma$ and all high-level strategies $\mathbf{H}$ consistent with some trace in Σ, there exists a trace τ' that is consistent with $\mathbf{H}$ and satisfies $\mathcal{L}(\tau') = \mathcal{L}(\tau)$. (Here a high-level strategy $\mathbf{H}$ can be seen as a function from high-level input/output trace prefixes to high-level input values.) Nondeducibility on strategies prevents the sort of problem described above.

To capture NOS is our framework is completely straightforward. If the strategy of the high-level agent is included as part of her local state, and f_{strat} is an H-information function that extracts the strategy of the high-level agent from any such local state, then it is almost trivial to show that NOS is just f_{strat}-S-secrecy.[3] We remark that, given that an agent's local state is intended to include all (the relevant parts of) the agent's knowledge, it seems rather natural to include the agent's strategy in its state. After all, the agent must know her strategy (otherwise the whole notion of NOS seems somewhat meaningless).

The results of Section 4 demonstrate that NOS, like separability and generalized noninterference, are closely related to knowledge. Nondeducibility on strategies means, in a precise sense, that low-level agents will never know (or learn) anything about the strategy of the high-level agent.

In [18] Roscoe identifies two scenarios where secrecy requirements can be violated: (1) the low-level agent is a spy who is trying to learn about the high-level agent's behavior without his knowledge; (2) the high-level agent is a "mole" who is trying to pass information to the low-level agent, possibly using some prearranged scheme for representing information. While we agree that there are quite different intuitions involved here, our framework accommodates both scenarios. The focus of our definitions has been on the first scenario. But we claim that NOS essentially cap-

[3] In fact, if we make the reasonable assumption that H's strategy is the same at every point of a run, so that it does not depend on time, NOS is also a special case of total f_{strat}-secrecy.

tures the second scenario. In order for the high-level agent to pass information to the low-level agent, there must be some correlation between the high-level agent's actions and the low-level agent's actions. NOS says that, in a precise sense, there is not.

6.3 Probabilistic Noninterference

Gray and Syverson [8] present a probabilistic version of noninterference. Their system model is described in terms of synchronous input/output traces, exactly in the spirit of the synchronous trace systems defined in Section 6.1. They assume low-level and high-level agents that use probabilistic strategies $\mathbf{L}$ and $\mathbf{H}$, respectively. In their terminology, an *adversary* $\mathbf{A}$ is a pair $(\mathbf{L}, \mathbf{H})$; the choice of an adversary factors out all the nondeterminism in the sense described in Section 5. Each adversary $\mathbf{A}$ determines a probability distribution on the runs generated by the joint strategy $\mathbf{A}$.

In our notation, Gray and Syverson's notion of probabilistic noninterference can essentially be captured by the following definition:

Definition 6.8: A system $(\mathcal{R}, \Delta)$, where Δ consists of the probability measures $\mu_{\mathbf{A}}$ for all adversaries $\mathbf{A}$, satisfies *probabilistic noninterference* if, for all low-level strategies $\mathbf{L}$, high-level strategies $\mathbf{H}$ and $\mathbf{H}'$, and points (r, m), we have $\mu_{(\mathbf{L},\mathbf{H})}(\mathcal{K}_L(r, m)) = \mu_{(\mathbf{L},\mathbf{H}')}(\mathcal{K}_L(r, m))$. ∎

As the following result shows, probabilistic noninterference is a special case of our definition of probabilistic f-C-secrecy. Assume, as we did in Section 6.2, that the high-level strategies are included in the local state of the high-level user, and consider the information function f_{strat} from Section 6.2 that extracts the high-level strategy from each point.

Theorem 6.9: *$(\mathcal{R}, \Delta)$ satisfies probabilistic noninterference iff H maintains probabilistic f_{strat}-S-secrecy with respect to L.*

Proof: (Sketch.) Suppose that H maintains probabilistic f_{strat}-S-secrecy with respect to L in $(\mathcal{R}, \Delta)$. Then for all runs $r, r' \in \mathcal{R}$, times m, and probability measures μ compatible with the probabilities $\mu_{\mathbf{A}}$, we have that

$$\mu_{r,m,L}(f_{strat}^{-1}(\mathbf{H})) = \mu_{r',m,L}(f_{strat}^{-1}(\mathbf{H})).$$

