Distrib. Comput. (1998) 11: 73–90

DISTRIBUTED COMPUTING

A logical approach to multilevel security of probabilistic systems

James W. Gray, III[⋆,1], **Paul F. Syverson**[⋆⋆,2]

[1] Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
[2] Center for High Assurance Computer Systems, Naval Research Laboratory, Washington, DC 20375, USA (e-mail: syverson@itd.nrl.navy.mil)

Summary. We set out a modal logic for reasoning about multilevel security of probabilistic systems. This logic contains expressions for time, probability, and knowledge. Making use of the Halpern-Tuttle framework for reasoning about knowledge and probability, we give a semantics for our logic and prove it is sound. We give two syntactic definitions of perfect multilevel security and show that their semantic interpretations are equivalent to earlier, independently motivated characterizations. We also discuss the relation between these characterizations of security and between their usefulness in security analysis.

Key words: Formal modeling – Verification – Knowledge – Security – Probabilistic Systems

1 Introduction

Multilevel security is the aspect of computer security concerned with protecting information that is classified with respect to a multilevel hierarchy (e.g., UNCLASSIFIED, SECRET, TOP SECRET). A *probabilistic system* is a hardware or software system that makes probabilistic choices (e.g., by consulting a random number generator) during its execution. Such probabilistic choices are useful in a multilevel security context for introducing noise to reduce the rate of (or eliminate) illicit communication between processes at different classification levels. In this paper, we are concerned with definitions of *perfect* (information-theoretic) multilevel security in the sense that the definitions rule out *all* illicit communication without relying on any complexity-theoretic assumptions. That is, our model allows the system penetrators to have unlimited computational power; yet, our definitions are sufficient to ensure there can be no illicit communication.

The systems we address can be depicted in the form shown in Fig. 1. This general form is intended to represent systems including physical hardware with hard-wired connections to other systems, an operating system kernel with connections to other processes provided by shared memory, and processes executing on a multiprocessor with connections to other systems provided by an interprocess communication (IPC) mechanism.

• There is a system, called Σ, that provides services to the other systems. For example, in the case of a multiuser relational database, Σ would store and control access to a set of relations. Σ is the system with respect to which we will be reasoning about multilevel security.

• There is a set of systems (labeled S_1, $S_2, \ldots$, S_i in the figure), called the "covert senders", that have access to secret information. These systems are called "covert senders" because they may attempt to covertly send secret information, via Σ, to other systems that are not authorized to see the information. It is these attempts with which we are concerned. As is commonly done in the literature, we will often refer to the covert senders as *high* systems (referring to the situation where the covert senders have access to *highly classified* information). We will also refer to the set of covert senders collectively as the high *environment*, denoted $\mathscr{H}$. These systems are part of "the environment" in the sense that they are in the environment of the central system, Σ.

• There is a second set of systems (labeled R_1, $R_2, \ldots$, R_j in the figure), called the "covert receivers", that are not authorized to see the secret information that is available to the covert senders. We will often refer to the covert receivers as *low* systems, or collectively as the low *environment*, denoted $\mathscr{L}$.

If the covert senders are able to use Σ to communicate information to the covert receivers, we will say that Σ has a *covert channel*, or equivalently, for our purposes, that Σ is *insecure*. A few notes are in order.

1. It is important to bear in mind that the threat that we are concerned with is *not* that the users (i.e., the *human* users) of the covert sender systems are attempting to send secret information to the covert receivers. We assume that if they wanted to, they could more easily pass notes in the park and entirely bypass Σ. Rather, we are concerned that the covert senders are actually trojan horses (i.e., they appear to be something that the user wants, but actually contain something else that is entirely undesirable to the user) and that these trojan

⋆ Supported by grant HKUST 608/94E from the Hong Kong Research Grants Council.
⋆⋆ Supported by ONR.
Correspondence to: P.F. Syverson

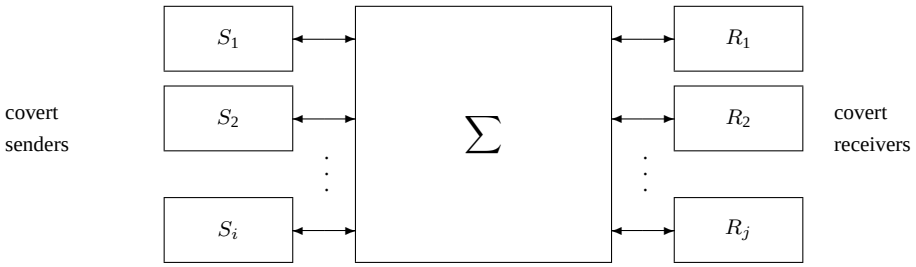

Fig. 1. The general form of a system

horses are attempting to send secret information to the covert receivers. This is a legitimate concern since system developers do not want to incur the cost of verifying every component of a conglomerate system with respect to multilevel security requirements. Ideally, only a small number of components in the system (e.g., in our case only Σ) have security requirements and thereby require verification; while the remaining components can be implemented by off-the-shelf hardware and software that are unverified with respect to security (and therefore may be trojan horses).

We assume a worst case scenario, where *all* of the covert senders and covert receivers are trojan horses. Indeed, we assume that *all* of the trojan horses are cooperating in an attempt to transmit information from the covert senders to the covert receivers.

2. It is also important to bear in mind that in our intended application, the covert senders will not be able to communicate directly to the covert receivers (i.e., by bypassing Σ). Typically, there are software, hardware or other physical controls to prevent this. For example, nonbypassability is one of the well-known principles of a "reference monitor" (see [13]), which is one of the typical applications we have in mind.
3. Our model contrasts sharply with much other work on security (e.g., [31], [11]) in that we consider a set of untrusted agents (viz, the covert senders and receivers) that are connected via a trusted agent, whereas these other works consider a set of *trusted* agents connected via an *untrusted* agent. This difference in our model reflects the difference in the respective applications. The work of Meadows in [31] and Dolev et al. in [11] is intended for the analysis of a set of legitimate (and trusted) agents that are attempting to establish secure communication over an untrusted network. In that work, the assumption is that the penetrator is able to subvert the network (i.e., the central component of the system), but not the trusted (lateral) agents.

   In contrast, our work is intended to be used to analyze a centralized server that serves a set of untrusted entities. Correspondingly, our assumption is that the penetrator may be able to subvert the untrusted (lateral) agents, but not the central server.
4. The fact that we have partitioned the set of systems external to Σ into two sets, high and low, may seem to indicate that we are limiting ourselves to two levels of information (e.g., SECRET and UNCLASSIFIED). However, this is not the case. In a more general setting, information is classified (users are cleared, resp.) according to a finite, partially ordered set (see, e.g., Denning's [9]); that is, there is a finite set of classification levels (clearance levels, resp.) that is ordered by a reflexive, transitive, and anti-symmetric relation, which we call *dominates*. (In fact this set forms a finite lattice.) A given user is permitted to observe a given piece of information only if the user's clearance dominates the classification of the information. In the case where there are more than two levels, a separate analysis would be performed for each level, x; in each analysis, the set of levels would be partitioned into those that are dominated by x (i.e., the "low" partition) and the set of levels that are not dominated by x (i.e., the "high" partition). Thus, we have lost no generality by restricting our attention to two levels.

The motivation for reasoning about the probabilistic behavior of systems has appeared in examples and discussions of many authors (cf. [4, 17, 26, 28, 30, 42]). Essentially, the motivation is that it is possible for a probabilistic system to satisfy many existing definitions of security (e.g., Sutherland's *Nondeducibility* [40], McCullough's *Restrictiveness* [29], etc.) and still contain probabilistic covert channels.

Our long term goal is to develop a logic that can be used to reason about the multilevel security of a given system Σ. The logical definition of security in this paper delineates *ideal* security for probabilistic systems. As such it does not apply to real systems, which are too complex to be simultaneously ideally secure and adequately functional. Nonetheless, it is important to establish the ideal in order to know what is possible. Further, we view the present work as a step in the direction of practical verification of multilevel security of real probabilistic systems (presumably based on definitions of security that allow some limited information flow).

In prior work ([18]), we gave a logic in which an information-theoretic definition of security—the first author's Probabilistic Noninterference (PNI) ([17])—was expressed as

$$K_L(\varphi) \rightarrow R_L(\varphi) \tag{1}$$

where $K_L(\varphi)$ is intuitively regarded as "L *knows* φ" and $R_L(\varphi)$ is intuitively regarded as "L *is permitted to know* φ." Thus, Formula 1 is intuitively interpreted as "If L knows φ then L is permitted to know φ", or in other words, "what L knows is a subset of what L is permitted to know".

This intuitively-appealing formula was proposed by Glasgow, MacEwen, and Panangaden [14] and further developed by Bieber and Cuppens [1, 2]. In other work [19] we extended their approach to a probabilistic framework, retaining the syntactic form of their definition of security, viz Formula 1. However, the knowledge operator (K_L) proposed by Bieber and Cuppens (and its probabilistic analog proposed by us) is nonstandard and rather unnatural. For example, what a subject "knows" does not change over time. In particular, in our probabilistic framework, subjects "know" the probability distribution over all of their future interactions, including all future outputs they will receive. This is in contrast with the intuitive notion of knowledge (as well as the standard formalizations of knowledge such as by Chandy and Misra [6] or Halpern [21]) wherein a subject can acquire knowledge as it interacts with its environment.

Another disadvantage of the prior work of Glasgow et al., Bieber and Cuppens, and the present authors is that Formula 1 makes use of a "permitted knowledge" operator (R_L). Such an operator has no standard semantics and seems, by its very nature, to be application specific. For example, see [8] wherein "permitted knowledge" is essentially formalized as "knowledge that is permitted, as defined in the present application".

In the present work, we develop a new formalization of PNI using the framework of Halpern and Tuttle [22]. In this framework, the knowledge operator is given the standard semantics. Also, our new formalization does not make use of a "permitted knowledge" operator; it is therefore free of the nonstandard operators that were used in our previous formalization. Thus, the present paper can be viewed as superseding our prior work.

In another sense, the present work can be viewed as a novel application of Halpern and Tuttle's framework, since we instantiate their framework with an adversary (see Definition 2.1) fundamentally different from those described in [22].

The remainder of the paper is organized as follows. In Sect. 2 we set out our model of computation. In Sects. 3 and 4, we set out the syntax and semantics of our logic and in Sect. 5, we prove its soundness. In Sect. 6 we state our primary definition of security and prove that it is equivalent to Probabilistic Noninterference. In Sect. 7 we state our verification condition and show that it is equivalent to the Applied Flow Model. Finally, in Sect. 8, we give some conclusions of this work.

2 System model

In this section, we describe our system model. This is the model by which we will (in Sect. 4) give semantics to our logic. First, we describe the general system model, which is taken from Halpern and Tuttle [22]. The framework of Halpern and Tuttle builds on the work of Fagin and Halpern in [12]. It also encompasses earlier work of Ruspini in [35]. After giving the general system model, we tailor the model to our needs by imposing some additional structure on the model and (in Halpern and Tuttle's terminology) choosing the "adversaries", resulting in our application-specific model.

2.1 General system model

We have a set of agents, $P_1, P_2, \ldots, P_n$, each with its own local state. The *global state* is an n-tuple of the local agents' states.[1] A *run* of the system is a mapping of times to global states. We assume that time is discrete because we are dealing with security at the digital level of the system. We are not, for example, addressing security issues such as analog channels in hardware. Therefore, we will assume that times are natural numbers.

The probabilities of moving among global states are represented in the model by means of labeled computation trees. The nodes of the trees represent global states. For any given node in a tree, the children of that node represent the set of global states that could possibly come next. Each arc from a node to one of its children is labeled with the probability of moving to that state. Thus, from any given node, the sum of the probabilites on its outgoing arcs must be one. We also assume the set of outgoing arcs is finite and that all arcs are labeled with *nonzero* probabilities. This final assumption can be viewed as a *convention* that if the probability of moving from state x to state y is zero, then state y is not included as a child of state x.

Certain events in a system may be regarded as *nonprobabilistic*, while still being nondeterministic. The typical example occurs when a user is to choose an input, and in the analysis of the system we do not wish to assign a probability distribution to that choice; in such cases, we regard that choice as nonprobabilistic. All nonprobabilistic choices in the system are lumped into a single choice that is treated as being made by an "adversary" prior to the start of execution. Thus, after this choice is made, the system's execution is purely probabilistic. In Halpern and Tuttle's words, the nonprobabilistic choices have been "factored out".

In the model of computation, each possible choice by the adversary corresponds to a labeled computation tree. In other words, a system is represented as a set of computation trees, each one corresponding to a different choice by the adversary. There is no indication how the adversary's choice is made, just that it is made once and for all, prior to the start of execution.

2.2 Application-specific system model

In this section, we impose some additional structure on the general model described in the previous section. We fix the set of agents, fix our model and intuitions regarding communication, place some (environmental) constraints on the agents, and fix the set of choices available to the adversary.

