Secrecy in Multiagent Systems

Joseph Y. Halpern & Kevin R. O'Neill
Department of Computer Science
Cornell University
halpern@cs.cornell.edu; kroneill@gmail.com

We introduce a general framework for reasoning about secrecy requirements in multiagent systems. Our definitions extend earlier definitions of secrecy and nondeducibility given by Shannon and Sutherland. Roughly speaking, one agent maintains secrecy with respect to another if the second agent cannot rule out any possibilities for the behavior or state of the first agent. We show that the framework can handle probability and nondeterminism in a clean way, is useful for reasoning about asynchronous systems as well as synchronous systems, and suggests generalizations of secrecy that may be useful for dealing with issues such as resource-bounded reasoning. We also show that a number of well-known attempts to characterize the absence of information flow are special cases of our definitions of secrecy.

Categories and Subject Descriptors: []:

General Terms: Security, theory

Additional Key Words and Phrases: Information flow, secrecy, security policies

1. INTRODUCTION

In the past two decades there have been many attempts to define what it means for a system to be perfectly secure, in the sense that one group of agents is unable to deduce anything at all about the behavior of another group. More generally, many papers in computer science have, in a variety of different settings, defined properties of "secrecy" and have discussed techniques for achieving these properties. In the computer-security literature, early definitions of "perfect security" were based on two different intuitions. *Noninterference* [Goguen and Meseguer 1982] attempted to capture the intuition that an agent at a high security level is unable to interfere with an agent at a lower security level, while *nondeducibility* [Sutherland 1986] attempted to capture the intuition that an agent at a low security level is unable to deduce anything about the state of agents at a higher security level. Others definitions have involved a notion of "information flow", and taken a system to be

Authors supported in part by NSF under grant IRI-96-25901 and IIS-0090145, by ONR under grants N00014-00-1-03-41 and N00014-01-10-511, and by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the ONR under grant N00014-01-1-0795. Kevin O'Neill was also supported in part by a graduate fellowship from the National Science and Engineering Research Council of Canada. A preliminary version of this paper appeared at the 15th IEEE Computer Security Foundations Workshop, 2002.

secure if it is impossible for information to flow from a high-level user to a low-level user. With these basic ideas in mind, definitions of security have been provided for a wide variety of system models, including semantic models that encode all possible input/output behaviors of a computing system and language-based models that deal with process algebras and with more traditional constructs such as imperative programming languages. (Focardi and Gorrieri [2001] provide a classification of security properties expressed using process algebras; Sabelfeld and Myers [2003] give a survey of language-based techniques.)

Sutherland's definition of nondeducibility was based on a simple idea: a system can be described as a set of "worlds" that encode the local states of users, and security is maintained if high-level and low-level states are independent in the sense that a low-level user can never totally rule out any high-level state based on his own local state. As we shall see in Section 4, nondeducibility is closely related to Shannon's [1949] probabilistic definition of secrecy in the context of cryptography, which requires high-level and low-level events to be probabilistically independent. This can be shown to imply that the low agent's posterior probability of a high event should be the same as his prior probability of that event before he began interacting with the system. (Nondeducibility is also closely related to Cohen's [**?**] definition of *strong dependency*. Cohen's work is concerned with information transmission over channels, which are represented as functions that transform inputs into outputs, whereas Sutherland's work is concerned with possible worlds that represent states of a system, but their definitions are essentially identical.)

Definitions of noninterference based on Goguen and Meseguer's early work (see, for example, McCullough [1987] and McLean [1994]) are quite different in flavor from the definitions of Shannon and Sutherland. Typically, they represent the system as a set of input/output traces, and deem the system secure if the set of traces is closed under operations that add or remove high-level events. Variants of this idea have been proposed to deal with issues such as verification, system composition, timing attacks, and so on. Although these definitions have been useful for solving a variety of technical problems, the complexity of some of this work has, in our view, obscured the simplicity of earlier definitions based on the notion of independence. We claim that the basic principle of nondeducibility captures notions of secrecy in an elegant and useful way.

In this paper we define secrecy in terms of an agent's knowledge, using the "runs-and-systems" framework [Fagin et al. 1995]. The runs-and-systems framework generalizes the standard input/output trace models that have been used in many definitions of noninterference. The runs-and-systems framework generalizes the standard input/output trace models that have been used in many definitions of noninterference. Trace-based approaches focus only on input and output values, which makes them poorly suited to modeling systems where agents have richer states than sequences of input and output events. For example, they are insufficient for modeling properties such as deadlock [Focardi and Gorrieri 2001] and for describing semisynchronous systems (where agents know the time to within some tolerance ϵ). The added generality of the runs-and-systems approach lets us deal with these issues in a straightforward way.

Many frameworks for reasoning about secrecy and information flow have assumed,

often implicitly, a very coarse notion of uncertainty. Either an agent knows, with certainty, that some fact is true, or she does not; a definition of secrecy (with respect to some agent) amounts to a characterization of which facts the agent must not know, or which facts she must think are possible. Indeed, this is precisely the intuition that we make precise in Section 3.2. In the literature, such definitions are called *possibilistic*, because they consider only what agents consider possible or impossible. In practice, however, such a coarse-grained notion of uncertainty is simply too weak; it is easy to concoct examples where one agent has possibilistic secrecy, but where intuition suggests that secrecy is not maintained. (See the introduction to Section 4 for such an example.) We extend our definitions of secrecy to incorporate probability, a much more fine-grained notion of uncertainty. Just as Shannon's definitions of secrecy can be viewed as a probabilistic strengthening of Sutherland's definition of nondeducibility, our definitions of probabilistic secrecy generalize the possibilistic definitions we give. In fact, there is a sense in which they are the same definitions, except with a different measure of uncertainty. (This intuition can be made precise using *plausibility measures*. See [Halpern and O'Neill 2005, Section 5] for details.)

Our approach has an additional advantage: it enables us to provide syntactic characterizations of secrecy, using a logic that includes modal operators for reasoning about knowledge and probability. We discuss what it means for a fact to "depend on" the state of an agent and show that secrecy can be characterized as the requirement that low-level agents never know any fact that depends on the state of a high-level agent. (In the probabilistic case, the requirement is that low-level agents must think that any such fact is equally likely at all points of the system.) This knowledge-based characterization lets us make precise the connection between secrecy (of the high-level agent with respect to the low-level agent) and the notion of a "secret", that is, a fact about the system that an agent is not allowed to know. This syntactic approach also opens the door to natural generalizations of information-flow properties that require secrecy for only some facts, as well as allowing us to consider notions of secrecy based on more computational notions of knowledge, which may be more appropriate for resource-bounded agents.

As we show in Section 5, our approach provides insight into a number of other definitions of secrecy and noninterference that have been proposed in the literature. We illustrate this point by considering *separability* [McLean 1994], *generalized noninterference* [McLean 1994], *nondeducibility on strategies* [Wittbold and Johnson 1990], and *probabilistic noninterference* [Gray and Syverson 1998]. One of our goals in this section, obviously, is to convince the reader that our definitions are in fact as general as we claim they are. More importantly, we hope that providing a unified framework for comparing definitions of secrecy will facilitate the cross-fertilization of ideas.

We believe that it is important to provide broad, general definitions of secrecy and noninterference for a general class of multiagent systems—definitions that emphasize the underlying unity of the notions we are trying to capture. Our work is intended to to do just that. Although our definitions should be appropriate for a wide variety of settings, to motivate our various definitions we discuss example programs written in a simple imperative language with input and output operators that allow

high-level users and low-level users to interact with the system at runtime. (The language is similar to that considered by O'Neill, Clarkson, and Chong[2005], who discuss secrecy-based properties for interactive, imperative programming languages and prove that such properties can be enforced using a well-known type system.) We hope the examples will make it clear that the definitions apply equally well in other settings.

The rest of the paper is organized as follows. Section 2 reviews the multiagent systems framework and the definition of knowledge in multiagent systems. In Section 3 we define secrecy and relate it to Sutherland's notion of nondeducibility. We also consider syntactic definitions of secrecy using a logic of knowledge. Section 4 considers probabilistic secrecy. In Section 5 we compare our definitions with others that have been given in the security literature. We conclude in section 6. Most proofs are deferred to the appendix.

2. KNOWLEDGE AND MULTIAGENT SYSTEMS

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

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

A *run* is a function from time to global states. Intuitively, a run is a complete description of what happens over time in one possible execution of the system. A *point* is a pair (r, m) consisting of a run r and a time m. For simplicity, we take time to range over the natural numbers. As pointed out in [Fagin et al. 1995], we think of time as being measured on some clock external to the system. We do *not* assume that agents in the system necessarily have access to this clock; at time m measured on the external clock, agent i need not know it is time m. This allows us to model asynchronous systems in a straightforward way. At a point (r, m), the system is in some global state $r(m)$. If $r(m) = (s_e, s_1, \ldots, s_n)$, then we take $r_i(m)$ to be s_i, agent i's local state at the point (r, m). Formally, a *system* consists of a set of runs (or executions). Let $\mathcal{PT}(\mathcal{R})$ denote the points in a system $\mathcal{R}$.

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

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

The set $\mathcal{K}_i(r, m)$ is often called an *i-information set* because, intuitively, it corresponds to the system-dependent information encoded in i's local state at the point (r, m).

A natural question to ask is where these runs come from. While the framework itself does not deal with this issue, in practice we are interested in systems where the runs are generated by a simple set of rules, such as a communication or security

protocol, a program written in some programming language, or a process described in a concurrent process language. Translating such rules to a set of runs is not always straightforward, but doing so is often useful inasmuch as it forces us to think carefully about what features of the system are essential and relevant to the safety or correctness issues that we are interested in. With respect to secrecy, the way in which we model the local states of various agents is especially important. In particular, if we want the actions or choices made by one agent not to affect the state of another agent, we should include sufficient information in the local state of the first agent to reason about those actions or choices. (We return to this issue in Sections 4.4 and 5.2.)

To reason formally about secrecy in multiagent systems, we use a logic of knowledge and time. Starting with a set Φ of primitive propositions, we close off under negation, conjunction, the modal operators K_i for $i = 1, \ldots, n$, and $\diamondsuit$. In the context of security protocols, the set Φ might consist of primitive propositions corresponding to facts such as "the key is n" or "agent A sent the message m to B". As usual, $K_i\varphi$ means that agent i knows φ; $K_i\varphi$ at a point (r, m) if φ is true at all points in $\mathcal{K}_i(r, m)$. Finally, $\diamondsuit\varphi$ is true at a point (r, m) if φ is true at some point on run r (either before, at, or after time m). While it is, of course, possible to define other temporal operators, the $\diamondsuit$ operator will prove particularly useful in our definitions.

We use the standard approach [Fagin et al. 1995] to give semantics to this language. An interpreted system $\mathcal{I}$ consists of a pair $(\mathcal{R}, \pi)$, where $\mathcal{R}$ is a system and π is an interpretation for the primitive propositions in Φ that assigns truth values to the primitive propositions at the global states. Thus, for every $p \in \Phi$ and global state s that arises in $\mathcal{R}$, we have $(\pi(s))(p) \in \{\mathbf{true}, \mathbf{false}\}$. Of course, π also induces an interpretation over the points in $\mathcal{PT}(\mathcal{R})$: simply take $\pi(r, m)$ to be $\pi(r(m))$. We now define what it means for a formula φ to be true at a point (r, m) in an interpreted system $\mathcal{I}$, written $(\mathcal{I}, r, m) \models \varphi$, by induction on the structure of formulas:

—$(\mathcal{I}, r, m) \models p$ iff $(\pi(r, m))(p) = \mathbf{true}$;
—$(\mathcal{I}, r, m) \models \varphi \wedge \psi$ iff $(\mathcal{I}, r, m) \models \varphi$ and $(\mathcal{I}, r, m) \models \psi$;
—$(\mathcal{I}, r, m) \models \neg\varphi$ iff $(\mathcal{I}, r, m) \not\models \varphi$;
—$(\mathcal{I}, r, m) \models K_i\varphi$ iff $(\mathcal{I}, r', m') \models \varphi$ for all $(r', m') \in \mathcal{K}_i(r, m)$;
—$(\mathcal{I}, r, m) \models \diamondsuit\varphi$ iff there exists n such that $(\mathcal{I}, r, n) \models \varphi$.

As usual, we say that φ is *valid in* $\mathcal{I}$ and write $\mathcal{I} \models \varphi$ if $(\mathcal{I}, r, m) \models \varphi$ for all points (r, m) in $\mathcal{I}$; similarly, φ is *satisfiable in* $\mathcal{I}$ if $(\mathcal{I}, r, m) \models \varphi$ for some point (r, m) in $\mathcal{I}$. We abbreviate $\neg K_i \neg\varphi$ as $P_i\varphi$. We read $P_i\varphi$ as "(according to agent i) φ is possible". Note that $(\mathcal{I}, r, m) \models P_i\varphi$ if there exists a point $(r', m') \in \mathcal{K}_i(r, m)$ such that $(\mathcal{I}, r', m') \models \varphi$.

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

Another natural assumption, often made in system models used in the security literature, is that agents have *perfect recall*. Roughly speaking, an agent with

perfect recall can reconstruct his complete local history. In synchronous systems, for example, an agent's local state changes with every tick of the external clock, so agent i's having perfect recall implies that the sequence $\langle r_i(0), \ldots, r_i(m)\rangle$ must be encoded in $r_i(m+1)$. In asynchronous systems, where the "stuttering" of states reflects an agent's ignorance of the passage of time, perfect recall implies that the agent's state encodes only the sequence of distinct states that she has observed.

To formalize this intuition, let *agent i's local-state sequence at the point* (r, m) be the sequence of local states she has gone through in run r up to time m, without consecutive repetitions.[1] Thus, if from time 0 through time 4 in run r agent i has gone through the sequence $\langle s_i, s_i, s_i', s_i, s_i\rangle$ of local states, where $s_i \neq s_i'$, then her local-state sequence at $(r, 4)$ is $\langle s_i, s_i', s_i\rangle$. Intuitively, an agent has perfect recall if her current local state encodes her local-state sequence. More formally, we say that *agent i has perfect recall in system $\mathcal{R}$* if, at all points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R})$, if $(r', m') \in \mathcal{K}_i(r, m)$, then agent i has the same local-state sequence at both (r, m) and (r', m'). Thus, agent i has perfect recall if she "remembers" her local-state sequence at all times. It is easy to check that perfect recall has the following key property: if $(r', m_1') \in \mathcal{K}_i(r, m_1)$, then for all $m_2 \leq m_1$, there exists $m_2' \leq m_1'$ such that $(r', m_2') \in \mathcal{K}_i(r, m_2)$. (See [Fagin et al. 1995] for more discussion of this definition.)

3. SECRECY IN NONPROBABILISTIC SYSTEMS

3.1 Defining Secrecy

In this section we give abstract definitions of secrecy for systems described using the runs-and-systems model. Roughly speaking, we define secrecy so as to ensure that low-level agents do not know anything about the state of high-level agents. In Section 3.2, we formalize these intuitions using the epistemic logic of Section 2.

To motivate our definitions, we use programs expressed in a simple imperative programming language. We assume that these programs are executed sequentially on a single machine, and that users with different security clearances can interact with the machine via channels appropriate to their security level. For example, the command **input** x **from** H prompts the high-level channel H for an input and stores the input value in program variable x, while the command **output** e **to** L outputs the value of the expression e on the low-level channel L.

All the programs that we consider determine systems in an obvious way, once we decide whether to model the systems synchronously or asynchronously. For example, in a synchronous system determined by the following program:

output 0 **to** L;
output 1 **to** L

the system consists of exactly one run.[2] In this run, L's local state is initially empty (i.e., $\langle\rangle$) and is then updated with a new output event $out(L, i)$ (denoting the output of value i on channel L) at each time step. At time 1, L's local state is

[1] Note that although we eliminate stuttering in the definition of a local-state sequence, which is used to define perfect recall, the stuttering may still be present in the underlying system!

[2] Though for simplicity we ignore timing variations arising due to blocking input commands, our system model can easily handle such timing issues.

$\langle out(L,1)\rangle$, and at time 2 it is $\langle out(L,1), out(L,2)\rangle$. Since there is no output event at subsequent steps, at time 4, L's local state is

$$\langle out(L,1), out(L,2), \_, \_\rangle.$$

L's local state is different at time 2 and time 4, since L is aware that time has passed. By way of contrast, one way to translate programs to asynchronous systems is to model agents' local states so that they are modified only when input and output event occurs on channels to which they have access. In an asynchronous system determined by the program above, L's state would be unchanged after time 2.

The strongest notion of secrecy that we consider in this section is the requirement that an agent, based on her local state, must not be able to infer anything about the local state of another agent. To guarantee that an agent i is unable to rule out any possible local states for another agent j, we require that every possible local state for j be compatible with every possible local state for i:

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

Total secrecy is a strong property. For almost any imaginable system, it is, in fact, too strong to be useful. There are two important respects in which it is too strong. The first respect has to do with the fact that total secrecy protects *everything* about the state of the high-level agent. In some systems, we might want only some part of the high-level agent's state to be kept secret from the low-level agent. For example, we might want the high-level agent to be able to observe details about the state of the low-level agent, in which case our definitions are too strong because they rule out any correlation between the states of the high-level and low-level agents.

To make this more concrete, consider the following program:

input x **from** L;
output x **to** H;
output 1 **to** L

H does not maintain total secrecy with respect to L because after L sees $out(L,1)$ he knows that H has already seen L's first input value as her output. (Note that in a synchronous setting the final output is irrelevant: L would know that H had seen L's input value at the second time step.) If we want to protect only the input values provided by H, total secrecy is too strong. We may be interested in a weaker notion of secrecy, which allows L to realize that H knows L's input value but still keeps all of the "significant" part of H's state secret. Rather than trying to define "significant", we characterize significance abstractly using what we call an "information function".

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

Thus, for example, if j's local state at any point (r,m) includes a list of input and output operations, $f(r,m)$ could consist of only the output values contained in j's local state. Intuitively, $f(r,m)$ is intended to represent that part of j's local state that is significant to whomever is doing the reasoning.

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

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

To see how f-secrecy handles the example program above, suppose that we introduce an information function f that extracts only the input events from H's state. Because $f(r, m)$ is always empty, it is easy to see that H maintains total f-secrecy with respect to L. If our goal is to protect only the input values provided by H, any program that never reads input values from H is trivially secure.

Total f-secrecy is a special case of *nondeducibility*, introduced by Sutherland [1986]. Sutherland considers "abstract" systems that are characterized by a set W of worlds. He focuses on two agents, whose views are represented by information functions g and h on W. Sutherland says that *no information flows from g to h* if, for all worlds $w, w' \in W$, there exists some world $w'' \in W$ such that $g(w'') = g(w)$ and $h(w'') = h(w')$. This notion is often called *nondeducibility (with respect to g and h)* in the literature. To see how total f-secrecy is a special case of nondeducibility, let $W = \mathcal{PT}(\mathcal{R})$, the set of all points of the system. Given a point (r, m), let $g(r, m) = r_i(m)$. Then total f-secrecy is equivalent to nondeducibility with respect to g and f.

