The Complexity of Intransitive Noninterference

Sebastian Eggert*, Ron van der Meyden†, Henning Schnoor*, Thomas Wilke*
*Institut für Informatik, Kiel University
†School of Computer Science and Engineering, University of New South Wales
Email: {eggert,schnoor,wilke}@ti.informatik.uni-kiel.de, meyden@cse.unsw.edu.au

***Abstract*—The paper considers several definitions of information flow security for intransitive policies from the point of view of the complexity of verifying whether a finite-state system is secure. The results are as follows. Checking (i) P-security (Goguen and Meseguer), (ii) IP-security (Haigh and Young), and (iii) TA-security (van der Meyden) are all in PTIME, while checking TO-security (van der Meyden) is undecidable. The most important ingredients in the proofs of the PTIME upper bounds are new characterizations of the respective security notions, which also enable the algorithms to return simple counter-examples demonstrating insecurity. Our results for IP-security improve a previous doubly exponential bound of Hadj-Alouane et al.**

***Keywords*-noninterference, information flow, verification**

I. Introduction

One of the fundamental methods in the construction of secure systems to high levels of assurance is to decompose the system into trusted and untrusted components, arranged in an architecture that constrains the possible causal effects and flows of information between these components. On the other hand, resource limitations and cost constraints may make it desirable for trusted and untrusted components to share resources. For example, it is cheaper for an intelligence analyst to handle high security and low security information on a single desktop machine than to use two physically separated machines. This leads to complex systems designs and implementations, in which the desired constraints on flows of information between trusted and untrusted components need to be enforced in spite of the fact that these components share resources. In order to provide high levels of assurance of implementations of this kind, it is desirable to have a formal theory of systems architecture and information flow, so that a design or implementation may be formally verified to conform to an information flow policy. Moreover, one would like, whenever possible, to automate the verification that a system satisfies such a formally defined policy. This motivates the problems we consider in this paper. We study the complexity of verification of a range of formally defined security policies that specify how a system is architecturally structured in terms of how information may flow between its components.

Attack model: The problems we consider in this paper address information flow and systems implementation attacks. We work in the paradigm of information flow security, where it is assumed that a (passive) adversary may attack the system by attempting to make subtle deductions from her possible observations of the system, exploiting covert channels that may exist in the system, in order to learn secrets that she is not authorized to possess. The automated analyses we consider aim to provide assurance that the system has been designed in such a way that such attacks are not possible, or to discover such attacks when they exist. The analysis can be applied both in circumstances where it is feared that a rogue systems developer may have deliberately constructed the system so as to contain such prohibited flows of information, as well as to ensure that such flows of information have not been inadvertently allowed to exist.

Policy model: Notions of *noninterference*—a first definition was given by Goguen and Meseguer [1]—are one approach to the formalisation of information flow and causal relationships. Noninterference was first proposed in the context of transitive information flow policies (with transitivity following from the partial order on security domains) but it was subsequently noted [2] that systems architectures often require intransitive policies. For example, a common architectural pattern is to restrict information flow

1081-6011/11 $26.00 © 2011 IEEE
DOI 10.1109/SP.2011.30

from a high-level domain to a low-level domain so as to be possible only via a trusted downgrader (e. g., a declassification guard or encryption device). This pattern motivates an intransitive information flow policy, stating that information flow is permitted from the high-level domain to the downgrader and from the downgrader to the low-level domain, but not directly from the high-level domain to the low-level domain.

Goguen and Meseguer's definition of noninterference, based on a "purge" function, does not yield the desired conclusions for intransitive policies. Haigh and Young proposed a variant for intransitive policies based on an "intransitive purge" function. Rushby [3] later refined their theory and developed connections to access control systems. Van der Meyden [4] has argued that the definitions of security for intransitive policies in these works suffer from some subtle flaws, and proposed some improved definitions, TA-security and TO-security, that first build an operational (full information protocol) model of the maximal permitted information flow in the system, and then compares the actual information flow to this maximal permitted information flow. The revised definitions can be shown to avoid the subtle flaws in the intransitive purge-based definition, and lead to a more satisfactory proof theory and connection to access control systems than in Rushby's work (e.g., yielding both soundness and completeness results, whereas Rushby proved only soundness.)

Verification: The goal of high assurance systems development by formal verification motivates the investigation of techniques whereby a systems design or implementation can be formally shown to satisfy a formal definition of security. The technique of unwinding relations [5], [3] provides a proof method that has been applied to establish that a system satisfies noninterference properties, but it requires significant human ingenuity to define an unwinding relation that forms the basis for the proof, and typically also has involved manual driving (proof rule selection) of the theorem proving tool within which the proof is conducted.

A better alternative, more acceptable to engineers when it can be applied, is for the property to be verified by fully automatic techniques. There is a substantial body of work on automated verification techniques for transitive noninterference properties (which we discuss in Section V), but there has been significantly less work on automated verification techniques for intransitive noninterference properties.

Contributions: Our contribution in this paper is to provide a basis for automated verification of definitions of intransitive noninterference, by developing a characterization of the computational complexity of deciding whether a given finite-state system is secure with respect to an intransitive information flow policy according to this definition. In particular, we consider Goguen and Meseguer's purge-based definition, Rushby's formulation of Haigh and Young's definition, and van der Meyden's definitions of TA-security and TO-security. We show that the last of these definitions is undecidable, but the others are decidable in polynomial time and even in nondeterministic logarithmic space. We give algorithms for the decidable cases and analyse their complexity.

The structure of the paper is as follows. In Section II we define the formal systems model that we work with, and recall the formal definitions of security for intransitive information flow policies that we study. New characterizations of Haigh and Young's definition and van der Meyden's notion of TA-security are presented in Section III. Section IV gives the complexity results for the four notions that we consider. Our results are positioned within the literature in Section V, where we discuss related work. Section VI concludes with a discussion of open problems and future research directions.

II. Basic Definitions and Notation

In this section, we introduce intransitive information flow policies and describe their motivation. We present a deterministic asynchronous systems model in which such policies may be interpreted, and then recall a number of different semantic interpretations of such policies in this system model that have been proposed in the literature.

A. Noninterference Policies

Noninterference policies are reflexive relations $\rightarrowtail \subseteq D \times D$, where D is a set of "domains". The intuitive reading of $u \rightarrowtail v$ is that "actions of domain u are permitted to interfere with domain v",

or "information is permitted to flow from domain u to domain v". For any set $U \subseteq D$ the image of U, denoted $U^{\rightarrowtail}$, is defined by $U^{\rightarrowtail} = \{v \in D | \exists u \in U : u \rightarrowtail v\}$. For a singleton set $\{u\}$ we also write $u^{\rightarrowtail}$ instead of $\{u\}^{\rightarrowtail}$.

The reason for the assumption of reflexivity is that, intuitively, a domain should be allowed to interfere with or have information about itself, since this cannot usually be prevented. In early work on noninterference [1], the relation $\rightarrowtail$ is also assumed to be transitive. This follows from the interpretation of domains as corresponding to security levels associated to classes of information and access rights, which have generally been taken to be partially ordered [6]. (In the classical multi-level security models, this partial order is derived from a linear order on security levels and the set containment order on sets of labels.)

One of the motivations for the consideration of policies $\rightarrowtail$ that are not transitive is that classical multilevel security policies are too restrictive for practical purposes, allowing flow of information from lower security levels to higher security levels, but prohibiting flow in the opposite direction. Such flows may be less frequent but are nevertheless required, e.g., for distribution of battle plans, in response to freedom of information requests, or for transmission of encrypted content across an insecure network. One of the ways this has been handled is to allow the general policy to be violated by a special downgrader component. A typical downgrader policy is depicted in Figure 1. Here the usual (transitive) multi-level policy for domains Public, Secret and Top-Secret is extended by the addition of two domains DownS and DownP, that are responsible for downgrading of information from Top-Secret to Secret, and from Secret to Public, respectively. These domains are *trusted* to enforce whatever policy constraints apply to the downgrading of information. Note that it would not be appropriate to apply an assumption of transitivity on this setting, since then, e.g., the edges involving DownS would imply that Top-Secret $\rightarrowtail$ Secret, i.e., a direct flow of information from Top-Secret to Secret is permitted.