Agents. As indicated in Fig. 1 and the surrounding discussion, we can limit our model to three agents: (1) the system

[1] Halpern and Tuttle also include the state of the environment as part of the global state. In their usage of the term, the "environment" is intended "to capture everything relevant to the state of the system that cannot be deduced from the agents' local states" [22, Sect. 2]. This typically includes messages in transit on the communication medium. However, we model such things as part of the covert senders' and receivers' local states; we therefore omit what they call the environment from our model. In contradistinction, we refer to everything external to Σ as "the environment"; viz, the covert senders and receivers constitute the environment.

under consideration, denoted Σ, (2) the covert senders (or alternatively, the high environment), denoted $\mathscr{H}$, and (3) the covert receivers (or alternatively, the low environment), denoted $\mathscr{L}$. In the remainder of the paper, we will tacitly assume that the global system is comprised of these three agents.

Model of communication. Our model of communication is similar to those of Bieber and Cuppens, Millen, and the first author (cf. [2], [32], and [17], respectively). We view Σ's interface as a collection of channels on which inputs and outputs occur. Since we consider the agent $\mathscr{H}$ (resp., $\mathscr{L}$) to consist of *all* processing that is done in the high (resp., low) environment, including any communication mechanism that delivers messages to Σ, we will not need to model messages in transit or, in Halpern and Tuttle's terminology, the state of the environment; rather, these components of the global state will be included as part of $\mathscr{H}$'s and $\mathscr{L}$'s state.

In many systems of interest, the timing of events is of concern. (See Lampson's [25] for an early description of covert communication channels that depend on timing.) In particular, some covert communication channels depend on a clock being shared between the covert senders and receivers. Such channels are typically called *timing channels*; see Wray's [43] for examples and discussion. To handle such cases, we take the set of times (i.e., the domain of the runs) to be the ticks of the best available shared clock.[2] Events occurring between two ticks are regarded as occurring on the latter tick. This is sufficient for the purposes of our analysis because, as far as the covert senders and receivers are concerned, this is the most accurate information available. Also note that if the timing of certain events (wrt the best available shared clock) is nonprobabilistic, we can consider the various possibilities to be choices that are made by the adversary and factor out that nondeterminism as discussed by Halpern and Tuttle [22].

Since the mechanisms of high-level[3] I/O routines may introduce covert channels (see, e.g., McCullough's [28, Sect. 2.3]), we take a very low-level view of I/O. In particular, we assume one input and one output per channel per unit time (where times are chosen according to the above considerations). That is, for each time we have a vector of inputs (one for each channel) and a vector of outputs (one for each channel). If a given agent produces no new data value at a given time, it may in fact serve as a signal in a covert channel exploitation. Hence, we treat such "no new signal" events as inputs. Similarly, we do not consider the possibility that the system can prevent an input from occurring. Rather, the system merely chooses whether to make use of the input or ignore it. Any acknowledgement that an input has been received is considered to be an output.

Given these considerations, we fix our model of communication as follows. We assume the following basic sets of symbols, all nonempty:

C: a finite set of input/output channel names, $c_1, \ldots, c_k$,
I: representing the set of input values,
O: representing the set of output values.
$\mathbb{N}^+$: representing the set of positive natural numbers. This set will be used as our set of "times".

Since there is one input per channel at each time, we will be talking about the vector of inputs that occurs at a given time. We will denote the set of all vectors of inputs by $I[C]$. Typical input vectors will be denoted $a, a', a_1, \ldots \in I[C]$.

Similarly, we will denote the set of all output vectors by $O[C]$ and typical output vectors will be denoted $b, b', b_1, \ldots \in O[C]$.

Now, to talk about the history of input vectors up to a given time, we introduce notation for traces. We will denote the set of input traces of length k by $I_{C,k}$. Mathematically, $I_{C,k}$ is a shorthand for the set of functions from $C \times \{1, 2, \ldots k\}$ to I. Therefore, for a trace $\alpha \in I_{C,k}$, we will denote the single input on channel $c \in C$ at time $k' \leq k$ by $\alpha(c, k')$.

We will also need to talk about infinite traces of inputs. For this we use the analogous notation $I_{C,\infty}$, which is short hand for the set of functions from $C \times \mathbb{N}^+$ to I.

Similarly, we will denote the set of output traces of length k by $O_{C,k}$ and the set of infinite output traces by $O_{C,\infty}$. Naturally, for an output trace β, $\beta(c, k)$ represents the output on channel c at time k.

There will be situations where we want to talk about vectors or traces of inputs or outputs on some subset of the channels, $S \subseteq C$. In such cases we will use the natural generalizations of the above notations, viz, $I[S]$, $I_{S,k}$, $I_{S,\infty}$, etc.

Environmental constraints. Any given agent will be able to see the inputs and outputs on a subset of the system's I/O channels. We make this precise by "restricting" vectors and traces to subsets of C. Given an input vector $a \in I[C]$ and a set of channels $S \subseteq C$, we define $a \restriction S \in I[S]$ to be the input vector on channels in S such that $a \restriction S(c) = a(c)$ for all $c \in S$.

Similarly, given an input trace $\alpha \in I_{C,k}$ and a set of channels $S \subseteq C$, we define $\alpha \restriction S \in I_{S,k}$ to be the input trace for channels in S such that $\alpha \restriction S(c, k') = \alpha(c, k')$ for all $c \in S$ and all $k' \leq k$.

We assume that the set of low channels, denoted L, is a subset of C. Intuitively, L is the set of channels that the low environment, $\mathscr{L}$, is able to directly see. In particular, $\mathscr{L}$ is able to see both the inputs and the outputs that occur on channels in L.

In practice, there will be some type of physical or procedural constraints on the agent $\mathscr{L}$ to prevent it from directly viewing the inputs and outputs on channels in $C - L$. For example, those channels may represent wires connected to workstations that are used for processing secret data. In this case, the secret workstations might be located inside a locked and guarded room. In addition, periodic checks of the wires might be made to ensure that there are no wiretaps on them. In this way, $\mathscr{L}$ is prevented from directly viewing the data that passes over the channels in $C - L$.

On the other hand, we place no constraints on the set of channels that $\mathscr{H}$ is able to see. In particular, we make the worst-case assumption that $\mathscr{H}$ is able to see all inputs and outputs on all channels.

[2] A shared clock may be an explicit clock supplied by Σ, e.g, the *system clock*, or it may be a clock manufactured by the covert senders and receivers for their own purposes; see [43] for examples and discussion.

[3] In this context, "high-level" means highly *abstract* rather than highly *classified*.

The above considerations are consistent with what we've called the "Secure Environment Assumption" in previous work [17, 18]. In the present paper, this assumption is made precise in terms of our definition of the adversary to be given next.

The adversary. As discussed above, in Halpern and Tuttle's framework, all nonprobabilistic choices are factored out of the execution of the system by fixing an adversary at the start of execution. To make use of this framework, we must define the set of possible adversaries from which this choice is made.

The "adversary" in our application is the pair of agents, $\mathscr{H}$ and $\mathscr{L}$, that are attempting to send data from the high environment across the system Σ to the low environment. To be fully general, we model these agents as mixed strategies (in the game-theoretic sense). That is, at each point in the execution of the system the strategy gives the probability distribution over the set of next possible inputs, conditioned on the history up to the current point. In the next section, we present an example to motivate the need for such generality. Before doing that, we make the adversary precise with the following two definitions.

Definition 2.1 *An* adversary *is a* conditional probability function, $\mathscr{A}(a \mid \alpha, \beta)$ *(where $a \in I[C]$ and for some time, k, $\alpha \in I_{C,k}$ and $\beta \in O_{C,k}$). Intuitively, the adversary describes the environment's conditional distribution on the next input vector, given the previous history of inputs and outputs. By saying that $\mathscr{A}(a \mid \alpha, \beta)$ is a conditional probability function we require that*

- $0 \leq \mathscr{A}(a \mid \alpha, \beta) \leq 1$, *and*
- $\sum_a \mathscr{A}(a \mid \alpha, \beta) = 1$

In fact, it is trivial to define a conditional probability mass function corresponding to $\mathscr{A}$ where a, α, and β are replaced with the values of the corresponding random variables [33]. Such a conditional probability mass function can be defined in terms of the probability measure $\mu_{T_{\mathscr{A}}}$ given in definition 4.4 below. □

Definition 2.2 *We say that an adversary $\mathscr{A}$ satisfies the* Secure Environment Assumption with respect to a set of channels $L \subseteq C$ *iff there exists a pair of conditional probability functions $\mathscr{H}$ and $\mathscr{L}$ such that for all $a \in I[C]$, all $k \in \mathbb{N}^+$, all $\alpha \in I_{C,k}$, and all $\beta \in O_{C,k}$,*

$$\mathscr{A}(a \mid \alpha, \beta) = \mathscr{H}(a \upharpoonright (C - L) \mid \alpha, \beta) \cdot \mathscr{L}(a \upharpoonright L \mid \alpha \upharpoonright L, \beta \upharpoonright L)$$

(where $\cdot$ denotes real multiplication). □

The Secure Environment Assumption can be intuitively understood as saying that the input on channels in $(C - L)$ at time k is (conditionally) statistically independent of the input on channels in L at time k, and the input on channels in L at time k depends only on previous inputs and outputs on channels in L. For the remainder of this paper, we will assume that all adversaries from which the initial choice is made satisfy the Secure Environment Assumption.

Later in this section, we describe how a given adversary $\mathscr{A}$ and the description of a particular system, Σ, are used to construct the corresponding computation tree $T_{\mathscr{A}}$. Since there is one tree for each possible adversary, we can think of the set of trees as being indexed by the adversaries. Therefore, we will often write $T_{\mathscr{A}}$, $T_{\mathscr{A}'}$, $T_{\mathscr{A}_i}$, etc.

It is clear that for an adversary $\mathscr{A}$ that satisfies the Secure Environment Assumption (wrt L), the conditional probability functions $\mathscr{H}$ and $\mathscr{L}$ are unique. Further, given $\mathscr{H}$ and $\mathscr{L}$, there is a unique adversary, $\mathscr{A}$, for which $\mathscr{H}$ and $\mathscr{L}$ are the probability functions that satisfy the corresponding constraint. There is therefore no ambiguity in writing $T_{\mathscr{H},\mathscr{L}}$, $T_{\mathscr{H}',\mathscr{L}'}$, etc. when we want to refer to the components of the adversary individually.

Note that our definition of an adversary is not meant to be as general as the adversary discussed by Halpern and Tuttle. (In fact, Halpern and Tuttle give no structure at all to their adversary.) Rather, our adversary is application-specific; in particular, it is for reasoning about multilevel security of probabilistic systems and is not designed to be used outside that domain.

On the other hand, this particular adversary represents a novel application of Halpern and Tuttle's framework. In Halpern and Tuttle's examples, the adversary represents one or both of two things:

- the initial input to the system; and
- the schedule according to which certain events (e.g., processors taking steps) occur.

In contrast, our adversary does not represent a given input to the system. Rather, it represents a mixed strategy for choosing the inputs to the system. In some sense, we can think of this as a generalization on the first item above; however, our application still fits within the framework set out by Halpern and Tuttle.

The state of the system. At any given point, P, in any given computation tree, $T_{\mathscr{A}}$, there should be a well-defined state of the system. For our purposes, the state includes the following information.

1. All inputs and outputs that have occurred on all channels up to the current time.
2. The adversary. In [22], Halpern and Tuttle make the assumption that all points in all trees are unique. They suggest (and we adopt) the following idea to ensure that this is true. The state encodes the adversary. That is, all nodes in tree $T_{\mathscr{A}}$ encode $\mathscr{A}$. Note that we do *not* assume that any given agent knows the adversary; just that it is somehow encoded in the state. We can think of the high part of the adversary, $\mathscr{H}$, as being encoded in the high environment and the low part, $\mathscr{L}$, as being encoded in the low environment.
3. Additional components of the global state represent the internal state of Σ. For example, in describing Σ, it is often convenient to use internal state variables. The state of these variables can be thought of as a vector of values, one value for each state variable. Thus, the internal state, when it exists, will be denoted c, and the history of internal states will be denoted γ.

Computation trees. Now that we have set out the possible states of the system (i.e., the points of computations), we can talk about the construction of the computation trees.

For each reachable point, P, we assume that Σ's probability distribution on outputs is given. For example, this can be given by a conditional probability distribution, $\mathscr{O}(b, c \mid \alpha, \beta, \gamma)$, where α, β, and γ give the history (up through some time k) of inputs, outputs, and internal states, respectively, c is a vector of internal state variables (i.e., the internal system state at time $k+1$), and b is the vector of outputs produced by the system (at time $k+1$).

Given $\mathscr{O}(b, c \mid \alpha, \beta, \gamma)$ and the adversary $\mathscr{A}$, we can construct the corresponding computation tree by starting with the initial state of the system (i.e., the point at the root of the tree with empty histories of inputs, outputs, etc.) and iteratively extending points as follows.

Let P be a point in the tree with internal system history γ, input history α, and output history β. We will make P' a child of P iff

1. P' is formed from P by modifying the internal system state to c and extending P's input history (output history, resp.) with a (b, resp.); and
2. both $\mathscr{O}(b, c \mid \alpha, \beta, \gamma)$ and $\mathscr{A}(a \mid \alpha, \beta)$ are positive.

In such cases, we label the arc from P to P' with $\mathscr{O}(b, c \mid \alpha, \beta, \gamma) \cdot \mathscr{A}(a \mid \alpha, \beta)$, i.e., the system, Σ, and the environment, $\mathscr{A}$, make their choices independently.