Note that nondeducibility is symmetric: no information flows from g to h iff no information flows from h to g. Since most standard noninterference properties focus only on protecting the state of some high-level agent, symmetry appears to suggest that if the actions of a high-level agent are kept secret from a low-level agent, then the actions of a low-level agent must also be kept secret from the high-level agent. Our definitions help to clarify this issue. Total secrecy as we have defined it is indeed symmetric: j maintains total secrecy with respect to i iff i maintains total secrecy with respect to j. However, total f-secrecy is not symmetric in general. If j maintains total f-secrecy with respect to i, it may not even make sense to talk about i maintaining total f-secrecy with respect to j, because f may not be an i-information function. Thus, although f-secrecy is an instantiation of nondeducibility (with respect to an appropriate g and h), the symmetry at the level of g and h does not translate to symmetry at the level of f-secrecy, which is where it matters.

While f-secrecy is useful conceptually, it is essentially a trivial technical generalization of the basic notion of secrecy, because for any agent j and j-information function f, we can reason about a new agent j_f whose local state at any point (r, m) is $r_{j_f}(m) = f(r_j, m)$. Therefore, every theorem we prove involving secrecy holds for f-secrecy as well. For this reason, and to simplify the definitions given in the remainder of the paper, we ignore information functions and deal only with secrecy of one agent with respect to another. Indeed, all our definitions hold without change for any agent "created" by identifying an agent with a function on global states.

The second respect in which total secrecy is too strong involves *time*. To understand the issue, consider synchronous systems (as defined in Section 2). In such

systems, the low-level agent knows the time and knows that the high-level agent knows it too. Thus, the low-level agent can rule out all high-level states except those that occur at the current time. Even in semisynchronous systems, where agents know the time to within some tolerance ϵ, total secrecy is impossible, because low-level agents can rule out high-level states that occur only in the distant past or future.

Total secrecy may be an unreasonable condition even in asynchronous systems. To see this, consider the following program:

input x **from** H;
output 1 **to** L

Even though L does not know which value was entered by H, H does not maintain total secrecy with respect to L in this program simply because L knows, after seeing his output value, that H has already entered *some* input value. Indeed, total secrecy—and also total f-secrecy, for an information function f that extracts high input values—rules out any program where low output events follow high input events.

We now consider two distinct ways of resolving this problem. The first way weakens total secrecy by considering runs instead of points. Total secrecy (of j with respect to i) says that at all times, agent i must consider all states of j to be (currently) possible. A weaker version of total secrecy says that at all times, i must consider it possible that every possible state of j either occurs at that time, or at some point in the past or future. We formalize this in the following definition. Given a set U of points, let $\mathcal{R}(U)$ consist of the runs in $\mathcal{R}$ going through a point in U. That is, $\mathcal{R}(U) = \{r \in \mathcal{R} : (r, m) \in U \text{ for some } m\}$.

Definition 3.4. Agent j maintains *run-based secrecy* with respect to j in system $\mathcal{R}$ if, for all points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R})$, $\mathcal{R}(\mathcal{K}_i(r, m)) \cap \mathcal{R}(\mathcal{K}_j(r', m')) \neq \emptyset$.

It is easy to check that j maintains run-based secrecy with respect to j in system $\mathcal{R}$ iff for all points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R})$, there exists a run r'' and times n and n' such that $r''_i(n) = r_i(m)$ and $r''_j(n') = r'_j(m')$. To relate the formal definition to its informal motivation, note that every state of j that occurs in the system has the form $r'_j(m')$ for some point (r', m'). Suppose that i's state is $r_i(m)$. If there exists a point (r'', n'') such that $r''_i(n'') = r_i(m)$ and $r''_j(n'') = r'_j(m')$, agent i considers it possible that j currently has state $r'_j(m')$. If instead $r''_j(n) = r'_j(m')$ for $n < n''$, then i currently considers it possible that j was in state $r'_j(m')$ at some point in the past; similarly, if $n > n''$, then i thinks that j could be in state $r'_j(m')$ at some point in the future. Note that total secrecy implies run-based secrecy, but the converse is not necessarily true (as shown in Example A.2). While run-based secrecy is still a very strong security property, it seems much more reasonable than total secrecy. In particular, H maintains run-based secrecy with respect to L in the system corresponding to the program **input** x **from** H;**output** 1 **to** L—as far as L is concerned, all the runs in this system look identical. However, run-based secrecy does not hold in systems derived from the kinds of programs typically used

to demonstrate indirect information flows, such as:

input x **from** H;
if $(x = 0)$ **then**
  output 0 **to** L
else
  output 1 **to** L

where run-based secrecy does not hold because L's output gives information about whether H's input value was greater than 0.

The second way to weaken total secrecy is to relax the requirement that the low-level agent cannot rule out any possible high-level states. We make this formal as follows.

Definition 3.5. An *i-allowability function* on $\mathcal{R}$ is a function C from $\mathcal{PT}(\mathcal{R})$ to subsets of $\mathcal{PT}(\mathcal{R})$ such that $\mathcal{K}_i(r,m) \subseteq C(r,m)$ for all $(r,m) \in \mathcal{PT}(\mathcal{R})$.

Intuitively, $\mathcal{PT}(\mathcal{R}) - C(r,m)$ is the set of points that i is allowed to "rule out" at the point (r,m). It seems reasonable to insist that the points that i considers possible at (r,m) not be ruled out, which is why we require that $\mathcal{K}_i(r,m) \subseteq C(r,m)$.

Definition 3.6. If C is an i-allowability function, then *j maintains C-secrecy with respect to i* if, for all points $(r,m) \in \mathcal{PT}(\mathcal{R})$ and $(r',m') \in C(r,m)$, we have $\mathcal{K}_i(r,m) \cap \mathcal{K}_j(r',m') \neq \emptyset$.

If $C(r,m) = \mathcal{PT}(\mathcal{R})$ for all points $(r,m) \in \mathcal{PT}(\mathcal{R})$, then C-secrecy reduces to total secrecy. Synchrony can be captured by the allowability function $S(r,m) = \{(r',m) : r' \in \mathcal{R}\}$. Informally, this says that agent i is allowed to know what time it is. We sometimes call S-secrecy *synchronous secrecy*. It is easy to see that H maintains synchronous secrecy with respect to L in the system generated by the program **input** x **from** H;**output** 1 **to** L.

In synchronous systems, synchronous secrecy has a simple characterization.

PROPOSITION 3.7. *Agent j maintains synchronous secrecy with respect to i in a synchronous system $\mathcal{R}$ iff, for all runs $r, r' \in \mathcal{R}$ and times m, we have that $\mathcal{K}_i(r,m) \cap \mathcal{K}_j(r',m) \neq \emptyset$.*

PROOF. This follows trivially from the definitions. □

In synchronous input/output trace systems, synchronous secrecy is essentially equivalent to the standard notion of *separability* [McLean 1994]. (Total secrecy can be viewed as an asynchronous version of separability. See Section 5.1 for further discussion of this issue.) The security literature has typically focused on either synchronous systems or completely asynchronous systems. One advantage of our framework is that we can easily model both of these extreme cases, as well as being able to handle in-between cases, which do not seem to have been considered up to now. Consider a semisynchronous system where agents know the time to within a tolerance of ϵ. At time 5, for example, an agent knows that the true time is in the interval $[5-\epsilon, 5+\epsilon]$. This corresponds to the allowability function $SS(r,m) = \{(r',m') : |m - m'| \leq \epsilon\}$, for the appropriate ϵ. We believe that any attempt to define security for semisynchronous systems will require something like allowability functions.

C-secrecy and run-based secrecy represent two quite different approaches to weakening total secrecy: allowability functions restrict the set of j-information sets that i must consider possible, while run-based secrecy focuses on runs rather than points. Even if we focus on synchronous secrecy, the two notions are distinct. In systems without perfect recall, for example, we may have synchronous secrecy without having run-based secrecy, while in asynchronous systems we may have run-based secrecy without having synchronous secrecy. (See Appendix A for examples.) However, there are contexts in which the definitions do coincide, suggesting that they are capturing some of the same intuitions. Consider, for example, our definition of synchronous secrecy. Intuitively it might at first seem that synchronous secrecy goes too far in weakening total secrecy. Informally, j maintains *total* secrecy with respect to i if i never learns anything not only about j's current state, but also his possible future and future states. Synchronous secrecy seems only to say that i never learns anything about j's state *at the current time*. However, when agents have perfect recall, it turns out that synchronous secrecy implies run-based secrecy, thus addressing this concern.

To make this precise for a more general class of allowability functions, we need the following definition, which captures the intuition that an allowability function depends only on timing. Given any two runs, we want the allowability function to map points on the first run to contiguous, nonempty sets of points on the second run in a way that respects the ordering of points on the first run and covers all points on the second run.

Definition 3.8. An allowability function C *depends only on timing* if it satisfies the following three conditions: (a) for all runs $r, r' \in \mathcal{R}$, and all times m', there exists m such that $(r', m') \in C(r, m)$; (b) if $(r', m') \in C(r, m)$, and $n \geq m$ (resp. $n \leq m$), there exists $n' \geq m'$ (resp. $n' \leq m'$) such that $(r', n') \in C(r, n)$; (c) if $(r', n_1) \in C(r, m)$, $(r', n_2) \in C(r, m)$, and $n_1 \leq m' \leq n_2$, then $(r', m') \in C(r, m)$.

It is easy to check that both synchronous and semisynchronous allowability functions depend only on timing. We now show that C-secrecy implies run-based secrecy if C depends only on timing.

PROPOSITION 3.9. *If $\mathcal{R}$ is a system where i and j have perfect recall, C depends only on timing, and j maintains C-secrecy with respect to i, then j maintains run-based secrecy with respect to i.*

In synchronous systems with perfect recall, synchronous secrecy and run-based secrecy agree. This reinforces our claim that both definitions are natural, useful weakenings of total secrecy.

PROPOSITION 3.10. *If $\mathcal{R}$ is a synchronous system where both i and j have perfect recall, then agent j maintains synchronous secrecy with respect to i iff j maintains run-based secrecy with respect to i.*

The requirement in Proposition 3.10 that both agents have perfect recall is necessary; see Example A.1 for details. Without perfect recall, two things can go wrong. First, if i does not have perfect recall, she might be able to determine at time n what j's state is going to be at some future time $n' > n$, but then forget about it by time n', so that j maintains synchronous secrecy with respect to i, but not

run-based secrecy. Second, if j does not have perfect recall, i might learn something about j's state in the past, but j might still maintain synchronous secrecy with respect to i because j has forgotten this information by the time i learns it. These examples suggest that secrecy is less interesting when agents can forget facts that they once knew. At any rate, it makes sense to model agents as if they have perfect recall, since not doing so requires us to trust that agents will forget facts when we need them to, leading to weaker security guarantees.

3.2 Characterizing Secrecy Syntactically

Our definition of secrecy is semantic; it is given in terms of the local states of the agents. However, it is helpful and instructive to reason about secrecy syntactically, using the logic of knowledge discussed in Section 2. Our goal in this section is to characterize secrecy in terms of the knowledge—or more precisely, the lack of knowledge—of the agent with respect to whom secrecy is maintained. To this end, we show that the state of an agent j is kept secret from an agent i exactly if i does not know any formulas that depend only on the state of j, or, dually, if i always thinks that any formula that depends on the state of j is currently possible.

For this characterization, we use the modal logic of knowledge described in Section 2. But first, we need to define what it means for a formula to depend on the local state of a particular agent. Given an agent j, a formula φ is *j-local* in an interpreted system $\mathcal{I}$ if, for all points (r, m) and (r', m') such that $r_j(m) = r'_j(m')$, $(\mathcal{I}, r, m) \models \varphi$ iff $(\mathcal{I}, r', m') \models \varphi$. It is easy to check that φ is j-local in $\mathcal{I}$ iff $\mathcal{I} \models K_j\varphi \vee K_j\neg\varphi$; thus, j-locality can be characterized syntactically. (See [Engelhardt et al. 1998] for an introduction to the logic of local propositions.) The notion of j-locality has another useful semantic characterization:

PROPOSITION 3.11. *A formula φ is j-local in an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ iff there exists a set Ω of j-information sets such that $(\mathcal{I}, r, m) \models \varphi$ exactly when $(r, m) \in \bigcup_{\mathcal{K}\in\Omega} \mathcal{K}$.*

The following theorem shows that the semantic characterizations of secrecy given in Section 3.1 correspond closely to our intuitions of what secrecy should mean: agent j maintains secrecy with respect to i precisely if i cannot rule out any satisfiable facts that depend only on the local state of j.

THEOREM 3.12. *Suppose that C is an i-allowability function. Agent j maintains C-secrecy with respect to agent i in system $\mathcal{R}$ iff, for every interpretation π and point (r, m), if φ is j-local and $(\mathcal{I}, r', m') \models \varphi$ for some $(r', m') \in C(r, m)$, then $(\mathcal{I}, r, m) \models P_i\varphi$.*

Since total secrecy is just C-secrecy for the allowability function C such that $C(r, m)$ consists of all point in $\mathcal{R}$, the following corollary, which gives an elegant syntactic characterization of total secrecy, is immediate.

COROLLARY 3.13. *Agent j maintains total secrecy with respect to agent i in system $\mathcal{R}$ iff, for every interpretation π, if the formula φ is j-local and satisfiable in $\mathcal{I} = (\mathcal{R}, \pi)$, then $\mathcal{I} \models P_i\varphi$.*

Corollary 3.13 says that total secrecy requires i not to know any j-local fact that is not valid in $\mathcal{I}$. A similar result holds for synchronous secrecy. For brevity, and because we prove more general results in later sections, we ignore the details here.

We can also give a similar syntactic characterization of run-based secrecy. For j to maintain total secrecy with respect to i, if φ is j-local, it is always necessary for i to think that φ was possible. For run-based secrecy, we require only that i think that φ is possible sometime in the current run. Recall that the formula $\Diamond\varphi$ means "φ is true at some point in the current run".

THEOREM 3.14. *Agent j maintains run-based secrecy with respect to agent i in system $\mathcal{R}$ iff, for every interpretation π, if φ is j-local and satisfiable in $\mathcal{I} = (\mathcal{R}, \pi)$, then $\mathcal{I} \models P_i \Diamond \varphi$.*

The results of this section show that secrecy has a syntactic characterization that is equivalent to the semantic characterization. There are several significant advantages to having such a syntactic characterization. For one thing, it suggests that secrecy can be checked by applying model-checking techniques (although techniques would have to be developed to allow checking $P_i\varphi$ for all formulas φ). Perhaps more importantly, it suggests some natural generalizations of secrecy that may be of practical interest. For example, it may not be relevant that i not know *all* satisfiable formulas. It may be enough for a system designer that i does not know only certain formulas. This may be particularly relevant for declassification or downgrading: if a noninterference property corresponds to a set of formulas that must be kept secret from the low-level agent, formulas can be declassified by removing them the set. Another significant generalization involves replacing knowledge by a more computational notion, such as *algorithmic knowledge* [Fagin et al. 1995; Halpern and Pucella 2003a]. Recall that the definition of knowledge described in Section 2 suffers from the *logical omniscience* problem: agents know all tautologies and know all logical consequences of their knowledge [Fagin et al. 1995]. In the context of security, we are more interested in what *resource-bounded* agents know. It does not matter that an agent with unbounded computational resources can factor and decrypt a message as long as a resource-bounded agent cannot decrypt the message. By requiring only that an agent does not *algorithmically know* various facts, we can capture secrecy with respect to resource-bounded agents.

4. SECRECY IN PROBABILISTIC SYSTEMS

The definitions of secrecy that we have considered up to now are *possibilistic*; they consider only whether or not an event is possible. They thus cannot capture what seem like rather serious leakages of information. As a motivating example, consider a system $\mathcal{R}$ with two agents Alice and Bob, who we think of as sitting at separate computer terminals. Suppose that $\mathcal{L}$ is a language with n words. At time 1, Bob inputs a string $x \in \mathcal{L}$ chosen uniformly at random. At time 2, with probability $1 - \epsilon$, the system outputs x directly to Alice's terminal. However, with probability ϵ, the system is struck by a cosmic ray as Bob's input is transmitted to Alice, and in this case the system outputs a random string from $\mathcal{L}$. (Bob receives no information about what Alice sees.) Thus, there are $n(n + 1)$ possible runs in this system: n runs where no cosmic ray hits, and n^2 runs where the cosmic ray hits. Moreover, it is immediate that Bob maintains (possibilistic) synchronous secrecy with respect to Alice even though, with very high probability, Alice sees exactly what Bob's input was.

Although this example may seem somewhat contrived, it is easy to implement in a programming language that includes operators for randomization. For example, suppose that we extend the input/output language from the last section to include an infix operator $\mathbf{8}_p$, where the command $c_0\ \mathbf{8}_p\ c_1$ executes c_0 with probability p and c_1 with probability $1-p$, and an operator **rand** that returns one of the n words in the language $\mathcal{L}$ with uniform probability. The following program implements the cosmic-ray system:

input w **from** B;
{**output rand**() **to** A $\mathbf{8}_\epsilon$ **output** w **to** A}

To reason about the unwanted information flow in this example, we need to add probability to the framework. We can do that by putting the obvious probability measure on the runs in $\mathcal{R}$:

—for each $x \in \mathcal{L}$, the run where Bob inputs x and no cosmic ray hits (so that Alice sees x) gets probability $(1-\epsilon)/n$.
—for each pair $(x, y) \in \mathcal{L} \times \mathcal{L}$, the run where the cosmic ray hits, Bob inputs x, and Alice sees y gets probability ϵ/n^2.

If Alice sees x, her posterior probability that Bob's input was x is

$$\Pr_{Alice}(\text{Bob typed } x \,|\, \text{Alice sees } x) = \frac{\epsilon + n - n\epsilon}{n} = 1 - \frac{n-1}{n}\epsilon.$$

If Alice sees x, her posterior probability that Bob's input was $y \neq x$ is

$$\Pr_{Alice}(\text{Bob typed } x \,|\, \text{Alice sees } y) = \frac{\epsilon}{n}.$$

Thus, if $\epsilon > 0$, even though Alice never learns *with certainty* that Bob's input was x, her probability that it was x rises from $1/n$ to almost 1 as soon as she sees an x.

In this section we introduce definitions of probabilistic secrecy. The definitions and the technical results we obtain closely resemble the definitions and results of the previous two sections. This is no coincidence. Probabilistic and possibilistic secrecy are instances of a definition of *plausibilistic* secrecy for which similar results can be proved in more generality [Halpern and O'Neill 2005, Section 5].

