The Complexity of Synchronous Notions of Information Flow Security

Franck Cassez[1,⋆], Ron van der Meyden[2,⋆⋆], and Chenyi Zhang[3]

[1] National ICT Australia & CNRS, Sydney, Australia
[2] University of New South Wales, Sydney, Australia
[3] University of Luxembourg, Luxembourg

Abstract. The paper considers the complexity of verifying that a finite state system satisfies a number of definitions of information flow security. The systems model considered is one in which agents operate synchronously with awareness of the global clock. This enables timing based attacks to be captured, whereas previous work on this topic has dealt primarily with asynchronous systems. Versions of the notions of nondeducibility on inputs, nondeducibility on strategies, and an unwinding based notion are formulated for this model. All three notions are shown to be decidable, and their computational complexity is characterised.

1 Introduction

Information flow security is concerned with the ability for agents in a system to deduce information about the activity and secrets of other agents. An information flow security policy prohibits some agents from knowing information about other agents. In an insecure system, an agent may nevertheless be able to make inferences from its observations, that enable it to deduce facts that it is not permitted to know. In particular, a class of system design flaws, referred to as covert channels, provide unintended ways for information to flow between agents, rendering a system insecure.

Defining what it is for a system to satisfy an information flow security policy has proved to be a subtle matter. A substantial literature has developed that provides a range of formal systems models and a range of definitions of security. In particular, in nondeterministic systems it has been found necessary to clarify the attack model, and distinguish between a passive attacker, which merely aims to deduce secret information from observations it is able to make from its position outside the security domain to be protected, and a more active attacker, that may have planted a Trojan Horse in the domain to be protected, and which seeks to use covert channels to pass information out of this domain. While this distinction turns out not to matter in asynchronous systems, in synchronous settings, it leads to two different definitions of security, known as Nondeducibility

⋆ Author supported by a Marie Curie International Outgoing Fellowship within the 7th European Community Framework Programme.

⋆⋆ Work supported by an Australian Research Council Discovery grant.

on Inputs (`NDI`), and Nondeducibility on Strategies (`NDS`). (The term strategies in the latter refers to the strategies that a Trojan Horse may employ to pass information out of the security domain.) Considerations of proof methods for security, and compositionality of these methods, has lead to the introduction of further definitions of security, such as *unwinding relations* and the associated definition of *restrictiveness* (`RES`).

One of the dimensions along which it makes sense to evaluate a definition of security is the practicality of verification techniques it enables. The early literature on the topic dealt primarily with theorem proving verification methods, but in recent years the feasibility of automated verification techniques has begun to be investigated. This recent work on automated verification of security has dealt primarily with asynchronous systems models. In this paper we investigate the complexity of automated verification for a range of definitions of information flow in a *synchronous* systems model, in which agents are aware of a global clock and may use timing information in their deductions. This model is significant in that a number of timing-based attacks have been demonstrated that, e.g., enable cryptographic keys to be deduced just from the amount of time taken to perform cryptographic operations [Koc96]. It is therefore desirable that systems designs are free of timing-based covert channels; the asynchronous definitions of security that have been the focus of much of the literature fail to ensure this.

We study three definitions of security in this paper: synchronous versions of Nondeducibility on Inputs (`NDI`), Nondeducibility on Strategies (`NDS`) and an unwinding based definition (`RES`). We consider just a two-agent setting, with agents L for a low security domain and H for a high security domain, and the (classical) security policy that permits H to know about L's activity, but prohibits L from knowing about the activity of H. We show that all three definitions are decidable in finite state systems, and with complexities of PSPACE-complete for `NDI`, EXPSPACE-complete for `NDS`, and polynomial time for `RES`.

The structure of the paper is as follows. Section 2 introduces our systems model, the definitions of security that we study, and states the main results of the paper. The following sections discuss the proofs of these results. Section 3 deals with Nondeducibility on Inputs, Section 4 deals with Nondeducibility on Strategies, and Section 5 deals with the unwinding-based definition. Related literature is discussed in Section 6, and Section 7 makes some concluding remarks.[4]

2 Semantic Model and Definitions

We work with a synchronous, nondeterministic state machine model for two agents, H and L. At each step of the computation, the agents (simultaneously) perform an action, which is resolved nondeterministically into a state transition. Both agents make (possibly incomplete) observations of the state of the system, and do so with awareness of the time.

A *synchronous machine* M is a tuple of the form $\langle S, A, s_0, \rightarrow, O, obs \rangle$ where

[4] Proof details excluded due to space limitations will appear in the full version.

- S is the set of states,
- $A = A_H \times A_L$ is a set of joint actions (or joint inputs), each composed of an action of H from the set A_H and an action of L from the set A_L,
- s_0 is the initial state,
- $\rightarrow \subseteq S \times A \times S$ defines state transitions resulting from the joint actions,
- O is a set of observations,
- $obs : S \times \{H, L\} \rightarrow O$ represents the observations made by each agent in each state.