Runs of the system. A run of the system is an infinite sequence of states along a path in one of the computation trees. When we want to talk about the particular run, ρ, and time, k, at which a point P occurs, we will denote the point by the pair (ρ, k). Further, if we wish to talk about the various components of the run, i.e., the trace of the inputs, α, outputs, β, or other variables, γ, we will denote the run by (α, β, γ) and denote the point, P, by $(\alpha, \beta, \gamma, k)$.

For a given tree, T, we denote the set of runs (i.e., infinite sequences of states), formed by tracing a path from the root, by $runs(T)$.

For security applications we are concerned with information flow into and out of the system rather than with information in the system per se. Thus, though our system model is adequate to represent internal states and traces thereof, in subsequent sections it will be adequate to represent systems entirely in terms of input and output. In particular, system behavior can be represented by '$\mathscr{O}(b \mid \alpha, \beta)$' rather than '$\mathscr{O}(b, c \mid \alpha, \beta, \gamma)$'.

3 Syntax

In this section we set out our formal language and use it to describe two simple systems. Then we give the axioms and rules of our logic.

3.1 Formation rules

To describe the operation of the system under consideration (viz, Σ), we use a variant of Lamport's Raw Temporal Logic of Actions (RTLA) [24].[4] The primary difference is that we add a modal operator $Pr_i(\varphi)$ that allows us to specify and reason about the probabilistic behavior of the system.

From the previous section, we assume the following basic sets of symbols, all nonempty: C, I, O, and $\mathbb{R}$. Members of $\mathbb{R}$ will have the usual representation—e.g., $43.5 \in \mathbb{R}$.

We will also be talking about the subjects (or agents) of the system. Formally, a *subject*, $S \subseteq C$, is identified with the process's view of the system, i.e. the set of channels on which it can *see* the inputs and outputs.

Formulae in the language are built up according to the following rules.

- constants from the set of basic symbols are terms.
- state variables (representing the value of that variable in the current state) are terms. Among the state variables, there are two reserved for each communication channel. For each $c \in C$, we have a state variable c_{in} that takes values from I, and another state variable c_{out} that takes values from O. Note that, implicitly, *inputs* are from the covert senders and receivers into the system (Σ) and *outputs* are from the system to the covert senders and receivers. This is because Σ is the system under consideration (i.e., with respect to which we are reasoning about security). We have no mechanism (and no need) to specify communication between agents not including the system under consideration.
- primed state variables (e.g., c'_{in}) are terms. (These represent the value of the variable in the next state.)
- We use standard operators among terms (e.g., + and $\cdot$ for addition and multiplication, respectively), with parentheses for grouping subterms, to form composite terms.
- an atomic *formula* is an equation or inequality among terms.
- For any formula φ, $\Box\varphi$ is a formula (to be read intuitively as *always* φ).
- We build up composite formulae, in the usual recursive fashion using $\wedge$, $\vee$, $\neg$, and $\rightarrow$.
- for any nonmodal formula[5] φ, and for any subject $S \subseteq C$, $Pr_S(\varphi)$ is a real-valued term. Intuitively, $Pr_S(\varphi)$ represents the *subjective probability* that S assigns to the formula φ, that is, the probability of φ, given the previous history of inputs and outputs on channels in S. We refer to $Pr_C(\varphi)$ (where C is the set of all communication channels) as the *objective probability* of φ, since it represents the probability of φ given all available information, i.e., the unbiased probability of φ.

To specify and reason about our security properties of interest, we also add a set of modal operators on formulae: $K_1, \ldots, K_n$, representing knowledge for each subject (represented by the subscript of the operator). Therefore, we add the following additional formation rule to our syntax.

- For any formula φ, and for any subject $S \subseteq C$, $K_S(\varphi)$ (representing that S knows φ) is a formula.

Note that this and previous rules are mutually recursive; so, we can express, e.g., that S always knows that $x = 5$.

[4] Roughly speaking, Raw Temporal Logic of Actions (RTLA) is the same as Lamport's Temporal Logic of Actions (TLA) without the treatment of stuttering [24]. Since we are not, in this paper, concerned with refinement, we omit the considerations of stuttering and use RTLA.

[5] A nonmodal formula is a formula that does not contain any knowledge or temporal operators.

3.2 Examples

We now give two simple examples of how to describe systems in our language. Ultimately, we will have sufficient formal machinery to show that one of these systems is secure and the other is not; however, here we simply set them out formally. These descriptions are meant to give the reader an intuitive feel for the meaning of expressions in the language. Precise meanings will be given in Sect. 4. Also, the second of these examples will motivate our choice to model adversaries as strategies.

Example 3.1 The first example is a simple encryption box that uses a "one-time pad" [10]. It has two channels, *high* and *low*. At each tick of the system clock, it inputs a 0 or 1 on the high channel and outputs a 0 or 1 on the low channel. The low output is computed by taking the "exclusive or" (XOR) (denoted $\oplus$) of the high input and a randomly generated bit.

Note that we are modeling only the sender's (encrypting) side of a one-time pad system. Thus, issues such as how the random bit string is distributed to, and used by, the receiver's (decrypting) side are out of the scope of this specification.

It is well known that the XOR of a data stream with an independent uniformly distributed bit stream results in an output stream that is uniformly distributed. Therefore, we can describe the encryption box as follows.

Let $C = \{h, l\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$. Then, the system is specified by the following formula.

$$\Box \left(Pr_C(l'_{out} = 0) = 0.5 \wedge Pr_C(l'_{out} = 1) = 0.5 \right)$$

In this formula, l_{out} is a state variable representing the output on the low channel, l. Therefore, l'_{out} is the output on l at the *next* time. Further, $Pr_C(l'_{out} = 0)$ denotes the probability that the output on l is a 0 at the next time. Hence, the entire formula says that at all times, the probability of Σ producing a one (1) on the next clock tick is equal to the probability of producing a zero (0), which is equal to 0.5. Note that we have not specified the probability distribution over inputs, since this constitutes environment behavior rather than system behavior.

□

Example 3.2 The second example is an insecure version of the simple encryption box. Shannon [37] gives an early description of this system.

As in the first example, the system computes the "exclusive or" of the current high input and a randomly generated bit and outputs that value on the low channel at each time. However, in this system, the randomly generated bit used at any given tick is actually generated and output on the high output channel during the *previous* tick of the clock.

This can be expressed in our formalism as follows. Let $C = \{h, l\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$. The following formula specifies the system.

$$\Box(Pr_C(h'_{out} = 0) = 0.5 \wedge Pr_C(h'_{out} = 1) = 0.5 \wedge l'_{out} = h_{out} \oplus h'_{in})$$

Note that in the third conjunct, h_{out} is unprimed, indicating that the output on l at the next time is the "exclusive or" of the *current* output on h with the *next* input on h.

Now note that if the high agent ignores its output, this system acts exactly as the system from the previous example (and can be used for perfect encryption). In particular, suppose we were to model an adversary as an input string—the input to be provided by the high agent. Then, it is straightforward to prove that for any adversary (i.e., any high input string) fixed prior to the start of execution, the output to low will be uniformly distributed and, in fact, will contain no information about the high input string.

However, the bit that will be used as the one-time pad at time k is available to the high agent at time $k-1$. Therefore, (due to the algebraic properties of "exclusive or", viz, $x \oplus x \oplus y = y$) the high agent can use this information to counteract the encryption. In particular, the high agent can employ a (game-theoretic) strategy to send any information it desires across the system to the low agent.

For example, suppose the high agent wishes to send a sequence of bits, $b_1, b_2, \ldots$. We'll denote the high input (resp., output) at time k by $h_{in}(k)$ (resp., $h_{out}(k)$). The appropriate strategy for the high agent is as follows.

> The high agent chooses its input for time $k + 1$ as $h_{in}(k + 1) = h_{out}(k) \oplus b_k$.

Thus, the output to low at time $k + 1$, denoted $l_{out}(k + 1)$ is computed as follows.

$$\begin{aligned} & l_{out}(k+1) \\ &= h_{out}(k) \oplus h_{in}(k+1) && \text{[by the system description]} \\ &= h_{out}(k) \oplus h_{out}(k) \oplus b_k && \text{[by the high strategy]} \\ &= b_k && \text{[by the properties of } \oplus\text{]} \end{aligned}$$

Thus, by employing the correct strategy, the high agent can noiselessly transmit an arbitrary message over Σ to the low agent. This, of course, motivates our choice to model adversaries as strategies, rather than, e.g., input strings.

□

We now have some sense of the formal language, with the exception of the knowledge operator K_S. As previously mentioned, this operator will be used to formalize the security properties that interest us. We will illustrate that use in a later section. For now we mention that in security analyses it is typical to assume that system users (and penetrators) know how the system works (i.e., its specifications are not secret); we make such assumptions explicit using our knowledge operator, in particular, if the system specification is given by a formula φ, we will assume that for every subject S, $K_S(\varphi)$.

3.3 The Logic

We now give the axioms of our logic. In the following, we will use 'φ' and 'ψ' to refer to formulae of our language.

Propositional reasoning. All instances of tautologies of propositional logic.

Temporal reasoning. The following are standard axioms for temporal reasoning about discrete systems. The logic they constitute is generally called **S4.3Dum**. (See Goldblatt's [16] for details.) We have labelled the axioms with their historical names. Let φ and ψ be formulae of our language.

K $(\Box(\varphi) \wedge \Box(\varphi \rightarrow \psi)) \rightarrow \Box\psi$

T $\Box\varphi \rightarrow \varphi$
4 $\Box\varphi \rightarrow \Box\Box\varphi$
L $\Box(\varphi \wedge \Box\varphi \rightarrow \psi) \vee \Box(\psi \wedge \Box\psi \rightarrow \varphi)$
Dum $\Box(\Box(\varphi \rightarrow \Box\varphi) \rightarrow \varphi) \rightarrow (\Diamond\Box\varphi \rightarrow \varphi)$

'$\Diamond\varphi$' can be interpreted roughly as saying that at some point φ is true. Formally, it is viewed as notational shorthand: for all formulae φ, $\Diamond\varphi \stackrel{\triangle}{=} \neg\Box\neg\varphi$. **K** guarantees that the temporal operator respects modus ponens. Each of the other axioms captures a feature of time that we desire. **4** gets us transitivity. **T** guarantees that we don't run out of time points (seriality) and that temporal references include the present. **L** guarantees that all points in time are connected (linearity). And, **Dum** guarantees that time is discrete. (Between any two points in time there are at most finitely many other points; see Goldblatt's [16] for further discussion.)

Real number axioms. Standard field and order axioms for the real numbers (to apply to members of $\mathbb{R}$ and function terms with range $\mathbb{R}$.) We will not enumerate these axioms. (See any elementary real analysis book for enumeration, e.g., [27] or [34].)

Epistemic reasoning. The (nonredundant) axioms of the Lewis system **S5** (cf. Chellas, [7], or Goldblatt. [16]) apply to the knowledge operators (K_S). As for temporal axioms, we give the axioms their historical names. Let S be a subject, and let φ and ψ be formulae of our language.

K $(K_S(\varphi) \wedge K_S(\varphi \rightarrow \psi)) \rightarrow K_S(\psi)$ (Knowledge respects modus ponens.)
T $K_S(\varphi) \rightarrow \varphi$ (What one knows is true.)
5 $\neg K_S(\varphi) \rightarrow K_S\neg K_S(\varphi)$ (If you don't know something, then you know that you don't know it.)

Random variable axioms. The standard requirements for random variables (in the probability theoretic sense).

PM (Positive Measure) for any formula, φ, and any subject, S, $Pr_S(\varphi) \geq 0$
(The probability of any event is greater than or equal to zero.)
NM (Normalized Measure) for any channel, c, and any subject, S,
$\sum_{a \in I} Pr_S(c_{in} = a) = 1$ (The probability of all possibilities sums to one.)
$\sum_{b \in O} Pr_S(c_{out} = b) = 1$

Additional axioms. Since our logic contains three different modalities, we need some axioms to describe the interactions among them. The following are not intended to be complete in any sense; they are merely sufficient for the present purposes.

K$\Box$ For any formula φ and any subject S,
$K_S(\Box\varphi) \rightarrow \Box(K_S\varphi)$
(If S knows something is always true, then S *always* knows it's true.)
KPr For any formula φ, any subject S, and any real number r,
$K_S(Pr_C(\varphi) = r) \rightarrow K_S(Pr_S(\varphi) = r)$
(If S knows the objective probability of φ, then S knows its subjective probability of φ and the two probabilities are the same.)

The above are all of our axioms. We now give the rules of our logic, which are both standard.

MP. (Modus Ponens)
From φ and $\varphi \rightarrow \psi$ infer ψ.

Nec. (Necessitation) This rule applies to both of our modal operators: $\Box$ and K_S. (It is called 'necessitation' because it was originally applied to a necessity operator.)
From $\vdash \varphi$ infer $\vdash \Box\varphi$
From $\vdash \varphi$ infer $\vdash K_S(\varphi)$

Note that in the above, '$\vdash \varphi$' indicates a derivation of φ from the axioms alone, rather than from a set of premises. (Derivations are defined below.) Thus, in the case of the knowledge operator (and analogously for $\Box$) **Nec** says that if φ is a theorem (derivable without any premises) then all subjects know φ.

We can now define formal derivations.