4.1 Defining Probabilistic Secrecy

To reason about probabilistic security, we need a way to introduce probability into the multiagent systems framework. There are actually two ways of doing this: we can either put a probability on points or a probability on runs. We consider putting a probability on points first, using a general approach described by Halpern [2003].

Given a system $\mathcal{R}$, define a *probability assignment* $\mathcal{PR}$ to be a function that assigns to each agent i and point (r, m) a probability space $\mathcal{PR}(r, m, i) = (W_{r,m,i}, \mathcal{F}_{r,m,i}, \mu_{r,m,i})$, where $W_{r,m,i} \subseteq \mathcal{PT}(\mathcal{R})$ is i's sample space at (r, m) and $\mu_{r,m,i}$ is a probability measure defined on the subsets of $W_{r,m,i}$ in $\mathcal{F}_{r,m,i}$. (That is, $\mathcal{F}_{r,m,i}$ is a σ-algebra that defines the measurable subsets of $W_{r,m,i}$.) We call a pair $(\mathcal{R}, \mathcal{PR})$ a *probability system*.

Given a probability system, we can give relatively straightforward definitions of probabilistic total secrecy and synchronous secrecy. Rather than requiring that an agent i think that all states of another j are *possible*, we require that all of

those states be measurable and equally likely according to i's subjective probability measure.

Definition 4.1. Agent j maintains *probabilistic total secrecy* with respect to agent i in $(\mathcal{R}, \mathcal{PR})$ if, for all points (r, m), (r', m'), and (r'', m'') in $\mathcal{PT}(\mathcal{R})$, we have $\mathcal{K}_j(r'', m'') \cap \mathcal{K}_i(r, m) \in \mathcal{F}_{r,m,i}$, $\mathcal{K}_j(r'', m'') \cap \mathcal{K}_i(r', m') \in \mathcal{F}_{r',m',i}$, and

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

Probabilistic total secrecy is a straightforward extension of total secrecy. Indeed, if for all points (r, m) we have $\mu_{r,m,i}(\{(r, m)\}) > 0$, then probabilistic total secrecy implies total secrecy (as in Definition 3.1).

PROPOSITION 4.2. *If $(\mathcal{R}, \mathcal{PR})$ is a probability system such that $\mu_{r,m,i}(\{(r, m)\}) > 0$ for all points (r, m) and j maintains probabilistic total secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$, then j also maintains total secrecy with respect to i in $\mathcal{R}$.*

Like total secrecy, probabilistic total secrecy is an unrealistic requirement in practice, and cannot be achieved in synchronous systems. To make matters worse, the sets $\mathcal{K}_j(r'', m'') \cap \mathcal{K}_i(r, m)$ are typically not measurable according to what is perhaps the most common approach for defining $\mathcal{PR}$, as we show in the next section. Thus, even in completely asynchronous systems, total secrecy is usually impossible to achieve for measurability reasons alone. Fortunately, the obvious probabilistic analogue of synchronous secrecy is a more reasonable condition.

Definition 4.3. Agent j maintains *probabilistic synchronous secrecy* with respect to agent i in $(\mathcal{R}, \mathcal{PR})$ if, for all runs r, r', r'' and all times m, we have $\mathcal{K}_j(r'', m) \cap \mathcal{K}_i(r, m) \in \mathcal{F}_{r,m,i}$, $\mathcal{K}_j(r'', m) \cap \mathcal{K}_i(r', m) \in \mathcal{F}_{r',m,i}$, and

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

Note that if we set up the cosmic-ray system of the previous section as a probability system in such a way that Alice's probability on points reflects the posterior probabilities we described for the system, it is immediate that Bob does *not* maintain probabilistic synchronous secrecy with respect to Alice.

We now consider definitions of probabilistic secrecy where we start with a probability on runs. Define a *run-based probability system* to be a triple $(\mathcal{R}, \mathcal{F}, \mu)$, where $\mathcal{R}$ is a system, $\mathcal{F}$ is a σ-algebra of subsets of $\mathcal{R}$, and μ is a probability measure defined on $\mathcal{F}$. Note that a run-based probability system requires only one probability measure, rather than a probability measure at each point for each agent. In practice, such a measure is often relatively easy to come by. In the same way that a set of runs can be generated by a protocol, a runs-based probability system can be generated by a probabilistic protocol: the probability of a set of runs sharing a common prefix can be derived by multiplying the probabilities of the protocol transitions necessary to generate the prefix (see [Halpern 2003; Halpern and Tuttle 1993] for further discussion).

Here and throughout the paper, we assume for simplicity that in a run-based probability system $(\mathcal{R}, \mathcal{F}, \mu)$, $\mathcal{F}$ contains all sets of the form $\mathcal{R}(\mathcal{K}_i(r, m))$, for all points (r, m) and all agents i. That is, if a set of runs is generated by an agent's local state, it is measurable. We also assume that $\mu(\mathcal{R}(\mathcal{K}_i(r, m))) > 0$, so that we can condition on information sets.

Recall from Section 3.1 that run-based total secrecy requires that, for all points (r, m) and (r', m'), we have $\mathcal{R}(\mathcal{K}_i(r, m)) \cap \mathcal{R}(\mathcal{K}_j(r', m')) \neq \emptyset$. In other words, run-based total secrecy is a property based on what can happen during runs, rather than points. In a run-based probability system where all information sets have positive measure, it is easy to see that this is equivalent to the requirement that $\mu(\mathcal{R}(\mathcal{K}_j(r', m')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) > 0$. We strengthen run-based total secrecy by requiring that these probabilities be *equal*, for all i-information sets.

Definition 4.4. Agent j maintains *run-based probabilistic secrecy* with respect to i in $(\mathcal{R}, \mathcal{F}, \mu)$ if for any three points $(r, m), (r', m'), (r'', m'') \in \mathcal{PT}(\mathcal{R})$,

$$\mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r', m'))).$$

The probabilities for the cosmic-ray system were defined on sets of runs. Moreover, facts such as "Alice sees x" and "Bob typed y" correspond to information sets, exactly as in the definition of run-based probabilistic secrecy. It is easy to check that Bob does not maintain run-based probabilistic secrecy with respect to Alice.

In Section 4.2, we consider the connection between probability measures on points and on runs, and the corresponding connection between probabilistic secrecy and run-based probabilistic secrecy. For the remainder of this section, we consider symmetry in the context of probabilistic secrecy. In Section 3.1, we mentioned that our definitions of secrecy were symmetric in terms of the agents i and j. Perhaps surprisingly, probabilistic secrecy is also a symmetric condition, at least in most cases of interest. This follows from a deeper fact: under reasonable assumptions, secrecy (of j with respect to i) implies the probabilistic independence of i-information sets and j-information sets. (See Lemma C.1 in the appendix for more details.)

Consider probabilities on points: if there is no connection whatsoever between $\mathcal{PR}(r, m, i)$ and $\mathcal{PR}(r, m, j)$ in a probability system $(\mathcal{R}, \mathcal{PR})$, there is obviously no reason to expect secrecy to be symmetric. However, if we assume that the probabilities of i and j at (r, m) are derived from a single common probability measure by conditioning, then symmetry follows. This assumption, which holds for the probability systems we will consider here (and is standard in the economics literature [Morris 1995]), is formalized in the next definition.

Definition 4.5. A probability system $(\mathcal{R}, \mathcal{PR})$ satisfies the *common prior assumption* if there exists a probability space $(\mathcal{PT}(\mathcal{R}), \mathcal{F}_{cp}, \mu_{cp})$ such that for all agents i and points $(r, m) \in \mathcal{PT}(\mathcal{R})$, we have $\mathcal{K}_i(r, m) \in \mathcal{F}_W$, $\mu_{cp}(\mathcal{K}_i(r, m)) > 0$, and

$$\mathcal{PR}_i(r, m) = (\mathcal{K}_i(r, m), \{U \cap K_i(r, m) \,|\, U \in \mathcal{F}_W\}, \mu_{cp} \,|\, \mathcal{K}_i(r, m)).^3$$

In probability systems that satisfy the common prior assumption, probabilistic secrecy is symmetric.

PROPOSITION 4.6. *If $(\mathcal{R}, \mathcal{PR})$ is a probability system (resp., synchronous probability system) that satisfies the common prior assumption with prior probability μ_{cp}, the following are equivalent:*

[3]Actually, it is more standard in the economics literature not to require that $\mu_{cp}(\mathcal{K}_i(r, m)) > 0$. No requirements are placed on $\mu_{r,m,i}$ if $\mu_{cp}(\mathcal{K}_i(r, m)) = 0$. See [Halpern 2002] for a discussion of this issue.

(a) Agent j maintains probabilistic total (resp., synchronous) secrecy with respect to i.

(b) Agent i maintains probabilistic total (resp., synchronous) secrecy with respect to j.

(c) For all points (r,m) and (r',m'), $\mu_{cp}(\mathcal{K}_j(r',m') \mid \mathcal{K}_i(r,m)) = \mu_{cp}(\mathcal{K}_j(r',m'))$ (resp., for all points (r,m) and (r',m), $\mu_{cp}(\mathcal{K}_j(r',m) \mid \mathcal{K}_i(r,m)) = \mu_{cp}(\mathcal{K}_j(r',m) \mid \mathcal{PT}(m))$, where $\mathcal{PT}(m)$ is the set of points occurring at time m; that is, the events $\mathcal{K}_i(r,m)$ and $\mathcal{K}_j(r',m)$ are conditionally independent with respect to μ_{cp}, given that the time is m).

In run-based probability systems there is a single measure μ that is independent of the agents, and we have symmetry provided that the system is synchronous or both agents have perfect recall. (If neither condition holds, secrecy may not be symmetric, as illustrated by Example A.2.)

PROPOSITION 4.7. *If $(\mathcal{R},\mathcal{F},\mu)$ is a run-based probability system that is either synchronous or one where agents i and j both have perfect recall, then the following are equivalent:*

(a) Agent j maintains run-based probabilistic secrecy with respect to i.

(b) Agent i maintains run-based probabilistic secrecy with respect to j.

(c) For all points $(r,m),(r',m') \in \mathcal{PT}(\mathcal{R})$, $\mathcal{R}(\mathcal{K}_i(r,m))$ and $\mathcal{R}(\mathcal{K}_j(r',m'))$ are probabilistically independent with respect to μ.

4.2 From Probability on Runs to Probability on Points

In the last section we described two ways of adding probability to systems: putting a probability on points and putting a probability on runs. In this section, we discuss an approach due to Halpern and Tuttle [1993] for connecting the two approaches.

Given an agent i and a point (r,m), we would like to derive the probability measure $\mu_{r,m,i}$ from μ by conditioning μ on $\mathcal{K}_i(r,m)$, the information that i has at the point (r,m). The problem is that $\mathcal{K}_i(r,m)$ is a set of *points*, not a set of *runs*, so straightforward conditioning does not work. To solve this problem, we condition μ on $\mathcal{R}(\mathcal{K}_i(r,m))$, the set of runs going through $\mathcal{K}_i(r,m)$, rather than on $\mathcal{K}_i(r,m)$. Conditioning is always well-defined, given our assumption that $\mathcal{R}(\mathcal{K}_i(r,m))$ has positive measure.

We can now define a measure $\mu_{r,m,i}$ on the points in $\mathcal{K}_i(r,m)$ as follows. If $\mathcal{S} \subseteq \mathcal{R}$ and $A \subseteq \mathcal{PT}(\mathcal{R})$, let $A(\mathcal{S})$ be the set of points in A that lie on runs in $\mathcal{S}$; that is,

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

In particular, $\mathcal{K}_i(r,m)(\mathcal{S})$ consists of the points in $\mathcal{K}_i(r,m)$ that lie on runs in $\mathcal{S}$. Let $\mathcal{F}_{r,m,i}$ consist of all sets of the form $\mathcal{K}_i(r,m)(\mathcal{S})$, where $\mathcal{S} \in \mathcal{F}$. Then define

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

It is easy to check that if $U \subseteq \mathcal{K}_i(r,m)$ is measurable with respect with respect to $\mu_{r,m,i}$, then $\mu_{r,m,i}(U) = \mu(\mathcal{R}(U) \mid \mathcal{R}(\mathcal{K}_i(r,m)))$. We say that the resulting probability system $(\mathcal{R},\mathcal{PR})$ is *determined* by the run-based probability system $(\mathcal{R},\mathcal{F},\mu)$, and call μ the *underlying measure*. We call a probability system *standard* if it is determined by a run-based probability system.

Note that *synchronous* standard probability systems satisfy the common prior assumption, as defined in the previous section. If we assume that all runs are measurable, then we can simply take $\mu_{cp}(r,m) = \mu(r)/2^{m+1}$. This ensures that the time m points have the same relative probability as the runs, which is exactly what is needed. More generally, if $\mathcal{PT}(m)$ is the set of time m points and $\mathcal{S}$ is a measurable subset of $\mathcal{R}$, we take $\mu_{cp}(\mathcal{PT}(m)(\mathcal{S})) = \mu(\mathcal{S})/2^{m+1}$. It follows from Proposition 4.6 that probabilistic synchronous secrecy is symmetric in synchronous standard systems.

In synchronous standard systems with perfect recall, probabilistic secrecy and run-based probabilistic secrecy coincide. (We remark that Example A.1 shows that the requirement of perfect recall is necessary.) This provides further evidence that our notion of synchronous secrecy is appropriate in synchronous systems.

PROPOSITION 4.8. *If $(\mathcal{R}, \mathcal{PR})$ is the standard system determined by the synchronous run-based probability system $(\mathcal{R}, \mathcal{F}, \mu)$, and agents i and j have perfect recall in $\mathcal{R}$, then agent j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mathcal{F}, \mu)$ iff j maintains probabilistic synchronous secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$.*

4.3 Characterizing Probabilistic Secrecy

We now demonstrate that we can characterize probabilistic secrecy syntactically, as in the nonprobabilistic case. To do so, we must first explain how to reason about probabilistic formulas. Define an *interpreted probability system* $\mathcal{I}$ to be a tuple $(\mathcal{R}, \mathcal{PR}, \pi)$, where $(\mathcal{R}, \mathcal{PR})$ is a probability system. In an interpreted probability system we can give semantics to syntactic statements of probability. We are most interested in formulas of the form $\Pr_i(\varphi) = \alpha$ (or similar formulas with $\leq$, $>$, etc., instead of $=$). Such formulas were given semantics by Fagin, Halpern, and Megiddo [1990]; we follow their approach here. Intuitively, a formula such as $\Pr_i(\varphi) = \alpha$ is true at a point (r,m) if, according to $\mu_{r,m,i}$, the probability that φ is true is given by α. More formally, $(\mathcal{I}, r, m) \models \Pr_i(\varphi) = \alpha$ if

$$\mu_{r,m,i}(\{(r',m') \in \mathcal{K}_i(r,m) : (\mathcal{I}, r', m') \models \varphi\}) = \alpha.$$

Similarly, we can give semantics to $\Pr_i(\varphi) \leq \alpha$ and $\Pr(\varphi) > \alpha$, etc., as well as conditional formulas such as $\Pr(\varphi \mid \psi) = \alpha$. Note that although these formulas talk about probability, they are either true or false at a given state.

The semantics for a formula such as $\Pr_i(\varphi)$ implicitly assumes that the set of points in $\mathcal{K}_i(r,m)$ where φ is true is measurable. While there are ways of dealing with nonmeasurable sets (see [Fagin et al. 1990]), here we assume that all relevant sets are measurable. This is certainly true in synchronous standard systems determined by a a run-based system where all sets of runs are measurable. More generally, it is true in a probability system $(\mathcal{R}, \mathcal{PR})$ where, for all r, m, i, all the sets in the probability space $\mathcal{PR}(r,m,i)$ are measurable.

The first result shows that we can characterize probabilistic total and synchronous secrecy.

THEOREM 4.9. *(a) If $(\mathcal{R}, \mathcal{PR})$ is a probabilistic system, then agent j maintains probabilistic total secrecy with respect to agent i iff, for every interpretation π*

and formula φ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\varphi) = \sigma$.

(b) If $(\mathcal{R}, \mathcal{PR})$ is a synchronous probabilistic system, then agent j maintains probabilistic synchronous secrecy with respect to agent i iff, for every interpretation π, time m, and formula φ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ_m such that $(\mathcal{I}, r, m) \models \Pr_i(\varphi) = \sigma_m$ for all runs $r \in \mathcal{R}$.

We can also characterize run-based secrecy in standard systems using the $\diamondsuit$ operator. For this characterization, we need the additional assumption of perfect recall.

THEOREM 4.10. *If $(\mathcal{R}, \mathcal{PR})$ is a standard probability system where agent j has perfect recall, then agent j maintains run-based probabilistic secrecy with respect to agent i iff, for every interpretation π and every formula φ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\diamondsuit \varphi) = \sigma$.*

Example A.3 demonstrates that the assumption of perfect recall is necessary in Theorem 4.10 and that synchrony alone does not suffice.

4.4 Secrecy in Adversarial Systems

It is easy to capture our motivating cosmic-ray system example using a synchronous standard system because we assumed a probability on the set of runs. Furthermore, it is not hard to show that Bob does not maintain synchronous secrecy with respect to Alice in this system. However, there is an important and arguably inappropriate assumption that was made when we modeled the cosmic-ray system, namely, that we were given the probability with which Bob inputs various strings. While we took that probability to be uniform, it was not necessary to do so: any other probability distribution would have served to make our point. The critical assumption was that there is some well-defined distribution that is known to the modeler. However, in many cases the probability distribution is *not* known. In the cosmic-ray example, if we think of the strings as words in natural language, it may not be reasonable to view all strings as equally likely. Moreover, the probability of a string may depend on the speaker: it is unlikely that a teenager would have the same distribution as an adult, or that people having a technical discussion would have the same distribution as people discussing a movie.

There are many settings in which it makes sense to reason about the nondeterminism of a system in terms of an initial nondeterministic step followed by a sequence of deterministic or probabilistic steps. The nondeterministic step could determine the choice of speaker, the adversary's protocol, or the input to a probabilistic protocol. Indeed, it has been argued [Rabin 1982; Vardi 1985] that any setting where there is a mix of nondeterministic, probabilistic, and deterministic moves can be reduced to one where there is an initial nondeterministic move followed by probabilistic or deterministic moves. In such a setting, we do not have one probability distribution over the runs in a system. Rather, we can partition the set of runs according to the nondeterministic initial step, and then use a separate probability distribution for the set of runs corresponding to each initial step. For example, consider a setting with a single agent and an adversary. Suppose that the agent uses a protocol p and the adversary uses one of a set $\{q_1, \ldots, q_n\}$ of protocols.