Subsequent work on intransitive noninterference has taken a somewhat extended interpretation of the term "domain," treating this more as akin to "component" in a systems architecture. Figure 2

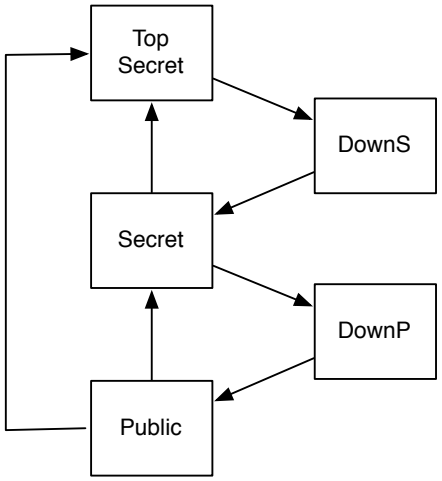

Fig. 1. Downgrader Policy

shows a systems architecture, discussed in [3] and [7], for a system in which messages are sent from a high security (Red) domain through a low security (Black) domain, with the global security policy stating that all content, except the message header, must be encrypted, and uncontrolled flow of information from Red to Black is prohibited. The architecture proposes to achieve this goal by having the Bypass component check a (more detailed) policy on the allowed header structure and content, and by having the Crypto component enforce a local policy stating that all output must be encrypted. These flows are recomposed into the encrypted message (with header) at the Black component. Crypto and Bypass are assumed to be trusted components of low enough complexity that they can be verified to enforce their local policies. Red (which may contain Trojans) and Black (which is at a low security level) are not assumed to be trusted. The argument for security of the system is intended to follow from the structure of the information flows in the architecture, plus the assumption that the trusted components correctly implement their local policies.

MILS security, as expounded in [7], proposes to base development of certifiably secure systems on design level arguments of this type, together with implementations in which mechanisms such as separation kernels or periods processing are used to enforce the systems architecture. We refer to [7] for a more detailed discussion of MILS security and the

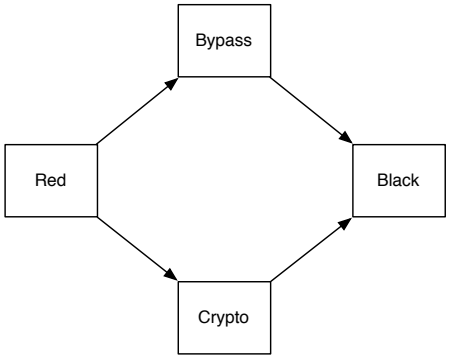

Fig. 2. Policy for Encrypted Message Transmission

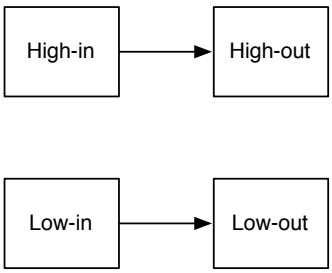

Fig. 3. A MILS Design Level Policy

proposed structure of the argument for security of the system in Figure 2.

We note that intransitive information flow policies are intended to express just the architectural structure of information flow, rather than encompass all the details of security policy. One key point is that implementations may involve resource sharing, which may mean that it is not immediately apparent that the design level architecture is enforced in the implementation.

For example, Figure 3 illustrates a design level policy for a system with multiple independent security levels that could be implemented, as shown in Figure 4, by a trusted multiplexer component that handles information from multiple security levels. One of the issues in the verification of such systems is to determine whether such a resource sharing implementation correctly enforces the design level architecture. The definitions in the following sections provide a number of distinct semantic interpretations of information flow policies that have been proposed to formalize what it means to implement

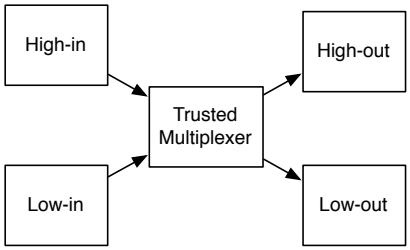

Fig. 4. A MILS Resource Sharing Implementation

the notion of correct enforcement.

B. State-Observed Machine Model

Several different types of semantic models have been used in the literature on noninterference. (See [8] for a comparison and a discussion of their relationships.) We work here with the state-observed machine model used by Rushby [3], but similar results would be obtained for other models.

This model consists of deterministic machines of the form $\langle S, s_0, A, \texttt{step}, \texttt{obs}, \texttt{dom}\rangle$, where S is a set of states, $s_0 \in S$ is the *initial state*, A is a set of actions, $\texttt{dom}\colon A \to D$ associates each action with an element of the set D of security domains, $\texttt{step}\colon S \times A \to S$ is a deterministic transition function, and $\texttt{obs}\colon S \times D \to O$ maps states to an observation in some set O, for each security domain. We may also refer to security domains more succinctly as "agents". We write $s \cdot \alpha$ for the state reached by performing the sequence of actions $\alpha \in A^*$ from state s, defined inductively by $s \cdot \epsilon = s$, and $s \cdot \alpha a = \texttt{step}(s \cdot \alpha, a)$ for $\alpha \in A^*$ and $a \in A$. Here, ϵ denotes the empty sequence. For any string α, we say a symbol a occurs in α if $\alpha = \beta a \beta'$ for some strings β, β'. We define $\text{alph}(\alpha)$ as the set of all symbols occuring in α.

C. The Purge Function

Noninterference is given a formal semantics in the transitive case [1] using a definition based on a "purge" function. Given a set $E \subseteq D$ of domains and a sequence $\alpha \in A^*$, we write $\alpha \upharpoonright E$ for the subsequence of all actions a in α with $\texttt{dom}(a) \in E$. Given a policy $\rightarrowtail$, we define the function $\texttt{purge}\colon A^* \times D \to A^*$ by

$$\texttt{purge}(\alpha, u) = \alpha \upharpoonright \{v \in D \mid v \rightarrowtail u\}.$$

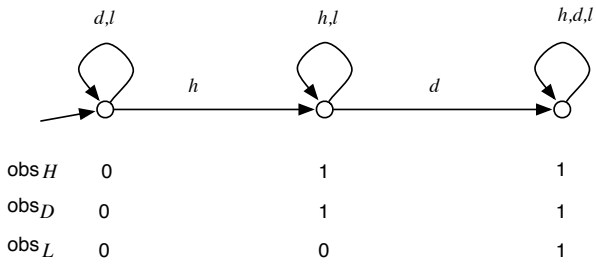

Fig. 5. A system that is TO-secure but not P-secure

(For clarity, we may use subscripting of agent arguments of functions, writing, e. g., $\mathtt{purge}(\alpha, u)$ as $\mathtt{purge}_u(\alpha)$.) The system M is said to be *secure with respect to the transitive policy* $\rightarrowtail$, when, for all $\alpha \in A^*$ and domains $u \in D$, we have $\mathtt{obs}_u(s_0 \cdot \alpha) = \mathtt{obs}_u(s_0 \cdot \mathtt{purge}_u(\alpha))$. That is, each agent's observations are as if only interfering actions had been performed. An equivalent formulation (which we state more generally for policies that are not necessarily transitive, in anticipation of later discussion) is the following:

Definition 1 (P-security): A system M is *P-secure* with respect to a policy $\rightarrowtail$ if for all sequences $\alpha, \alpha' \in A^*$ such that $\mathtt{purge}_u(\alpha) = \mathtt{purge}_u(\alpha')$, we have $\mathtt{obs}_u(s_0 \cdot \alpha) = \mathtt{obs}_u(s_0 \cdot \alpha')$.

This can be understood as saying that agent u's observation depends only on the sequence of interfering actions that have been performed.

D. The Intransitive Purge Function

While P-security is a reasonable definition of security for transitive information flow policies, it works less well for intransitive policies. Figure 5 illustrates a system that is, intuitively, secure for the downgrader policy $H \rightarrowtail D \rightarrowtail L$, but which does not satisfy P-security. Here h, d, l are actions of domains H, D, L, respectively, and the observations in each domain are depicted below the states. Intuitively, the observations convey a single bit of information: "has H ever performed the action h?". Domains H and D learn that H has performed h as soon as this action is performed (by their observations turning to value 1), but L does not learn this until after D subsequently performs the downgrading action d. Since the policy permits D to transmit information about H, the system is secure. However, this system does not satisfy P-security, since we have $\mathtt{purge}_L(hdl) = dl = \mathtt{purge}_L(dl)$ but $\mathtt{obs}_L(s_0 \cdot hdl) = 1 \neq 0 = \mathtt{obs}_L(s_0 \cdot \mathtt{purge}_L(dl))$. Intuitively, P-security says that L observations depend only on what D and L actions have been performed, so cannot contain information about H, even though the policy, intuitively, permits D to transmit such information.