Definition 3.3 *Let Γ be a finite set of formulae of our language. A finite sequence of formulae $\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_n$ is called a derivation (of φ_n from Γ) iff each φ_k ($k = 1, \ldots, n$) satisfies one of the following:*

- *$\varphi_k \in \Gamma$*
- *φ_k is an axiom.*
- *φ_k follows from some theorem by* **Nec**.
- *For some $i, j < k$, φ_k results from φ_i and φ_j by* **MP**.

We write '$\Gamma \vdash \varphi$' to indicate a derivation of φ from Γ, and we write '$\vdash \varphi$' to indicate a derivation of φ from the axioms alone. □

This completes our statement of the formal system.

4 Semantics

In the previous section we presented a syntactic system. So far we have only intuitive meanings to attach to this formalism. In this section we provide semantics for our system in terms of the Halpern-Tuttle framework and our application-specific model set out in Sect. 2.

4.1 Semantic model

A *model* M is a tuple of the form:

$$\langle \mathbb{R}, +, \cdot, \leq, W, \mathscr{T}, C, I, O, v, \ \kappa_1, \ldots, \kappa_{|\mathscr{P}(C)|} \ \rangle$$

Here, $\mathbb{R}$ and its operations and ordering relation gives us the real numbers; W is the set of points (i.e., global states or "worlds"); $\mathscr{T}$ is the set of labeled computation trees (with nodes from W); C, I, and O are the sets of channels, possible inputs, and possible outputs, respectively; v is the assignment function, which assigns semantic values to syntactic expressions at each point; (values of v at a particular point P, will be indicated by the projection 'v_P'); and the κ_{i_S} are knowledge accessibility relations, one each for each subject S. Essentially, two points are accessible for a given subject if that subject cannot distinguish between those two points. (i.e., the subject does not "know" which of the points he is in.) We describe these accessibility relation precisely

in the next section. In the remainder of this paper we will generally denote the accessibility relations corresponding to subject S by 'κ_S'.

In assigning meaning to our language, it is of fundamental importance to associate a probability space with each labeled computation tree. In particular, for each labeled computation tree $T_{\cdot\mathscr{L}}$ we will construct a sample space of runs, $\mathscr{R}_{\cdot\mathscr{L}}$, an event space, $\mathscr{X}_{\cdot\mathscr{L}}$ (i.e., those subsets of $\mathscr{R}_{\cdot\mathscr{L}}$ to which a probability can be assigned), and a probability measure $\mu_{\cdot\mathscr{L}}$ that assigns probabilities to members of $\mathscr{X}_{\cdot\mathscr{L}}$.

Our construction of this probability space is quite natural and standard (see, e.g., Seidel's [36] as well as [22] for two instances). We will not go into detail explaining the basic concepts of probability and measure theory here (cf. [20] or [39]).

Definition 4.1 *For a labeled computation tree $T_{\cdot\mathscr{L}}$, the associated* **sample space** *$\mathscr{R}_{\cdot\mathscr{L}}$ is the set of all infinite paths starting from the root of $T_{\cdot\mathscr{L}}$.* □

Definition 4.2 *For any sample space $\mathscr{R}_{\cdot\mathscr{L}}$, the set $e \subseteq \mathscr{R}_{\cdot\mathscr{L}}$, is called a* **generator** *iff it consists of the set of all traces with some common finite prefix. Intuitively, generators are probability-theoretic events corresponding to finite traces.* □

Definition 4.3 *For any sample space $\mathscr{R}_{\cdot\mathscr{L}}$, we define the* **event space**, *$\mathscr{X}_{\cdot\mathscr{L}}$, to be the (unique) field of sets generated by the set of all generators. That is, $\mathscr{X}_{\cdot\mathscr{L}}$ is the smallest subset of $\mathscr{P}(\mathscr{R}_{\cdot\mathscr{L}})$ that contains all of the generators and is closed under countable union and complementation.* □

Definition 4.4 *We define the* **probability measure**, *$\mu_{\cdot\mathscr{L}}$, on $\mathscr{X}_{\cdot\mathscr{L}}$ in the standard way. Suppose e is a generator corresponding to the finite prefix given by (ρ, k). Then, $\mu_{\cdot\mathscr{L}}(e)$ is defined as the product of the transition probabilities from the root of the tree, along the path ρ, up to time k. Further, it is well known that there is a unique extension of $\mu_{\cdot\mathscr{L}}$ to the entire event space (cf. [20]).* □

We will be rather abusive in the use of our probability measures $\mu_{\cdot\mathscr{L}}$. In particular, when we have a finite set of points, x, we will write $\mu_{\cdot\mathscr{L}}(x)$ to denote the probability (as assigned by $\mu_{\cdot\mathscr{L}}$) of passing through one of the points in x. Technically, this is wrong, since $\mu_{\cdot\mathscr{L}}$ is defined for sets of runs; not for sets of points. However, the mapping between the two is extremely natural; the set of runs corresponding to a set of points is the set of all runs that *pass through* those points. Further, by the construction of our probability spaces, all sets of runs corresponding to finite sets of points are measureable. Therefore, there is no danger in this abuse of notation and it greatly simplifies our presentation.

4.2 Assignment function

Given the above semantic model, the main technical question we need to address in assigning meaning to formulae in our logic is:

> For a given subject at a given point in its execution (i.e., at a given node in a given computation tree), what sample space should be used in evaluating the probability that subject assigns to a given formula?

As discussed by Halpern and Tuttle, after choosing these sample spaces, assigning meaning to probability formulae is straightforward. Further, assigning meanings to nonprobability formulae will be done in the standard ways, so that too will be straightforward.

We denote the sample space for subject S at point P by $\mathscr{S}_{S,P}$. Our approach in assigning these sample spaces is discussed by Halpern and Tuttle, where they describe it as "correspond[ing] to what decision theorists would call an agent's *posterior* probability" [22, Sect. 6]. In particular, we choose $\mathscr{S}_{S,P}$ to be the set of points within *tree*(P) that have the same history of inputs and outputs on channels in S as occur on the path to point P. Essentially, this means that S's probability space takes into account all inputs and outputs that S has seen up to the current point; S does not forget anything it has seen. More precisely, we have the following definitions.

Definition 4.5 *Let $S \subseteq C$ be a subject and let $\rho_1 = (\alpha_1, \beta_1, \gamma_1)$ and $\rho_2 = (\alpha_2, \beta_2, \gamma_2)$ be two runs (not necessarily in the same tree). We say that ρ_1 and ρ_2 have the same S-history up to time k if and only if* [6]

$$\forall i, 1 \leq i \leq k, \forall c \in S, \alpha'(c,i) = \alpha(c,i) \wedge \beta'(c,i) = \beta(c,i)$$

□

Definition 4.6 *Let $S \subseteq C$ be a subject and let $P_1 = (\rho_1, k_1)$ and $P_2 = (\rho_2, k_2)$ be two points (not necessarily in the same tree). We say that P_1 and P_2 have the same S-history if and only if the following two conditions hold.*

1. *$k_1 = k_2$.*
2. *ρ_1 and ρ_2 have the same S-history up to time k_1.*

□

Definition 4.7 *Since points are unique even across trees, for a given point P, there is no ambiguity in referring to "the tree that contains P". In the following, we will use $tree(P)$ to denote that tree.* □

Definition 4.8 *Let $S \subseteq C$ be a subject and P be a point; the sample space for S at point P is given by*

$$\mathscr{S}_{S,P} \stackrel{\triangle}{=} \{\ P' \mid tree(P') = tree(P) \wedge P' \text{ and } P \text{ have the same } S\text{-history}\}$$

□

Now, for a given point P, we will assign truth values to temporal formulae φ at that point. In addition, we assign values to variables, for example the input on a channel, at that point. The assignment function that does both of these is denoted by v_P.

To define v_P, we will need to assign truth values to formulae containing primed variables. Therefore we will also define functions $v_{(P_1,P_2)}$ (where P_1 and P_2 are points and we

[6] In other settings, we might also consider the possibility that a subject S has internal state variables and could use these to make finer distinctions between points. However, in our application, all of the internal processing of the relevant subjects (viz, $\mathscr{H}$ and $\mathscr{D}$) is encoded in the adversary and is thus factored out of the computation tree. We therefore do not lose any needed generality in making this definition.

think of P_2 as being a child of P_1 in some tree) to assign truth values to formulae over a pair of points.

We define v_P and $v_{(P_1,P_2)}$ mutually recursively below. First we present some additional notation.

Notation. Since there is a one-to-one correspondence from trees to adversaries, we can refer to "the adversary corresponding to $tree(P)$". We denote that adversary by $\mathscr{A}(P)$.

We use the notation $succ(P)$ to denote the set of nodes that immediately succeed P in $tree(P)$ (i.e., the children of P).

We use the notation *extensions*(P) to denote the set of infinite sequences of states starting at P in $tree(P)$. □

We now define v_P and $v_{(P_1,P_2)}$. Let P be a point at time k in the execution $\rho = (\alpha, \beta, \gamma)$ in computation tree $T_{\mathscr{A}}$.

- Numbers are assigned to number names.
- Members of I and O are assigned to their syntactic identifiers.
- For any channel $c \in C$,

  $v_P(c_{in}) \triangleq \alpha(c,k)$

- For any channel $c \in C$,

  $v_P(c_{out}) \triangleq \beta(c,k)$

- For any variable name, X, excluding channel variables (such as c_{in} or c_{out})

  $v_P(X) \triangleq \gamma(X,k)$

- Members of $\mathbb{R}$, I, and O are assigned values at a pair of points by referring to their values in the first of the points, e.g.,

  $v_{(P_1,P_2)}(0.5) \triangleq v_{P_1}(0.5)$

  In contrast, variables may change their value from one point to the next, so unprimed variables are evaluated by referring to the first point, e.g., for a state variable X,

  $v_{(P_1,P_2)}(X) \triangleq v_{P_1}(X)$

  whereas primed variables are evaluated by referring to the second point, e.g.,

  $v_{(P_1,P_2)}(X') \triangleq v_{P_2}(X)$

- Composite terms are assigned values at a pair of points by evaluating the constituent parts at the same pair of points and applying the corresponding semantic operator, e.g.,

  $v_{(P_1,P_2)}(X' + Y) \triangleq v_{(P_1,P_2)}(X') + v_{(P_1,P_2)}(Y)$

- Similary, nonmodal formulae are assigned truth values at a pair of points by evaluating the constituent parts at the same pair of points, e.g.,

  $v_{(P_1,P_2)}(X \leq Y) =$ **true** iff $v_{(P_1,P_2)}(X) \leq v_{(P_1,P_2)}(Y)$

  and

  $v_{(P_1,P_2)}(\varphi \wedge \psi) =$ **true** iff $v_{(P_1,P_2)}(\varphi) =$ **true** and $v_{(P_1,P_2)}(\psi) =$ **true**

- To interpret the *probability* of a nonmodal formula φ at a point P, we will take the set of all pairs of points, (P_1, P_2) where P_1 is in $\mathscr{S}_{S,P}$ and P_2 emanates from P_1. Restricting to this set, we compute the probability of those pairs such that $v_{(P_1,P_2)}(\varphi)$ evaluates to true. More precisely, for any nonmodal formula, φ, and for any subject $S \subseteq C$,

  $v_P(Pr_S(\varphi)) \triangleq \mu_{\mathscr{A}(P)}(\mathscr{S}_{S,P}(\varphi) \mid \mathscr{S}_{S,P})$

  where

  $\mathscr{S}_{S,P}(\varphi) \triangleq \{P_2 \mid \exists P_1 \in \mathscr{S}_{S,P}$ such that $P_2 \in succ(P_1)$ and $v_{(P_1,P_2)}(\varphi) =$ **true** $\}$

- An atomic formula, φ, is true at a point, P, iff it is true for all pairs of points emanating from P. More precisely,

  $v_P(\varphi) =$ **true** iff $\forall P' \in succ(P),\ v_{(P,P')}(\varphi) =$ **true**

  (Since we have not needed to include quantification in our language we are free to use '∀' and '∃' as metalinguistic shorthand.)
- For any formula, φ,

  $v_P(\Box\varphi) =$ **true** iff $\forall\ \rho \in$ *extensions*(P), $\forall\ iv_{(\rho,i)}(\varphi) =$ **true**

- Composite formulae are assigned truth values at points in the natural way.
  For example,

  $v_P(\varphi \wedge \psi) =$ **true** iff $v_P(\varphi) =$ **true** and $v_P(\psi) =$ **true**

- Our knowledge operator is an **S5** modal operator and is given semantics in terms of the accessibility relation (on points) in the standard way; viz, for any two points, P_1 and P_2 (not necessarily in distinct trees) and any subject, $S \subseteq C$, we say that P_2 is *accessible* from P_1, denoted '$\kappa_S(P_1, P_2)$' if and only if P_1 and P_2 have the same S-history; further, we use these accessibility relations to assign truth values to formulae of the form $K_S(\varphi)$ as follows.

  $v_P(K_S(\varphi)) =$ **true** iff $\forall P',\ \kappa_S(P,P')$ implies $v_{P'}(\varphi) =$ **true**

In the remainder of the paper, for a model $M = \langle \mathbb{R}, +, \cdot, \leq, W, \mathscr{T}, C, I, O, v,\ \kappa_1, \ldots, \kappa_{|\mathscr{P}(C)|} \rangle$, formula φ, and set of formulae Γ, we will use '$M \models \varphi$' to indicate that φ evaluates to true at the roots of all trees in $\mathscr{T}$ and $M \models \Gamma$ to indicate that all members of Γ evaluate to true at the roots of all trees in $\mathscr{T}$.[7] Finally, we will use '$\Gamma \models \varphi$' to indicate that $M \models \Gamma$ implies $M \models \varphi$ for every model M.