The system $\mathcal{R}$ consists of all the runs generated by running (p, q_k) for $k = 1, \ldots, n$. $\mathcal{R}$ can then be partitioned into n subsets $D_1, \ldots, D_n$, where D_j consists the runs of the joint protocol (p, q_j). While we may not want to assume a probability on how likely the adversary is to use q_j, typically there is a natural probability distribution on each set D_j. Note that we can capture uncertainty about a speaker's distribution over natural language strings in the same way; each protocol corresponds to a different speaker's "string-production algorithm".

Situations where there is a nondeterministic choice followed by a sequence of probabilistic or deterministic choices can be characterized by an *adversarial probability system*, which is a tuple $(\mathcal{R}, \mathcal{D}, \Delta)$, where $\mathcal{R}$ is a system, $\mathcal{D}$ is a countable partition of $\mathcal{R}$, and $\Delta = \{(D, \mathcal{F}_D, \mu_D) : D \in \mathcal{D}\}$ is a set of probability spaces, where μ_D is a probability measure on the σ-algebra $\mathcal{F}_D$ (on $D \in \mathcal{D}$) such that, for all agents i, points (r, m), and cells D, $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \in \mathcal{F}_D$ and, if $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$, then $\mu_D(\mathcal{R}(\mathcal{K}_i(r, m))) > 0$.[4]

There are several ways of viewing the cosmic-ray example as an adversarial probability system. If we view the input as a nondeterministic choice, then we can take $D(x)$ to consist of all runs where the input is x, and let $\mathcal{D} = \{D(x) : x \in \mathcal{L}\}$. The measure μ_x on $D(x)$ is obvious: the one run in $D(x)$ where the cosmic ray does not strike gets probability $1 - \epsilon$; the remaining n runs each get probability ϵ/n. Note that we can assign a probability on $D(x)$ without assuming anything about Bob's input distribution. Alternatively, we can assume there are k "types" of agents (child, teenager, adult, etc.), each with their own distribution over inputs. Then the initial nondeterministic choice is the type of agent. Thus, the set of runs is partitioned into sets D_j, $j = 1, \ldots, k$. We assume that agents of type j generate inputs according to probability $\Pr_j$. In each set D_j, there is one run where Bob inputs x and the cosmic ray does not strike; it has probability $\Pr_j(x)(1-\epsilon)$. There are n runs where Bob inputs x and the cosmic ray strikes; each gets probability $\Pr_j(x)\epsilon/n$.

For another example of a system where initial nondeterministic choices play an important role, consider the following program, which implements a system similar to one first described by Wittbold and Johnson [1990]:

while (**true**) **do**
  $x := 0$ $\mathbf{8}_{0.5}$ $x := 1$;
  output x **to** H;
  input y **from** H;
  output $x \oplus y$ **to** L

(We assume that H can input only 0s and 1s; $\oplus$ is the exclusive-or operator.) Note that, in every iteration of this loop, H can transmit a bit b to L by choosing $x \oplus b$ as his input value. More generally, given a bitstring $z = z_1 \ldots z_k$, H can transmit z to L by inputting $x_i \oplus z_i$ at the ith iteration (where x_i is H's output value). Thus, even though H's input values are kept completely secret from L—they are encrypted with a one-time pad that L cannot see—H can transmit arbitrary messages to L.

[4]We actually should have written $\mu_D(\mathcal{R}(\mathcal{K}_i(r, m)) \cap D)$ rather than $\mu_D(\mathcal{R}(\mathcal{K}_i(r, m)))$ here, since $\mathcal{R}(\mathcal{K}_i(r, m))$ is not necessarily in $\mathcal{F}_D$ (and is certainly not in $\mathcal{F}_D$ if $\mathcal{R}(\mathcal{K}_i(r, m))$ is not a subset of D). For brevity we shall continue to abuse notation and write $\mu_D(U)$ as shorthand for $\mu_D(U \cap D)$.

Clearly there is a sense in which this program is completely insecure.

To model the system generated by this program in a way that demonstrates the lack of secrecy, we need somehow to reason about the "intention" of H. One way to do so is to assume that a string z is encoded in H's initial local state and that H follows the protocol suggested above, choosing the input value $x_i \oplus z_i$. If f_{string} is an H-information function that extracts the string from H's local state, then requiring H to maintain some form of f_{string}-secrecy with respect to L would disallow the information leak in the program above.

Although it may seem strange to be concerned about preserving the secrecy of an agent who is actively attempting to transmit secret information, this turns out to be a reasonable way to capture threats such as "rogue agents" or Trojan-horse programs whose goal is to leak confidential information to public users. Such a threat model has been the motivation for many definitions of noninterference. Intuitively, an agent j can interfere with another agent i if i's state might depend on what j does. Though some papers have suggested otherwise [McCullough 1987], we claim that nondeducibility-based definitions of secrecy provide a sensible way to reason about noninterference. If i's state depends on what j does, a reasonable model of j's local state should include information about the actions she can take that affect i. When this is the case, i's local state is correlated with j's local state if j interferes with i, so j preserves secrecy with respect to i only if j does not interfere with i.

We can identify an adversarial probability system with a set of run-based probability systems, by viewing the measures in Δ as constraints on a single measure on $\mathcal{R}$. Let $\mathcal{F}_{\mathcal{D}} = \sigma(\bigcup_{D \in \mathcal{D}} \mathcal{F}_D)$, the σ-algebra generated by the measurable sets of the probability spaces of Δ. (It is straightforward to check that $U \in \mathcal{F}_{\mathcal{D}}$ iff $U = \bigcup_{D \in \mathcal{D}} U_D$, where $U_D \in \mathcal{F}_D$.) Let $\mathcal{M}(\Delta)$ consist of all measures μ on $\mathcal{F}$ such that (1) for all $D \in \mathcal{D}$, if $\mu(D) > 0$ then $\mu \mid D = \mu_D$ (i.e., μ conditioned on D is μ_D) and (2) for all agents i and points (r, m), there exists some cell D such that $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$ and $\mu(D) > 0$. It follows from these requirements and our assumption that if $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$ then $\mu_D(\mathcal{R}(\mathcal{K}_i(r, m) \cap D) > 0$ that $\mu(\mathcal{R}(\mathcal{K}_i(r, m)) > 0$ for all agents i and points (r, m). We can thus associate $(\mathcal{R}, \mathcal{D}, \Delta)$ with the set of run-based probability systems $(\mathcal{R}, \mathcal{F}_{\mathcal{D}}, \mu)$, for $\mu \in \mathcal{M}(\Delta)$.

Rather than defining secrecy in adversarial systems directly, we give a slightly more general definition. Define a *generalized run-based probability system* to be a tuple $(\mathcal{R}, \mathcal{F}, \mathcal{M})$, where $\mathcal{M}$ is a set of probability measures on the σ-algebra $\mathcal{F}$. Similarly, define a *generalized probability system* to be a tuple $(\mathcal{R}, \mathbf{PR})$, where $\mathbf{PR}$ is a set of probability assignments. We can define secrecy in generalized (run-based) probability systems by considering secrecy with respect to each probability measure/probability assignment.

Definition 4.11. Agent j maintains *probabilistic total (resp. synchronous) secrecy* with respect to agent i in the generalized probabilistic system $(\mathcal{R}, \mathbf{PR})$ if, for all $\mathcal{PR} \in \mathbf{PR}$, j maintains probabilistic total (resp. synchronous) secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$. Agent j maintains *run-based secrecy* with respect to agent i in the generalized probabilistic run-based system $(\mathcal{R}, \mathcal{F}, \mathcal{M})$ if, for all $\mu \in \mathcal{M}$, j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mathcal{F}, \mu)$.

It is now straightforward to define secrecy in an adversarial systems by reducing

it to a generalized probabilistic system. Agent j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mathcal{D}, \Delta)$ if j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R}, \mathcal{F}_{\mathcal{D}}, \mathcal{M}(\Delta))$. Similarly, agent j maintains total (resp. synchronous) secrecy with respect to i in $(\mathcal{R}, \mathcal{D}, \Delta)$ if j maintains total (resp. synchronous) secrecy with respect to i in $(\mathcal{R}, \mathbf{PR})$, where $\mathbf{PR}$ consists of all the probability assignments determined by the run-based probability systems $(\mathcal{R}, \mathcal{F}_{\mathcal{D}}, \mu)$ for $\mu \in \mathcal{M}(\Delta)$. A straightforward analogue of Proposition 4.7 holds for adversarial systems; again, secrecy is symmetric in the presence of assumptions such as perfect recall or synchrony.

4.5 Secrecy and Evidence

Secrecy in adversarial probability systems turns out to be closely related to the notion of *evidence* in hypothesis testing (see [Kyburg 1983] for a good overview of the literature). Consider this simple example: someone gives you a coin, which may be fair or may be double-headed. You have no idea what the probability is that the coin is fair, and it may be exceedingly unlikely that the coin is double-headed. But suppose you then observe that the coin lands heads on each of 1,000 consecutive tosses. Clearly this observation provides strong *evidence* in favor of the coin being double headed.

In this example there are two hypotheses: that the coin is fair and that it is double-headed. Each hypothesis places a probability on the space of observations. In particular, the probability of seeing 1000 heads if the coin is fair is $1/2^{1000}$, and the probability of seeing 1000 heads if the coin is double-headed is 1. While we can talk of an observation being more or less likely with respect to each hypothesis, making an observation does *not* tell us how likely an hypothesis is. No matter how many heads we see, we do not know the probability that the coin is double-headed unless we have the prior probability of the coin being double headed. In fact, a straightforward computation using Bayes' Rule shows that if the prior probability of the coin being double-headed is α, then the probability of the coin being double-headed after seeing 1000 heads is $\frac{\alpha}{\alpha+((1-\alpha)/2^{1000}}$.

In an adversarial probability system $(\mathcal{R}, \mathcal{D}, \Delta)$, the initial nondeterministic choice plays the role of an hypothesis. For each $D \in \mathcal{D}$, μ_D can be thought of as placing a probability on observations, given that choice D is made. These observations then give evidence about the choice made. Agent i does not obtain evidence about which choice was made if the probability of any sequence of observations is the same for all choices.

Definition 4.12. Agent i *obtains no evidence for the initial choice* in the adversarial probability system $(\mathcal{R}, \mathcal{D}, \Delta)$ if, for all $D, D' \in \mathcal{D}$ and all points (r, m) such that $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$ and $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D' \neq \emptyset$, we have

$$\mu_D(\mathcal{R}(\mathcal{K}_i(r, m))) = \mu_{D'}(\mathcal{R}(\mathcal{K}_i(r, m))).$$

Roughly speaking, i obtains no evidence for initial choices if the initial choices (other than i's own choice) are all secret. The restriction to cells such that $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D \neq \emptyset$ and $\mathcal{R}(\mathcal{K}_i(r, m)) \cap D' \neq \emptyset$ ensures that D and D' are both compatible with i's initial choice.

To relate this notion to secrecy, we consider adversarial probability systems with

a little more structure. Suppose that for each agent $i = 1, \ldots, n$, there is a set $INIT_i$ of possible initial choices. (For example, $INIT_i$ could consist of a set of possible protocols or a set of possible initial inputs.) Let $INIT = INIT_1 \times \cdots \times INIT_n$ consist of all tuples of initial choices. For $y_i \in INIT_i$, let D_{y_i} consist of all runs in $\mathcal{R}$ where agent i's initial choice is y_i; if $y = (y_1, \ldots, y_n) \in INIT$, then $D_y = \cap_{i=1}^n D_{y_i}$ consists of all runs where the initial choices are characterized by y. Let $\mathcal{D} = \{D_y : y \in INIT\}$. To model the fact that i is aware of his initial choice, we require that for all points (r, m) and agents i, there exists y such that $\mathcal{R}(\mathcal{K}_i(r, m)) \subseteq D_y$. If $\mathcal{D}$ has this form and each agent i is aware of his initial choice, we call $(\mathcal{R}, \mathcal{D}, \Delta)$ the adversarial system *determined by INIT*.

If i obtains no evidence for the initial choice, she cannot learn anything about the initial choices of other agents. To make this precise in our framework, let $\mathcal{M}_i^{INIT}(\Delta)$ consist of the measures $\mu \in \mathcal{M}(\Delta)$ such that for all cells $D_{(y_1,\ldots,y_n)}$, we have $\mu(D_{(y_1,\ldots,y_n)}) = \mu(D_{y_i}) \cdot \mu(\cap_{j \neq i} D_{y_j})$, that is, such that the initial choices made by agent i are independent of the choices made by other agents. Intuitively, if the choices of i and the other agents are correlated, i learns something about the other agents' choices simply by making his own choice. We want to rule out such situations. Note that because all the information sets have positive probability (with respect to all $\mu \in \mathcal{M}(\Delta)$) and, for all i, there exists an information set $\mathcal{K}_i(r, m)$ such that $D_{y_i} \supseteq \mathcal{R}(\mathcal{K}_i(r, m))$, the sets D_{y_i} must also have positive probability. It follows that $INIT$ and $\mathcal{D}$ must be countable.

Given i, let i^- denote the "group agent" consisting of all agents other than i. (In particular, if the system consists of only two agents, then i^- is the agent other than i.) The local state of i^- is just the tuple of local states of all the agents other than i. Let f_{i^-} be the i^--information function that maps a global state to the tuple of (i^-)'s initial choice. As we observed in Section 3.1, our definitions apply without change to new agents that we "create" by identifying them with functions on global states. In particular, our definitions apply to i^-.

THEOREM 4.13. *Let $(\mathcal{R}, \mathcal{D}, \Delta)$ be the adversarial probability system determined by INIT and suppose that $\mathcal{R}$ is either synchronous or a system where i has perfect recall. Agent i obtains no evidence for the initial choice in $(\mathcal{R}, \mathcal{D}, \Delta)$ iff agent i^- maintains generalized run-based probabilistic f_{i^-}-secrecy with respect to i in $(\mathcal{R}, \mathcal{M}_i^{INIT}(\Delta))$.*

The assumption of either synchrony or perfect recall is necessary because the proof relies on the symmetry of run-based secrecy (as established by Proposition 4.7). We do not need to assume perfect recall for agent i^- because the theorem deals with f_{i^-}-secrecy and, on every run, f_{i^-} is constant. It therefore follows that the "agent" associated with f_{i^-} (in the sense described in Section 3.1) has perfect recall even if i^- does not.

Thinking in terms of evidence is often simpler than thinking in terms of run-based probabilistic secrecy. Moreover, the evidence-based definition of secrecy is well-defined even when the set $INIT$ of initial choices is uncountable. The connection between evidence and secrecy is particularly relevant when it comes to relating our work to that of Gray and Syverson [1998]; see Section 5.2.

5. RELATED WORK

We are certainly not the first to discuss formal definitions of secrecy: many definitions have been proposed over the last two decades. One reason for this is that researchers have sought an "ideal" definition of security that has a variety of useful properties (such as verifiability and composability). While we certainly agree that verifiability and composability are important properties, we believe that the intuition behind secrecy should be isolated from stronger properties that happen to imply secrecy—especially when we have to worry about subtle issues such as probability and nondeterminism.

In this section we consider how our definitions relate to other information-flow conditions. We show in particular how they can capture work that has been done in the synchronous setting, the asynchronous setting, and the probabilistic setting. Because there are literally dozens of papers that have, in one way or another, defined notions of secrecy or confidentiality, this section is in no way meant to be comprehensive or representative. Rather, we have chosen examples that inspired our definitions, or examples for which our definitions give some insight. In light of our earlier comments, we also focus on definitions that have tried to capture the essence of secrecy rather than notions that have been more concerned with issues like composability and verification.

One important strand of literature to which we do not compare our work directly here is the work on defining information flow and noninterference using process algebras related to CCS and CSP [Focardi and Gorrieri 1994; 2001; Ryan and Schneider 1999; Ryan et al. 2001]. Although we believe that the intuitions behind many of these definitions are closely related to our notions of secrecy, a careful discussion of this issue would take us too far afield. One possibility for future work is a careful consideration of how processes can be translated to the runs-and-systems framework in a way that captures their semantics, and then shows how some of the process-algebraic definitions can be recast as examples of secrecy. In [Halpern and O'Neill 2003] we give one instance of such a translation: we show how definitions of anonymity given using CSP by Schneider and Sidiropoulos [1996] can be captured in the runs-and-systems framework.

5.1 Secrecy in Trace Systems

Many papers in computer security define notions of secrecy (often referred to as "noninterference") using using trace-based models. Traces are usually defined as sequences of input and output events, where each event is associated with some agent (either as an input that she provides or an output that she sees). However, there have been some subtle differences among the trace-based models. In some cases, infinite traces are used; in others, the traces are finite. Similarly, some models assume that the underlying systems are synchronous while others do not. Although asynchronous system models have been more common, we first consider synchronous trace-based systems.

Both McLean [1994] and Wittbold and Johnson [1990] present their definitions of security in the context of synchronous input/output traces. These traces are essentially restricted versions of the runs introduced in this paper. Here we consider a slightly simplified version of McLean's framework and describe two well-known

noninterference properties within the framework.

Let HI be a set of possible high-level values, let LI be a set of possible low-level input values, let HO be a set of possible high-level output values, and LO be a set of possible low-level output values. We assume that these sets are pairwise disjoint and finite. A tuple $t = \langle l_i, h_i, l_o, h_o \rangle$ (with $l_i \in LI, h_i \in HI$, $l_o \in LO$, and $h_o \in HO$) represents a snapshot of a system at a given point in time; it describes the input provided to the system by a low-level agent L and a high-level agent H, and the output sent by the system to L and H. A *synchronous trace* $\tau = \langle t_1, t_2, \ldots \rangle$ is an infinite sequence of such tuples. It represents an infinite execution sequence of the entire system by describing the input/output behavior of the system at any given point in time.[5] A *synchronous trace system* is a set Σ of synchronous traces, representing the possible execution sequences of the system.

In a synchronous trace system, the local state of an agent can be defined using a *trace projection function*. For example, let $|_L$ be the function projecting τ onto the low-level input/output behavior of τ, so that if

$$\tau = \langle \langle l_i^{(1)}, h_i^{(1)}, l_o^{(1)}, h_o^{(1)} \rangle, \langle l_i^{(2)}, h_i^{(2)}, l_o^{(2)}, h_o^{(2)} \rangle, \ldots \rangle,$$

then

$$\tau|_L = \langle \langle l_i^{(1)}, l_o^{(1)} \rangle, \langle l_i^{(2)}, l_o^{(2)} \rangle, \ldots \rangle.$$

Similarly, we can define a function $|_H$ that extracts high-level input/output traces and a function $|_{HI}$ that extracts just high-level *input* traces.

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

It is easy to see that synchronous trace systems can be viewed as systems in the multiagent systems framework. Given a trace τ, we can define the run r^τ such that $r^\tau(m) = (\tau_m|_L, \tau_m|_H)$. (For simplicity, we have omitted the environment state from the global state in this construction, since it plays no role.) Given a synchronous trace system Σ, let $\mathcal{R}(\Sigma) = \{r^\tau : \tau \in \Sigma\}$. It is easy to check that $R(\Sigma)$ is synchronous, and that both agents L and H have perfect recall.