To address this deficiency, Haigh and Young [2] generalized the definition of the purge function to intransitive policies. Intuitively, the intransitive purge of a sequence of actions with respect to a domain u is the largest subsequence of actions that could form part of a causal chain of effects (permitted by the policy) ending with an effect on domain u. More formally (we follow the presentation from [3]), the definition makes use of a function $\mathtt{sources}\colon A^* \times D \to \mathcal{P}(D)$ defined inductively by $\mathtt{sources}(\epsilon, u) = \{u\}$ and, for $a \in A$ and $\alpha \in A^*$, if there exists $v \in \mathtt{sources}(\alpha, u)$ with $\mathtt{dom}(a) \rightarrowtail v$, then

$$\mathtt{sources}(a\alpha, u) = \mathtt{sources}(\alpha, u) \cup \{\mathtt{dom}(a)\} \ ,$$

and else

$$\mathtt{sources}(a\alpha, u) = \mathtt{sources}(\alpha, u) \ .$$

Intuitively, $\mathtt{sources}(\alpha, u)$ is the set of domains v such that there exists a sequence of permitted interferences from v to u within α. The *intransitive purge* function $\mathtt{ipurge}\colon A^* \times D \to A^*$ is then defined inductively by $\mathtt{ipurge}(\epsilon, u) = \epsilon$ and, for $a \in A$ and $\alpha \in A^*$, if $\mathtt{dom}(a) \in \mathtt{sources}(a\alpha, u)$, then

$$\mathtt{ipurge}(a\alpha, u) = a\,\mathtt{ipurge}(\alpha, u) \ ,$$

and else

$$\mathtt{ipurge}(a\alpha, u) = \mathtt{ipurge}(\alpha, u) \ .$$

The intransitive purge function is then used in place of the purge function in Haigh and Young's definition:

Definition 2 (IP-security): A system M is IP-secure with respect to a (possibly intransitive) policy $\rightarrowtail$ if for all sequences $\alpha \in A^*$, and $u \in D$, we have $\mathtt{obs}_u(s_0 \cdot \alpha) = \mathtt{obs}_u(s_0 \cdot \mathtt{ipurge}_u(\alpha))$.

Since the function $\mathtt{ipurge}_u$ on A^* is idempotent, this definition, like the definition for the transitive

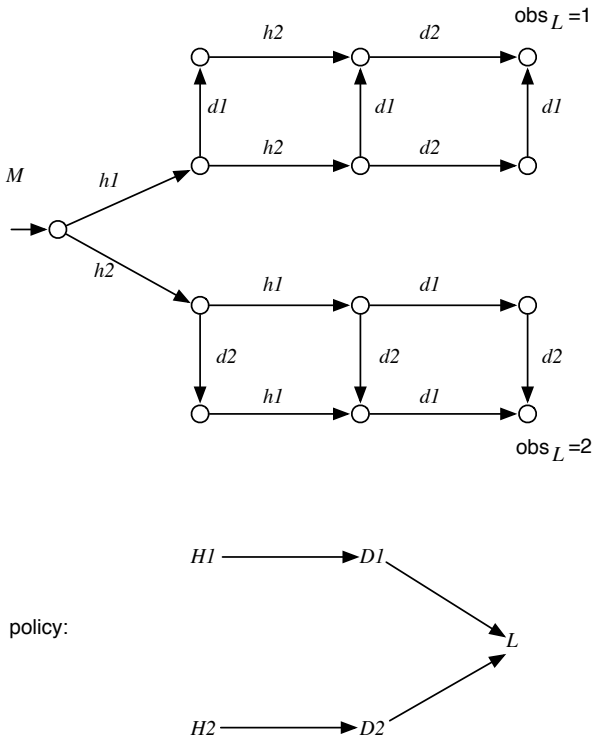

Fig. 6. A system that is IP-secure but not TA-secure

case, can be formulated as: M is IP-secure with respect to a policy $\rightarrowtail$ if for all $u \in D$ and all sequences $\alpha, \alpha' \in A^*$ with $\texttt{ipurge}_u(\alpha) = \texttt{ipurge}_u(\alpha')$, we have $\texttt{obs}_u(s_0{\cdot}\alpha) = \texttt{obs}_u(s_0{\cdot}\alpha')$. It can be seen that $\texttt{ipurge}_u(\alpha) = \texttt{purge}_u(\alpha)$ when $\rightarrowtail$ is transitive, so IP-security is in fact a generalisation of the definition of security for transitive policies.

E. The `ta` *Function*

It has been noted by van der Meyden [4] that IP-security classifies some systems as secure where there is, intuitively, an insecure flow of information that relates to a domain learning ordering information about the actions of other domains that it should not have.

Figure 6 depicts part of a system M and a policy $\rightarrowtail$ such that M is IP-secure, but for which the conclusion that the system is secure is questionable. We sketch the argument for this here, and refer the reader to [4] for a more rigorous presentation. Intuitively, the system is comprised of two High security level domains $H1, H2$, each with a downgrader ($D1, D2$, respectively) to the Low security domains L. The actions $h1, h2, d1, d2$ are associated to the domains $H1, H2, D1, D2$, respectively, and state transitions are depicted only when there is a change of state. The observations of L are depicted at two of the states; at all other states we assume that L makes observation 0. All other agents may be assumed to make observation 0 at all states. Intutively, at the state where L observes 1, it is possible for L to deduce that there has been an occurrence of $h1$ followed by an occurrence of $h2$; the state where L observes 2, it is possible for L to deduce that these actions have occurred in the opposite order.

We show that this system is IP-secure: Suppose we have $\texttt{ipurge}_L(\alpha) = \texttt{ipurge}_L(\beta)$, and one of $\texttt{obs}_L(s_0 \cdot \alpha)$ or $\texttt{obs}_L(s_0 \cdot \beta)$ is 1 or 2, say the former is equal to 1. Then this sequence must contain an occurrence of $h1$ before an occurrence of $h2$, and each is followed by d_1 and d_2, respectively. This observation shows, in fact, that L *knows* the order of the first h_1 and h_2 actions in the sequence α. Because $\texttt{ipurge}_L$ preserves $h1$ when it is followed by $d1$, and similarly for $h2$ and $d2$, and also preserves the order of actions that it retains, the same statement must hold for β, and it then follows that also $\texttt{obs}_L(s_0 \cdot \beta) = 1 = \texttt{obs}_L(s_0 \cdot \alpha)$. If neither observation is in $1, 2$, then both are equal to 0, and again we have the required equality of observations.

On the other hand, the conclusion that the system is secure is somewhat peculiar. Each of the downgraders is individually permitted by the policy to know only about activity in its associated High level domain, and its own activity. Thus, individually, neither $D1$ nor $D2$ can know the order of the first two $H1$ and $H2$ actions. Moreover, since the system is asynchronous, even if we were to combine all the information that the downgraders are permitted to know, we would still not be able to deduce the order on the $H1, H2$ actions. We therefore have the peculiar conclusion that the system is classified by IP-security to be secure, but it allows L to learn information that would not be permitted to be known to the two domains $D1, D2$, which are supposed to filter all flow of information from $H1, H2$, even if these domains were to combine their information.

To address this peculiarity, van der Meyden has proposed some other interpretations of intransitive policies. Both proceed by first defining a concrete

operational model of the maximal amount of information that an agent is permitted to have after some sequence of actions has been performed. Security of the system is then defined by requiring that an agent's observation may not contain more than this maximal amount of information.

In the first operational model, when an agent performs an action, it transmits what it is permitted to know to other agents, subject to constraints in the policy. The following definition expresses this in a weaker way than the ipurge function.

Given sets X and A, let the set $\mathcal{T}(X, A)$ be the smallest set containing X and such that if $x, y \in \mathcal{T}$ and $z \in A$ then $(x, y, z) \in \mathcal{T}$. Intuitively, the elements of $\mathcal{T}(X, A)$ are binary trees with leaves labelled from X and interior nodes labelled from A.

Given a policy $\rightarrowtail$, define, for each agent $u \in D$, the function $\mathtt{ta}_u \colon A^* \to \mathcal{T}(\{\epsilon\}, A)$ inductively by $\mathtt{ta}_u(\epsilon) = \epsilon$, and, for $\alpha \in A^*$ and $a \in A$,

$$\mathtt{ta}_u(\alpha a) = \begin{cases} \mathtt{ta}_u(\alpha) & \text{if } \mathtt{dom}(a) \not\rightarrowtail u, \\ (\mathtt{ta}_u(\alpha), \mathtt{ta}_{\mathtt{dom}(a)}(\alpha), a) & \text{otherwise.} \end{cases}$$

Intuitively, $\mathtt{ta}_u(\alpha)$ captures the maximal information that agent u may, consistently with the policy $\rightarrowtail$, have about the past actions of other agents. Initially, an agent has no information about what actions have been performed. The recursive clause describes how the maximal information $\mathtt{ta}_u(\alpha)$ permitted to flow to u after the performance of α changes when the next action a is performed. If a may not interfere with u, then there is no change, otherwise, u's maximal permitted information is increased by adding the maximal information permitted to $\mathtt{dom}(a)$ at the time a is performed (represented by $\mathtt{ta}_{\mathtt{dom}(a)}(\alpha)$), as well the fact that a has been performed. Thus, this definition captures the intuition that an agent may only transmit information that it is permitted to have, and then only to agents with which it is permitted to interfere.