[7] Typically, semantics for modal logics treat truth in a model as truth in all possible worlds in that model. Those more familiar with this usage than with ours should note that on a computational view the primary notion is that of a run rather than a world (state). Thus, truth in a model is more naturally thought of as truth in all runs in that model (hence, at the initial state of all runs).

5 Soundness

In Sect. 6 and Sect. 7 below we give two syntactic characterizations of security and show that the semantic interpretations of our syntactic characterizations are equivalent to certain previously developed definitions. However, the significance of these results is greatly reduced unless the logic is sound. For, without soundness there is no guarantee that any formal proof of security implies any independently motivated notion of security. A soundness theorem gives us just such a correspondence.

Theorem 5.1 *[Soundness] Given a set of formulae of our language Γ and a formula φ,*

If $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$.

□

Proof. In order to prove soundness we must show that the axioms are valid and the rules are truth preserving (except **Nec** which need only be theorem preserving). For most of the axioms and all of the rules the results are completely standard. (Cf. Chellas [7] and Goldblatt [16].) Hence, we do not set them out here. We specifically assumed a semantics in which all the rules and axioms concerning logical connectives preserve soundness. Since we assume the real numbers are part of our models, the axioms concerning them must all be valid. Likewise, because the $Pr(\varphi)$ terms are interpreted as conditional probabilities of events, the **RV** axioms are valid in our semantics since they reflect basic facts about probability measures. The accessibility relations, set out above in Sect. 4, are clearly equivalence relations. Thus, by a standard result of modal logic, the **S5** axioms are all valid and **Nec** (for the knowledge operators) is theorem preserving (cf. [7]). The temporal reasoning axioms are similarly valid and **Nec** for the temporal operator is theorem preserving based on the time structure of our model of computation (cf. [16]).

All that remains is to show the soundness of our two additional axioms. To show the validity of **K**□, let P_1 be a point where

$$v_{P_1}(\Box(K_S\varphi)) = \textbf{false}$$

Then, by our definition of the semantic assignment function, there exist P_2, ρ_2, and k_2 such that the following three conditions hold.

$$\kappa_S(P_1, P_2) \tag{2}$$

$$\rho_2 \in \mathit{extensions}(P_2) \tag{3}$$

$$v_{(\rho_2,k_2)}(\varphi) = \textbf{false} \tag{4}$$

Now, (ρ_2, k_2) is a point in an extension of P_2. Hence, Equation 4 implies

$$v_{P_2}(\Box\varphi) = \textbf{false} \tag{5}$$

which, along with Formula 2, implies

$$v_{P_1}(K_S(\Box\varphi)) = \textbf{false}$$

and **K**□ is valid.

To show the validity of **KPr**, we'll assume P_1 is a point such that

$$v_{P_1}(K_S(Pr_C(\varphi))) = r \tag{6}$$

and show that

$$v_{P_1}(K_S(Pr_S(\varphi))) = r \tag{7}$$

Applying the semantic assignment function to Equation 6 implies that for all P', $\kappa_S(P_1, P')$ implies

$$\mu_{\mathscr{A}(P')}(\mathscr{S}_{C,P'}(\varphi) \mid \mathscr{S}_{C,P'}) = r \tag{8}$$

To show Equation 7, let P_2 be a point such that $\kappa_S(P_1, P_2)$. By the semantic assignment function, we have the following.

$$v_{P_2}(Pr_S(\varphi)) = \mu_{\mathscr{A}(P_2)}(\mathscr{S}_{S,P_2}(\varphi) \mid \mathscr{S}_{S,P_2}) \tag{9}$$

By the definition of conditional probability and the additive property of probability measures, we can expand the right-hand side of Equation 9 to get:

$$v_{P_2}(Pr_S(\varphi)) = \frac{\sum_{P'} \mu_{\mathscr{A}(P_2)}(\mathscr{S}_{S,P_2}(\varphi) \mid \mathscr{S}_{C,P'})\mu_{\mathscr{A}(P_2)}(\mathscr{S}_{C,P'})}{\mu_{\mathscr{A}(P_2)}(\mathscr{S}_{S,P_2})} \tag{10}$$

(where the summation is taken over all P' such that P' is in the same tree and has the same S-history as P_2).

Limiting $\mathscr{S}_{S,P_2}(\varphi)$ to those points emanating from $\mathscr{S}_{C,P'}$ results in $\mathscr{S}_{C,P'}(\varphi)$, so Equation 10 can be rewritten as:

$$v_{P_2}(Pr_S(\varphi)) = \frac{\sum_{P'} \mu_{\mathscr{A}(P_2)}(\mathscr{S}_{C,P'}(\varphi) \mid \mathscr{S}_{C,P'})\mu_{\mathscr{A}(P_2)}(\mathscr{S}_{C,P'})}{\mu_{\mathscr{A}(P_2)}(\mathscr{S}_{S,P_2})} \tag{11}$$

Since P_2 and P' are in the same tree, $\mathscr{A}(P_2) = \mathscr{A}(P')$. Also, since $\kappa_S(P_1, P_2)$, all of the P' in the above equation have the same S-history as P_1. Therefore, by Equation 8,

$$v_{P_2}(Pr_S(\varphi)) = \frac{r \sum_{P'} \mu_{\mathscr{A}(P_2)}(\mathscr{S}_{C,P'})}{\mu_{\mathscr{A}(P_2)}(\mathscr{S}_{S,P_2})} \tag{12}$$

which, again by the additive property of probability measures, implies

$$v_{P_2}(Pr_S(\varphi)) = r \tag{13}$$

and **KPr** is valid, which completes the proof. □

This completes our discussion of the logic itself. In the remainder of the paper we focus on security and applications of the logic thereto.

6 Formal definition of security

In this section, we give our primary definition of security—which we call the *Formal Security Condition* (FSC)—and show that its meaning is equivalent to our own *Probabilistic Noninterference* (PNI) [17], which is itself equivalent to Browne's independently-developed *Stochastic Non-Interference* [3]. As described in [17], PNI is motivated by previous work on Noninterference by Goguen and Meseguer [15] and by connections to information theory. In particular, when the system is modeled as a *two-way channel with memory* ([38]) PNI implies that there is no information flow over the channel from the covert senders to the covert receivers ([17]). (See Browne's [4] for other connections to classical information theory.)

In contrast, the intuition for our definition of security in the present paper derives from an understanding of what the low subject *knows* when using a secure system versus an insecure system. The intuition for our definition is as follows.

> The system under consideration is "secure" iff *before* the low subject receives any given output (b) with any given probability (r), it will already *know* that b is about to occur with probability r.

In essence, since the low subject already knows the probability distribution over its upcoming outputs, it cannot learn any new information when it actually receives those outputs. To make this precise, we introduce the following shorthand.

Notation. Recall that a subject is formalized as a subset of C, the set of Σ's communication channels; this subset represents the subject's view of the system. For a given subject $L = \{l_1, l_2, \ldots l_n\} \subseteq C$ we often require a formula specifying what L receives as output at the next time. This can be specified as

$$(l_1)'_{out} = b_1 \wedge (l_2)'_{out} = b_2 \wedge \ldots \wedge (l_n)'_{out} = b_n$$

(where $b_i \in O$ for $1 \leq i \leq n$). We will specify this more compactly as

$$L'_{out} = b_L$$

(where $L \subseteq C$ is the subject equal to $\{l_1, l_2, \ldots l_n\}$ and $b_L \in O[L]$ is the output vector equal to $[b_1, b_2, \ldots, b_n]$). □

We now define the Formal Security Condition as follows.

Definition 6.1 *Let $L \subseteq C$ be a subject. Suppose a system Σ is described by a set of formulae in our logic, Γ. We say that Γ satisfies the Formal Security Condition (FSC) with respect to L if and only if, for every $b_L \in O[L]$, the formula*

$$\Box(Pr_L(L'_{out} = b_L) = r \rightarrow K_L(Pr_L(L'_{out} = b_L) = r))$$

is derivable from Γ. □

Note that this definition refers only to L's next output. Nevertheless, we will see below, in Theorem 6.9, that this is sufficient to insure that high behavior has no effect on any L-events, including all future outputs visible to L.

At first glance, this property may appear too strong to be satisfied by useful systems. In particular, the reader may wonder:

> if users know the probability distribution over their outputs before they get them, why would they bother to use the system at all? After all, they won't learn anything by using it.

To see why this is not a concern, we need to keep in mind that the low subject L represents not a single user, but rather, the entire low environment. For example, suppose the system we are analyzing is a two-level database containing unclassified and secret information. In this case, L represents all users and processes that are operating at the unclassified level, including the users and processes involved in entering and updating unclassified data. Thus, an individual low user may not, in practice, know the answer to his query before submitting it, but in principle the information is available to him, since he can (in principle) know the entire history of the low environment, including all low inputs.

On the other hand, the reader may now wonder:

> in what way can a system fail to satisfy FSC? That is, in what case does the low environment *fail* to know the probability distribution on its next output?

The answer is: in precisely those cases where that probability distribution is affected by the high environment. That is, if the high environment can influence the probability with which the low environment gets certain outputs, then the low environment will not *know* that probability distribution (except, e.g., by statistical inference *after* it has received those outputs). Further, from information theory we know this is precisely the situation in which the high environment can send information to the low environment, i.e., this is the situation in which the system has a covert channel.

Now we would like to show that FSC is equivalent to PNI. To do so, we will talk about the "meaning" of FSC, or more precisely, the *semantic interpretation* of FSC, which we define as follows.

Definition 6.2 *We say that Γ satisfies the Semantic Interpretation of FSC with respect to L if and only if, for every $b_L \in O[L]$,*

$$\Gamma \models \Box(Pr_L(L'_{out} = b_L) = r \rightarrow K_L(Pr_L(L'_{out} = b_L) = r))$$

□

To prove the semantic interpretation of FSC is equivalent to PNI, we also need to recast the latter in terms of our model. We do this as follows.

Definition 6.3 *Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be two adversaries that satisfy the Secure Environment Assumption. We will say that $\mathscr{A}_1$ and $\mathscr{A}_2$ agree on L behavior iff there exist $\mathscr{H}_1$, $\mathscr{H}_2$, and $\mathscr{L}$ such that $\mathscr{H}_1$ and $\mathscr{L}$ are the unique probability functions that describe $\mathscr{A}_1$ (as in Definition 2.2) and $\mathscr{H}_2$ and $\mathscr{L}$ are the unique probability functions that describe $\mathscr{A}_2$.* □

Definition 6.4 *Let $S \subseteq C$ be a subject and let e be a set of runs, $\{\rho_i\}$, (not necessarily taken from any one computation tree). We say that e is an S-event if and only if there exists a time $k \in \mathbb{N}^+$ such that for any two runs, ρ_1 and ρ_2, having the same S-history up to time k, $\rho_1 \in e$ iff $\rho_2 \in e$.*
For an S-event, e, we will refer to the least k such that above condition holds as the length of e. □

Intuitively, e is an S-event if and only if there is some finite time k (i.e., its length) after which S can always determine whether or not e has occurred.

Note that in general, an S-event contains runs from more than one computation tree. Therefore, such "events" will not be measurable in any of our probability spaces. Rather, we think of them as meta events and we will be interested in the measure of the subset of the runs that are contained in a given computation tree. To make this precise, we introduce the following definition.

Definition 6.5 *Given a computation tree, $T_{\mathscr{A}}$, and an S-event, e, the projection of e onto $T_{\mathscr{A}}$, denoted $e_{\mathscr{A}}$, is given by:*

$e_{\cdot\mathscr{A}} \triangleq runs(T_{\cdot\mathscr{A}}) \cap e$

□

When it is clear from context what is meant, we ignore the distinction between meta-events and their projections, e.g., we write '$\mu_{\cdot\mathscr{A}}(e)$' for '$\mu_{\cdot\mathscr{A}}(e_{\cdot\mathscr{A}})$'.

Observation 6.6 Every projection of every S-event is measurable. That is, for any S-event, e, and any computation tree, $T_{\cdot\mathscr{A}}$,

$e_{\cdot\mathscr{A}} \in \mathscr{X}_{\cdot\mathscr{A}}$

This is due to the restriction on S-events that they be observable within some finite time. In particular, the projection of an S-event onto a tree, T, must also be observable within a finite time, and so it must be formable from a finite number of unions and complementations of the generators of T. □

Definition 6.7 *Let Σ be a system with computation trees $\mathscr{T}(\Sigma)$. We say that Σ satisfies Probabilistic Noninterference (PNI) with respect to a subject $L \subseteq C$ iff for any two trees satisfying the Secure Environment Assumption, $T_{\cdot\mathscr{A}}, T_{\cdot\mathscr{A}'} \in \mathscr{T}(\Sigma)$ and any L-event, e, if $\mathscr{A}$ and $\mathscr{A}'$ agree on L behavior, then*

$\mu_{\cdot\mathscr{A}}(e) = \mu_{\cdot\mathscr{A}'}(e)$

□

Now we are in a position to state the main theorem of this section. Before doing so, we state and prove a lemma.