McLean defines a number of notions of secrecy in his framework. We consider two of the best known here: *separability* [McLean 1994] and *generalized noninterference* [McCullough 1987]. Separability, as its name suggests, ensures secrecy between the low-level and high-level agents, whereas generalized noninterference ensures that the low-level agent is unable to know anything about high-level input behavior.

Definition 5.1. A synchronous trace system Σ satisfies *separability* if, for every pair of traces $\tau, \tau' \in \Sigma$, there exists a trace $\tau'' \in \Sigma$ such that $\tau''|_L = \tau|_L$ and $\tau''|_H = \tau'|_H$.

Definition 5.2. A synchronous trace system Σ satisfies *generalized noninterference* if, for every pair of traces $\tau, \tau' \in \Sigma$, there exists a trace $\tau'' \in \Sigma$ such that $\tau''|_L = \tau|_L$ and $\tau''|_{HI} = \tau'|_{HI}$.

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

These definitions are both special cases of nondeducibility, as discussed in Section 3.1: take the set of worlds W to be Σ, the information function g to be $|_L$, and the information function h to be $|_H$ (for separability) and $|_{HI}$ (for generalized noninterference).[6] In our framework, separability essentially corresponds to synchronous secrecy, whereas generalized noninterference corresponds to synchronous $|_{HI}$-secrecy. The following proposition makes this precise. Let f_{hi} be the information function that extracts a high-level input trace prefix from a point in exactly the same way that $|_{HI}$ extracts it from the infinite trace.

PROPOSITION 5.3. *If a synchronous trace system Σ satisfies separability (resp., generalized noninterference), then H maintains synchronous secrecy (resp., synchronous f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

PROOF. We prove the result for separability. The proof for generalized noninterference is similar and left to the reader. Suppose that Σ satisfies separability. Let r^τ and $r^{\tau'}$ be runs in $\mathcal{R}(\Sigma)$. We want to show that, for all times m, we have that $\mathcal{K}_L(r^\tau, m) \cap \mathcal{K}_H(r^{\tau'}, m) \neq \emptyset$. Since σ satisfies separability, there exists a trace $\tau'' \in \Sigma$ such that $\tau''|_L = \tau|_L$ and $\tau''|_H = \tau'|_H$. It follows immediately that $\tau''_m|_L = \tau_m|_L$ and $\tau''_m|_H = \tau'_m|_H$. Thus, $(r^{\tau''}, m) \in \mathcal{K}_L(r^\tau, m) \cap \mathcal{K}_H(r^{\tau'}, m)$. □

The converse to Proposition 5.3 is not quite true. There is a subtle but significant difference between McLean's framework and ours. McLean works with *infinite* traces; separability and generalized noninterference are defined with respect to *traces* rather than sets of *points* (i.e., trace prefixes). To see the impact of this, consider a system Σ where the high-level agent inputs either infinitely many 0s or infinitely many 1s. The output to the low-level agent is always finitely many 0s followed by infinitely 1s, except for a single trace where the high-level agent inputs infinitely many 0s and the low-level agent inputs infinitely many 0s. Thus, the system consists of the following traces, where we have omitted the low-level inputs since they do not matter, and the high-level outputs, which can taken to be constant:

$$\begin{array}{ll} (0^k 1^\infty, 0^\infty), & k = 0, 1, 2, 3, \ldots \\ (0^k 1^\infty, 1^\infty), & k = 0, 1, 2, 3, \ldots \\ (0^\infty, 0^\infty). & \end{array}$$

In the system $\mathcal{R}(\Sigma)$, H maintains synchronous secrecy and thus synchronous f_{hi}-secrecy with respect to L, because by looking at any finite trace prefix, L cannot tell whether the high-level inputs have been 0s or 1s. However, Σ does not satisfy separability or generalized interference. If L "sees" infinitely many 0s, he immediately knows that the high-level inputs have been 0s. This seems unreasonable. After all, agents only makes observations at finite points in time.

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

[6] Actually, it is not difficult to see that if the information functions g and h are restricted to trace projection functions, then nondeducibility is essentially equivalent in expressive power to *selective interleaving functions*, the mechanism for defining security properties introduced by McLean [1994].

Definition 5.4. A synchronous trace system Σ is *limit closed* [Emerson 1983] if, for all synchronous traces τ, we have $\tau \in \Sigma$ iff for every time k there exists a trace $\tau' \in \Sigma$ such that $\tau'_k = \tau_k$.

Under the assumption of limit closure, we do get the converse to Proposition 5.3.

PROPOSITION 5.5. *A limit-closed synchronous trace system Σ satisfies separability (resp. generalized noninterference) iff H maintains synchronous secrecy (resp., synchronous f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

While we believe that it is unreasonable in general to assume that an agent's view includes the entire run (as McLean's definitions implicitly do), these results nonetheless demonstrate the close connection between our definition of synchronous f-secrecy and security properties such as separability and generalized noninterference.

Up to now we have considered a synchronous trace model, where the input and output events of high and low users occur in lockstep. However, many trace-based definitions of security are given in an asynchronous setting. We consider a number of definitions of secrecy in this setting. For uniformity we use the terminology of Mantel [2003], who has carefully compiled a variety of well-known trace-based properties into a single framework.

In Mantel's framework, traces are not infinite sequences of input/output value tuples, but finite sequences of input/output events. For example, if l and l' are low-level events while h and h' are high-level events, a possible system trace could be

$$\tau = \langle l, h, l, h', h', l', l', l, h \rangle.$$

As with synchronous trace systems, we denote a projection function for a set A by $|_A$. Thus, if τ is defined as above, we have

$$\tau|_L = \langle l, l, l', l', l \rangle,$$

where $|_L$ is the low-level projection function. Note that because asynchronous traces are sequences of events rather than tuples, the low-level projection function ignores high-level events altogether. This means that a low-level view of the system may remain completely unchanged even as many, many high-level input events occur.

An *asynchronous trace system* is a set of traces that is closed under trace prefixes. There is a straightforward way of associating with each system a set of runs. A set T of traces is *run-like* if, for all traces τ_1 and τ_2 in T, either τ_1 is a prefix of τ_2 or τ_2 is a prefix of τ_1. Intuitively, a run corresponds to a maximal run-like set of traces. More formally, let T be a maximal set of run-like traces. Note that if T is infinite, then for all $n \geq 0$ there exists exactly one trace in T of length n (where the length of $\langle t_0, \ldots, t_{n-1} \rangle$ is n); if T is finite, then there is some $N \geq 0$ such that T has exactly one trace of length n for all $n \leq N$. If T is infinite, let the run r^T be such that $r^T(m) = \langle \tau^m|_L, \tau^m|_H \rangle$, where τ^m is the unique trace in T of length m. If T is finite, let r^T be such that $r^T(m) = \langle \tau^m|_L, \tau^m|_H \rangle$ if $m \leq N$, where N is the length of the longest trace in T, and $r^T(m) = r^T(N)$ if $m \geq N$; that is, the final state repeats forever. Given an asynchronous trace system Σ, let $\mathcal{R}(\Sigma)$ denote the set of all runs of the form r^T, where T is a maximal set of run-like traces in Σ.

Trace-based security properties are usually expressed as closure properties on sets of traces, much like our possibilistic definitions of secrecy; see [Mantel 2000] for more details. We focus here on the definitions of asynchronous separability and generalized noninterference given by Zakinthinos and Lee [1997].

Definition 5.6. An asynchronous trace system Σ satisfies *asynchronous separability* if, for all traces $\tau, \tau' \in \Sigma$, if τ'' is a trace that results from an arbitrary interleaving of the traces $\tau|_L$ and $\tau'|_H$, then $\tau'' \in \Sigma$.

The definition of generalized noninterference is slightly more complicated, because the trace that results from interleaving does not include high inputs:

Definition 5.7. An asynchronous trace system Σ satisfies *asynchronous generalized noninterference* if, for all traces $\tau, \tau' \in \Sigma$, if τ'' is a trace that results from an arbitrary interleaving of the traces $\tau|_L$ and $\tau'|_{HI}$, there exists a trace τ''' such that $\tau'''|_{L \cup HI} = \tau''|_{L \cup HI}$.

It is straightforward to relate these definitions to secrecy. Exactly as in the synchronous case, let f_{hi} be an information function that extracts a high-level input trace prefix from a point: if $r^T(m) = \langle \tau|_L, \tau|_H \rangle$, let $f_{hi}(r^T, m) = \tau|_{HI}$.

PROPOSITION 5.8. *If Σ is an asynchronous trace system that satisfies asynchronous separability (resp. asynchronous generalized noninterference), then H maintains total secrecy (resp. total f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

The converse of Proposition 5.8 does not necessarily hold. We demonstrate this by providing a counterexample that works for both separability and generalized noninterference. Suppose that there are no high output events, only one low output event l_o, and arbitrary sets LI and HI of low and high input events, respectively. Consider the system consisting of all traces τ involving these events such that l_o occurs at most once in τ, and when it occurs, it does not follow any high input events. In $\mathcal{R}(\Sigma)$, H maintains total secrecy and f_{hi}-secrecy with respect to L, because any local state for L is compatible with any local state for H. (Because the system is asynchronous, L learns nothing by seeing l_o: when L sees l_o, he thinks it possible that arbitrarily many high input events could have occurred *after* l_o. Furthermore, L learns nothing about H when he does *not* see l_o: it is always possible that no high input events have occurred and that l_o may yet occur.) However, Σ does not satisfy asynchronous separability or asynchronous generalized noninterference, because interleavings where a high input event precedes l_o are ruled out by construction.

This example illustrates a potential weakness of our approach to secrecy. Although H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$, there is a sense in which L learns something about H. Consider a point (r, m) in $\mathcal{R}(\Sigma)$ at which L has not seen l_o. At that point, L knows that *if* a high event has occurred, he will never see l_o. This knowledge does not violate secrecy, because it does not depend on the local state of H; it is not an H-local fact. But there is a sense in which this fact can be said to be "about" H: it is information about a correlation between high events and a particular low event. Is such information leakage a problem? We have not been able to construct an example where it is. But it is worth pointing out that all of our definitions of secrecy aim to protect the local state of some particular user,

and therefore that any "secret information" that cannot be characterized as a local proposition is not protected.

In any case, we can show that total secrecy and separability are equivalent if we assume a particularly strong form of asynchrony that rules out a temporal dependence between high and low events. Formally, Σ is *closed under interleavings* if for all asynchronous traces τ and τ', if $\tau \in \Sigma$, $\tau'|_L = \tau|_L$ and $\tau'|_H = \tau'|_H$, then $\tau' \in \Sigma$. Though this requirement allows L to learn about high events that may occur in the future (or that have possibly occurred in the past), it rules out any knowledge of the ordering of high and low events in a given run. With this requirement, total secrecy and asynchronous separability coincide.

PROPOSITION 5.9. *If Σ is an asynchronous trace system that is closed under interleavings, then Σ satisfies asynchronous separability iff H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$.*

A similar result is true for generalized noninterference and f_{hi}-secrecy if we modify the definition of closure under interleavings to allow L to learn something about the ordering of high output events; we omit the details here.

5.2 Secrecy and User Protocols

In Section 4.4 we described a program that allows a high-level agent H to transmit arbitrarily long data strings directly to a low-level agent L even though H's actual input values remain secret. We then described one possible way to model the lack of secrecy in the system, by assuming that the string that H wants to transmit is included in his initial local state. This example demonstrates that generalized noninterference (as defined in the previous section) is insufficient to ensure that one agent cannot interfere with another, even if we restrict our concern to possibilistic information flows. By using a clever-enough protocol, an agent may be able to exploit the nondeterminism of a system to transmit data to another agent.

This problem was noted by Wittbold and Johnson [1990], who introduced *nondeducibility on strategies* to protect the protocol (or "strategy") employed by high-level agents. We modify their definition slightly so that it is compatible with McLean's framework of synchronous traces. A *protocol* $\mathbf{H}$ is a function from a high-level input/output trace prefix $\tau_k|_H$ to a high-level input value $h_i \in HI$. Intuitively, a protocol tells the agent H what to do at each step, given what he has already seen and done. A trace τ is *consistent* with a protocol $\mathbf{H}$ if, for all k, $\mathbf{H}(\tau_{k-1}|_H) = h_i^{(k)}$, where $h_i^{(k)}$ is the high-level input value of the kth tuple in τ. A synchronous trace system Σ *satisfies nondeducibility on strategies* if, for all traces $\tau \in \Sigma$ and every high-level protocol $\mathbf{H}$ consistent with some trace in Σ, there exists a trace τ' that is consistent with $\mathbf{H}$ such that $\tau'|_L = \tau|_L$. If the protocol of the high-level agent is included as part of her local state, and f_{prot} is an H-information function that extracts the protocol of the high-level agent from the local state, then it is straightforward to show that nondeducibility on strategies is simply synchronous f_{prot}-secrecy.

Gray and Syverson [1998] extend nondeducibility on strategies to probabilistic systems using the Halpern-Tuttle framework [Halpern and Tuttle 1993]. In Gray and Syverson's terminology, low-level and high-level agents use probabilistic protocols $\mathbf{L}$ and $\mathbf{H}$, respectively. Again, the protocols ($\mathbf{H}$ and $\mathbf{L}$) determine what the

agents H and L will input next, given what they have seen and done so far. The system is assumed to have a fixed probability distribution $\mathbf{O}$ that determines its output behavior, given the inputs and outputs seen so far. Formally, for each trace prefix τ of length k, $\mathbf{H}(\cdot \,|\, (\tau|_H))$ is a probability measure on high-level input events that occur at time $k+1$, given the projection of τ onto the high-level input/output; similarly, $\mathbf{L}(\cdot \,|\, (\tau|_L))$ is a probability measure on low-level input events that occur at time $k+1$ and $\mathbf{O}(\cdot \,|\, \tau)$ is a probability measure on output events that occur at time $k+1$, given τ. Gray and Syverson require that the choices made by H, L, and the system at each time step be probabilistically independent. With this assumption, $\mathbf{H}$, $\mathbf{L}$, and $\mathbf{O}$ determine a conditional distribution that we denote $\mu_{\mathbf{H},\mathbf{L},\mathbf{O}}$, where

$$\mu_{\mathbf{L},\mathbf{H},\mathbf{O}}(\langle l_i, h_i, l_o, h_o\rangle \,|\, \tau) = \mathbf{L}(l_i \,|\, (\tau|_L)) \cdot \mathbf{H}(h_i \,|\, (\tau|_H)) \cdot \mathbf{O}(l_o, h_o \,|\, \tau).$$

Let Λ and Γ be countable sets of protocols for the low-level and high-level agents, respectively.[7] Given Λ, Γ, and $\mathbf{O}$ (and, implicitly, sets of low and high input and output values), we can define an adversarial probability system $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O})$ in a straightforward way. Let Σ consist of all synchronous traces over the input and out values. For each joint protocol $(\mathbf{L}, \mathbf{H}) \in \Lambda \times \Gamma$, let $\mathcal{R}(\Sigma, \mathbf{L}, \mathbf{H})$ consist of all runs defined as in our earlier mapping from synchronous traces to runs, except that now we include $\mathbf{L}$ in the low-level agent's local state and $\mathbf{H}$ in the high-level agent's local state. Let $\mathcal{R}(\Sigma, \Lambda \times \Gamma) = \cup_{(\mathbf{L},\mathbf{H}) \in \Lambda \times \Gamma} \mathcal{R}(\Sigma, \mathbf{L}, \mathbf{H})$. We can partition $\mathcal{R}(\Sigma, \Lambda \times \Gamma)$ according to the joint protocol used; let $\mathcal{D}(\Lambda, \Gamma)$ denote this partition. Given the independence assumptions, the joint protocol $(\mathbf{L}, \mathbf{H})$ also determines a probability $\mu_{\mathbf{L},\mathbf{H},\mathbf{O}}$ on $\mathcal{R}(\Sigma, \mathbf{L}, \mathbf{H})$. Let $\Delta(\Lambda, \Gamma, \mathbf{O}) = \{\mu_{\mathbf{L},\mathbf{H},\mathbf{O}} : \mathbf{L} \in \Lambda, \mathbf{H} \in \Gamma\}$. Let $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O}) = (\mathcal{R}(\Sigma, \Lambda \times \Gamma), \mathcal{D}, \Delta\}$. We can now define Gray and Syverson's notion of secrecy in the context of these adversarial systems.

Definition 5.10. An adversarial system $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O})$ satisfies *probabilistic noninterference* if, for all low-level protocols $\mathbf{L} \in \Lambda$, points (r, m) where L's protocol is $\mathbf{L}$, and high-level protocols $\mathbf{H}, \mathbf{H}' \in \Gamma$, we have $\mu_{(\mathbf{L},\mathbf{H},\mathbf{O})}(\mathcal{K}_L(r, m)) = \mu_{(\mathbf{L},\mathbf{H}',\mathbf{O})}(\mathcal{K}_L(r, m))$.

THEOREM 5.11. *The following are equivalent:*

(a) $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O})$ *satisfies probabilistic noninterference;*

(b) L obtains no evidence about H's protocol (in the sense of Definition 4.12) in $\mathcal{R}^*(\Lambda, \Gamma, \mathbf{O})$*;*

(c) H maintains generalized run-based probabilistic f_{prot}*-secrecy with respect to L in* $(\mathcal{R}(\Sigma, \Lambda \times \Gamma), \mathcal{M}^{INIT}(\Delta(\Lambda, \Gamma, \mathbf{O})))$*, where* f_{prot} *is the information function that maps from H's local state to H's protocol;*

(d) H maintains generalized probabilistic synchronous f_{prot}*-secrecy with respect to L in the standard generalized probability system determined by* $(\mathcal{R}(\Sigma, \Lambda \times \Gamma), \mathcal{M}^{INIT}(\Delta(\Lambda, \Gamma, \mathbf{O})))$*.*

[7] Gray and Syverson take Λ and Γ to consist of *all possible* probabilistic protocols for the low-level and high-level agent, respectively, but their approach still makes sense if Λ and Γ are arbitrary sets of protocols, and it certainly seems reasonable to assume that there are only countably many protocols that H and L can be using.

PROOF. The fact that (a) implies (b) is immediate from the definitions, since H's initial choice is the protocol $\mathbf{H}$. The equivalence of (b) and (c) follows from Theorem 4.13. Finally, since the traces in Σ are synchronous, the equivalence of (c) and (d) follows from Proposition 4.8. □

6. CONCLUSION

We have defined general notions of secrecy for systems where multiple agents interact over time, and have given syntactic characterizations of our definitions that connect them to logics of knowledge and probability. We have applied our definitions to the problem of characterizing the absence of information flow, and have shown how our definitions can be viewed as a generalization of a variety of information-flow definitions that have been proposed in the past.