Definition 3 (TA-security): A system M is TA-secure with respect to a policy $\rightarrowtail$ if for all agents u and all $\alpha, \alpha' \in A^*$ such that $\mathtt{ta}_u(\alpha) = \mathtt{ta}_u(\alpha')$, we have $\mathtt{obs}_u(s_0 \cdot \alpha) = \mathtt{obs}_u(s_0 \cdot \alpha')$.

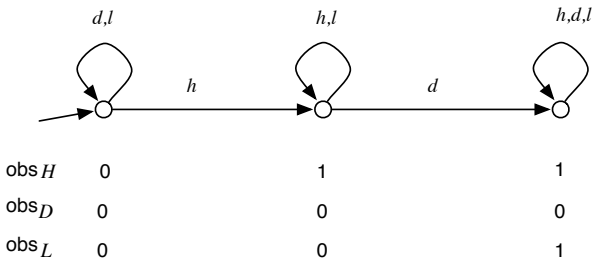

Fig. 7. A system that is TA-secure but not TO-secure

Intuitively, this says that each agent's observations provide the agent with no more than the maximal amount of information that may have been transmitted to it, as expressed by the functions ta.

F. The to Function

In the definition of TA-security, the operational model of information flow given by the function ta permits a domain to transmit information that it *may* have, even if it has never observed anything from which it could deduce that information. Arguably, this is too liberal.

Figure 7 shows a system for the downgrader policy $H \rightarrowtail D \rightarrowtail L$, similar to that in Figure 5. It can be argued that the system is TA-secure; we leave the details to the reader. Again, when L observes 1, it can deduce that H has performed the action h, and indeed, this observation can only occur after D has performed the action d, thereby downgrading the information about H. On the other hand, note that in this system, D's observation is always 0, so D cannot know, on the basis of its observations, whether H has performed h. D is therefore transmitting to L information that it does not itself have.

Van der Meyden [4] therefore also considers a variant operational model in which a domain transmits only what it has actually observed. This yields the alternate notion of TO-security.

The sequence of all observations and actions of a domain is denoted as its view. Formally, the notion of *view* is defined as follows. The definition uses an absorptive concatenation function $\circ$, defined over a set X by $s \circ x = s$ if x is equal to the final element of s (if any), and $s \circ x = s \cdot x$ (ordinary concatenation) otherwise, for every $s \in X^*$ and $x \in X$. Define

the view of domain u with respect to a sequence $\alpha \in A^*$ using the function $\mathtt{view}_u \colon A^* \to (A \cup O)^*$ (where O is the set of observations in the system) defined by

$$\mathtt{view}_u(\epsilon) = \mathtt{obs}_u(s_0), \text{ and}$$
$$\mathtt{view}_u(\alpha a) = (\mathtt{view}_u(\alpha) \cdot b) \circ \mathtt{obs}_u(s_0 \cdot \alpha) \ ,$$

where $b = a$ if $\mathtt{dom}(a) = u$ and $b = \epsilon$ otherwise. That is, $\mathtt{view}_u(\alpha)$ is the sequence of all observations and actions of domain u in the run generated by α, compressed by the elimination of stuttering observations. Intuitively, $\mathtt{view}_u(\alpha)$ is the complete record of information available to agent u in the run generated by the sequence of actions α. The reason we apply the absorptive concatenation is to capture that the system is asynchronous, with agents not having access to a global clock. The effect of this operation is to reduce any stuttering of an observation in the run to a single copy. Thus, two sequences that only differ from each other in repetitions of a single observation are not distinguishable by the agent.

Given a policy $\rightarrowtail$, for each domain $u \in D$, define the function $\mathtt{to}_u \colon A^* \to \mathcal{T}((A \cup O)^*, A)$ by $\mathtt{to}_u(\epsilon) = \mathtt{obs}_u(s_0)$ and

$$\mathtt{to}_u(\alpha a) = \begin{cases} \mathtt{to}_u(\alpha) & \text{if } \mathtt{dom}(a) \not\rightarrowtail u, \\ (\mathtt{to}_u(\alpha), \mathtt{view}_{\mathtt{dom}(a)}(\alpha), a) & \text{otherwise.} \end{cases}$$

Intuitively, this definition takes the model of the maximal information that an action a may transmit after the sequence α to be the fact that a has occurred, together with the information that $\mathtt{dom}(a)$ *actually* has, as represented by its view $\mathtt{view}_{\mathtt{dom}(a)}(\alpha)$. By contrast, TA-security uses in place of this the maximal information that $\mathtt{dom}(a)$ *may* have. We may now base the definition of security on the function $\mathtt{to}$ rather than $\mathtt{ta}$.

Definition 4 (TO-security): The system M is TO-secure with respect to $\rightarrowtail$ if for all domains $u \in D$ and all $\alpha, \alpha' \in A^*$ with $\mathtt{to}_u(\alpha) = \mathtt{to}_u(\alpha')$, we have $\mathtt{obs}_u(s_0 \cdot \alpha) = \mathtt{obs}_u(s_0 \cdot \alpha')$.

It is possible to give a flatter representation of the information in $\mathtt{to}_u(\alpha)$ that clarifies the relationship of this definition to P-security. Define the *possibly transmitted view* of domain u for a sequence of actions α to be the largest prefix $\mathtt{tview}_u(\alpha)$ of $\mathtt{view}_u(\alpha)$ that ends in an action a with $\mathtt{dom}(a) = u$. Then we have the following result, which intuitively says that u's observations depend only on (1) the parts of the views of other agents which are permitted to pass information to u that they have actually acted to transmit, and (2) u's knowledge of the ordering of its own actions and the actions of these other agents.

Proposition 1: (Characterization of TO-security [4]) M is TO-secure with respect to a policy $\rightarrowtail$ iff for all sequences $\alpha, \alpha' \in A^*$, and domains $u \in D$, if $\mathtt{purge}_u(\alpha) = \mathtt{purge}_u(\alpha')$ and $\mathtt{tview}_v(\alpha) = \mathtt{tview}_v(\alpha')$ for all domains $v \neq u$ such that $v \rightarrowtail u$, then $\mathtt{obs}_u(s_0 \cdot \alpha) = \mathtt{obs}_u(s_0 \cdot \alpha')$.

The definitions introduced above are shown in [4] to be related as follows: P-security implies TO-security implies TA-security implies IP-security. The converse of each of these implications does not hold: Figures 5-7 provide counter-examples. In the special case of transitive policies $\rightarrowtail$, all these notions are equivalent.

III. Characterization of IP-security and TA-security

A. Characterization of IP-security

We present a new characterization of IP-security. This characterization is the main tool for our later algorithm that verifies IP-security in polynomial time.

Intuitively, the ipurge function that defines IP-security removes actions that should be irrelevant for the domain u from its "visible trace." This leads us to the definition of the relation $\rightarrow_u^{irr}$: for $u \in D$ and $\alpha, \alpha' \in A^*$, we define $\alpha \rightarrow_u^{irr} \alpha'$ if $\mathtt{ipurge}_u(\alpha) = \mathtt{ipurge}_u(\alpha')$ and there exist $\beta, \beta' \in A^*$, and $a \in A$ such that $\alpha = \beta a \beta'$ and $\alpha' = \beta\beta'$. That is, $\alpha \rightarrow_u^{irr} \alpha'$ if α' is obtained from α by removing a single action that is "irrelevant" in the sense that (according to the information flow allowed by the policy) u should not be able to observe whether the removed action has occurred at all. The symmetric closure of $\rightarrow^{irr}$ is denoted with $\leftrightarrow^{irr}$.