Lemma 6.8 *If $T_{\cdot\mathscr{A}}$ and $T_{\cdot\mathscr{A}'}$ are two trees such that $\mathscr{A}$ and $\mathscr{A}'$ agree on L behavior (and satisfy the Secure Environment Assumption) then the following two conditions are equivalent.*

1. *For any low output vector, $b_L \in O[L]$, and any two points $P_1 \in T_{\cdot\mathscr{A}}$ and $P_2 \in T_{\cdot\mathscr{A}'}$ such that $\kappa_S(P_1, P_2)$,*

   $\mu_{\cdot\mathscr{A}(P_1)}(\cdot\mathscr{S}_{L,P_1}(L'_{out} = b_L) \mid \cdot\mathscr{S}_{L,P_1}) =$
   $\mu_{\cdot\mathscr{A}(P_2)}(\cdot\mathscr{S}_{L,P_2}(L'_{out} = b_L) \mid \cdot\mathscr{S}_{L,P_2})$

2. *For any L-event, e,*

   $\mu_{\cdot\mathscr{A}}(e_{\cdot\mathscr{A}}) = \mu_{\cdot\mathscr{A}'}(e_{\cdot\mathscr{A}'})$

□

Proof. We begin by observing that $\cdot\mathscr{S}_{L,P_1}(L'_{out} = b_L)$ and $\cdot\mathscr{S}_{L,P_2}(L'_{out} = b_L)$ are projections of the same L-event and that $\cdot\mathscr{S}_{L,P_1}$ and $\cdot\mathscr{S}_{L,P_2}$ are projections of another L-event. Therefore the backward direction of the lemma (i.e., that condition 2 implies condition 1) follows easily. In particular, let $P_1 \in T_{\cdot\mathscr{A}}$ and $P_2 \in T_{\cdot\mathscr{A}'}$ be two points such that $\kappa_S(P_1, P_2)$. Then condition 2 implies

$\mu_{\cdot\mathscr{A}(P_1)}(\cdot\mathscr{S}_{L,P_1}) = \mu_{\cdot\mathscr{A}(P_2)}(\cdot\mathscr{S}_{L,P_2})$

and further—since the intersection of two L-events is again an L-event—that

$\mu_{\cdot\mathscr{A}(P_1)}(\cdot\mathscr{S}_{L,P_1}(L'_{out} = b_L) \cap \cdot\mathscr{S}_{L,P_1}) =$
$\mu_{\cdot\mathscr{A}(P_2)}(\cdot\mathscr{S}_{L,P_2}(L'_{out} = b_L) \cap \cdot\mathscr{S}_{L,P_2})$

Therefore condition 1 holds by the definition of conditional probability.

To prove the forward part of the lemma, we start by showing that it holds for a certain subset of L-events, namely those L-events corresponding to finite L-histories.

Let e be an L-event such that there exists a time, k, (the length of e) and a characteristic run, ρ, such that for any run, ρ',

$\rho' \in e$ iff ρ' has the same L-history as ρ up to time k

That is, e corresponds to the finite L-history characterized by ρ up to time k.

We now prove the forward part of the lemma for this subclass of L-events by induction on the length of e.

Base case. The length of e is zero.

Since all runs have the same L-history up to time 0, the only two L-events of length 0 are the empty set, $\emptyset$, and the set of all runs from all trees, $\mathscr{R}$. In the former case,

$\mu_{\cdot\mathscr{A}}(\emptyset_{\cdot\mathscr{A}}) = 0 = \mu_{\cdot\mathscr{A}'}(\emptyset_{\cdot\mathscr{A}'})$

and in the latter case,

$\mu_{\cdot\mathscr{A}}(\cdot\mathscr{R}_{\cdot\mathscr{A}}) = 1 = \mu_{\cdot\mathscr{A}'}(\cdot\mathscr{R}_{\cdot\mathscr{A}'})$

Thus, the base case is proved.

Induction case. Assume condition 2 holds for all L-events (corresponding to finite L-histories) of length k. Let e be an L-event corresponding to a finite L-history of length $k+1$. Suppose that ρ is a run that (up to time $k+1$) characterizes e.

Now, let $\bar{e}$ be the L-event characterized by ρ up to time k. Intuitively, $\bar{e}$ corresponds to the finite L-history obtained by truncating e at time k. By the induction hypothesis,

$$\mu_{\cdot\mathscr{A}}(\bar{e}_{\cdot\mathscr{A}}) = \mu_{\cdot\mathscr{A}'}(\bar{e}_{\cdot\mathscr{A}'}) \tag{14}$$

If $\mu_{\cdot\mathscr{A}}(\bar{e}_{\cdot\mathscr{A}}) = 0$, then $\mu_{\cdot\mathscr{A}}(e) = 0 = \mu_{\cdot\mathscr{A}'}(e)$ and the induction case holds trivially, so we assume $\mu_{\cdot\mathscr{A}}(\bar{e}_{\cdot\mathscr{A}}) > 0$.

By Equation 14, we also have that $\mu_{\cdot\mathscr{A}'}(\bar{e}_{\cdot\mathscr{A}}) > 0$. Thus, by the definition of conditional probability,

$$\mu_{\cdot\mathscr{A}}(e) = \mu_{\cdot\mathscr{A}}(\bar{e}) \cdot \mu_{\cdot\mathscr{A}}(e \mid \bar{e}) \tag{15}$$

and

$$\mu_{\cdot\mathscr{A}'}(e) = \mu_{\cdot\mathscr{A}'}(\bar{e}) \cdot \mu_{\cdot\mathscr{A}'}(e \mid \bar{e}) \tag{16}$$

Let $\alpha \in I_{L,k}$ and $\beta \in O_{L,k}$ be the low input and output history, resp., that characterize $\bar{e}$ and let $a_L \in I[L]$ and $b_L \in O[L]$ be the low input and output vectors at time $k+1$ that are needed to additionally characterize e. Then, by the construction of our probability measures (as described in Sect. 2.2) and by the Secure Environment Assumption, we have that

$$\mu_{\cdot\mathscr{A}}(e \mid \bar{e}) = \mu_{\cdot\mathscr{A}}(\hat{b_L}, \mid \bar{e}) \cdot \mathscr{S}(a_L \mid \alpha, \beta) \tag{17}$$

and

$$\mu_{\cdot\mathscr{A}'}(e \mid \bar{e}) = \mu_{\cdot\mathscr{A}'}(\hat{b_L} \mid \bar{e}) \cdot \mathscr{S}'(a_L \mid \alpha, \beta) \tag{18}$$

where $\hat{b_L}$ is the meta-event representing that the low output vector at time $k+1$ is b and $\mathscr{S}$ and $\mathscr{S}'$ are the low environments of $\mathscr{A}$ and $\mathscr{A}'$, respectively.

Since $\mathscr{A}$ and $\mathscr{A}'$ agree on L behavior,

$$\mathscr{S}(a_L \mid \alpha, \beta) = \mathscr{S}'(a_L \mid \alpha, \beta) \tag{19}$$

Further, since $\mu_{\cdot\mathscr{A}}(\bar{e})$ and $\mu_{\cdot\mathscr{A}'}(\bar{e})$ are both greater than zero, there are points in both trees, $P_1 \in T_{\cdot\mathscr{A}}$ and $P_2 \in T_{\cdot\mathscr{A}'}$, each of whose L-histories are (α, β). By condition 1,

$$\mu_{\mathscr{A}(P_1)}(\mathscr{S}_{L,P_1}(L'_{out} = \hat{b_L}) \mid \mathscr{S}_{L,P_1}) = \mu_{\mathscr{A}(P_2)}(\mathscr{S}_{L,P_2}(L'_{out} = \hat{b_L}) \mid \mathscr{S}_{L,P_2})$$

But notice that $\mathscr{S}_{L,P_1}$ and $\mathscr{S}_{L,P_2}$ are projections of $\overline{e}$ and $\mathscr{S}_{L,P_1}(L'_{out} = b_L)$ and $\mathscr{S}_{L,P_2}(L'_{out} = b_L)$ are projections of $\hat{b_L}$. Therefore,

$$\mu_{\mathscr{A}}(\hat{b_L} \mid \overline{e}) = \mu_{\mathscr{A}'}(\hat{b_L} \mid \overline{e}) \tag{20}$$

Thus, by Equations 17, 18, 19, and 20, we have that

$$\mu_{\mathscr{A}}(e \mid \overline{e}) = \mu_{\mathscr{A}'}(e \mid \overline{e}) \tag{21}$$

and finally, by Equations 14, 15, 16, and 21, we have that

$$\mu_{\mathscr{A}}(e) = \mu_{\mathscr{A}'}(e)$$

and the induction case is proved.

Now, we can complete the proof by observing that every L-event can be constructed by taking a finite number of unions and complementations of L-events that correspond to finite L-histories. That is, the L-events that correspond to finite L-histories are analogous to the *generators* of our event spaces. Thus, the desired result that $\mu_{\mathscr{A}}(e) = \mu_{\mathscr{A}'}(e)$ for arbitrary L-events follows from the fact that the measures are equal on all of the L-events that correspond to finite L-histories. □

We can now prove the following theorem relating PNI and FSC.

Theorem 6.9 *Let Γ be a set of formulae describing Σ and let $L \subseteq C$ be a subject. Then, Σ satisfies PNI with respect to L iff Γ satisfies the semantic interpretation of FSC with respect to L.* □

Proof. Let $M = \langle \mathbb{R}, +, \cdot, \leq, W, \mathscr{T}, C, I, O, v, \kappa_1, \ldots, \kappa_{|\mathscr{P}(C)|} \rangle$ be a model such that $M \models \Gamma$.

Γ satisfies the semantic interpretation of FSC (wrt L) iff for every $b_L \in O[L]$,

$$M \models \Box(Pr_L(L'_{out} = b_L) = r \rightarrow K_L(Pr_L(L'_{out} = b_L) = r))$$

which holds iff for every $b_L \in O[L]$ and every root, P, of any tree in $\mathscr{T}$,

$$v_P(\Box(Pr_L(L'_{out} = b_L) = r \rightarrow K_L(Pr_L(L'_{out} = b_L) = r))) = \textbf{true}$$

which, by applying the semantic assignment function, holds iff for every $b_L \in O[L]$ and every two points $P_1, P_2 \in W$ such that $\kappa_L(P_1, P_2)$,

$$v_{P_1}(Pr_L(L'_{out} = b_L) = r) = \textbf{true} \quad \text{implies}$$
$$v_{P_2}(Pr_L(L'_{out} = b_L) = r) = \textbf{true}$$

which holds iff for every $b_L \in O[L]$ and every two points $P_1, P_2 \in W$ such that $\kappa_L(P_1, P_2)$,

$$v_{P_1}(Pr_L(L'_{out} = b_L)) = v_{P_2}(Pr_L(L'_{out} = b_L))$$

which, again by the semantic assignment function, holds iff for every $b_L \in O[L]$ and every two points $P_1, P_2 \in W$ such that $\kappa_L(P_1, P_2)$,

$$\mu_{\mathscr{A}(P_1)}(\mathscr{S}_{L,P_1}(L'_{out} = b_L) \mid \mathscr{S}_{L,P_1}) = \mu_{\mathscr{A}(P_2)}(\mathscr{S}_{L,P_2}(L'_{out} = b_L) \mid \mathscr{S}_{L,P_2}) \tag{22}$$

It is therefore sufficient to show that Σ satisfies PNI iff Equation 22 holds. That Equation 22 implies PNI follows easily from Lemma 6.8. In particular, let $\mathscr{A}$ and $\mathscr{A}'$ be two adversaries that agree on L behavior. From Equation 22, Lemma 6.8 implies that for any L-event e,

$$\mu_{\mathscr{A}}(e) = \mu_{\mathscr{A}'}(e)$$

Hence, Σ satisfies PNI.

To show the reverse direction, assume Σ satisfies PNI, let $b_L \in O[L]$ be arbitrary, and let $P_1, P_2 \in W$ be two points such that $\kappa_L(P_1, P_2)$.

Note that we cannot apply PNI immediately, since it may be the case that $\mathscr{A}(P_1)$ and $\mathscr{A}(P_2)$ do not agree on L behavior. For this reason, for each point P, we construct a new adversary $\overline{\mathscr{A}}(P)$ as follows. (Note that $\overline{\mathscr{A}}(P)$ is an adversary corresponding to a tree that does not, in general, contain P.)

Suppose $P = (\alpha, \beta, k)$ and $\mathscr{A}(P) = (\mathscr{H}, \mathscr{L})$. Then $\overline{\mathscr{A}}(P) = (\mathscr{H}, \overline{\mathscr{L}})$, where $\overline{\mathscr{L}}$ is defined as follows.

For $k' \leq k$, $\alpha' \in I_{L,k'}$, and $\beta' \in O_{L,k'}$,

$$\overline{\mathscr{L}}(a \restriction L \mid \alpha', b') = \begin{cases} 1, & \text{if } a \restriction L = (\alpha, k') \restriction L; \\ 0, & \text{otherwise.} \end{cases}$$

For $k' > k$, $\alpha' \in I_{L,k'}$, and $\beta' \in O_{L,k'}$,

$$\overline{\mathscr{L}}(a \restriction L \mid \alpha', b') = \begin{cases} 1, & \text{if } a \restriction L = a_0; \\ 0, & \text{otherwise.} \end{cases}$$

(where a_0 is some constant input on channels in L). That is, $\overline{\mathscr{L}}$ blindly and deterministically follows $\alpha \restriction L$ up to time k and then blindly and deterministically inputs a_0 from then on.