We write obs_u for the mapping $obs(\cdot, u) : S \rightarrow O$, and $s \xrightarrow{a} s'$ for $\langle s, a, s' \rangle \in \rightarrow$. We assume that machines are *input-enabled*, by requiring that for all $s \in S$ and $a \in A$, there exists $s' \in S$ such that $s \xrightarrow{a} s'$. We write $\mathbb{M}^s$ for the set of synchronous machines.

A *run* r of M is a finite sequence $r = s_0 a_1 s_1 \ldots a_n s_n$ with: $a_i \in A$ and $s_i \xrightarrow{a_{i+1}} s_{i+1}$ for all $i = 0 \ldots n-1$. We write $\mathcal{R}(M)$ for the set of all runs of M. We denote the sequence of joint actions $a_1 \ldots a_n$ in the run r by $Act(r)$. For each agent $u \in \{H, L\}$ we define $p_u : A \rightarrow A_u$ to be the projection of joint actions onto agent u's actions. We write $Act_u(r)$ for the sequence of agent u's actions in $Act(r)$, e.g., if $Act(r) = a_1 \ldots a_n$ then $Act_u(r) = p_u(a_1) \ldots p_u(a_n)$.

For a sequence w, and $1 \leq i \leq |w|$, we write w_i for the i-th element of w, and $w[i]$ for the prefix of w up to the i-th element. We assume agents have a *synchronous* view of the machine, making an observation at each moment of time and being aware of each of their own actions (but not the actions of the other agent, which are given simultaneously and independently). Given a synchronous machine M, and $u \in \{H, L\}$, we define the mappings $\texttt{view}_u : \mathcal{R}(M) \rightarrow O(A_u O)^*$ by:

$$\texttt{view}_u(s_0 a_1 s_1 a_2 \cdots a_n s_n) = obs_u(s_0)\, p_u(a_1)\, obs_u(s_2)\, p_u(a_2) \cdots p_u(a_n)\, obs_u(s_n).$$

Intuitively, this says that an agent's view of a run is the history of all its state observations as well as its own actions in the run. We say that a sequence v of observations and actions is a *possible u view in a system M* if there exists a run r of M such that $v = \texttt{view}_u(r)$. The mapping $\texttt{view}_u$ extends straightforwardly to sets of runs $R \subseteq \mathcal{R}(M)$, by $\texttt{view}_u(R) = \{\texttt{view}_u(r) \mid r \in R\}$. We define the length $|v|$ of a view v to be the number of actions it contains.

We remark that the model is sufficiently expressive to represent an alternate model in which agents act in turn under the control of a scheduler. We say that a machine is *scheduled* if for each state $s \in S$ either

- for all actions $a \in A_H$ and $b, b' \in A_L$, and states $t \in S$, $s \xrightarrow{(a,b)} t$ iff $s \xrightarrow{(a,b')} t$, or
- for all actions $a, a' \in A_H$ and $b \in A_L$, and states $t \in S$, $s \xrightarrow{(a,b)} t$ iff $s \xrightarrow{(a',b)} t$.

This definition says that state transitions in a scheduled machine are determined by the actions of *at most one* of the agents (the agent scheduled at that state); the other agent has no control over the transition. The model involving machines under the control of a scheduler of [vdMZ08], in which at most one agent acts at each step of the computation, can be shown to be interpretable as scheduled synchronous machines.

We consider a number of different notions of information flow security. Each definition provides an interpretation for the security policy $L \rightarrow H$, which states that information is permitted to flow from L to H, but not from H to L. Our definitions are intended for synchronous systems, in which the agents share a clock and are able to make deductions based on the time. (Much of the prior literature has concentrated on asynchronous systems, in which an agent may not know how many actions another agent has performed.) The first definition we consider states that L should not be able to infer H actions from its view.

Definition 1. *A synchronous machine M satisfies Non-Deducibility on Inputs ($M \in \mathtt{NDI}$) if for every possible L view v in M and every sequence of H actions $\alpha \in A_H^*$ with $|\alpha| = |v|$, there exists a run $r \in \mathcal{R}(M)$ such that $Act_H(r) = \alpha$ and $\mathtt{view}_L(r) = v$.*

Intuitively, in a synchronous system, L always knows how many actions H has performed, since this is always identical to the number of actions that L has itself performed. In particular, if L has made view v, then L knows that H has performed $|v|$ actions. The definition says that the system is secure if this is *all* that L can learn about what sequence of actions H has performed. Whatever L observes is consistent with any sequence of actions by H of this length. More precisely, define $K_L(v)$ for an L view v to be the set of H action sequences $Act_H(r)$ for r a run with $v = \mathtt{view}_L(r)$; this represents what L knows about H's actions in the run. Then $M \in \mathtt{NDI}$ iff for all possible L views v we have $K_L(v) = A_H^{|v|}$.