Note that if $\mathtt{ipurge}_u(\alpha) = \mathtt{ipurge}_u(\alpha')$, then there exists a sequence $\alpha = \alpha_0 \leftrightarrow^{irr} \alpha_1 \leftrightarrow^{irr} \ldots \leftrightarrow^{irr} \alpha_n = \alpha'$. If $\mathtt{obs}_u(s_0 \cdot \alpha) \neq \mathtt{obs}_u(s_0 \cdot \alpha')$, then we must have $\mathtt{obs}_u(s_0 \cdot \alpha_k) \neq \mathtt{obs}_u(s_0 \cdot \alpha_{k+1})$ for some

k. Thus, directly from the definition of IP-security, we obtain that a system is IP-insecure iff there exists a domain $u \in D$, a reachable state $q \in S$, $a \in A$ and $\alpha \in A^*$ such that $\texttt{ipurge}_u(a\alpha) = \texttt{ipurge}_u(\alpha)$ and $\texttt{obs}_u(q \cdot a\alpha) \neq \texttt{obs}_u(q \cdot \alpha)$. We now state a lemma that shows that we can put some restrictions on α. The lemma shows that if a system is not IP-secure, then an α and a as above exist such that additionally, α does not contain any action c whose domain $\texttt{dom}(c)$ can be influenced by $\texttt{dom}(a)$. This allows us to reduce the search space for a witness of insecurity significantly when designing our algorithms.

In the following, we will always assume that every state $s \in S$ is reachable, i.e., there is a sequence $\alpha \in A^*$ such that $s_0 \cdot \alpha = s$.

Lemma 1: A system M is IP-insecure iff there exist $u \in D$, $q \in S$, $a \in A$ and $\alpha \in A^*$ such that

(i) $\texttt{ipurge}_u(a\alpha) = \texttt{ipurge}_u(\alpha)$,
(ii) $\texttt{obs}_u(q \cdot a\alpha) \neq \texttt{obs}_u(q \cdot \alpha)$, and
(iii) $\texttt{dom}(a)^{\rightarrowtail} \cap \{\texttt{dom}(c) | c \in \text{alph}(\alpha)\} = \emptyset$.

B. Characterization of TA-security

We present a new characterization of TA-security that also makes precise its relationship to IP-security. As seen earlier, the latter is concerned with the question which actions an agent u may observe at all, hence $\texttt{ipurge}_u(\alpha)$ is obtained from α by removing from α actions that should be "unobservable" for u, provided that information only flows as specified by the security policy. The definition of TA-security in [4] was motivated by the observation that the security-relevant information that should be unobservable to some agents is not just which actions appear at all, but also information about the *order* in which certain actions are performed. This type of information-flow is not prohibited by the definition of IP-security.

In this section we show that what separates the definition of IP-security from that of TA-security is the question how much information is known about execution orders of actions. TA-security can essentially be seen as IP-security plus the requirement that an agent should only have access to "timing information" (i.e., information about the order of the occurrence of actions) insofar as permitted by the security policy.

To formalize this, we require a few technical definitions. The following definition captures the situation in which an agent u should not have information about the order in which certain actions are performed, although it may know whether these actions have been performed, and how often.

Definition 5 (swappable): Let $\alpha, \alpha' \in A^*$ and $a, b \in A$ and $u \in D$. We write $\alpha ab\alpha' \leftrightarrow_u^{swap} \alpha ba\alpha'$ iff $\texttt{dom}(a)^{\rightarrowtail} \cap \texttt{dom}(b)^{\rightarrowtail} \cap \{u, \texttt{dom}(c) | c \in \text{alph}(ab\alpha')\} = \emptyset$. In this case, we call the actions a and b *swappable* in $\alpha ab\alpha'$.

For any relation $\rightarrow$, we define $\stackrel{=}{\rightarrow}$ as the reflexive closure of $\rightarrow$ and $\stackrel{*}{\rightarrow}$ as the reflexive, transitive closure of $\rightarrow$.

We will see later that definition 5 captures exactly the issue mentioned above: If $\alpha ab\alpha' \leftrightarrow_u^{swap} \alpha ba\alpha'$, then the action sequences $\alpha ab\alpha'$ and $\alpha ba\alpha'$ should be indistinguishable for agent u, even though it is allowed to know whether actions a and b have been performed. The reason why, intuitively, u should not have access to this "timing information" is that only agents $w \in \texttt{dom}(a)^{\rightarrowtail} \cap \texttt{dom}(b)^{\rightarrowtail}$ can directly observe whether a or b is performed first. If no agent that can observe this information directly performs any action in α, then, after performing α, the agent u should not have this information either (unless of course, u is in the intersection.)

We call strings $\alpha, \alpha' \in A^*$ *order indistinguishable for* u, and write $\alpha \equiv_u^{oi} \alpha'$, if $\alpha \stackrel{*}{\leftrightarrow}_u^{swap} \alpha'$.

The following lemma shows that our definition correctly captures the above intuition. It states that information about the order of "swappable" operations are indeed hidden from an agent by the definition of TA-security.

Lemma 2: Let $u \in D$, $\alpha, \alpha' \in A^*$ with $\alpha \leftrightarrow_u^{swap} \alpha'$, then $\texttt{ta}_u(\alpha) = \texttt{ta}_u(\alpha')$.

The following corollary combines the above result and the fact that TA-security implies IP-security:

Corollary 1: Let be $u \in D$ and $\alpha, \alpha' \in A^*$ with $\texttt{ipurge}_u(\alpha) \equiv_u^{oi} \texttt{ipurge}_u(\alpha')$, then $\texttt{ta}_u(\alpha) = \texttt{ta}_u(\alpha')$.

We now state the result mentioned earlier: TA-security is, in a very precise sense, IP-security plus the requirement that agents should not be able to distinguish between action sequences that are order

indistinguishable. The following theorem shows that the information that an agent is not permitted to have in the definition of IP-security, in addition to information already forbidden to it by IP-security, is exactly the information about the orders of actions that are "swappable."

Theorem 1: Let be $u \in D$ and $\alpha, \alpha' \in A^*$. Then $\mathtt{ta}_u(\alpha) = \mathtt{ta}_u(\alpha')$ if and only if $\mathtt{ipurge}_u(\alpha) \equiv_u^{oi} \mathtt{ipurge}_u(\alpha')$.

The characterization obtained by the above theorem is now stated in the following corollary:

Corollary 2: A system M is TA-secure if and only if it is IP-secure and for every state q, every agent u, and every $a, b \in A$, $\alpha \in A^*$, if a and b are swappable in $ab\alpha$, then $\mathtt{obs}_u(q \cdot ab\alpha) = \mathtt{obs}_u(q \cdot ba\alpha)$.

IV. Complexity

In this section, we consider the complexity of algorithmic verification of the notions we have discussed, in the case of finite state systems.

We show that three of these notions (P-security, IP-security, and TA-security) are decidable, and in fact can be decided in polynomial time, and we prove that TO-security is undecidable.

A. P-security

We show that P-security can be verified in polynomial time. The proof uses a "doubling" construction similar to those that have been applied elsewhere [9] to show that verifying the simple (transitive) policy $L \rightarrowtail H$ can be done in PTIME.

SC: If $s \sim_u t$ then $s \cdot a \sim_u t \cdot a$,
LR: if $\mathtt{dom}(a) \rightarrowtail u$ then $s \sim_u s \cdot a$.

(Here SC abbreviates "step consistency" and LR stands for "left respect") also satisfies "output consistency" defined as

OC: If $s \sim_u t$, then $\mathtt{obs}_u(s) = \mathtt{obs}_u(t)$.

This characterization allows us to prove the following:

Theorem 2: Given a finite system $M = \langle S, s_0, A, \mathtt{step}, \mathtt{obs}, \mathtt{dom}\rangle$ and a (possibly intransitive) noninterference policy $\rightarrowtail$, it can be decided in time $O(|S|^2 \cdot |A| \cdot |D|)$ whether M is P-secure with respect to $\rightarrowtail$.

Proof: Let R be the smallest subset of $S^2 \times D$ such that

1) $(s_0, s_0, u) \in R$
2) if $(s, t, u) \in R$ and $\mathtt{dom}(a) \rightarrowtail u$, then $(s \cdot a, t \cdot a, u) \in R$,
3) if $(s, t, u) \in R$ and $\mathtt{dom}(a) \not\rightarrowtail u$, then $(s \cdot a, t, u) \in R$ and $(s, t \cdot a, u) \in R$.

Then we have $(s, t, u) \in R$ iff there exists a sequence $\alpha \in A^*$ such that $s = s_0 \cdot (\alpha)$ and $t = s_0 \cdot \mathtt{purge}_u(\alpha)$. Thus M is not P-secure iff there exists $(s, t, u) \in R$ such that $\mathtt{obs}_u(s) \neq \mathtt{obs}_u(t)$. The complexity bound is attained by a depth first search following the construction of R. ∎

B. IP-security

We present a polynomial time algorithm for verifying IP-security. The algorithm uses the characterization of IP-security given by Lemma 1: It checks if for any $u \in D$, $q \in S$, $a \in A$ there is an $\alpha \in A^*$ that leads to states with different observations and α does not contain an action of a domain from $\mathtt{dom}(a)^{\rightarrowtail}$.