Now, note that according to our construction of the computation trees, there is a point $P = (\alpha, \beta, k)$ in a given tree T if and only if at each point leading up to P, say $P' = (\alpha, \beta, k')$ ($k' < k$), the following three conditions all hold.

$$\mathscr{O}(\beta(k'+1) \mid \alpha \restriction k', \beta \restriction k') > 0 \tag{23}$$

$$\mathscr{H}(\alpha(k'+1) \restriction (C - L) \mid \alpha \restriction k', \beta \restriction k') > 0 \tag{24}$$

$$\mathscr{L}(\alpha(k'+1) \restriction L \mid \alpha \restriction k' \restriction L, \beta \restriction k' \restriction L) > 0 \tag{25}$$

(where $\alpha \restriction k'$ and $\beta \restriction k'$ are the restrictions of α and β to $I_{C,k'}$ and $O_{C,k'}$, respectively.

Further, since in constructing $T_{\overline{\mathscr{A}}(P)}$ we have retained the output probability function $\mathscr{O}$ and high behavior $\mathscr{H}$ used in $T_{\mathscr{A}(P)}$ and chosen $\overline{\mathscr{L}}$ to ensure Equation 25 holds appropriately, there is a point $\overline{P} \in T_{\overline{\mathscr{A}}(P)}$ that has the identical I/O history as $P \in T_{\mathscr{A}(P)}$. Further, for any point $P' \in T_{\mathscr{A}(P)}$ having the same L-history as P, there is a corresponding point $\overline{P'} \in T_{\overline{\mathscr{A}}(P)}$ with the same I/O history as P'.

Finally, note that by our construction of the computation trees, since P' and $\overline{P'}$ have the same I/O history, say (α', β'), the sum of the probabilities on arcs emanating from P', where $L'_{out} = b_L$ is equal to the sum of the probabilities on arcs emanating from $\overline{P'}$, where $L'_{out} = b_L$. In particular, they are both equal to

$$\sum_{b \,\restriction\, L = b_L} \mathscr{O}(b \mid \alpha', \beta')$$

Since this is the case for all points in $T_{\mathscr{A}(P)}$ having the same L-history as P, we have that

$$\mu_{\mathscr{A}(P)}(\mathscr{S}_{L,P}(L'_{out} = b_L) \mid \mathscr{S}_{L,P}) = \mu_{\overline{\mathscr{A}}(P)}(\mathscr{S}_{L,\overline{P}}(L'_{out} = b_L) \mid \mathscr{S}_{L,\overline{P}}) \quad (26)$$

In particular, Equation 26 holds for both P_1 and P_2.

Now note that $\overline{\mathscr{A}}(P_1)$ and $\overline{\mathscr{A}}(P_2)$ agree on L behavior. Therefore, since $\mathscr{S}_{L,P}(L'_{out} = b_L) \cap \mathscr{S}_{L,P}$ and $\mathscr{S}_{L,P}$ are L-events, PNI implies

$$\mu_{\overline{\mathscr{A}}(P_1)}(\mathscr{S}_{L,\overline{P_1}}(L'_{out} = b_L) \mid \mathscr{S}_{L,\overline{P_1}}) = \mu_{\overline{\mathscr{A}}(P_2)}(\mathscr{S}_{L,\overline{P_2}}(L'_{out} = b_L) \mid \mathscr{S}_{L,\overline{P_2}}) \quad (27)$$

Finally, Equations 26 and 27 imply Equation 22, which completes the proof. □

The significance of this theorem is that (given soundness) verifying that a system satisfies FSC is sufficient to show that it satisfies PNI, which (as was previously mentioned) is a necessary and sufficient condition for a system to be free of covert channels. In the next section, we discuss the issue of verifying FSC.

7 Verification

7.1 Syntactic statement

In [30], McLean defines the Flow Model (FM) with the motivation of providing an abstract, but precise, explication of information flow security. McLean's intent for FM is to provide a characterization of security against which more concrete security models can be evaluated. In [17], the first of the present authors studies a more concrete version of FM, called the Applied Flow Model (AFM), and shows that AFM captures a strictly stronger notion of security than PNI.

In this paper, we have another reason for studying AFM: it is more easily verified than FSC.

Definition 7.1 *Let $L \subseteq C$ be a subject. Suppose Γ is a set of premises that describe a system Σ. We say that Γ satisfies the Syntactic Verification Condition (SVC) with respect to L if and only if, for every $b_L \in O[L]$, the formula*

$$\Box(Pr_C(L'_{out} = b_L) = r \rightarrow K_L(Pr_C(L'_{out} = b_L) = r))$$

is derivable from Γ. □

Intuitively, SVC says that at all times, the low environment knows the objective probability distribution on its next output.

In the next section, we show this statement is equivalent to a statement about conditional statistical independence. Namely, conditioned on the previous L-history, the next output on L is statistically independent of the previous non-L (i.e., high) history.

7.2 Relationship to AFM

In this section we show that $\Gamma \models$ SVC if and only if the system specified by Γ satisfies AFM (i.e., the relationship between SVC and AFM is analogous to the relationship between FSC and PNI).

Definition 7.2 *Let Σ be a system with computation trees $\mathscr{T}(\Sigma)$ and let $L \subseteq C$ be a subject. We will say that Σ satisfies the Applied Flow Model (AFM) with respect to L iff for any tree, $T_{\mathscr{A}} \in \mathscr{T}(\Sigma)$ (satisfying the Secure Environment Assumption with respect to L), any point $P \in T_{\mathscr{A}}$, and any low output vector, $b_L \in O[L]$,*

$$\mu_{\mathscr{A}}(\mathscr{S}_{C,P}(L'_{out} = b_L) \mid \mathscr{S}_{C,P}) = \mu_{\mathscr{A}}(\mathscr{S}_{L,P}(L'_{out} = b_L) \mid \mathscr{S}_{L,P}) \quad (28)$$

□

This definition is, except for notational differences, exactly the definition of AFM as given in [17]. Now we can prove the following theorem.

Theorem 7.3 *Let Γ be a set of formulae describing Σ and let $L \subseteq C$ be a subject. Then, Σ satisfies AFM with respect to L iff Γ satisfies the semantic interpretation of SVC with respect to L.* □

Proof. Let $M = \langle \mathbb{R}, +, \cdot, \leq, W, \mathscr{T}, C, I, O, v, \kappa_1, \ldots, \kappa_{|\mathscr{P}(C)|} \rangle$ be a model such that $M \models \Gamma$.

Γ satisfies the semantic interpretation of SVC (wrt L) iff for every $b_L \in O[L]$,

$$M \models \Box(Pr_C(L'_{out} = b_L) = r \rightarrow K_L(Pr_C(L'_{out} = b_L) = r))$$

which holds iff for every $b_L \in O[L]$ and every root, P, of any tree in $\mathscr{T}$,

$$v_P(\Box(Pr_C(L'_{out} = b_L) = r \rightarrow K_L(Pr_C(L'_{out} = b_L) = r))) = \textbf{true}$$

which, by applying the semantic assignment function, holds iff for every $b_L \in O[L]$ and every two points $P_1, P_2 \in W$ such that $\kappa_L(P_1, P_2)$,

$$v_{P_1}(Pr_C(L'_{out} = b_L) = r) = \textbf{true} \quad \text{implies}$$
$$v_{P_2}(Pr_C(L'_{out} = b_L) = r) = \textbf{true}$$

which holds iff for every $b_L \in O[L]$ and every two points $P_1, P_2 \in W$ such that $\kappa_L(P_1, P_2)$,

$$v_{P_1}(Pr_C(L'_{out} = b_L)) = v_{P_2}(Pr_C(L'_{out} = b_L))$$

which, again by the semantic assignment function, holds iff for every $b_L \in O[L]$ and every two points $P_1, P_2 \in W$ such that $\kappa_L(P_1, P_2)$,

$$\mu_{\mathscr{A}(P_1)}(\mathscr{S}_{C,P_1}(L'_{out} = b_L) \mid \mathscr{S}_{C,P_1}) = \mu_{\mathscr{A}(P_2)}(\mathscr{S}_{C,P_2}(L'_{out} = b_L) \mid \mathscr{S}_{C,P_2}) \quad (29)$$

Now, to show that the semantic interpretation of SVC implies AFM, we assume Equation 29 and show that Equation 28 holds. Consider the right-hand side of Equation 28.

$$\mu_{\mathscr{A}}(\mathscr{S}_{L,P}(L'_{out} = b_L) \mid \mathscr{S}_{L,P})$$

By the definition of conditional probability and the additive property of probability measures, this is equal to:

$$\frac{\sum_{P'} \mu_{\mathscr{A}}(\mathscr{S}_{L,P}(L'_{out} = b_L) \mid \mathscr{S}_{C,P'}) \mu_{\mathscr{A}}(\mathscr{S}_{C,P'})}{\mu_{\mathscr{A}}(\mathscr{S}_{L,P})}$$

(where the summation is taken over all P' such that P' is in the same tree and has the same L-history as P).
Limiting $\mathscr{S}_{L,P}(L'_{out} = b_L)$ to those points emanating from $\mathscr{S}_{C,P'}$ results in $\mathscr{S}_{C,P'}(L'_{out} = b_L)$, so the above is equal to:

$$\frac{\sum_{P'} \mu_{\mathscr{A}}(\mathscr{S}_{C,P'}(L'_{out} = b_L) \mid \mathscr{S}_{C,P'})\mu_{\mathscr{A}}(\mathscr{S}_{C,P'})}{\mu_{\mathscr{A}}(\mathscr{S}_{L,P})}$$

Now, by Equation 29, $\mu_{\mathscr{A}}(\mathscr{S}_{C,P'}(L'_{out} = b_L) \mid \mathscr{S}_{C,P'}) = \mu_{\mathscr{A}}(\mathscr{S}_{C,P}(L'_{out} = b_L) \mid \mathscr{S}_{C,P})$ for all P' having the same L-history as P, so the above is equal to

$$\frac{\mu_{\mathscr{A}}(\mathscr{S}_{C,P}(L'_{out} = b_L) \mid \mathscr{S}_{C,P}) \sum_{P'} \mu_{\mathscr{A}}(\mathscr{S}_{C,P'})}{\mu_{\mathscr{A}}(\mathscr{S}_{L,P})}$$

which, again by the additive property of probability measures, is equal to

$$\mu_{\mathscr{A}}(\mathscr{S}_{C,P}(L'_{out} = b_L) \mid \mathscr{S}_{C,P})$$

which is precisely the left-hand side of Equation 28 and therefore, the system satisfies AFM.

To show that AFM implies the semantic interpretation of SVC, we assume Equation 28, let $b_L \in O[L]$ be arbitrary, and let $P_1, P_2 \in W$ be such that $\kappa_L(P_1, P_2)$. We want to show that Equation 29 holds; however, P_1 and P_2 may not be in the same computation tree, so we cannot apply Equation 28 directly. We therefore define an adversary $\mathscr{A}_0$ such that $T_{\mathscr{A}_0}$ is guaranteed to contain two points P'_1 and P'_2 such that P'_1 has the same C-history as P_1 and P'_2 has the same C-history as P_2.

Suppose $\mathscr{A}(P_1) = (\mathscr{H}_1, \mathscr{S}_1)$ and $\mathscr{A}(P_2) = (\mathscr{H}_2, \mathscr{S}_2)$. We define $\mathscr{A}_0$ to be $(\mathscr{H}_0, \mathscr{S}_0)$, where for all $b_H \in I[C-L]$, $\alpha \in I_{C,k}$, and $\beta \in O_{C,k}$,

$$\mathscr{H}_0(b_H \mid \alpha, \beta) = \frac{1}{2}\mathscr{H}_1(b_H \mid \alpha, \beta) + \frac{1}{2}\mathscr{H}_2(b_H \mid \alpha, \beta)$$

and for all $a_L \in I[C - L]$, $\alpha \in I_{L,k}$, and $\beta \in I_{L,k}$,

$$\mathscr{S}_0(a_L \mid \alpha, \beta) = \frac{1}{2}\mathscr{S}_1(b_H \mid \alpha, \beta) + \frac{1}{2}\mathscr{S}_2(b_H \mid \alpha, \beta)$$

Since all arcs leading to P_1 (resp., P_2) are labelled with positive probabilities, there will be corresponding positive-probability arcs in $T_{\mathscr{A}_0}$ leading up to a point P'_1 (resp., P'_2) with the same C-history.