It follows from Proposition 5.4 that for all high-level strategies $\mathbf{H}$ and $\mathbf{H}'$ and points (r, m), we have that $\mu(\mathcal{K}_L(r, m)) \,|\, \mathcal{R}(\mathbf{H})) = \mu(\mathcal{K}_L(r, m) \,|\, \mathcal{R}(\mathbf{H}'))$ (where $\mathcal{R}(\mathbf{H})$ consists of the runs where H uses strategy $\mathbf{H}$). ¿From this it follows that $\mu(\mathcal{K}_L(r, m)) \,|\, \mathcal{R}((\mathbf{L}, \mathbf{H})) = \mu(\mathcal{K}_L(r, m) \,|\, \mathcal{R}((\mathbf{L}, \mathbf{H}'))$ for every low-level strategy $\mathbf{L}$. Since μ is compatible with $\mu_{\mathbf{L},\mathbf{H}}$, it follows that $\mu_{(\mathbf{L},\mathbf{H})}(\mathcal{K}_L(r, m)) = \mu_{(\mathbf{L},\mathbf{H}')}(\mathcal{K}_L(r, m))$. Hence probabilistic f_{strat}-S-secrecy implies Syverson and Gray's probabilistic noninterference. The converse follows from another application of Proposition 5.4. We leave details to the full paper. ∎

Note that we used f_{strat}-S-secrecy, not f_{strat}-S-secrecy$'$, in Theorem 6.9. Indeed, it is easy to see that f_{strat}-S-secrecy$'$ does not hold in Syverson and Gray's setting. The problem is that then $\mu_{r,m,L}$ is defined by conditioning $\mu_{\mathbf{H_r},\mathbf{L_r}}$ on $\mathcal{R}(\mathcal{K}_L(r, m))$, where $\mathbf{H_r}$ and $\mathbf{L_r}$ are the high- and low-level strategies used in the run r. Thus, $\mu_{r,m,L}(f_{strat}^{-1}(\mathbf{H}))$ is either 1 (if $\mathbf{H} = \mathbf{H}_r$) or 0 (otherwise). It follows that it will not be the case that $\mu_{r,m,L}(f_{strat}^{-1}(\mathbf{H})) = \mu_{r',m,L}(f_{strat}^{-1}(\mathbf{H}))$ for all points (r, m) and (r, m').

6.4 Asynchronous System Models

Zakinthinos and Lee [24] present a general theory of possibilistic security properties. In their framework, traces are not sequences of input/output tuples, but rather sequences of input/output events. Furthermore, traces are finite. If a and b are low-level input events while c and d are high-level input events, a possible system trace could be

$$\tau = \langle a, c, a, d, d, b, b, a, c\rangle.$$

Since traces are sequences of input/output events—so that high-level and low-level events do not occur in lockstep, as in McLean's framework—the underlying system model is asynchronous. A low-level projection function $\mathcal{L}$ simply extracts all the low-level events from a trace. Thus, if τ is defined as above, we have

$$\mathcal{L}(\tau) = \langle a, a, b, b, a\rangle.$$

This means that the local state of a low-level agent might not change even as the high-level agent experiences hundreds or thousands of input events. An *asynchronous trace system* is a set of traces. Clearly we can associate with each such system a set of runs (where the last state gets repeated infinitely often, since our runs are infinite).

The security properties presented by Zakinthinos and Lee are stated in terms of *low-level equivalence sets*, very much in the spirit of nondeducibility. For example, they define separability as the following requirement: for every two traces τ and τ', every possible interleaving τ'' of the subtraces $\mathcal{L}(\tau)$ and $\mathcal{H}(\tau')$ is a valid trace in the system. Here, separability says that the low-level agent is unable to rule out any possible high-level input/output sequences, as in Definition 6.1. Notice that since traces can be of arbitrary length, and since arbitrary interleavings of low-level and high-level traces must be valid traces of the system, the systems considered by Zakinthinos and Lee must be completely asynchronous, so that agents have absolutely no idea of what the time is. As we show in the full paper, this asynchronous notion of separability corresponds to our notion of total secrecy; similarly Zakinthinos and Lee's notion of generalized noninterference corresponds to f_{hi}-secrecy.

6.5 Process Algebras and Noninterference

Recent work in properties related to information flow and noninterference has focused on systems that have been specified using process algebras related to CCS [5] and CSP [19]. (Milner [16] gives an excellent introduction to CCS, while Focardi and Gorrieri [6] provide an overview of recent work on noninterference properties expressed using CCS. Ryan et al. [20] give a thorough introduction to CSP and describe how it can be used to describe a number of security properties.)

Definitions of noninterference properties that are based on process algebras have several important advantages. To begin with, process algebras allow us to describe systems compactly, even if the resulting systems have trace sets that are large or infinite. In addition, the process-algebra approach provides an elegant way of constructing systems compositionally. Finally, the process-algebra approach can express features of systems that cannot be expressed using traces that focus on input/output relations. Moreover, these features can be quite significant in the context of security (see [6, 19]).