Theorem 3: Given a finite system $M = \langle S, s_0, A, \mathtt{step}, \mathtt{obs}, \mathtt{dom}\rangle$ and a policy $\rightarrowtail$, it can be decided in time $O(|S|^3 \cdot |A|^2 \cdot |D|)$ whether M is IP-secure with respect to $\rightarrowtail$.

Proof: For any $u \in D$, $q \in S$ and $a \in A$ we define the following deterministic finite-state automaton:

$$\mathcal{A}(u, q, a) = (A', S \times S, (q, q \cdot a), \Delta, F)$$

where we set

$$A' = A \setminus \{b \in A \mid \mathtt{dom}(a) \rightarrowtail \mathtt{dom}(b)\}$$
$$\Delta = \{((s, s'), b, (t, t')) | b \in A', s \cdot b = t, s' \cdot b = t'\}$$
$$F = \{(s, s') | \mathtt{obs}_u(s) \neq \mathtt{obs}_u(s')\} \ .$$

Applying Lemma 1 it is sufficient to check emptiness of the language generated by $\mathcal{A}(u, q, a)$ for every $u \in D$, $q \in S$, $a \in A$. For fixed u, q, a this can be done in time linear in the number of transitions, which are bounded by $|S|^2 \cdot |A|$. ∎

Note that the above algorithm is essentially a reduction to the reachability problem in directed graphs. It is therefore easy to see that IP-security can be verified in nondeterministic logarithmic space. Analogous arguments hold for P-security and TA-security. Since graph search trivially reduces to

these problems, verification of all three security notions is complete for nondeterministic logarithmic space.

C. TA-security

We show that, in a similar fashion as IP-security, TA-security can be verified in polynomial time. The algorithm relies on the characterization of IP-security given in Section III-B.

Theorem 4: Given a finite system $M = \langle S, s_0, A, \mathtt{step}, \mathtt{obs}, \mathtt{dom}\rangle$ and a policy $\rightarrowtail$, it can be decided in time $O(|S|^3 \cdot |A|^3 \cdot |D|)$ whether M is TA-secure with respect to $\rightarrowtail$.

Proof: We use the characterization of TA-security given by Corollary 2. We first apply the algorithm from Theorem 3 to determine whether M is IP-secure. If this is not the case, then M also is not TA-secure. It remains to verify that there is no state q, actions $a, b \in A$, and sequence $\alpha \in A^*$ that do not satisfy the conditions from Corollary 2. To verify this, we proceed in the same way as in the proof of Theorem 3: For every choice of q, a, b and u such that $\mathtt{dom}(a) \not\rightarrowtail \mathtt{dom}(b)$ and $\mathtt{dom}(b) \not\rightarrowtail \mathtt{dom}(a)$, we consider the finite automaton that simulates two copies of the system M in parallel, one starting in the state $q \cdot ab$ and the other starting in $q \cdot ba$. We check whether in this automaton, there is a state reachable (via the same action sequence α that does not contain any element from $\mathtt{dom}(a)^{\rightarrowtail} \cap \mathtt{dom}(b)^{\rightarrowtail}$) that has different observations in the two copies.

The considered automaton has a state space of cardinality $|S|^2$, and has $|S|^2 \cdot |A|$ many edges. Hence a search for a single automaton requires time $|S|^2 \cdot |A|$. Since the procedure has to be performed for every possible choice of a, b, q, u, an additional factor of $|S| \cdot |A^2| \cdot |D|$ occurs. Since this running time dominates the running time of the algorithm from Theorem 3, this concludes the proof. ■

D. TO-security

We now prove that TO-security is undecidable. The proof also shows that the source of the undecidability does not lie in using complex policies, in fact the problem remains undecidable for a very simple, small policy.

Theorem 5: It is undecidable whether M is TO-secure with respect to $\rightarrowtail$, even for a fixed policy containing 4 domains.

Proof: We prove the undecidability of TO-security by a reduction from the Post Correspondence Problem [10]. An instance of this problem consists of a pair of sequences $\mathcal{U} = U_1, \ldots, U_n$ and $\mathcal{W} = W_1, \ldots, W_n$ of words over an alphabet Σ with at least two letters. The problem PCP is the set of such pairs $(\mathcal{U}, \mathcal{W})$ such that there exists a sequence of indices $i_1, \ldots, i_k$ with $1 \leq i_j \leq n$ for each $j = 1, \ldots, k$, such that $U_{i_1} \ldots U_{i_k} = W_{i_1} \ldots W_{i_k}$. We encode an instance of this problem as a machine $M(\mathcal{U}, \mathcal{W})$ for the (intransitive) policy for agents A, B, C, D given by $A \rightarrowtail C$, $A \rightarrowtail D$, $B \rightarrowtail C$ and $C \rightarrowtail D$, such that $(\mathcal{U}, \mathcal{W}) \in PCP$ iff $M(\mathcal{U}, \mathcal{W})$ is not TO-secure with respect to $\rightarrowtail$.

Intuitively, in the machine $M(\mathcal{U}, \mathcal{W})$, agent A guesses a word over Σ, and agent B chooses whether this word is to be compared to a sequence of U_i or W_i, and guesses a sequence of indices used to make the comparison. Agent C observes the indices guessed by B, and guesses when the word being constructed is complete. Agent D observes nothing until C declares the end of the construction, and then observes whether the word guessed by A does in fact correspond to the sequence of indices guessed by B. The definition of TO-secure will be guaranteed to hold with respect to agents A, B and C, so the determination as to whether $M(\mathcal{U}, \mathcal{W})$ is TO-secure depends on how the observations of agent D relate to the actions and observations of A and C. More precisely, $M(\mathcal{U}, \mathcal{W})$ has

1) states of the form (p, V, i, x), where
   a) $p \in \{U, U', W\}$ indicates whether the sequence of letters guessed by A is to be compared with a sequence of U_i (when $p \in \{U, U'\}$) or as a sequence of W_i (when $p = W$).
   b) V is either a word over Σ which is a prefix (possibly the empty word ϵ) of one of the U_i or W_i, or $\top$. Intuitively, this indicates a part of the word guessed by A that will be compared to an index guessed by B. The case of $\top$ represents that an inconsistency

has been detected.[1]

c) $i \in \{0, \ldots, n\}$ is either 0 (no activity so far) or the last index guessed by B,

d) $x \in \{0, 1\}$ is used to represent the state of the computation, with 0 meaning ongoing and 1 meaning complete.

2) initial state $(U, \epsilon, 0, 0)$,

3) actions

a) of A: an action a for each $a \in \Sigma$, corresponding to guessing the letter a

b) of B: an action w (corresponding to the selection of W) plus an action g_i for each $i \in \{1, \ldots, n\}$ (corresponding to a guess of the index i)

c) of C: an action end

d) of D: none

The transition function is defined as follows. For all actions b and states $s = (p, V, i, x)$, if $x = 1$ then we will have $\texttt{step}(s, b) = s$, i.e., once the computation has terminated, no action changes the state. We therefore confine the definitions below to the case $x = 0$. We make use of two functions $G\colon \{U, U', W\} \to \{U, W\}$ defined by $G(U) = G(U') = U$ and $G(W) = W$, and $F\colon \{U, U', W\} \to \{U', W\}$ defined by $F(U) = F(U') = U'$ and $F(W) = W$.

In a state (p, V, i, x), the value $G(p)$ captures the choice of $\mathcal{U}$ or $\mathcal{W}$ with which to compare the word being generated by A. Intuitively, both $p = U$ and $p = U'$ represent that the word being processed is to be compared with the $\mathcal{U}$. This the default, as indicated in the initial state. The reason for including U' is that agent B is given an opportunity to switch the system to comparing with $\mathcal{W}$ only in the first step of a run. If it does not act, then the choice remains with $\mathcal{U}$ for the remainder of the run.

In the case of action w, we define $\texttt{step}((p, V, i, 0), w) = (W, V, i, 0)$ if $p = U$ and $\texttt{step}((p, V, i, 0), w) = (p, V, i, 0)$ otherwise. This says that w switches the choice of comparison to $\mathcal{W}$. That the choice can be made only if w is the initial action of a run is captured by defining all other actions $b \neq w$ so that if $\texttt{step}((p, V, i, 0), b) = (p', V', i', x')$ then $p' = F(p)$.