The definition of $\mathtt{NDI}$ takes the viewpoint that a system is secure if it is not possible for L to make any nontrivial deductions about H behaviour, provided that H does not actively seek to communicate information to L. This is an appropriate definition when H is trusted not to deliberately act so as to communicate information to L, and the context is one where H is equally likely to engage in any of its possible behaviours. In some circumstances, however, $\mathtt{NDI}$ proves to be too weak a notion of security. In particular, this is the case if the attack model against which the system must be secure includes the possibility of Trojan Horses at the H end of the system, which must be prevented from communicating H secrets to L. The following example, due in essence to Wittbold and Johnson [WJ90] shows that it is possible for a system to satisfy $\mathtt{NDI}$, but still allow for L to deduce H information.

Example 1. We present a synchronous machine that satisfies $\mathtt{NDI}$ in Fig. 1. We use the convention in such figures that the observations are shown on a state s in the form of $obs_H(s)/obs_L(s)$. Edges are labelled with joint actions (a, b) where $a \in A_H$ and $b \in A_L$. When a is x this means that there is such an edge for all $a \in A_H$. In this example the action sets are $A_H = \{0, 1\}$, $A_L = \{0\}$. Note that in state s_1 and s_2, L's observation in the next state is determined as the *exclusive-or* of H's current observation and H's action. The system is in $\mathtt{NDI}$ since every H action sequence is compatible with every L view of the same

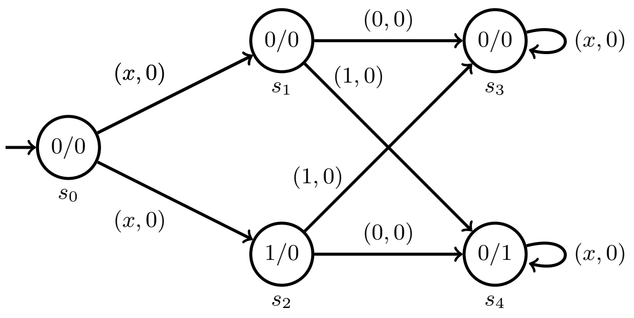

Fig. 1. A synchronous machine in `NDI`, but not in `NDS`, where $x \in \{0,1\}$.

length. For example, the L view 00000 is consistent with H action sequence $x0$ (path $s_0s_1s_3$) and with H action sequence $x1$ (path $s_0s_2s_3$). Nevertheless, H can communicate a bit b of information to L, as follows. Note that H is able to distinguish between state s_1 and s_2 by means of the observation it makes on these states (at time 1). Suppose $b = 1$, then H chooses action 1 at s_1 and action 0 at s_2; in either case the next state is s_4, and L observes 1. Alternately, if $b = 0$, then H chooses action 0 at s_1 and action 1 at s_2; in either case the next state is s_3, and L observes 0. Whatever the value of b, H has guaranteed that L observes b at time 2, so this bit has been communicated. Intuitively, this means that the system is unable to block Trojan Horses at H from communicating with L, even though it satisfies `NDI`. (The structure can be repeated so that H can communicate a message of any length to L in plaintext.) □

The essence of this example is that L is able to deduce H secrets based not just on its knowledge of the system, but also its knowledge that H is following a particular strategy for communication of information to L. In response to this example, Wittbold and Johnson proposed the following stronger definition of security.

First, define an H *strategy* in a system M to be a function π mapping each possible view of H in M to an H action. Intuitively, H's behaviour must depend on what H has been able to observe in the system. Say that a run $r = s_0a_1s_1 \ldots a_ns_n$ is *consistent with* an H strategy π if for all $i = 0 \ldots n-1$, we have $p_H(a_{i+1}) = \pi(\mathtt{view}_H(s_0a_1s_1 \ldots a_is_i))$. We write $\mathcal{R}(M,\pi)$ for the set of runs of M that are consistent with the H strategy π.

Definition 2. *A synchronous system M satisfies* Nondeducibility on Strategies *($M \in$ `NDS`), if for all H strategies π_1, π_2 in M, we have* $\mathtt{view}_L(\mathcal{R}(M,\pi_1)) = \mathtt{view}_L(\mathcal{R}(M,\pi_2))$.

Intuitively, this definition says that the system is secure if L is not able to distinguish between different H strategies by means of its views. In Example 1, H uses a strategy when $b = 1$ that produces the L view 00001 that is not produced when H uses the strategy for $b = 0$. Thus, the sets of L views differ for these two strategies, so the system is not in `NDS`.