With regard to the last point, it may seem that our approach suffers from the limitations of other trace-based approaches, since runs are essentially traces. We show that, by using local states in a natural way, our approach can in fact capture some of the standard examples that are problematic for other trace-based approaches. Consider the following example (essentially discussed in by both Ryan and Schneider [19] and Focardi and Gorrieri [6]). Let E and F be the following two processes:

$$E = a.b.\underline{0} + a.b.\underline{0}$$

and

$$F = a.(b.\underline{0} + c.\underline{0}),$$

represented graphically in Figure 1. These two systems have the same set of input/output traces, namely the set $\{\langle a\rangle, \langle a, b\rangle, \langle a, c\rangle\}$. However, the behavior of the two systems is different. If b and c represent input events, the two systems have different *refusal sets* after executing the trace $\langle a\rangle$. After F executes a, it moves to a state where it can execute the event b *or* the event c, while E, after executing a, moves to a state where it has no choice about the next event it executes (which must be either b or c). These systems are *not* equivalent according to the standard notion of bisimulation-based observational equivalence.

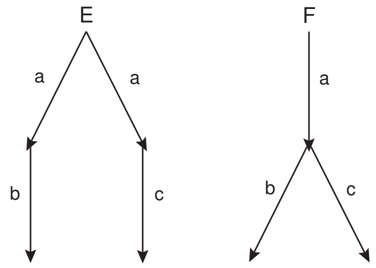

Figure 1. Processes E and F.

There are several ways to capture the processes E and F as systems in our framework. Perhaps the easiest is to assume that the system consists of one agent, who performs the actions a, b, and c. The agent decides (nondeterministically) what protocol she is following. In both cases, there are two runs, one corresponding to the trace $a; b$ and the other corresponding to the trace $a; c$. (There may also be a run or runs corresponding to the trace a if we assume that the system is deadlocked for some reason after the agent performs a. We start by considering the case where there are two traces.) For simplicity, assume that the environment state just records the actions that were taken. (Thus, the environment records the input-output relation.) The agent's nondeterministic choice is reflected in her local state. In the case of E, the agent's initial local state must reflect her choice of whether to run protocol $a; b$ or protocol $a; c$. This choice continues to be reflected in her state at time 1. In the case of process F, at time 0, she does not have a choice to make yet. Her initial local state just reflects the fact that her next action is a. At time 1, she must choose between b and c. Thus, in the system corresponding to E, the agent can distinguish the initial points of the two runs (since she has made different choices); on the other hand, in the system corresponding to F, she cannot distinguish the initial states of the runs. If we also want to consider the trace a, the system corresponding to E has two more runs (depending on the initial choice of the agent); the system corresponding to F has just one more run.

The model just discussed is synchronous. However, we can also take an asynchronous version of the model. In this case, there would be infinitely many runs, corresponding to the different times the actions were performed. We can, of course, also capture other constraints on timing (such as upper bounds on how long it takes to perform actions), which seem to be less easy to model in the process-algebra framework.[4] In any case, the key point is that that there are quite

[4]There are yet other ways of modeling E and F. For example, we could encode the CSP notion of *refusals* by having runs where the agent tries to input b or c and is refused by the system. However, this approach seems less natural in the context of systems.

natural ways to capture E and F as *different* systems.

We in fact conjecture that our framework can capture all the distinctions captured by the process-algebra approach, by appropriately choosing local and global states. (More precisely, we conjecture that there will be canonical translations from the process-algebra approach to the systems approach, where the translation depends on the notion of equivalence under consideration. For example, different types of bisimulation equivalence or testing equivalence will correspond to different translations.) A careful examination of this issue is beyond the scope of the paper; we hope to return to it in future work.

This example suggests that our approach is at least as expressive as the process-algebra approach(es), although we have not formalized this, let alone verified it. What about the converse? Ryan and Schneider [19, Section 3.5] claim to capture nondeducibility using their approach, so it would seem that they are capturing (at least) our notion of total f-secrecy. However, they identify "nondeducibility" with the instance of nondeducibility discussed by Sutherland at the end of his paper, which is essentially generalized noninterference. Thus, the way they capture nondeducibility does *not* capture our notion of f-total secrecy. Their "most liberal generalisation" [19, Section 4.2], presented in terms of an abstraction operator and a *Constrain* operator, does seem to correspond to f-total secrecy, although we have not formalized the relationship. However, it does not seem that their paper (or any of the other papers on the process-algebra approach) has an analogue to f-C-secrecy, nor is there an analogue to the notion of syntactic security. It would be interesting to see whether notions like semisynchronous secrecy and resource-bounded secrecy could be handled cleanly in a process-algebra approach; it would also be interesting to see how our definitions of probabilistic secrecy relate to recent work adding probability to definitions of noninterference based on the process algebra approach [1, 21].

7 Discussion

We have defined general notions of secrecy in multiagent systems, and shown the connection between our definitions and standard epistemic notions. We then applied the definition to the problem of characterizing absence of information flow.