We are not the first to attempt to provide a general framework for analyzing secrecy; see, for example, [Focardi and Gorrieri 2001; Mantel 2000; McLean 1994; Ryan and Schneider 1999; Zakinthinos and Lee 1997] for some other attempts. However, we believe that our definitions are more closely related to the intuitions that people in the field have had, because those definitions have often been expressed in terms of the knowledge of the agents who interact with a system.

Our definitions of probabilistic secrecy (and their plausibilistic generalizations) demonstrate the underlying simplicity and unity of our definitions. Likewise, our results on the symmetry of secrecy illustrate the close connection between notions of secrecy and independence. The definitions and results that we have presented, and their underlying intuitions of knowledge and independence, do not depend on the particular system representation that we describe here, so they should be broadly applicable. Indeed, one major theme of our work is the importance of having good system models. Given the right system model and the right measure of uncertainty, a reasonable definition of secrecy—often some form of run-based f-secrecy—usually follows quite easily. By providing a general, straightforward way to model systems, the runs-and-systems framework provides a useful foundation for appropriate definitions of security.

Although we have discussed secrecy largely with respect to the kinds of input and output systems that have been popular with the theoretical security community, our definitions of secrecy apply in other contexts, such as protocol analysis, semantics for imperative programming languages, and database theory. Chor, Goldreich, Kushilevitz, and Sudan [1998], for example, consider the situation where a user wants to query a replicated database for some specific database item, but wants a guarantee that no one will be able to determine, based on his query, which item he wants. It is not hard to show that the definition of privacy given by Chor et al. is a special case of secrecy in an adversarial system with a cell corresponding to each possible item choice.

There are several possible directions for future work. One is the verification of secrecy properties. Because we have provided syntactic characterizations of several secrecy properties in terms of knowledge and local propositions, it would seem that model-checking techniques could be applied directly. (Van der Meyden [1998] gives some recent results on model-checking in the runs-and-systems framework.) However, verifying a secrecy property requires verifying an infinite set of formulas, and

developing techniques to do this efficiently would seem to require some nontrivial advances to the state of the art in model checking. Of course, to the extent that we are interested in more limited forms of secrecy, where an agent is restricted from knowing a small set of formulas, knowledge-based model-checking techniques may be immediately applicable. At any rate, we emphasize that it is our goal in this paper to provide general techniques for the specification, rather than the verification, of secrecy.

Another direction for future work is a careful consideration of how secrecy definitions can be weakened to make them more useful in practice. Here we briefly consider some of the issues involved:

—*Declassification:* Not all facts can be kept secret in a real-world computer system. The canonical example is password checking, where a system is forced to release information when it tells an attacker that a password is invalid. Declassification for information-flow properties has been addressed by, for example, Myers, Sabelfeld, and Zdancewic [2004]. It would be interesting to compare their approach to our syntactic approach to secrecy, keeping in mind that our syntactic definitions can be easily weakened simply by removing facts from the set of facts that an agent is required to think are possible.

—*Computational secrecy:* Our definitions of secrecy are most appropriate for attackers with unlimited computational power, since agents "know" any fact that follows logically from their local state, given the constraints of the system. Such an assumption is unreasonable for most cryptographic systems, where secrecy depends on the inability of attackers to solve difficult computational problems. The process-algebraic approach advocated by Mitchell, Ramanathan, Scedrov, and Teague [2004] and the work on probabilistic algorithm knowledge of Halpern and Pucella [2003b] may help to shed light on how definitions of secrecy can be weakened to account for agents with computational limitations.

—*Quantitative secrecy:* Our definitions of probabilistic secrecy require independence: an agent's posterior probability distribution on the possible local states of a secret agent must be exactly the same as his prior distribution. This requirement can be weakened using the information-theoretic notions of entropy and mutual information. Rather than requiring that no information flow from one user to another, we can quantitatively bound the mutual information between their respective local states. Information-theoretic approaches to secrecy have been discussed by Wittbold and Johnson [1990], and more recently by Clark, Hunt, and Malacaria [2002], Lowe [2002], and Di Pierro, Hankin, and Wiklikcy [2002].

—*Statistical privacy:* In some systems, such as databases that release aggregate statistical information about individuals, our definitions of secrecy are much too strong because they rule out the release of any useful information. Formal definitions of secrecy and privacy for such systems have recently been proposed by Evfimievski, Gehrke, and Srikant [2003] and by Chawla, Dwork, McSherry, Smith and Wee [2005]. These definitions seek to limit the information that an attacker can learn about a user whose personal information is stored in the database. It would be interesting to cast those definitions as weakenings of secrecy.

These weakenings of secrecy are all conceptually different, but it seems highly

likely that there are relations and connections among them. We hope that our work will help to clarify some of the issues involved.

A. EXAMPLES OF SYSTEMS

In this section, we give examples of simple systems that show the limitations of various theorems. All the systems involve only two agents, and we ignore the environment state. We describe each run using the notation

$$\langle (X_{i,1}, X_{j,1}), (X_{i,2}, X_{j,2}), (X_{i,3}, X_{j,3}), \ldots \rangle,$$

where $X_{i,k}$ is the local state of agent i at time k. For asynchronous systems, we assume that the final global state—$(X_{i,3}, X_{j,3})$, in the example above—is repeated infinitely. For synchronous systems we need different states at each time step, so we assume that global states not explicitly listed encode the time in some way, so change at each time step. For notational simplicity, we use the same symbol for a local state and its corresponding information set.

EXAMPLE A.1**:** Suppose that the synchronous system $\mathcal{R}$ consists of the following two runs:

$$\begin{aligned} r_1 &= \langle (X, A), (Y_1, B_1), (Y_2, B_2), \ldots \rangle \\ r_2 &= \langle (Z, A), (Y_1, C_1), (Y_2, C_2), \ldots \rangle \end{aligned}$$

Note that agent 2 has perfect recall in $\mathcal{R}$, but agent 1 does not (since at time 0 agent 1 knows the run, but at all later times, he does not). It is easy to check that agent 2 maintains synchronous secrecy with respect to 1, but not run-based secrecy, since $\mathcal{R}(B_1) \cap \mathcal{R}(Z) = \emptyset$.

For the same reasons, if we take the probability measure μ on $\mathcal{R}$ with $\mu(r_1) = \mu(r_2) = 1/2$, probabilistic synchronous secrecy and run-based probabilistic secrecy do not coincide. This shows that the perfect recall requirement is necessary in both Propositions 3.10 and 4.8. □

EXAMPLE A.2**:** Suppose that the $\mathcal{R}$ consists of the following three runs (where, in each case, the last state repeats infinitely often):

$$\begin{aligned} r_1 &= \langle (X, A) \ldots \rangle \\ r_2 &= \langle (X, B), (Y, A) \ldots \rangle \\ r_3 &= \langle (Y, A) \ldots \rangle, \end{aligned}$$

It is easy to see that agent 2 maintains run-based secrecy with respect to agent 1 in $\mathcal{R}$, but not total secrecy or synchronous secrecy (since, for example, $Y \cap B = \emptyset$).

Now consider a probability measure μ on $\mathcal{R}$ such $\mu(r_1) = \mu(r_3) = 2/5$, and $\mu(r_2) = 1/5$. Then $\mu(\mathcal{R}(A) \,|\, \mathcal{R}(X)) = \mu(\mathcal{R}(A) \,|\, \mathcal{R}(Y)) = 1$ and $\mu(\mathcal{R}(B) \,|\, \mathcal{R}(X)) = \mu(\mathcal{R}(B) \,|\, \mathcal{R}(Y)) = 1/3$, so agent 2 maintains run-based probabilistic secrecy with respect to 1 in $\mathcal{R}$. 1 does not maintain probabilistic secrecy with respect to 2 in $(\mathcal{R}, \mu)$, since $\mu(\mathcal{R}(X) \,|\, \mathcal{R}(A)) = 3/5$, while $\mu(\mathcal{R}(X) \,|\, \mathcal{R}(B)) = 1$. Thus, if the agents do not have perfect recall and the system is not synchronous, then run-based probabilistic secrecy is not necessarily symmetric. □

EXAMPLE A.3.: Suppose that the synchronous system $\mathcal{R}$ consists of the following four runs:

$$\begin{aligned} r_1 &= \langle (X,A),(Y_1,C_1),(Y_2,C_2),\ldots\rangle \\ r_2 &= \langle (X,B),(Y_1,D_1),(Y_2,D_2),\ldots\rangle \\ r_3 &= \langle (Q,A),(R_1,D_1),(R_2,D_2),\ldots\rangle \\ r_4 &= \langle (Q,B),(R_1,C_2),(R_2,C_2),\ldots\rangle \end{aligned}$$

Note that agent 2 does not have perfect recall in $\mathcal{R}$, although agent 1 does. Let μ give each of these runs equal probability. It is easy to check that for all $i \geq 1$, $\mu(\mathcal{R}(A)\,|\,\mathcal{R}(X)) = \mu(\mathcal{R}(A)\,|\,\mathcal{R}(Q)) = 1/2$, $\mu(\mathcal{R}(B)\,|\,\mathcal{R}(X)) = \mu(\mathcal{R}(B)\,|\,\mathcal{R}(Q)) = 1/2$, $\mu(\mathcal{R}(C_i)\,|\,\mathcal{R}(X)) = \mu(\mathcal{R}(C_i)\,|\,\mathcal{R}(Q)) = 1/2$, and $\mu(\mathcal{R}(D_i)\,|\,\mathcal{R}(X)) = \mu(\mathcal{R}(D_i)\,|\,\mathcal{R}(Q)) = 1/2$. Because $\mathcal{R}(X) = \mathcal{R}(Y_i)$ and $\mathcal{R}(Q) = \mathcal{R}(R_i)$ for all $i \geq 1$, it follows that agent 2 maintains run-based probabilistic secrecy with respect to 1 in $(\mathcal{R},\mu)$.

Now, let p be a primitive proposition and let π be an interpretation such that p is true if 2's local state is either A or D_1. Thus, p is 2-local in $\mathcal{I} = (\mathcal{R},\mu,\pi)$. Since $\mu(\mathcal{R}(A) \cup \mathcal{R}(D_1)\,|\,\mathcal{R}(X)) = 1$ while $\mu(\mathcal{R}(A) \cup \mathcal{R}(D_1)\,|\,\mathcal{R}(Q)) = 1/2$, there is no constant σ such that $\mathcal{I} \models \Pr_1(\diamondsuit p) = \sigma$. This shows that the assumption that agent j has perfect recall is necessary in Theorem 4.10. □

B. PROOFS FOR SECTION 3

PROPOSITION 3.9. *If $\mathcal{R}$ is a system where i and j have perfect recall, C depends only on timing, and j maintains C-secrecy with respect to i, then j maintains run-based secrecy with respect to i.*

PROOF. Given (r,m) and (r',m'), we must find a run r'' and times m_1 and m_2 such that $r''_i(m_1) = r_i(m)$ and $r''_j(m_2) = r'_j(m')$. Because C depends only on timing, there exists a point (r,n) such that $(r',m') \in C(r,n)$. The proof now splits into two cases:

—Suppose that $n \geq m$. By C-secrecy, there exists a point (r'',m_2) such that $r''_i(m_2) = r_i(n)$ and $r''_j(m_2) = r'_j(m')$. Because i has perfect recall, there exists some $m_1 \leq m_2$ such that $r''_i(m_1) = r_i(m)$.

—Suppose that $m > n$. Because C depends only on timing, there exists $n' \geq m'$ such that $(r',n') \in C(r,m)$. By C-secrecy, there exists a point (r'',m_2) such that $r''_i(m_2) = r_i(m)$ and $r''_j(m_2) = r'_j(n')$. Because j has perfect recall, there exists some $m_1 \leq m_2$ such that $r''_j(m_1) = r'_j(m')$.

□

PROPOSITION 3.10. *If $\mathcal{R}$ is a synchronous system where both i and j have perfect recall, then agent j maintains synchronous secrecy with respect to i iff j maintains run-based secrecy with respect to i.*

PROOF. Suppose that agent j maintains synchronous secrecy with respect to j in $\mathcal{R}$. Because both i and j have perfect recall, j maintains run-based secrecy with respect to i by Proposition 3.9.

Conversely, suppose that j maintains run-based secrecy with respect to i. Given runs $r, r' \in \mathcal{R}$ and any time m, there exists a run r'', and times n and n', such

that $r_i''(n) = r_i(m)$ and $r_j''(n') = r_j'(m)$. By synchrony, $m = n = n'$, and we have $r_i''(m) = r_i(m)$ and $r_j''(m) = r_j'(m)$. Thus j maintains synchronous secrecy with respect to i. □

PROPOSITION 3.11. *A formula φ is j-local in an interpreted system $\mathcal{I} = (\mathcal{R}, \pi)$ iff there exists a set Ω of j-information sets such that $(\mathcal{I}, r, m) \models \varphi$ whenever $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$.*

PROOF. Suppose that φ is j-local. Let

$$\Omega = \{\mathcal{K}_j(r, m) \mid (\mathcal{I}, r, m) \models \varphi\}.$$

If $(\mathcal{I}, r, m) \models \varphi$, then $\mathcal{K}_j(r, m) \in \Omega$ by definition, so $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$. Likewise, if $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$, then $(r, m) \in \mathcal{K}_j(r', m')$ for some (r', m') such that $(\mathcal{I}, r', m') \models \varphi$. By j-locality, $(\mathcal{I}, r, m) \models \varphi$.

Conversely suppose that there exists a set of j-information sets Ω such that $(\mathcal{I}, r, m) \models \varphi$ whenever $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$. We need to show that φ is j-local. Suppose that $r_j(m) = r_j'(m')$. If $(\mathcal{I}, r, m) \models \varphi$, then $(r, m) \in \mathcal{K}_j(r'', m'')$ for some $\mathcal{K}_j(r'', m'') \in \Omega$, and clearly $(r', m') \in \mathcal{K}_j(r'', m'') \subseteq \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$ too, so $(\mathcal{I}, r', m') \models \varphi$ by assumption. □

THEOREM 3.12. *Suppose that C is an i-allowability function. Agent j maintains C-secrecy with respect to agent i in system $\mathcal{R}$ iff, for every interpretation π and point (r, m), if φ is j-local and $(\mathcal{I}, r', m') \models \varphi$ for some $(r', m') \in C(r, m)$, then $(\mathcal{I}, r, m) \models P_i\varphi$.*

PROOF. Suppose that j maintains C-secrecy with respect to i in $\mathcal{R}$. Let π be an interpretation, let (r, m) be a point, and let φ be a formula that is j-local such that $(\mathcal{I}, r', m') \models \varphi$ for some $(r', m') \in C(r, m)$. By C-secrecy, there exists a point $(r'', m'') \in \mathcal{K}_i(r, m) \cap \mathcal{K}_j(r', m')$. Because φ is j-local, $(\mathcal{I}, r'', m'') \models \varphi$. Thus $(\mathcal{I}, r, m) \models P_i\varphi$, as required.

For the converse, given $(r, m) \in \mathcal{PT}(\mathcal{R})$ and $(r', m') \in C(r, m)$, let π be an interpretation such that $\pi(r'', m'')(p) = \mathbf{true}$ iff $(r'', m'') \in \mathcal{K}_j(r', m')$. Let $\mathcal{I} = (\mathcal{R}, \pi)$. Clearly, p is j-local. By assumption, $(\mathcal{I}, r, m) \models P_i p$. Thus, there exists some point $(r'', m'') \in \mathcal{K}_i(r, m)$ such that $(\mathcal{I}, r'', m'') \models p$. By definition, $(r'', m'') \in \mathcal{K}_j(r', m')$. Because $(r'', m'') \in \mathcal{K}_i(r, m) \cap \mathcal{K}_j(r', m')$, j maintains C-secrecy with respect to i in $\mathcal{R}$. □

THEOREM 3.14. *Agent j maintains run-based secrecy with respect to agent i in system $\mathcal{R}$ iff, for every interpretation π, if φ is j-local and satisfiable in $\mathcal{I} = (\mathcal{R}, \pi)$, then $\mathcal{I} \models P_i \Diamond \varphi$.*

PROOF. Suppose that j maintains run-based secrecy with respect to i. Let π be an interpretation and let φ be a j-local formula formula that is satisfiable in $\mathcal{I} = (\mathcal{R}, \pi)$. Choose a point (r, m). Because φ is satisfiable, there exists a point (r', m') such that $(\mathcal{I}, r', m') \models \varphi$. Because j maintains run-based secrecy with respect to i, there exist a run r'' and times n and n' such that $r_i''(n) = r_i(m)$ and $r_j''(n') = r_j'(m')$. By j-locality, $(\mathcal{I}, r'', n') \models \varphi$. It follows that $(\mathcal{I}, r'', n) \models \Diamond \varphi$, and that $(\mathcal{I}, r, m) \models P_i \Diamond \varphi$, as desired.

For the converse, given points (r, m) and (r', m'), let π be an interpretation such that $\pi(r'', m'')(p) = \mathbf{true}$ iff $(r'', m'') \in \mathcal{K}_j(r', m')$. We must show that

$\mathcal{R}(\mathcal{K}_i(r,m)) \cap \mathcal{R}(\mathcal{K}_j(r',m')) \neq \emptyset$. Clearly p is j-local and satisfiable, so $(\mathcal{I}, r, m) \models P_i \diamondsuit p$. Thus, there exists a point $(r'', n) \in \mathcal{K}_i(r,m)$ such that $(\mathcal{I}, r'', n) \models \diamondsuit p$. By definition of p, there exists n' such that $(r'', n') \in \mathcal{K}_j(r', m')$. It follows that $r'' \in \mathcal{R}(\mathcal{K}_i(r,m)) \cap \mathcal{R}(\mathcal{K}_j(r',m'))$. □

C. PROOFS FOR SECTION 4

PROPOSITION 4.2. *If $(\mathcal{R}, \mathcal{PR})$ is a probability system such that $\mu_{r,m,i}(\{(r,m)\}) > 0$ for all points (r,m) and j maintains probabilistic total secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$, then j also maintains total secrecy with respect to i in $\mathcal{R}$.*

PROOF. Suppose that j maintains probabilistic total secrecy with respect to i in $(\mathcal{R}, \mathcal{PR})$, and (r,m) and (r',m') are arbitrary points. Then (taking $(r'', m'') = (r', m')$ in the definition) we have $\mu_{r,m,i}(\mathcal{K}_j(r',m') \cap \mathcal{K}_i(r,m)) = \mu_{r',m',i}(\mathcal{K}_j(r',m') \cap \mathcal{K}_i(r',m'))$. But $(r',m') \in \mathcal{K}_j(r',m') \cap \mathcal{K}_i(r',m')$, so $\mu_{r',m',i}(\mathcal{K}_j(r',m') \cap \mathcal{K}_i(r',m')) \geq \mu_{r',m',i}(\{(r',m')\}) > 0$, by assumption. Thus, $\mu_{r,m,i}(\mathcal{K}_j(r',m') \cap \mathcal{K}_i(r,m)) > 0$, from which it follows that $\mathcal{K}_j(r',m') \cap \mathcal{K}_i(r,m) \neq \emptyset$. □

The following result is proved by Gill, van der Laan, and Robins [1997]; see also Grünwald and Halpern [2003, Theorem 3.1].