An alternate formulation of the definition can be obtained by noting that for every possible L view v, there is an H strategy π such that $v \in \mathtt{view}_L(\mathcal{R}(M, \pi))$, viz., if $v = \mathtt{view}_L(r)$, we take π to be a strategy that always performs the same action at each time $i < |r|$ as H performs at time i in r. Thus, we can state the definition as follows:

Proposition 1. *$M \in \mathtt{NDS}$ iff for all H strategies π in M, we have $\mathtt{view}_L(\mathcal{R}(M, \pi)) = \mathtt{view}_L(\mathcal{R}(M))$.*

This formulation makes clear that H cannot communicate any information to L by means of its strategies. It is also apparent that allowing H strategies to be nondeterministic (i.e., functions from H views to a *set* of H actions) would not lead to a different definition of NDS, since the more choices H has in a strategy the more L-views are compatible with that strategy. We remark that in asynchronous systems (in which we use an asynchronous notion of view), similarly defined notions of non-deducibility on inputs and non-deducibility on strategies turn out to be equivalent [FG95a,vdMZ06]. The example above shows that this is not the case in synchronous machines, where the two notions are distinct.

Nondeducibility-based definitions of security are quite intuitive, but they turn out to have some disadvantages as a basis for secure systems development. One is that they are not compositional: combining two systems, each secure according to such a definition, can produce a compound system that is not secure [McC88]. For this reason, some stronger, but less intuitive definitions have been advocated in the literature.

One of these, McCullough's notion of restrictiveness [McC88], is closely related to an approach to formal proof of systems security based on what are known as "unwinding relations." A variety of definitions of unwinding relations have been proposed in the literature [GM84,Rus92,Man00b,BFPR03], in the context of a number of different underlying systems models and associated definitions of security for which they are intended to provide a proof technique. We propose here a variant of such definitions that is appropriate to the machine model we consider in this paper, drawing on definitions proposed by van der Meyden and Zhang [vdMZ08] for machines acting under the control of a scheduler.

A *synchronous unwinding relation* on a system M is a symmetric relation $\sim \subseteq S \times S$ satisfying the following:

- $s_0 \sim s_0$,
- $s \sim t$ implies $obs_L(s) = obs_L(t)$.
- $s \sim t$ implies for all $a_1, a_2 \in A_H$ and $a_3 \in A_L$, if $s \xrightarrow{(a_1,a_3)} s'$ then there exists a state t' such that $t \xrightarrow{(a_2,a_3)} t'$, and $s' \sim t'$.

Intuitively, an unwinding relation is a bisimulation-like relation over S that shows L observations are locally uncorrelated with H actions.

Definition 3. *A synchronous machine M satisfies restrictiveness ($M \in \mathtt{RES}$), if there exists a synchronous unwinding relation on M.*

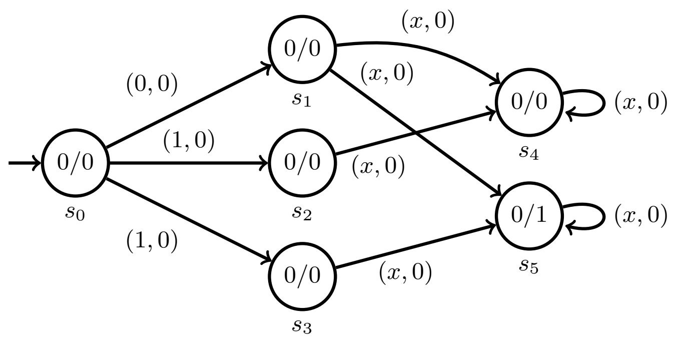

Fig. 2. A synchronous machine M in NDS, but not in RES, where $x \in \{0, 1\}$. Every state s is labelled with a pair $obs_H(s)/obs_L(s)$.

Part of the significance of RES is that it provides a proof technique for our notions of nondeducibility, as shown by the following result, which relates the three notions of security we have introduced:

Theorem 1. *The following containments hold and are strict:* RES $\subset$ NDS $\subset$ NDI.

Example 2. We present a machine in fig. 2 that satisfies NDS but does not satisfy RES. In this system we let $A_H = \{0, 1\}$, $A_L = \{0\}$. We use the conventions from Example 1. One may easily observe that L's view is in the pattern of $000((00)^* + (01)^*)$ all of which are compatible with every possible H strategy. However, there does not exist a synchronous unwinding relation to relate s_0 to s_0. Suppose there were such a relation $\sim$ such that $s_0 \sim s_0$, then for joint actions $(0, 0)$ and $(1, 0)$, we have $s_0 \xrightarrow{(0,0)} s_1$, $s_0 \xrightarrow{(1,0)} s_2$ and $s_0 \xrightarrow{(1,0)} s_3$, and we would require s_1 to be related to either s_2 or s_3. However, neither s_2 nor s_3 can be related to s_1: from s_2 user L can only observe $(00)^*$ in the future, and from s_3 only $(01)^*$ can be observed by L. Note from s_1 both $(00)^*$ and $(01)^*$ are possible for L. □

Our main contribution in this paper is to characterise the complexity of checking the three notions of security we have introduced above. The results are given in the following theorem:

Theorem 2. *Restricted to finite state synchronous machines, and with respect to PTIME reductions,*

1. NDI *is PSPACE-complete,*
2. NDS *is EXPSPACE-complete, and*
3. RES *is in PTIME.*

We remark that the lower bound results for NDI and NDS require only scheduled machines, so these problems are already PSPACE-hard and EXPSPACE-hard, respectively, on this subclass. We describe the proof of these three claims in the following three sections, in turn.