Of course, we are not the first to attempt to provide a general framework for analyzing security properties. (See, for example, [6, 12, 15, 19, 24] for some other attempts.) However, we believe that our definitions come closer to the intuitions that people in the field have had, since those intuitions are often expressed in terms of (a lack of) knowledge. Our approach has a number of other advantages over previous approaches:

- As we have shown, it can be easily extended to deal with probabilistic secrecy.
- It can deal with the whole spectrum from synchrony to total asynchrony, without assuming a specific underlying system model.
- It can be generalized to deal with more specific security concerns, so that we can talk about the secrecy of a particular formula as well as total secrecy.
- The characterization in terms of knowledge allows extensions to resource-bounded knowledge, which seems appropriate for dealing with security in the presence of resource-bounded adversaries.

There are a number of possibilities for future work; we highlight a few of them here.

- It would be interesting to apply model-checking algorithms to checking whether (finite-state) systems preserved security properties expressed in terms of knowledge.
- Compositionality has been an important concern in security that we have not addressed at all. (See [13, 15] for some early work on compositionality.) It would be interesting to understand the extent to which our notions are preserved under composition.
- An investigation of the general relationship between our systems-based approach and the process-algebra approaches discussed in Section 6.5 could yield numerous benefits. For one thing, as we said before, it would be useful to compare formally the expressive power of the various approaches. It may also be possible to combine the benefits of the process-algebra approach with those of the approach that we are advocating here. For example, a process can be viewed as a compact way of specifying a system. This suggests that perhaps the tools of process algebra can be brought to bear on verifying secrecy as we have defined it. This seems like a promising line of further research.

Acknowledgments: We thank Riccardo Pucella and Andrei Sabelfeld for useful discussions, and Dan Grossman for a thorough copy-editing job of an earlier draft of this paper.

References

[1] A. Aldini. Probabilistic information flow in a process algebra. In *Proceedings of the 12th International Conference on Concurrency Theory*, pages 152–168, 2001.

[2] E. A. Emerson. Alternative semantics for temporal logics. *Theoretical Computer Science*, 26:121–130, 1983.

[3] K. Engelhardt, R. van der Meyden, and Y. Moses. Knowledge and the logic of local propositions. In *Proc. Seventh Conference on Theoretical Aspects of Reasoning about Knowledge*, pages 29–41, 1998.

[4] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, Mass., 1995.

[5] R. Focardi and R. Gorrieri. A classification of security properties for process algebra. *Journal of Computer Security*, 3(1):5–33, 1994.

[6] R. Focardi and R. Gorrieri. Classification of security properties (Part I: Information flow). In *Foundations of Security Analysis and Design*, pages 331–396. Springer, 2001.

[7] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proc. IEEE Symposium on Security and Privacy*, pages 11–20. 1982.

[8] J. W. Gray and P. F. Syverson. A logical approach to multilevel security of probabilistic systems. *Distributed Computing*, 11(2):73–90, 1998.

[9] J. W. Gray III. Toward a mathematical foundation for information flow security. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 21–35, 1991.

[10] J. Halpern and R. Pucella. Security protocol analysis, intruder models, and algorithmic knowledge. 2002.

[11] J. Y. Halpern and M. R. Tuttle. Knowledge, probability, and adversaries. *Journal of the ACM*, 40(4):917–962, 1993.

[12] H. Mantel. Possibilistic Definitions of Security—An Assembly Kit. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 185–199, 2000.

[13] D. McCullough. Specifications for multi-level security and a hook-up property. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 161–166, 1987.

[14] J. McLean. Security models and information flow. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 180–187, 1990.

[15] J. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 79–93, 1994.

[16] R. Milner. *Communication and Concurrency*. Prentice Hall, Hertfordshire, 1989.

[17] A. Roscoe. Csp and determinism in security modelling. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 114–127, 1995.

[18] A. W. Roscoe. *The Theory and Practice of Concurrency*. Prentice Hall, Hertfordshire, 1997.

[19] P. Y. A. Ryan and S. A. Schneider. Process algebra and noninterference. In *Proceedings of The 12th Computer Security Foundations Workshop*, pages 214–227, 1999.

[20] P. Y. A. Ryan, S. A. Schneider, M. H. Goldsmith, G.Lowe, and A.W.Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley, Harlow, England, 2001.

[21] A. Sabelfeld and D. Sands. Probabilistic noninterference for multi-threaded programs. In *Proceedings of The 13th Computer Security Foundations Workshop*, pages 200–214, 2000.

[22] D. Sutherland. A model of information. In *Proceedings of the 9th National Security Conference*, pages 175–183, 1986.

[23] J. T. Wittbold and D. M. Johnson. Information flow in nondeterministic systems. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 144–161, 1990.

[24] A. Zakinthinos and E. S. Lee. A general theory of security properties. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 94–102, 1997.