[1]Throughout, we use $\perp$ to represent undetermined information and $\top$ to represent inconsistency.

For the actions a of A, we define $\texttt{step}((p, V, i, x), a) = (F(p), V', i, x)$, where $V' = V \cdot a$ if $V \cdot a$ is a prefix of $G(p)_j$ for some j, and $V' = \top$ otherwise. Intuitively, V is used to collect a fragment of the sequence being generated by A for comparison with the $G(p)_j$. We accumulate the fragment while it is a prefix of such a string, and as soon as this is no longer the case we flag the inconsistency.

For the actions g_j of B, we define $\texttt{step}((p, V, i, 0), g_j) = (F(p), V', j, 0)$, where

1) if $G(p)_j = V$ then $V' = \epsilon$, and

2) if $G(p)_j \neq V$ then $V' = \top$.

Intuitively, this captures that the effect of the action g_j is to compare $G(p)_j$ with the current fragment of the string being generated by A. If they are equal, we reset V to ϵ in order to check the next fragment. Otherwise, we flag the inconsistency.

For the action end of C, we define $\texttt{step}((p, V, i, 0), end) = (F(p), V', i, 1)$, where

1) if $V = \epsilon$ then $V' = \epsilon$, and

2) if $V \neq \epsilon$ then $V' = \top$.

Intuitively, this action checks that the end is declared at a time when there is no fragment currently being processed, and flags an inconsistency otherwise.

The observations are now defined as follows. The observations of A and B are trivial: $\texttt{obs}_A(s) = \texttt{obs}_B(s) = \perp$ for all states s. For C, we define

$$\texttt{obs}_C((p, V, i, x)) = \begin{cases} i & \text{if } V = \epsilon \\ \top & \text{if } V = \top \\ \perp & \text{otherwise.} \end{cases}$$

Note that this means that for the initial state s_0 we have $\texttt{obs}_C(s_0) = 0$. Intuitively, since i records the last (successful) guess of index for a fragment of the word being generated by A, we have that C becomes aware of a guess whenever it is correct, and can see from its observation $\perp$ that a further fragment is in the process of being constructed.

For D we define

$$\texttt{obs}_D((p, V, i, x)) = \begin{cases} \perp & \text{when } x = 0, \\ G(p) & \text{when } x = 1, \text{ and } V = \epsilon \text{ and } i \neq 0 \\ \top & \text{when } x = 1 \text{ and either } V \neq \epsilon \text{ or } i = 0. \end{cases}$$

Intuitively, this means that D observes $\perp$ until C declares the end of the string, and learns whether

$\mathcal{U}$ or $\mathcal{V}$ fragments were being checked when a decomposition has been successfully guessed. Otherwise, it learns that the guesses do not match. This completes the definition of the system $M(\mathcal{U}, \mathcal{W})$.

It can be verified that this reduction is correct and hence establishes undecidability of TO-security. It is easy to check that the conditions for TO-security are satisfied in $M(\mathcal{U}, \mathcal{W})$ for the agents $u = A, B, C$. For $u = D$, the definition is violated iff there exists a sequence of indices $i_1, \ldots i_k$ such that $U_{i_1} \ldots U_{i_k} = W_{i_1} \ldots W_{i_k}$. ■

We note that the undecidability result for TO-security implies that there are no simple unwinding conditions that are complete for this notion of security. In particular, any first-order set of conditions on a collection of binary relations on states can be checked in PTIME, hence cannot be both sound and complete.

V. Related Work

The notion of *noninterference* was first proposed by Goguen and Meseguer [1]. Early work in this area was motivated by multi-level secure systems, and dealt with partially ordered (hence transitive) information flow policies. The simplest of these is the two-domain policy with domains L and H and $L \rightarrowtail H$, but not $H \rightarrowtail L$. Much of the literature is confined to this simple policy. Even with this restriction, there exists a large set of proposed definitions of noninterference [11], [12], [13], [14], [15]. These definitions generally agree when applied to deterministic systems, and the differences relate to how the definitions should behave on nondeterministic systems. In addition to state-observed systems model used in the present paper, a variety of systems models have been considered, including action-observed systems, trace semantics, and process algebraic semantics (both CCS and CSP flavours). A number of works have sought to classify the definitions when formulated in a very general process algebraic setting [14], as well as to establish formal relations between definitions cast in different semantic models [8].

The main point of overlap of our work with this literature is to consider how our results concerning P-security, when applied to transitive policies, relate to other algorithmic verification approaches in the literature for such policies. Our approach here is similar to other work in the area. In particular, the idea of running two copies of the system in parallel, in order to compare two different runs, has been used before [9]. Other approaches have been developed for automated verification of noninterference based on process algebraic bisimulation techniques [14], [16]. Mantel [17] has characterised many of the existing definitions of noninterference as compositions of a set of Basic Security Properties. The complexity of verifying these basic properties has been studied [18]. A few works have considered richer systems models than finite state systems, e.g. pushdown systems [19]. We note that our results in this paper, and in much of the literature, is concerned with asynchronous systems in which agents are unaware of the passage of time. Some of the literature deals with synchronous systems, where a similar spectrum of definitions of noninterference exists for nondeterminisitic systems. Some recent work has investigated verification of synchronous notions of noninterference [20], [21].

Some work on development of tools based on decidable cases of such definitions of noninterference has been performed. Focardi et al. describe a tool based on process algebraic techniques [22]. Whalen et al. [23] present an approach to model checking noninterference that is in use at Rockwell Collins for verification of MILS systems. Their approach is a mix of model checking and static analysis, in which a model checker is used to search through an enriched version of the model in which labels computed by static analysis are associated to systems components. They formally prove it to be sound with respect to a definition of noninterference from work by Greve, Wilding and van Fleet [24]. While they discuss examples requiring intransitive policies, they leave details of this for future work.

Since we have confined ourselves in this paper to deterministic systems, but focus on richer intransitive policies, much of the work discussed above, which is confined to transitive policies and nondeterministic systems, is orthogonal to our concerns. Algorithmic verification of intransitive noninterference has had less attention in the literature. After the work of Rushby [3], IP-security has generally been taken to be the definition studied.

Pinsky [25] presents a PTIME procedure for

deciding IP-security that, in effect, generates a relation that is claimed to satisfy Rushby's unwinding conditions for transitive noninterference just when the system is secure. However, in fact the relation may fail to satisfy the Output Consistency condition, so this claim is incorrect. (Pinsky's argument supporting the claim that the relation satisfies Output Consistency, in the corollary to Theorem 2, states that $SA(basis_\pi(z), \alpha)$ is a subset of $view(state\_action(z, \alpha))$. This is correct for transitive policies, but could be false for intransitive policies.) That such an approach cannot work for IP-security also follows from results in [4], where it is shown that Rushby's unwinding conditions are sound also for TA-security, which is a stronger notion than IP-security. Moreover, an example in [4] shows that a system may be TA-secure, but no Rushby unwinding exists on the system (although one will exist on the infinite state unfolded system, when the system satisfies TA-security).

Hadj-Alouane et al. [26] also present a decision procedure for IP-security, but it has complexity $O(2^{|S|.2^{|D|}})$, which is less efficient than our procedure by two exponentials.

Roscoe and Goldsmith [27] have presented a critique of IP-security (arguing that it is too liberal in the information flows it permits), and have proposed two alternate definitions cast in the process algebra CSP, based on what they call *lazy* and *mixed* abstraction operators. It has been shown by van der Meyden [28] that the definition based on lazy abstraction corresponds to P-security, and the version based on mixed abstraction corresponds to a definition closely related to TO-security. Roscoe and Goldsmith give an informal discussion, without precise complexity results or proof, of algorithms for deciding "the generalised noninterference condiiton". Based on van der Meyden's characterization of the definition based on the mixed abstraction as related to TO-security, we would conjecture that this definition is undecidable, and their comments should be interpreted as concerned (like most of the preceding content in their paper) just with the lazy abstraction based definition, and hence comparable to our PTIME result for P-security.

In practice, definitions of the kind we have studied are very liberal in the information flows that they permit: when a (source) domain acts, everything that it knows (in some sense of knowledge) may be transmitted to any domain with which the source is permitted to interfere. In practice, one generally wants to limit the information that flows from one domain to another to be just a subset of the information available to the source.