3 Synchronous Nondeducibility on Inputs

In this section we establish the complexity of NDI, Theorem 2(1).

3.1 NDI: Upper bound

Stating the definition in the negative, a system is not in NDI if there exists an L view v and a sequence of H actions α with $|\alpha| = |v|$ such that there exists no run r with $Act_H(r) = \alpha$ and $\mathtt{view}_L(r) = v$. We show that NDI is decidable by a procedure that searches for such an L view v and H action sequence α. The key element of the proof is to show that we need to maintain only a limited amount of information during this search, so that we can bound the length of the witness (v, α), and the amount of space needed to show that such a witness exists.

To show this, given a sequence $\alpha \in A_H^*$ and a possible L view v, with $|\alpha| = |v|$, we define the set $K(\alpha, v)$ to be the set of all final states of runs r such that $Act_H(r) = \alpha$ and $\mathtt{view}_L(r) = v$. For the system M define the labelled transition system $L(M) = (Q, q_0, \Rightarrow)$ as follows:

1. $Q = S \times \mathcal{P}(S)$,
2. $q_0 = (s_0, \{s_0\})$,
3. $\Rightarrow$ is the labelled transition relation on Q with labels in $A_H \times A_L \times A_H$, defined by $(s, T) \Rightarrow^{(a,b,a')} (s', T')$ if $a \in A_H$, $b \in A_L$, $a' \in A_H$ such that $s \xrightarrow{(a,b)} s'$ and $T' = \{t' \in S \mid \text{for some } t \in T \text{ we have } t \xrightarrow{(a',b)} t' \text{ and } obs_L(t') = obs_L(s')\}$.

Intuitively, the component s in a state $(s, T) \in Q$ is used to ensure that we generate an L view v that is in fact possible. The components a, b in a transition $(s, T) \Rightarrow^{(a,b,a')} (s', T')$ represent the actions used to generate the run underlying v, and the component a' is used to generate a sequence α. The set T represents $K(\alpha, v)$. More precisely, we have the following result:

Lemma 1. *If $q_0 \Rightarrow^{(a_1,b_1,a_1')} (s_1, T_1) \Rightarrow \cdots \Rightarrow^{(a_n,b_n,a_n')} (s_n, T_n)$, then the sequence $v = obs_L(s_0) b_1 obs_L(s_1) \ldots b_n obs_L(s_n)$ is a possible L view, and $\alpha = a_1' \ldots a_n'$ is a sequence of H actions such that $|v| = |\alpha|$ and $K(\alpha, v) = T_n$.*

Conversely, for every possible L view v with $|v| = n$, and sequence of H actions $\alpha = a_1' \ldots a_n'$, there exists a path $q_0 \Rightarrow^{(a_1,b_1,a_1')} (s_1, T_1) \Rightarrow \cdots \Rightarrow^{(a_n,b_n,a_n')} (s_n, T_n)$ such that $v = obs_L(s_0) b_1 obs_L(s_1) \ldots b_n obs_L(s_n)$ and $K(\alpha, v) = T_n$.

We now note that for an H action sequence α and a possible L view v, with $|v| = |\alpha|$, there exists no run r such that $Act_H(r) = \alpha$ and $\mathtt{view}_L(r) = v$ iff $K(\alpha, v) = \emptyset$. The existence of such a pair (α, v), is therefore equivalent, by Lemma 1, to the existence of a path in $L(M)$ from q_0 to a state (s, T) with $T = \emptyset$. This can be decided in $NSPACE(O(|M|)) = SPACE(O(|M|^2)) \subseteq PSPACE$. This proves the following theorem.

Theorem 3. *$M \in$ NDI is decidable in $PSPACE$.*

We note, moreover, that since there are at most $|S| \times 2^{|S|}$ states in Q, if there exists a pair (α, v) witnessing that $M \notin$ NDI there exists such a pair with $|\alpha| \leq |S| \times 2^{|S|}$.

3.2 NDI: Lower bound

We show that NDI is PSPACE-hard already in the special case of scheduled machines. The proof is by a polynomial time reduction from the problem of deciding if the language $L(\mathcal{A})$ accepted by a nondeterministic finite state automaton $\mathcal{A}$ on alphabet Σ is equal to $\Sigma^* \setminus \{\epsilon\}$. This is easily seen to be a PSPACE-hard problem, since testing $L(\mathcal{A}) = \Sigma^*$ is known to be PSPACE-hard [SM73].