LEMMA C.1. *Suppose that μ is a probability on W, $X, Y \subseteq W$, $Y_1, Y_2, \ldots$ is a countable partition of $Y \subseteq W$, and $X, Y_1, Y_2, \ldots$ are all measurable. The following are equivalent:*

(a) $\mu(X \mid Y_i) = \mu(X \mid Y_j)$ for all Y_i, Y_j such that $\mu(Y_i) > 0$ and $\mu(Y_j) > 0$.
(b) $\mu(X \mid Y_i) = \mu(X \mid Y)$ for all Y_i such that $\mu(Y_i) > 0$, so that Y_i is conditionally independent of X given Y.

PROPOSITION 4.6. *If $(\mathcal{R}, \mathcal{PR})$ is a probability system (resp., synchronous probability system) that satisfies the common prior assumption with prior probability μ_{cp}, the following are equivalent:*

(a) Agent j maintains probabilistic total (resp., synchronous) secrecy with respect to i.
(b) Agent i maintains probabilistic total (resp., synchronous) secrecy with respect to j.
(c) For all points (r,m) and (r',m'), $\mu_{cp}(\mathcal{K}_j(r',m') \mid \mathcal{K}_i(r,m)) = \mu_{cp}(\mathcal{K}_j(r',m'))$ (resp., for all points (r,m) and (r',m), $\mu_{cp}(\mathcal{K}_j(r',m) \mid \mathcal{K}_i(r,m)) = \mu_{cp}(\mathcal{K}_j(r',m) \mid \mathcal{PT}(m))$, where $\mathcal{PT}(m)$ is the set of points occurring at time m; that is, the events $\mathcal{K}_i(r,m)$ and $\mathcal{K}_j(r',m)$ are conditionally independent with respect to μ_{cp}, given that the time is m).

PROOF. We prove the synchronous case here. The proof for total secrecy is almost identical and left to the reader. Recall that j maintains probabilistic synchronous secrecy with respect to i if, for all times m and all runs r, r', r'',

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

Because $(\mathcal{R}, \mathcal{PR})$ satisfies the common prior assumption with prior probability μ_{cp}, this requirement can be restated as

$$\mu_{cp}(\mathcal{K}_j(r'',m) \mid \mathcal{K}_i(r,m)) = \mu_{cp}(K_j(r'',m) \mid \mathcal{K}_i(r',m)).$$

By Lemma C.1, taking $Y = \mathcal{PT}(m)$ and the Y_i's to be the i-information sets at time m, it follows that j maintains probabilistic synchronous secrecy with respect to i iff $\mathcal{K}_j(r'', m)$ is conditionally independent of $\mathcal{K}_i(r, m)$ conditional on $\mathcal{PT}(m)$ for all runs r and r''. By the symmetry of conditional independence, it immediately follows that this is true iff i maintains probabilistic synchronous secrecy with respect to j. □

LEMMA C.2. *If $\mathcal{R}$ is a system where agent i has perfect recall and Ω is an arbitrary set of i-information sets, then there exists a set $\Omega' \subseteq \Omega$ such that $\{R(\mathcal{K}) \mid \mathcal{K} \in \Omega'\}$ is a partition of $\bigcup_{\mathcal{K}\in\Omega} \mathcal{R}(\mathcal{K})$.*

PROOF. Define a set $\mathcal{K} \in \Omega$ to be *dominated* by a set $\mathcal{K}' \in \Omega$ if $\mathcal{K} \neq \mathcal{K}'$ and there exists a run r and times $m' < m$ such that $(r, m) \in \mathcal{K}$ and $(r, m') \in \mathcal{K}'$. Let Ω' consist of the information sets in Ω that are not dominated by another set in Ω. Note that if $r \in \cup_{\mathcal{K}\in\Omega}\mathcal{R}(\mathcal{K})$, then $r \in \mathcal{R}(\mathcal{K}')$ for some $\mathcal{K}' \in \Omega'$. To see this, consider the set $\Omega(\mathcal{K})$ consisting of $\mathcal{K}$ and all information sets in Ω that dominate $\mathcal{K}$. By perfect recall, i's local state sequence at each information set in $\Omega(\mathcal{K})$ is a (not necessarily strict) prefix of i's local state sequence in $\mathcal{K}$. Let $\mathcal{K}'$ be the information set in $\Omega(\mathcal{K})$ where i's local state sequence is shortest. It follows that $\mathcal{K}'$ is not dominated by another information set in $\Omega(\mathcal{K})$. Furthermore, if there exists an information set $\mathcal{K}'' \in \Omega - \Omega(\mathcal{K})$ that dominates $\mathcal{K}'$, then $\mathcal{K}''$ would dominate $\mathcal{K}$ as well, contradicting the construction of $\Omega(K)$. Therefore, $\mathcal{K}' \in \Omega'$ and $r \in \mathcal{K}'$. Thus $\bigcup_{\mathcal{K}\in\Omega'} \mathcal{R}(\mathcal{K}) = \bigcup_{\mathcal{K}\in\Omega} \mathcal{R}(\mathcal{K})$. Moreover, if $\mathcal{K}$ and $\mathcal{K}'$ are different sets in Ω', then $\mathcal{R}(\mathcal{K})$ and $\mathcal{R}(\mathcal{K}')$ must be disjoint, for otherwise one of $\mathcal{K}$ or $\mathcal{K}'$ would dominate the other. □

PROPOSITION 4.7. *If $(\mathcal{R}, \mathcal{F}, \mu)$ is a run-based probability system that is either synchronous or one where agents i and j both have perfect recall, then the following are equivalent:*

(a) Agent j maintains run-based probabilistic secrecy with respect to i.

(b) Agent i maintains run-based probabilistic secrecy with respect to j.

(c) For all points (r, m) and (r', m'), $\mathcal{R}(\mathcal{K}_i(r, m))$ and $\mathcal{R}(\mathcal{K}_j(r', m'))$ are probabilistically independent with respect to μ.

PROOF. First, note that if $\mathcal{R}$ is synchronous or if i has perfect recall, then there exists a collection Ω of i-information sets such that the set $\{R(\mathcal{K}) \mid \mathcal{K} \in \Omega\}$ is a partition of $\mathcal{R}$. In the case of perfect recall, this follows by Lemma C.2 applied to the set of all information sets (whose union is clearly $\mathcal{R}$). With synchrony we can take Ω to consist of sets of the form $\mathcal{R}(\mathcal{K}_i(r, m))$, for some fixed time m.

Now, suppose j maintains run-based probabilistic secrecy with respect to i. By definition,

$$\mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \mid \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \mid \mathcal{R}(\mathcal{K}_i(r', m')))$$

for all points (r, m), (r', m'), and (r'', m''). In particular, for all $\mathcal{K}, \mathcal{K}' \in \Omega$, and all points (r', m'), $\mu(\mathcal{R}(\mathcal{K}_j(r', m') \mid \mathcal{R}(\mathcal{K}))) = \mu(\mathcal{R}(\mathcal{K}_j(r', m')) \mid \mathcal{R}(\mathcal{K}'))$. By Lemma C.1 it follows that $\mu(\mathcal{R}(\mathcal{K}_j(r', m'')) \mid \mathcal{R}(\mathcal{K})) = \mu(\mathcal{R}(\mathcal{K}_j(r', m'')))$ for all information sets $\mathcal{K} \in \Omega$. But then it follows by secrecy that $\mu(\mathcal{R}(\mathcal{K}_j(r', m')) \mid \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(\mathcal{R}(\mathcal{K}_j(r', m'))$ for all i-information sets $\mathcal{R}(\mathcal{K}_i(r, m))$. Therefore $\mathcal{R}(\mathcal{K}_j(r', m'))$

and $\mathcal{R}(\mathcal{K}_i(r,m))$ are independent for all information sets $\mathcal{K}_j(r',m')$ and $\mathcal{K}_i(r,m)$. Thus secrecy implies independence, and this holds if we reverse the roles of i and j.

It is also clear that independence implies secrecy. For suppose that (c) holds. Then, for all points (r,m), (r',m'), and (r'',m''), we have

$$\mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'',m''))) = \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r',m'))),$$

so that j maintains run-based probabilistic secrecy with respect to i. Similarly, i maintains secrecy with respect to j. □

PROPOSITION 4.8. *If $(\mathcal{R},\mathcal{PR})$ is the standard system determined by the synchronous run-based probability system $(\mathcal{R},\mathcal{F},\mu)$ and agents i and j have perfect recall in $\mathcal{R}$, then agent j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R},\mathcal{F},\mu)$ iff j maintains probabilistic synchronous secrecy with respect to i in $(\mathcal{R},\mathcal{PR})$.*

PROOF. Clearly if j maintains run-based probabilistic secrecy with respect to i in $(\mathcal{R},\mu)$ and $(\mathcal{R},\mathcal{PR})$ is the standard system determined by $(\mathcal{R},\mu)$ then, at all times m,

$$\begin{aligned}\mu_{r,m,i}(\mathcal{K}_j(r'',m) \cap \mathcal{K}_i(r,m)) &= \mu(\mathcal{K}_j(r'',m) \,|\, \mathcal{K}_i(r,m)) \\ &= \mu(\mathcal{K}_j(r'',m) \,|\, \mathcal{K}_i(r',m)) \\ &= \mu_{r',m,i}(\mathcal{K}_j(r'',m) \cap \mathcal{K}_i(r',m)),\end{aligned}$$

so j maintains probabilistic synchronous secrecy with respect to i in $(\mathcal{R},\mathcal{PR})$.

For the converse, suppose that j maintains probabilistic synchronous secrecy with respect to i in $(\mathcal{R},\mathcal{PR})$. We want to show that, for all points (r,m), (r',m'), and (r'',m''),

$$\mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r',m'))). \tag{1}$$

We first show that, for all runs r and r'' and times m and m'',

$$\mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m'))). \tag{2}$$

Since (2) also holds with r replaced by r', (1) easily follows from (2) and the assumption that j maintains probabilistic synchronous secrecy with respect to i.

To prove (2), we consider two cases: $m \le m''$ and $m'' < m$. If $m \le m''$ then, by synchrony and perfect recall, we can partition the runs in $\mathcal{R}(\mathcal{K}_i(r,m))$ according to i's local state at time m''. Let $\Omega = \{\mathcal{K}_i(r^*,m'') \,|\, r^* \in \mathcal{R}(\mathcal{K}_i(r,m))\}$. By perfect recall and synchrony, $\mathcal{R}(\mathcal{K}_i(r,m))$ is the disjoint union of the sets in Ω. Thus,

$$\begin{aligned}&\mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) \\ &= \textstyle\sum_{\mathcal{K}\in\Omega} \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \cap \mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) \\ &= \textstyle\sum_{\mathcal{K}\in\Omega} \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K})) \cdot \mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) \\ &= \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m''))) \cdot \textstyle\sum_{\mathcal{K}\in\Omega} \mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) \text{ [by synchronous secrecy]} \\ &= \mu(\mathcal{R}(\mathcal{K}_j(r'',m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r,m''))).\end{aligned}$$

The argument is similar if $m'' < m$. We now partition the runs in $\mathcal{R}(\mathcal{K}_i(r,m''))$ according to i's local state at time m and the runs in $\mathcal{R}(\mathcal{K}_j(r'',m''))$ according to j's local state at time m. Define

$$\Omega_i = \{\mathcal{K}_i(r^*,m) \,|\, r^* \in \mathcal{R}(\mathcal{K}_i(r,m''))\}.$$

and

$$\Omega_j = \{\mathcal{K}_j(r^*, m) \,|\, r^* \in \mathcal{R}(\mathcal{K}_j(r'', m''))\}.$$

We now have

$$\begin{aligned}
& \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m'''))) \\
= \; & \textstyle\sum_{\mathcal{K}_j \in \Omega_j} \mu(\mathcal{R}(\mathcal{K}_j) \,|\, \mathcal{R}(\mathcal{K}_i(r, m'''))) \\
= \; & \textstyle\sum_{\mathcal{K}_j \in \Omega_j} \sum_{\mathcal{K}_i \in \Omega_i} \mu(\mathcal{R}(\mathcal{K}_j) \,|\, R(\mathcal{K}_i)) \cdot \mu(\mathcal{R}(\mathcal{K}_i) \,|\, \mathcal{R}(\mathcal{K}_i(r, m'''))) \\
= \; & \textstyle\sum_{\mathcal{K}_j \in \Omega_j} \mu(\mathcal{R}(\mathcal{K}_j) \,|\, R(\mathcal{K}_i)) \cdot \sum_{\mathcal{K}_i \in \Omega_i} \mu(\mathcal{R}(\mathcal{K}_i) \,|\, \mathcal{R}(\mathcal{K}_i(r, m'''))) \\
= \; & \textstyle\sum_{\mathcal{K}_j \in \Omega_j} \mu(\mathcal{R}(\mathcal{K}_j) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) \\
= \; & \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))),
\end{aligned}$$

as needed. □

THEOREM 4.9. *(a) If $(\mathcal{R}, \mathcal{PR})$ is a probabilistic system, then agent j maintains probabilistic total secrecy with respect to agent i iff, for every interpretation π and formula φ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\varphi) = \sigma$.*

(b) If $(\mathcal{R}, \mathcal{PR})$ is a synchronous probabilistic system, then agent j maintains probabilistic synchronous secrecy with respect to agent i iff, for every interpretation π, time m, and formula φ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ_m such that $(\mathcal{I}, r, m) \models \Pr_i(\varphi) = \sigma_m$ for all runs $r \in \mathcal{R}$.

PROOF. We prove part (b) here. The proof of (a) is similar.

Suppose $\mathcal{R}$ is synchronous and that j maintains synchronous probabilistic secrecy with respect to i. Let π be an interpretation, m be an arbitrary time, and φ be a j-local formula in $\mathcal{I} = (\mathcal{R}, \mu, \pi)$. Because φ is j-local, by Proposition 3.11, there exists a set Ω of j-information sets such that $(\mathcal{I}, r, m) \models \varphi$ iff $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$. Let $\Psi = \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$. Let $S = \{r' \in \mathcal{R} \,|\, (r', m) \in \Psi\}$, and let $\Omega(m) = \{\mathcal{K} \in \Omega \,|\, (r', m) \in \mathcal{K} \text{ for some } r' \in \mathcal{R}\}$. Since j maintains synchronous probabilistic secrecy with respect to i, for every element $\mathcal{K} \in \Omega(m)$, there is a constant $\sigma(\mathcal{K}, m)$ such that, for all runs $r \in \mathcal{R}$, $\mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \sigma(\mathcal{K}, m)$. Let $\sigma_m = \sum_{\mathcal{K} \in \Omega(m)} \sigma(\mathcal{K}, m)$, and fix $r \in \mathcal{R}$. By synchrony, the set $\{\mathcal{R}(\mathcal{K}) \,|\, \mathcal{K} \in \Omega(m)\}$ partitions S, and

$$\mu(S \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \sum_{\mathcal{K} \in \Omega(m)} \mu(\mathcal{R}(\mathcal{K}) \,|\, \mathcal{R}(\mathcal{K}_i(r, m))) = \sigma_m.$$

Because $\Psi \cap \mathcal{K}_i(r, m) = \mathcal{K}_i(r, m)(S)$, we have $\mu_{r,m,i}(\Psi) = \mu(S \,|\, \mathcal{R}(\mathcal{K}_i(r, m)))$, and it follows that $(\mathcal{I}, r, m) \models \Pr_i(\varphi) = \sigma_m$, as desired.

For the converse, suppose that for every interpretation π and time m, if φ is j-local in $\mathcal{I} = (\mathcal{R}, \mu, \pi)$, then there exists a constant σ_m such that $(\mathcal{I}, r, m) \models \Pr_i(\varphi) = \sigma_m$ for all runs $r \in \mathcal{R}$. Fix a time m. Suppose that $r, r', r'' \in \mathcal{R}$ and that π is an interpretation such that $\pi(r^*, n)(p) = \mathbf{true}$ iff $(r^*, n) \in \mathcal{K}_j(r'', m)$. The proposition p is j-local, so there exists a constant σ_m such that $(\mathcal{I}, r, m) \models \Pr_i(p) = \sigma_m$ and $(\mathcal{I}, r', m) \models \Pr_i(p) = \sigma_m$. It follows that

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

as desired. □

THEOREM 4.10. *If $(\mathcal{R}, \mathcal{PR})$ is a standard probability system where agent j has perfect recall, then agent j maintains run-based probabilistic secrecy with respect to agent i iff, for every interpretation π and every formula φ that is j-local in $\mathcal{I} = (\mathcal{R}, \mathcal{PR}, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\diamondsuit \varphi) = \sigma$.*

PROOF. Suppose that j maintains probabilistic secrecy with respect to agent i in $(\mathcal{R}, \mu)$. Given an interpretation π and a formula φ that is j-local in $\mathcal{I} = (\mathcal{R}, \mu, \pi)$, by Proposition 3.11 there exists a set Ω of j-information sets such that $(\mathcal{I}, r, m) \models \varphi$ whenever $(r, m) \in \bigcup_{\mathcal{K} \in \Omega} \mathcal{K}$. Let $\Psi = \bigcup_{\mathcal{K} \in \Omega} \mathcal{R}(\mathcal{K})$. Note that $(\mathcal{I}, r, m) \models \diamondsuit \varphi$ iff $r \in \mathcal{R}(\bigcup_{\mathcal{K} \in \Omega} \mathcal{K}) = \Psi$. By Lemma C.2, there exists a set $\Omega' \subseteq \Omega$ such that $\{\mathcal{R}(\mathcal{K}) : \mathcal{K} \in \Omega\}$ is a partition of Ψ. By probabilistic secrecy, for each $\mathcal{K} \in \Omega'$, there exists a constant $\sigma_{\mathcal{K}}$ such that

$$\mu(\mathcal{R}(\mathcal{K}) \mid \mathcal{R}(\mathcal{K}_i(r, m))) = \sigma_{\mathcal{K}}$$

for all points (r, m). Let $\sigma = \sum_{\mathcal{K} \in \Omega'} \sigma_{\mathcal{K}}$. Because $\{\mathcal{R}(\mathcal{K}) \mid \mathcal{K} \in \Omega'\}$ is a partition of Ψ, for all points (r, m),

$$\mu(\Psi \mid \mathcal{R}(\mathcal{K}_i(r, m))) = \sum_{\mathcal{K} \in \Omega} \mu(\mathcal{R}(\mathcal{K}) \mid \mathcal{R}(\mathcal{K}_i(r, m))) = \sigma.$$

Because $\mu_{r,m,i}(\mathcal{K}_i(r, m)(\Psi)) = \mu(\Psi \mid \mathcal{R}(\mathcal{K}_i(r, m)))$, it follows that $\mathcal{I} \models \Pr_i(\diamondsuit \varphi) = \sigma$.