Approaches to stating policies expressing such limitations have been developed in the context of language-based approaches to security, where they are generally supported by means of sound but incomplete static analyses [29], [30], [31], [32], [33]. In the existing work, the policy is generally taken to be $L \rightarrowtail H$ with exceptions allowed to $H \not\rightarrowtail L$, or more generally, a partial order with exceptions. The system is given by a single, typically deterministic program, and the focus is on relating initial values of input variables to final values of output variables, rather than on what can be deduced from ongoing observations in the state machine approach we have considered here.

VI. Conclusion

In this paper, we have determined the computational complexity of verifying whether a finite-state system satisfies an intransitive noninterference security property. The polynomial-time upper bounds build on new characterizations of two of the four notion of noninterference dealt with. They also allow counterexamples (which can be used to improve the system in question) to be found when the system is insecure.

We have considered only deterministic systems: although there have been some attempts [27], [34], to define intransitive noninterference in nondeterministic systems would seem to require further attention, particularly in view of issues raised in the work of van der Meyden [4]. Another open question is our conjecture that Roscoe and Goldsmith's mixed abstraction-based definition of intransitive noninterference is undecidable.

It would also be desirable to investigate algorithms and complexity for information flow policies of the richer types studied in the literature on programming languages approaches to declassification, in order to obtain sound and complete approaches for such specifications. Since intransitive noninterference policies provide a format for specifying

architectural structure of a system, it would be interesting to combine the strengths of the programing languages perspective and the state machine model approach we have followed in this paper.

Acknowledgments: Work supported by Australian Research Council Discovery grant DP1097203. The second author thanks the Courant Institute, New York University, for hosting a sabbatical visit during which this research was initiated.

REFERENCES

[1] J. Goguen and J. Meseguer, "Security policies and security models," in *Proc. IEEE Symp. on Security and Privacy*, Oakland, 1982, pp. 11–20.

[2] J. Haigh and W. Young, "Extending the noninterference version of MLS for SAT," *IEEE Trans. on Software Engineering*, vol. SE-13, no. 2, pp. 141–150, Feb 1987.

[3] J. Rushby, "Noninterference, transitivity, and channel-control security policies," SRI International, Tech. Rep. CSL-92-02, Dec 1992. [Online]. Available: http://www.csl.sri.com/papers/csl-92-2/

[4] R. van der Meyden, "What, indeed, is intransitive noninterference?" in *European Symposium On Research In Computer Security (ESORICS)*, ser. Lecture Notes in Computer Science, J. Biskup and J. Lopez, Eds., vol. 4734. Springer, 2007, pp. 235–250.

[5] J. Goguen and J. Meseguer, "Unwinding and inference control," in *IEEE Symp. on Security and Privacy*, 1984.

[6] D. E. Denning, "A lattice model of secure information flow," *Communications of the ACM*, vol. 19, no. 5, pp. 263–243, 1976.

[7] C. Boettcher, R. DeLong, J. Rushby, and W. Sifre, "The MILS component integration approach to secure information sharing," in *Proc. 27th IEEE/AIAA Digital Avionics Systems Conference*, Oct. 2008, pp. 1.C.2–1–1.C.2–14.

[8] R. van der Meyden and C. Zhang, "A comparison of semantic models for noninterference," *Theoretical Computer Science*, vol. 411, no. 47, pp. 4123–4147, Oct. 2010.

[9] G. Barthe, P. R. D'Argenio, and T. Rezk, "Secure information flow by self-composition," in *CSFW*. IEEE Computer Society, 2004, pp. 100–114.

[10] E. Post, "A variant of a recursively unsolvable problem," *Bulletin of the American Mathematical Society*, vol. 52, 1946.

[11] D. Sutherland, "A model of information," in *Proc. 9th National Computer Security Conf.*, 1986, pp. 175–183.

[12] J. T. Wittbold and D. M. Johnson, "Information flow in nondeterministic systems." in *IEEE Symposium on Security and Privacy*, 1990, pp. 144–161.

[13] D. McCullough, "Noninterference and the composability of security properties," in *Proc. IEEE Symp. on Security and Privacy*, 1988, pp. 177–186.

[14] R. Focardi and R. Gorrieri, "Classification of security properties (Part I: information flow)," in *Foundations of Security Analysis and Design, FOSAD 2000, Bertinoro, Italy, September 2000*, ser. LNCS, R. Focardi and R. Gorrieri, Eds. Springer, 2001, vol. 2171, pp. 331–396.

[15] P. Ryan, "Mathematical models of computer security," in *Foundations of Security Analysis and Design, FOSAD 2000, Bertinoro, Italy, September 2000*, ser. LNCS, R. Focardi and R. Gorrieri, Eds. Springer, 2001, vol. 2171, pp. 1–62.

[16] R. Focardi and R. Gorrieri, "Automatic compositional verification of some security properties," in *Proceedings of Second International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, ser. Springer LNCS, vol. 1055, 1996, pp. 167–186.

[17] H. Mantel, "A uniform framework for the formal specication and verication of information flow security," Ph.D. dissertation, Universitat des Saarlandes, 2003.

[18] D. D'Souza, K. R. Raghavendra, and B. Sprick, "An automata based approach for verifying information flow properties," *Electr. Notes Theor. Comput. Sci.*, vol. 135, no. 1, pp. 39–58, 2005.

[19] D. D'Souza, R. Holla, J. Kulkarni, R. K. Ramesh, and B. Sprick, "On the decidability of model-checking information flow properties," in *ICISS*, ser. Lecture Notes in Computer Science, R. Sekar and A. K. Pujari, Eds., vol. 5352. Springer, 2008, pp. 26–40.

[20] B. Köpf and D. A. Basin, "Timing-sensitive information flow analysis for synchronous systems," in *Proc. European Symp. on Research in Computer Security*, ser. Springer LNCS, vol. 4189, 2006, pp. 243–262.

[21] F. Cassez, R. van der Meyden, and C. Zhang, "The complexity of synchronous notions of information flow security," in *FoSSaCS 2010, 13th International Conference on Foundations of Software Science and Computation Structures*, ser. Springer LNCS, vol. 6014, 2010, pp. 282–296.

[22] R. G. Riccardo Focardi and V. Panini, "The security checker: a semantics-based tool for the verification of security properties," in *Proceedings of Eighth IEEE Computer Security Foundations Workshop (CSFW'95)*, 1995, pp. 60–69.

[23] M. W. Whalen, D. A. Greve, and L. G. Wagner, "Model checking information flow," in *Design and Verification of Microprocessor Systems for High-Assurance Applications*, D. Hardin, Ed. Springer-Verlag, Berlin Germany, March 2010.

[24] D. Greve, M. Wilding, and M. Vanfleet, "A separation kernel formal security policy," in *Proc. Fourth International Workshop on the ACL2 Theorem Prover and Its Applications*, 2003.

[25] S. Pinsky, "Absorbing covers and intransitive non-interference." in *Proc. IEEE Symp. on Security and Privacy*, 1995, pp. 102–113.

[26] N. Hadj-Alouane, S. Lafrance, F. Lin, J. Mullins, and M. Yeddes, "On the verification of intransitive noninterference in multilevel security," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 35, no. 5, pp. 948– 958, Oct. 2005.

[27] A. W. Roscoe and M. H. Goldsmith, "What is intransitive noninterference?" in *IEEE Computer Security Foundations Workshop*, 1999, pp. 228–238.

[28] R. van der Meyden, "A comparison of semantic models for intransitive noninterference," Dec 2007, unpublished manuscript, available at `http://www.cse.unsw.edu.au/~meyden`.

[29] H. Mantel and D. Sands, "Controlled declassification based on intransitive noninterference," in *Proc. Asian Symp. on Programming Languages and Systems*, ser. LNCS, vol. 3302. Springer-Verlag, Nov. 2004, pp. 129–145.

[30] A. Sabelfeld and D. Sands, "Dimensions and principles of declassification," in *Proceedings of the 18th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2005, pp. 255–269.

[31] S. Chong and A. C. Myers, "Security policies for downgrading," in *11th ACM Conf. on Computer and Communications Security (CCS)*, Oct 2004.

[32] A. Banerjee, D. A. Naumann, and S. Rosenberg, "Expressive declassification policies and modular static enforcement," in

IEEE Symposium on Security and Privacy. IEEE Computer Society, 2008, pp. 339–353.

[33] A. C. Myers, A. Sabelfeld, and S. Zdancewic, "Enforcing robust declassification," in *CSFW*. IEEE Computer Society, 2004, pp. 172–186.

[34] D. von Oheimb, "Information flow control revisited: Noninfluence = Noninterference + Nonleakage," in *Computer Security – ESORICS 2004*, ser. LNCS, vol. 3193, 2004, pp. 225–243.