Let $\mathcal{A} = \langle Q, Q_0, \Sigma, \delta, F\rangle$ be a nondeterministic finite state automaton (without ϵ-transitions), with states Q, initial states $Q_0 \subseteq Q$, alphabet Σ, transition function $\delta : Q \times \Sigma \to \mathcal{P}(Q)$, and final states F. Without loss of generality, we assume $Q_0 \cap F = \emptyset$.[5] We define $M(\mathcal{A}) = \langle S, A, s_0, \to, obs, O\rangle$ to be a scheduled machine, and use a function $sched : S \to \{H, L\}$ to indicate the agent (if any) whose actions determine transitions. In view of this, when $sched(s) = u$ and $a \in A_u$, we may write $s \xrightarrow{a} t$ to represent that $s \xrightarrow{b} t$ for all joint actions b with $p_u(b) = a$. The components of $M(A)$ are defined as follows.

- $S = Q \cup \{s_0, s_1, s_2, s_3\}$, where $Q \cap \{s_0, s_1, s_2, s_3\} = \emptyset$,
- $sched(s_0) = H$ and $sched(s) = L$ for all $s \in S \setminus \{s_0\}$,
- $A = A_H \cup A_L$ where $A_L = \Sigma$ and $A_H = \{h, h'\}$,
- $O = \{0, 1\}$,
- $obs : \{H, L\} \times S \to O$ with $obs_H(s) = 0$ for all $s \in S$ and $obs_L(s) = 0$ for all $s \in S \setminus \{s_2\}$, and $obs_L(s_2) = 1$.
- $\longrightarrow \subseteq S \times A \times S$ is defined as consisting of the following transitions (using the convention noted above)
  - $s_0 \xrightarrow{h} q$ for all $q \in Q_0$, and $s_0 \xrightarrow{h'} s_1$.
  - $s_1 \xrightarrow{a} s_1$ and $s_1 \xrightarrow{a} s_2$ for all $a \in \Sigma$,
  - $s_2 \xrightarrow{a} s_2$ for all $a \in \Sigma$,
  - $s_3 \xrightarrow{a} s_3$ for all $a \in \Sigma$,
  - for $q, q' \in Q$ and $a \in A_L = \Sigma$ we have $q \xrightarrow{a} q'$ for all $q' \in \delta(q, a)$
  - for $q \in Q$ and $a \in A_L = \Sigma$ such that $\delta(q, a) \cap F \neq \emptyset$, we have $q \xrightarrow{a} s_2$,
  - for $q \in Q$ and $a \in A_L = \Sigma$ such that $\delta(q, a) = \emptyset$, we have $q \xrightarrow{a} s_3$.

The construction of $M(\mathcal{A})$ from $\mathcal{A}$ can be done in polynomial time.

Proposition 2. *$L(\mathcal{A}) = \Sigma^* \setminus \{\varepsilon\}$ iff $M(\mathcal{A}) \in$ NDI.*

Proof. Intuitively, the runs of $M(\mathcal{A})$ produce two sets of L views. Runs in which H does h' in the first step produce all L views in $0\Sigma0(\Sigma0)^*(\Sigma1)^*$. Runs in which H does h in the first step correspond to simulations of $\mathcal{A}$ and produce L views in $0\Sigma0(\Sigma0)^*$ or of the form $0\Sigma0a_10\ldots a_{n-1}0a_n1(\Sigma1)^*$ with $a_1\ldots a_n \in L(\mathcal{A})$. Note that L may obtain any view in $0\Sigma0(\Sigma0)^*$ by means of a run that stays in Q for as long as possible, and moves to s_3 whenever an action is not enabled. Views in $0\Sigma0a_10\ldots a_{n-1}0a_n1(\Sigma1)^*$ come from runs that pass through Q and then jump to s_2. Note that since H is not scheduled after the first step, replacing

[5] This is just to let $\varepsilon \notin L(\mathcal{A})$. If not, we apply unfolding on the initial states, and study the resulting automaton which accepts the language $L(\mathcal{A}) \setminus \{\varepsilon\}$.

any action by H after the first step in a run by any other action of H results in another run, with no change to the L view. Thus the only thing that needs to be checked to determine whether $M(\mathcal{A}) \in \texttt{NDI}$ is whether the same can be said for the first step.

- For the 'only if' part, suppose $L(\mathcal{A}) = \Sigma^* \setminus \{\varepsilon\}$. We show that $M(\mathcal{A}) \in \texttt{NDI}$. Let $r = s_0(b_1, a_1)t_1 \ldots (b_n, a_n)t_n$ be a run of $M(\mathcal{A})$, with the $b_i \in A_H$ and the $a_i \in A_L$. Let $b'_1 \ldots b'_n$ be any sequence of actions in A_H. If $b'_1 = h'$, then it is clear that we can find a run $r = s_0(b'_1, a_1)t'_1 \ldots (b'_n, a_n)t'_n$ with $\texttt{view}_L(r) = \texttt{view}_L(r')$. On the other hand, if $b'_1 = h$ then the same is true, since $L(\mathcal{A}) = \Sigma^* \setminus \{\varepsilon\}$. Thus, the run required by $M(\mathcal{A}) \in \texttt{NDI}$ has been shown to exist.
- For the 'if' part, suppose there is a word $w = a_1a_2 \ldots a_n \notin L(\mathcal{A})$. Then for an arbitrary $a_0 \in \Sigma$, the L view $0a_00a_10a_2 \ldots a_n1$ cannot be obtained from runs in which the first H action is h, because otherwise w would be accepted by $\mathcal{A}$. However this view is obtained from a run in which the first action is h'. Therefore $M(\mathcal{A}) \notin \texttt{NDI}$. $\square$

4 Nondeducibility on Strategies

In this section we establish the complexity of NDS, Theorem 2(2).