Note that the probabilities of reaching P_1 and P_2 will, in general, be different than the probabilities of reaching P'_1 and P'_2, respectively. However, due to our construction of the computation trees, from any given point, the conditional probability of receiving a particular output at the next time step is determined solely by the system (and not by the adversary). Therefore, we have the following.

$$\mu_{\mathscr{A}(P_1)}(\mathscr{S}_{C,P_1}(L'_{out} = b_L) \mid \mathscr{S}_{C,P_1}) = \mu_{\mathscr{A}_0}(\mathscr{S}_{C,P'_1}(L'_{out} = b_L) \mid \mathscr{S}_{C,P'_1}) \quad (30)$$

$$\mu_{\mathscr{A}(P_2)}(\mathscr{S}_{C,P_2}(L'_{out} = b_L) \mid \mathscr{S}_{C,P_2}) = \mu_{\mathscr{A}_0}(\mathscr{S}_{C,P'_2}(L'_{out} = b_L) \mid \mathscr{S}_{C,P'_2}) \quad (31)$$

Now, we can apply Equation 28 to the right-hand sides of Equations 30 and 31; in particular, since P'_1 and P'_2 have the same L-history, both right-hand sides are equal to $\mu_{\mathscr{A}_0}(\mathscr{S}_{L,P'_1}(L'_{out} = b_L) \mid \mathscr{S}_{L,P'_1})$. Therefore, Equations 28, 30, and 31 imply

$$\mu_{\mathscr{A}(P_1)}(\mathscr{S}_{C,P_1}(L'_{out} = b_L) \mid \mathscr{S}_{C,P_1}) = \mu_{\mathscr{A}(P_2)}(\mathscr{S}_{C,P_2}(L'_{out} = b_L) \mid \mathscr{S}_{C,P_2})$$

and the proof is complete. □

7.3 FSC versus SVC

We introduced SVC by claiming it is easier to formally verify than FSC. To see why, consider the structure of the formulae that need to be derived in verifying FSC, viz,

$$\Box(Pr_L(L'_{out} = b_L) = r \rightarrow K_L(Pr_L(L'_{out} = b_L) = r))$$

The primary difficulty with deriving such a formula is that it requires us to reason about L's *subjective* probabilities (i.e., formulae of the form $Pr_L(\varphi)$, where $L \neq C$). We expect systems will typically be described entirely in terms of objective probabilities (i.e., where all probability formulae are of the form $Pr_C(\varphi)$). Therefore, deriving a formula in the above form requires us to reason about how various objective probabilities give rise to other subjective probabilities. This is a topic we have not pursued in any depth in the present work. (In fact, the reader may note there is only one axiom in the present logic addressing the interaction between objective and subjective probabilities, viz, **KPr**.) However, our intuition is that the relationship is closely related to the Secure Environment Assumption.

There are two special cases of verifying FSC that are worth pointing out. First, in the case where we can derive

$$\Box\,(Pr_L(L'_{out} = b_L) = r \rightarrow Pr_C(L'_{out} = b_L) = r) \quad (32)$$

verifying FSC reduces to the problem of verifying SVC. That is, if (as part of verifying SVC) we have derived

$$\Box\,(Pr_C(L'_{out} = b_L) = r \rightarrow K_L(Pr_C(L'_{out} = b_L) = r)) \quad (33)$$

then we can use Formulae 32 and 33 in conjuction with Axiom **KPr** to conclude

$$\Box(Pr_L(L'_{out} = b_L) = r \rightarrow K_L(Pr_L(L'_{out} = b_L) = r))$$

and thus prove FSC. Of course, in such cases we can also simply verify SVC and avoid the extra work of verifying Formula 32.

The other special case is when the system behavior can be described without any reference to inputs. In this case, if SVC is provable, then it will be based on the truth value of the consequent of SVC without concern for the antecedent. Because of the **KPr** axiom, proving FSC (by showing the truth of the consequent) will then be exactly as easy as proving SVC.

7.4 Examples, continued

We note here that the security of the encryption box of Example 3.1 with respect to a subject $L \subseteq C$ is formally derivable using SVC. Recall the system specification: If $C = \{h, l\}$, $I = \{0, 1\}$, and $O = \{0, 1\}$, then, the system is specified by the following formula.

$$\Box\left(Pr_C(l'_{out} = 0) = 0.5 \wedge Pr_C(l'_{out} = 1) = 0.5\right)$$

Recall also that subjects are assumed to know the system description. Thus,

$$\Gamma = \{K_L \Box\left(Pr_C(l'_{out} = 0) = 0.5 \wedge Pr_C(l'_{out} = 1) = 0.5\right)\}$$

The only $b_L \in O[L]$ are 0 and 1; hence, the only antecedents for the SVC schema that are consistent with Γ are

$Pr_C(L'_{out} = 0)$ and $Pr_C(L'_{out} = 1)$. Thus, SVC with respect to L for this system consists of the following two formulae.

$\Box(Pr_C(L'_{out} = 0) = 0.5 \rightarrow K_L(Pr_C(L'_{out} = 0) = 0.5))$
$\Box(Pr_C(L'_{out} = 1) = 0.5 \rightarrow K_L(Pr_C(L'_{out} = 1) = 0.5))$

Each of these is derivable from Γ using propositional reasoning, Modus Ponens and Necessitation and, axioms **K**, **4**, and **K**$\Box$. Further, since SVC is stronger than FSC, such a proof is sufficient to show this system satisfies FSC. (In the typical case one would proceed through SVC to prove FSC, as we have done. As noted above, however, for special cases such as this it is equally easy to derive FSC directly.)

We also observe that for the insecure encryption box of Example 3.2 $\Gamma \nvdash FSC$ (where Γ includes those formulae that describe the system as well as the assumptions about knowledge thereof). It is obvious that the insecure encryption box fails to satisfy PNI. By the attack described in the original example, we can easily find two adversaries that satisfy the Secure Environment Assumption and agree on low behavior; yet, disagree on the probability of certain low events. Indeed, the low environment can assign $0/1$ probabilities to any output sent by the high part of the adversary. By theorem 6.9, we thus have that $\Gamma \not\models FSC$. And, by soundness (theorem 5.1), it follows that $\Gamma \nvdash FSC$.

8 Conclusions

We have given a logic for specifying and reasoning about the multilevel security of probabilistic computer systems. We have established connections between information-theoretic formulations of security and logical formulations of knowledge and probability in distributed systems.

To date, we have only been able to specify and verify toy systems using our logic. Our SVC takes one small step towards practically verifiable security. However, it is unlikely that one could ever use FSC, or even SVC, for verifying real systems since real multilevel-secure systems (e.g., as in Karger et al. [23]) are too complex to be completely free of covert channels, even at the specification level (e.g., as in Browne [5]). Therefore, they cannot satisfy our ideal notions of security. Nevertheless, we feel it is important to cast ideal security in a precise logical framework. It is our hope that extensions of this work—using less ideal notions of security allowing some limited information flow—will ultimately lead to machine checkable proofs of security for real systems.

Acknowledgements. Portions of this work appeared in [18] and [41]. We would like to thank Jeremy Jacob, Mike Kaminski, John McLean, Catherine Meadows, and Ira Moskowitz for helpful comments on the first of those papers. We would also like to thank the anonymous referees for their careful reading and helpful comments on the previous draft.

References

1. Bieber P, Cuppens F: A definition of secure dependencies using the logic of security. In Proc. Computer Security Foundations Workshop IV, Franconia, NH, June 1991
2. Bieber P, Cuppens F: A logical view of secure dependencies. J Comput Security, 1:99–129 (1992)
3. Browne R: Stochastic Non-Interference: Temporal Stochastic Processes Without Covert Channels. Odyssey Research Associates, Ithaca, NY, November 1989. unpublished manuscript
4. Browne R: The Turing Test and non-information flow. In Proc. 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1991
5. Browne R: Mode security: An infrastructure for covert channel suppression. In Proc. 1994 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1994
6. Chandy KM, Misra J: How processes learn. Distrib Comput, 1:40–52 (1986)
7. Chellas BF: Modal Logic: An Introduction. Cambridge University Press, Cambridge 1980
8. Cuppens F: A logical analysis of authorized and permitted information flows. In Proc. 1993 IEEE Computer Society Symposium on Research in Security and Privacy, pages 100–109, Oakland, CA, May 1993
9. Denning DE: A lattice model of secure information flow. Communications of the ACM, 19(5):236–243, May 1976
10. Denning DE: Cryptography and Data Security. Addison-Wesley, Reading, MA 1982
11. Dolev D, Dwork C, Waarts O, Yung M: Perfectly secure message transmission. JACM, 40(1):17–47, January 1993
12. Fagin R, Halpern JY: Reasoning about knowledge and probability. JACM, 41(2):340–367, March 1994
13. Gasser M: Building a Secure Computer System. Van Nostrand Reinhold, New York 1988
14. Glasgow J, MacEwen G, Panangaden P: A logic for reasoning about security. ACM Transactions on Computer Systems, 10(3):226–264, August 1992
15. Goguen JA, Meseguer J: Security policies and security models. In Proceedings of the 1982 IEEE Computer Society Symposium on Security and Privacy, Oakland, CA 1982
16. Goldblatt R: Logics of Time and Computation (2nd edn) Vol 7 of CSLI Lecture Notes. CSLI Publications, Stanford, CA 1992
17. Gray, III JW: Toward a mathematical foundation for information flow security. J Comput Security 1:255–294 (1992) A preliminary version appears in Proc. 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1991.
18. Gray, III JW, Syverson PF: A logical approach to multilevel security of probabilistic systems. In Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy, pp 164–176, Oakland, CA, May 1992
19. Gray, III JW, Syverson PF: Epistemology of information flow in the multilevel security of probabilistic systems. Technical Report NRL/MR/5540–95-7733, Naval Research Laboratory, May 1995
20. Halmos PR: Measure Theory. Springer, New York 1950
21. Halpern JY: Using reasoning about knowledge to analyze distributed systems. In Traub J, Grosz B, Lampson B, Nilson N (eds) Annu Rev Comput Sci 2:37–68 (1987)
22. Halpern JY, Tuttle MR: Knowledge, probability, and adversaries. JACM, 40:917–962 (1993)
23. Karger PA, Zurko ME, Bonin DW, Mason AH, Kahn CE: A retrospective on the vax vmm security kernel. IEEE Trans Softw Eng, 17:1147–1165 (1991)
24. Lamport L: The temporal logic of actions. Technical Report 79, DEC Systems Research Center, Palo Alto, CA, December 1991
25. Lampson BW: A note on the confinement problem. Communications of the ACM, 16(10), October 1973
26. L.Marcus, Redmond T: A model-theoretic approach to specifying, verifying, and hooking up security policies. In Proceeding of the Computer Security Foundations Workshop, Franconia, NH, 1988
27. Marsden JE: Elementary Classical Analysis. Freeman, San Francisco, 1974
28. McCullough D: Noninterference and the composability of security properties. In Proceedings of the 1988 IEEE Computer Society Symposium on Security and Privacy, Oakland, CA, 1988
29. McCullough D: A hookup theorem for multilevel security. IEEE Transactions on Software Engineering, 16(6):563–568, June 1990
30. McLean J: Security models and information flow. In Proc. 1990 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 1990

31. Meadows C: Applying formal methods to the analysis of a key management protocol. J Comput Security, 1:5–35 (1992)
32. Millen JK: Hookup security for synchronous machines. In Proceedings of the Computer Security Foundations Workshop III, Franconia, NH, June 1990
33. Ross SM: Introduction to Probability Models. (3rd edn) Academic Press, Orlando, FL 1985
34. Rudin W: Principals of Mathematical Analysis. McGraw-Hill, New York
35. Ruspini EH: Epistemic logics, probability, and the calculus of evidence. In Proceedings of the 10^{th} International Joint Conference on Artificial Intelligence, pp 924–931 (1987)
36. Seidel K: Probabilistic communicating processes. Technical report, University of Oxford, Lady Margaret Hall PhD thesis, 1992
37. Shannon CE: Channels with side information at the transmitter. IBM Journal of Research and Development 2:289–293 (1958) Republished in: David Slepian (ed.), "Key Papers in the Development of Information Theory", IEEE Press, 1974
38. Shannon CE: Two-way communication channels. In Proc. 4th Berkeley Symp. Math. Stat. and Prob., Vol. 1: pp 611–644 (1961) Republished in: David Slepian (ed.), "Key Papers in the Development of Information Theory", IEEE Press, 1974
39. Shiryayev AN: Probability, Vol. 95 of Graduate Texts in Mathematics. Springer Berlin Heidelberg New York 1984
40. D.Sutherland: A model of information. In Proceeding of the 9th National Computer Security Conference, Baltimore, MD, September 1986
41. Syverson PF, Gray, III JW: The epistemic representation of information flow security in probabilistic systems. In Proc. 8^{th} IEEE Computer Security Foundations Workshop, pp 152–166, County Kerry, Ireland, June 1995
42. Wittbold JT, Johnson DM: Information flow in nondeterministic systems. In Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, 1990
43. Wray JC: An analysis of covert timing channels. J Comput Security 1:219–232 (1992)

James Wesley Gray, III was born in 1962. Between 1983 and 1993 he pursued the intersection of computer science and formal methods, working along the way in the Formal Methods section at the United States Naval Research Lab, as well as receiving his PhD in Computer Science at the University of Maryland at College Park. Since 1993, Jim has been an Assistant Professor in the Department of Computer Science at the Hong Kong University of Science and Technology, where he has become more of an engineer, often doing things with no justification other than that "it works". As of January 1998, Jim is in the midst of a move to RSA Labs in Redwood City, California, where he will work as a Research Scientist.

Paul Syverson received a Ph.D. in philosophy (specializing in logic) from Indiana University in 1993, master's degrees in philosophy and mathematics from Indiana University, both in 1988, and an A.B. in philosophy from Cornell University in 1981. Since 1989 he has worked at the U.S. Naval Research Laboratory, primarily on epistemic and temporal logics for analyzing cryptographic protocols and secure computing systems. His current focus is the design and analysis of protocols and systems for anonymity and for accountability.