For the converse, suppose that for every interpretation π and formula φ that is j-local in $\mathcal{I} = (\mathcal{R}, \mu, \pi)$, there exists a constant σ such that $\mathcal{I} \models \Pr_i(\diamondsuit \varphi) = \sigma$. Given points (r, m), (r', m'), and (r'', m''), let π be an interpretation such that $\pi(r^*, n)(p) = \mathbf{true}$ iff $(r^*, n) \in \mathcal{K}_j(r'', m'')$. The proposition p is j-local, so $\mathcal{I} \models \Pr_i(\diamondsuit p) = \sigma$. It follows that

$$\mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \mid \mathcal{R}(\mathcal{K}_i(r, m))) = \mu_{r,m,i}(\mathcal{K}_i(r, m)(\mathcal{R}(\mathcal{K}_j(r'', m'')))) = \sigma,$$

and the same holds if we replace (r, m) with (r', m'), so

$$\mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \mid \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(\mathcal{R}(\mathcal{K}_j(r'', m'')) \mid \mathcal{R}(\mathcal{K}_i(r', m'))).$$

This gives us probabilistic secrecy. □

THEOREM 4.13. *Let $(\mathcal{R}, \mathcal{D}, \Delta)$ be the adversarial probability system determined by INIT and suppose that $\mathcal{R}$ is either synchronous or a system where i has perfect recall. Agent i obtains no evidence for the initial choice in $(\mathcal{R}, \mathcal{D}, \Delta)$ iff agent i^- maintains generalized run-based probabilistic f_{i^-}-secrecy with respect to i in $(\mathcal{R}, \mathcal{M}_i^{INIT}(\Delta))$.*

PROOF. For the forward direction, we want to show that i^- maintains generalized run-based probabilistic f_{i^-}-secrecy with respect to i in $(\mathcal{R}, \mathcal{M}_i^{INIT}(\Delta))$. Suppose that $\mu \in \mathcal{M}_i^{INIT}(\Delta)$. The information function f_{i^-} maps an i^--information set to the choices made by the agents other than i. Let an *i^--choice set* be a set of runs of the form $\cap_{j \neq i} D_{y_j}$. We must show that for arbitrary points (r, m) and (r', m') and i^--choice sets D_{i^-}, we have

$$\mu(D_{i^-} \mid \mathcal{R}(\mathcal{K}_i(r, m))) = \mu(D_{i^-} \mid \mathcal{R}(\mathcal{K}_i(r', m'))). \tag{3}$$

Since, by assumption, i's choice is encoded i's local state, there exists a unique y_i such that $\mathcal{R}(\mathcal{K}_i(r, m)) \subseteq D_{y_i}$. Since i obtains no evidence for the initial choice, we

have that for all i^--choice sets D_{i^-} and D'_{i^-},

$$\mu_{D_{y_i} \cap D_{i^-}}(\mathcal{R}(\mathcal{K}_i(r,m))) = \mu_{D_{y_i} \cap D'_{i^-}}(\mathcal{R}(\mathcal{K}_i(r,m))). \tag{4}$$

Thus, whenever $\mu(D_{y_i} \cap D_{i^-}) > 0$ and $\mu(D_{y_i} \cap D'_{i^-}) > 0$, we have

$$\begin{aligned} \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i} \cap D_{i^-}) &= \mu_{D_{y_i} \cap D_{i^-}}(\mathcal{R}(\mathcal{K}_i(r,m))) \\ &= \mu_{D_{y_i} \cap D'_{i^-}}(\mathcal{R}(\mathcal{K}_i(r,m))) \\ &= \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i} \cap D'_{i^-}). \end{aligned}$$

It now follows by Lemma C.1 that $\mathcal{R}(\mathcal{K}_i(r,m))$ is conditionally independent of every i^--choice set given D_{y_i}. (Though Lemma C.1 actually shows only that $\mathcal{R}(\mathcal{K}_i(r,m))$ is conditionally independent of every i^- choice set D_{i^-} such that $\mu(D_{i^-} \cap D_{y_i}) > 0$, conditional independence is immediate if $\mu(D_{i^-} \cap D_{y_i}) = 0$) Thus, for any i^--choice set D_{i^-}, we have

$$\mu(D_{i^-} \,|\, \mathcal{R}(\mathcal{K}_i(r,m))) = \mu(D_{i^-} \,|\, \mathcal{R}(\mathcal{K}_i(r,m)) \cap D_{y_i}) = \mu(D_{i^-} \,|\, D_{y_i}) = \mu(D_{i^-}),$$

where the last equality follows because we have assumed that i's choice is independent of the choices made by other agents. Similarly, $\mu(D_{i^-} \,|\, \mathcal{R}(\mathcal{K}_i(r',m'))) = \mu(D_{i^-})$, so (3) follows, and i^- does indeed maintain generalized run-based probabilistic f_{i^-}-secrecy with respect to i.

For the converse, suppose that i^- maintains generalized run-based probabilistic f_{i^-}-secrecy with respect to i. Thus, for all points (r,m), i^--choice sets D_{i^-}, and measures $\mu \in \mathcal{M}_i^{INIT}(\Delta)$, we have (3). Given two i^--choice sets D_{i^-} and D'_{i^-} and an i-information set $\mathcal{K}_i(r,m)$ such that $\mathcal{R}(\mathcal{K}_i(r,m)) \subseteq D_{y_i}$, we want to show (4). To do so we first show that there exists a measure $\mu \in \mathcal{M}_i^{INIT}(\Delta)$ that places positive probability on all the cells. (We will make use of this particular measure for the duration of the proof.) Our strategy is to take a countable linear combination of the cell-specific probability measures, such that the set of runs in each cell is assigned positive probability by μ. Let $y_{i1}, y_{i2}, \ldots$ be a countable enumeration of $INIT_i$, and let $D_1, D_2, \ldots$ be a countable enumeration of the possible i^--choice sets. Define the function μ such that for $U \in \mathcal{F}$,

$$\mu(U) = \sum_{j \geq 1, k \geq 1} \frac{\mu_{D_{y_{ij}} \cap D_k}(U \cap D_{y_{ij}} \cap D_k)}{2^{jk}}.$$

It is straightforward to check that $\mu \in \mathcal{M}_i^{INIT}(\Delta)$ and that it places a positive probability on all the cells in $\mathcal{D}$. Furthermore, we have $\mu_{D_{y_i} \cap D_{i^-}}(\mathcal{R}(\mathcal{K}_i(r,m))) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i} \cap D_{i^-})$, and the same holds if we replace D_{i^-} with D'_{i^-}.

Given an i-information set $\mathcal{K}_i(r,m)$, let y_i be the initial choice for i such that $\mathcal{R}(\mathcal{K}_i(r,m)) \subseteq D_{y_i}$. For all i^- choice sets D_{i^-}, we have

$$\mu_{D_{y_i} \cap D_{i^-}}(\mathcal{R}(\mathcal{K}_i(r,m))) = \mu(\mathcal{R}(\mathcal{K}_i(r,m) \,|\, D_{y_i} \cap D_{i^-}).$$

Thus, to prove (4), it suffices to show that

$$\mu(\mathcal{R}(\mathcal{K}_i(r,m) \,|\, D_{y_i} \cap D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m) \,|\, D_{y_i} \cap D'_{i^-}).$$

Standard probabilistic manipulations show that

$$\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i} \cap D_{i^-}) \cdot \mu(D_{y_i} \,|\, D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D_{y_i} \,|\, D_{i^-}); \tag{5}$$

a similar equation holds if we replace D_{i^-} by D'_{i^-}. Since either $\mathcal{R}$ is synchronous or i has perfect recall in $\mathcal{R}$, there exists a set Ω of i-information sets such that $\{\mathcal{R}(\mathcal{K}) : \mathcal{K} \in \Omega\}$ partitions $\mathcal{R}$. By Lemma C.1 and (3), it follows that i^--choice sets are independent of the i-information sets in Ω. Applying (3) again, it follows that i^--choice sets are independent of *all* i-information sets. Thus, $\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \cap D_{y_i} \,|\, D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)))$. Since D_{i^-} and D_{y_i} are independent by assumption, it follows that $\mu(D_{y_i} \,|\, D_{i^-}) = \mu(D_{y_i})$. Thus, (5) reduces to

$$\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i} \cap D_{i^-}) \cdot \mu(D_{y_i}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m))).$$

The same is true for D'_{i^-}, so because $\mu(D_{y_i}) > 0$ it follows that $\mu(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i} \cap D_{i^-}) = \mu(\mathcal{R}(\mathcal{K}_i(r,m)) \,|\, D_{y_i} \cap D'_{i^-})$. (4) is now immediate. □

D. PROOFS FOR SECTION 5

PROPOSITION 5.5. *A limit-closed synchronous trace system Σ satisfies separability (resp. generalized noninterference) iff H maintains synchronous secrecy (resp., synchronous f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

PROOF. We give the argument for separability here; the argument for generalized noninterference is similar and left to the reader. The forward direction follows from Proposition 5.3. For the converse, suppose that H maintains synchronous secrecy with respect to L in $\mathcal{R}(\Sigma)$. Given $\tau, \tau' \in \Sigma$, let τ'' be the trace such that $\tau''|_L = \tau|_L$ and $\tau''|_H = \tau'|_H$. We must show that $\tau'' \in \Sigma$. Since H maintains synchronous secrecy with respect to L in $\mathcal{R}(\Sigma)$, for all m, there exists a run $r^m \in \mathcal{R}(\Sigma)$ such that $r^m_L(m) = r^\tau_L(m)$ and $r^m_H(m) = r^{\tau'}_H(m)$. Thus, for all m, there exists a trace $\tau^m \in \Sigma$ such that $\tau^m_m|_L = \tau|_L$ and $\tau^m_m|_H = \tau'|_H$. It follows that $\tau''_m = \tau^m_m$ for all m. Since $\tau^m \in \Sigma$ for all m, it follows by limit closure that $\tau'' \in \Sigma$, as desired. □

PROPOSITION 5.8. *If Σ is an asynchronous trace system that satisfies asynchronous separability (resp. asynchronous generalized noninterference), then H maintains total secrecy (resp. total f_{hi}-secrecy) with respect to L in $\mathcal{R}(\Sigma)$.*

PROOF. Suppose that Σ satisfies asynchronous separability, and let (r,m) and (r',m') be arbitrary points. By the construction of $\mathcal{R}(\Sigma)$, there exist traces $\tau, \tau' \in T$ such that $r_L(m) = \tau|_L$ and $r_H(m) = \tau'|_H$. Let τ'' be an interleaving of $\tau|_L$ and $\tau'|_H$. Since Σ satisfies asynchronous separability, $\tau'' \in \Sigma$. Let T'' be a run-like set of traces that contains τ''. (Such a set must exist because Σ is closed under trace prefixes.) By definition, $r^{T''} \in \mathcal{R}(\Sigma)$. Taking m to be the length of τ'', it follows that $r''_L(m'') = r_L(m)$ and $r''_H(m'') = r'_H(m')$. Thus, H maintains total secrecy with respect to L.

The proof for asynchronous generalized noninterference (and total f_{hi}-secrecy) is analogous, and left to the reader. □

PROPOSITION 5.9. *If Σ is an asynchronous trace system that is closed under interleavings, then Σ satisfies asynchronous separability iff H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$.*

PROOF. We have already established the forward direction. For the converse, suppose that H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$, and that Σ is

closed under interleavings. Given $\tau, \tau' \in \Sigma$, there exist points (r, m) and (r', m') in $\mathcal{PT}(\mathcal{R}(\Sigma))$ such that $r_L(m) = \tau|_L$ and $r'_H(m') = \tau'|_H$. Since H maintains total secrecy with respect to L in $\mathcal{R}(\Sigma)$, there exists a point (r'', m'') such that $r''_L(m'') = r_L(m)$ and $r''_H(m'') = r'_H(m')$. By the construction of $\mathcal{R}(\Sigma)$, there exists a run-like set T of traces such that $r'' = r^T$. Taking τ'' to be the traces of length m'' in T, it follows that $\tau''|_L = \tau|_L$ and $\tau''|_H = \tau'|_H$. Because Σ is closed under interleavings, $\tau'' \in \Sigma$ as required. □

ACKNOWLEDGMENTS

We thank Riccardo Pucella, Andrei Sabelfeld, and Ron van der Meyden for useful discussions. We also thank the CSFW reviewers, who provided helpful feedback and criticism, and Niranjan Nagarajan, who suggested a probability measure that led to Example A.2.

REFERENCES

Chawla, S., Dwork, C., McSherry, F., Smith, A., and Wee, H. 2005. Towards privacy in public databases. To appear, *Theory of Cryptography*.

Chor, B., Goldreich, O., Kushilevitz, E., and Sudan, M. 1998. Private information retrieval. *Journal of the ACM 45,* 6, 965–982.

Clark, D., Hunt, S., and Malacaria, P. 2002. Quantitative analysis of the leakage of confidential data. *Electronic Notes in Theoretical Computer Science 59,* 3. (Proc. *Workshop on Quantitative Aspects of Programming Laguages (QAPL '01).*).

Di Pierro, A., Hankin, C., and Wiklicky, H. 2002. Approximate non-interference. In *Proc. 15th IEEE Computer Security Foundations Workshop*. 3–17.

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

Engelhardt, K., van der Meyden, R., and Moses, Y. 1998. Knowledge and the logic of local propositions. In *Theoretical Aspects of Rationality and Knowledge: Proc. Seventh Conference (TARK 1998)*. 29–41.

Evfimievski, A., Gehrke, J. E., and Srikant, R. 2003. Limiting privacy breaches in privacy preserving data mining. In *Proc. 22nd ACM Symposium on Principles of Database Systems*. 211–222.

Fagin, R., Halpern, J. Y., and Megiddo, N. 1990. A logic for reasoning about probabilities. *Information and Computation 87,* 1/2, 78–128.

Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. Y. 1995. *Reasoning About Knowledge*. MIT Press, Cambridge, Mass. A slightly revised paperback version was published in 2003.

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

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

Gill, R. D., van der Laan, M., and Robins, J. 1997. Coarsening at random: Characterisations, conjectures and counter-examples. In *Proc. First Seattle Conference on Biostatistics*. 255–294.

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

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

Grünwald, P. D. and Halpern, J. Y. 2003. Updating probabilities. *Journal of A.I. Research 19*, 243–278.

Halpern, J. Y. 2002. Characterizing the common prior assumption. *Journal of Economic Theory 106,* 2, 316–355.

Halpern, J. Y. 2003. *Reasoning About Uncertainty*. MIT Press, Cambridge, Mass.

Halpern, J. Y. and O'Neill, K. 2003. Anonymity and information hiding in multiagent systems. In *Proc. 16th IEEE Computer Security Foundations Workshop*. 75–88.

Halpern, J. Y. and O'Neill, K. 2005. Secrecy in multiagent systems. Available at http://www.cs.cornell.edu/people/oneill.

Halpern, J. Y. and Pucella, R. 2003a. Modeling adversaries in a logic for security protocol analysis. In *Proc. Formal Aspects of Security (FASec 2002)*. Lecture Notes in Computer Science, Volume 2629. Springer-Verlag, Berlin/Heidelberg/New York, 115–132.

Halpern, J. Y. and Pucella, R. 2003b. Probabilistic algorithmic knowledge. In *Theoretical Aspects of Rationality and Knowledge: Proc. Ninth Conference (TARK 2003)*. 118–130.

Halpern, J. Y. and Tuttle, M. R. 1993. Knowledge, probability, and adversaries. *Journal of the ACM 40,* 4, 917–962.

Kyburg, H. 1983. Recent work in inductive logic. In *Recent Work in Philosophy*, T. Machan and K. Lucey, Eds. Rowman & Allanheld, Totowa, NJ, 87–150.

Lowe, G. 2002. Quantifying information flow. In *Proc. 15th IEEE Computer Security Foundations Workshop*. 18–31.

Mantel, H. 2000. Possibilistic definitions of security—an assembly kit. In *Proc. IEEE Computer Security Foundations Workshop*. 185–199.

Mantel, H. 2003. A uniform framework for the formal specification and verification of information flow security. Ph.D. thesis, Universität des Saarlandes.

McCullough, D. 1987. Specifications for multi-level security and a hook-up property. In *Proc. IEEE Symposium on Security and Privacy*. 161–166.

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

Mitchell, J. C., Ramanathan, A., Scedrov, A., and Teague, V. 2004. A probabilistic polynomial-time calculus for the analysis of cryptographic protocols. Submitted for publication.

Morris, S. 1995. The common prior assumption in economic theory. *Economics and Philosophy 11*, 227–253.

Myers, A. C., Sabelfeld, A., and Zdancewic, S. 2004. Enforcing robust declassification. In *Proc. 17th IEEE Computer Security Foundations Workshop*. 172–186.

O'Neill, K., Clarkson, M., and Chong, S. 2005. Information-flow security for interactive programs. Unpublished manuscript.

Rabin, M. O. 1982. n-process mutual exclusion with bounded waiting by $4 \cdot \log_2 N$-valued shared variable. *Journal of Computer and System Sciences 25,* 1, 66–75.

Ryan, P. Y. A. and Schneider, S. A. 1999. Process algebra and non-interference. In *Proc. 12th Computer Security Foundations Workshop*. 214–227.

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

Sabelfeld, A. and Myers, A. C. 2003. Language-based information-flow security. *IEEE J. Selected Areas in Communications 21,* 1, 5–19.

Schneider, S. and Sidiropoulos, A. 1996. CSP and anonymity. In *European Symposium on Research in Computer Security*. 198–218.

Shannon, C. E. 1949. Communication theory of secrecy systems. *Bell System Technical Journal 28-4*, 656–715.

Sutherland, D. 1986. A model of information. In *Proc. 9th National Security Conference*. 175–183.

van der Meyden, R. 1998. Common knowledge and update in finite environments. *Information and Computation 140,* 2, 115–157.

Vardi, M. Y. 1985. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symposium on Foundations of Computer Science*. 327–338.

Wittbold, J. T. and Johnson, D. M. 1990. Information flow in nondeterministic systems. In *Proc. IEEE Symposium on Research in Security and Privacy*. 144–161.

Zakinthinos, A. and Lee, E. S. 1997. A general theory of security properties. In *Proc. IEEE Symposium on Security and Privacy*. 94–102.

Received Month Year; revised Month Year; accepted Month Year