4.1 NDS: upper bound

For the proof that NDS is decidable in EXPSPACE, we show that the problem is in DSPACE($2^{O(n)}$).

We use characterization of NDS given in Proposition 1. Let π be an H strategy, let α be an H view, and let β be an L view, with $|\alpha| \leq |\beta|$. Say that π *excludes* β if there does not exist a run r consistent with π such that $\beta = \texttt{view}_L(r)$. Since always $\mathcal{R}(M, \pi) \subseteq \mathcal{R}(M)$, by Proposition 1, a system M satisfies NDS if it is not the case that there exists a possible L view β in M and a strategy π such that π *excludes* β. We first give a lemma that enables strategies excluding a particular L view to be put into a uniform structure.

Given an H view $\alpha \in \texttt{view}_H(\mathcal{R}(M, \pi))$, define $K(\alpha, \pi, \beta)$ to be the set of all final states of runs r consistent with π such that $\texttt{view}_H(r) = \alpha$ and $\texttt{view}_L(r)$ is a prefix of β.

Lemma 2. *If there exists a strategy π that excludes β, then there exists a strategy π' that also excludes β, and has the property that $K(\alpha, \pi', \beta) = K(\alpha', \pi', \beta)$ and $|\alpha| = |\alpha'|$ implies $\pi'(\alpha) = \pi'(\alpha')$ for all H views α and α'.*

Proof. Suppose that π excludes β. For purposes of the proof, note that we can assume without loss generality that β is infinite — this helps to avoid mention of views longer than β as a separate case.[6] It is convenient to consider K

[6] Note that it is equivalent to say that π excludes some prefix of β.

and strategies π to be defined over the larger set $P = O(A_H O)^*$ rather than $\mathtt{view}_H(\mathcal{R}(M, \pi))$. In case of K, we take $K(\alpha, \pi, \beta) = \emptyset$ when there is no run r consistent with π such that $\mathtt{view}_H(r) = \alpha$ and $\mathtt{view}_L(r)$ is a prefix of β.

Let f be any mapping from P to P such that for all $\alpha, \alpha' \in P$ we have (1) $|f(\alpha)| = |\alpha|$ and (2) $K(\alpha, \pi, \beta) = K(f(\alpha), \pi, \beta)$, and (3) if $|\alpha| = |\alpha'|$ and $K(\alpha, \pi, \beta) = K(\alpha', \pi, \beta)$, then $f(\alpha) = f(\alpha')$. Such a mapping always exists; intuitively, it merely picks, at each length, a representative $f(\alpha) \in [\alpha]_\sim$ of the equivalence classes of the equivalence relation defined by $\alpha \sim \alpha'$ if $K(\alpha, \pi, \beta) = K(\alpha', \pi, \beta)$.

Now define the mapping g on P as follows. Let $\alpha_0 = O_H(s_0)$ be the only possible H view of length 0. For $\alpha \in P$ of length 0, we define $g(\alpha) = \alpha_0$ if $\alpha = \alpha_0$ and $g(\alpha) = \alpha$ otherwise. For longer α, we define $g(\alpha a o) = f(g(\alpha))\pi(f(g(\alpha))o$. Also, define the strategy π' by $\pi'(\alpha) = \pi(f(g(\alpha)))$.

We claim that for all $\alpha \in P$ we have $K(\alpha, \pi', \beta) = K(g(\alpha), \pi, \beta)$. The proof is by induction on the length of α. The base case is straightforward, since α_0 is consistent with all strategies, so $K(\alpha_0, \pi', \beta) = \{s_0\} = K(\alpha_0, \pi, \beta)$, and $K(\alpha, \pi', \beta) = \emptyset = K(\alpha, \pi, \beta)$ for $\alpha \neq \alpha_0$. Suppose the claim holds for $\alpha \in P$ of length i. Let $\alpha a o \in P$. By induction and (2), $K(\alpha, \pi', \beta) = K(g(\alpha), \pi, \beta) = K(f(g(\alpha)), \pi, \beta)$. Since action $a = \pi'(\alpha) = \pi(f(g(\alpha))$, $K(\alpha a o, \pi', \beta)$ is equal to $K(f(g(\alpha))ao, \pi, \beta) = K(g(\alpha a o), \pi, \beta)$, as required.

To see that π' has the required property, if $K(\alpha, \pi', \beta) = K(\alpha', \pi', \beta)$ with $|\alpha| = |\alpha'|$, then we have $K(g(\alpha), \pi, \beta) = K(g(\alpha'), \pi, \beta)$. By (3) we have $f(g(\alpha)) = f(g(\alpha'))$. Therefore $\pi'(\alpha) = \pi(f(g(\alpha))) = \pi(f(g(\alpha'))) = \pi'(\alpha')$ by definition.

Since π excludes β, there exists a length n such that for all $\alpha \in P$ with $|\alpha| = n$, we have $K(\alpha, \pi, \beta) = \emptyset$. Thus, we also have for all α of length n that $K(\alpha, \pi', \beta) = K(g(\alpha), \pi, \beta) = \emptyset$. This means that π' also excludes β. □

Based on Lemma 2, we construct a transition system $(Q, q_0 \Rightarrow)$ that simultaneously searches for the strategy π and an L view β that is omitted by π. The states Q are sets of sets $k \subseteq S$. The initial state q_0 is $\{\{s_0\}\}$. We use an "update" function $\delta_{a_L, o_L, a_H, o_H} : \mathcal{P}(S) \to \mathcal{P}(S)$, for each $a_L \in A_L$, $a_H \in A_H$, and $o_L, o_H \in O$, defined by

$$\delta_{a_L,o_L,a_H,o_H}(k) = \{t \in S \mid \text{there exists } s \in k \text{ with } s \xrightarrow{(a_L,a_H)} t \text{ and } \\ obs_H(t) = o_H \text{ and } obs_L(t) = o_L\}.$$

The transitions are defined as follows: $q \Rightarrow^{(\rho, a_L, o_L)} q'$ if $\rho : q \to A_H$, $a_L \in A_L$ and $o_L \in O$, and $q' = \{\delta_{a_L,o_L,a_H,o_H}(k) \mid k \in q \text{ and } a_H = \rho(k) \text{ and } o_H \in O\}$. Intuitively, each state q represents a collection of all possible knowledge sets that H can be in at a certain point of time, while attempting to omit some sequence β. More specifically, each set k in $q \in Q$ corresponds to an H view α such that $k = K(\alpha, \pi, \beta)$. In a transition, we both determine the next phase of π, by extending π so that $\pi(\alpha) = \rho(K(\alpha, \pi, \beta))$, and extend β to $\beta a_L o_L$.

The following results justify the correspondence between this transition system and NDS.

Lemma 3. *There exists a strategy π and a Low view β such that π excludes β, iff there exists a path $q_0 \Rightarrow^* q_n = \{\emptyset\}$.*

We obtain the claimed complexity bound from Lemma 3, simply by noting that it reduces `NDS` to a reachability problem in the transition system. Since the states of the system can be represented in space $|S| \cdot 2^{|S|} = 2^{O(|S|)}$, we obtain from Savitch's theorem that we can do the search in DSPACE($2^{O(|S|)}$).

4.2 `NDS`: Lower Bound

To show that `NDS` is EXPSPACE-hard, we show how to encode the game BLIND-PEEK of Reif [Rei84]. We need only scheduled machines for the encoding, so the problem is EXPSPACE-hard already for this subclass.

BLIND-PEEK is a variant of the two-player game PEEK introduced by Stockmeyer and Chandra [SC79]. A PEEK game consists of a box with two open sides and containing horizontally stacked plates; the players sit at opposite sides of the box. Each plate has two positions, 'in' and 'out', and contains a knob at one side of the box, so that this plate can be controlled by one of the players. At each step, one of the two players may grasp a knob from his side and push it 'in' or 'out'. The player may also pass. Both the top of the box and the plates have holes in various positions, and each hole is associated to a player. If, just after a move of player $a \in \{1, 2\}$, the plates are positioned so that for one of the player's holes, each plate has a hole positioned directly underneath that hole, so that the player can peek through a sequence of holes from the top of the box to the bottom, then player a wins. In PEEK, both players can observe the position of all plates at all times. BLIND-PEEK [Rei84] is a modification of PEEK in which player 1's side of the box is partially covered, so that it is not possible for player 1 to see the positions of the plates controlled by player 2. Deciding whether there exists a winning strategy for player 1 in a PEEK game is EXPTIME-hard, and it is EXPSPACE-hard in the case of BLIND-PEEK. We make a reduction from a BLIND-PEEK game to achieve the lower bound result for `NDS`.

Due to space limitations, we just give a brief sketch of the construction. Given an instance G of BLIND-PEEK, we construct a synchronous system $M(G)$, with the following property: player 1 has a winning strategy in G iff there exists an L view v_L and an H strategy π that excludes v_L in $M(G)$. Note that since player 1 plays blindfold in G, a player 1 strategy can be represented as simply a sequence of player 1 moves, rather than a function from player 1 views to player 1 moves. This sequence of player 1 moves will be encoded in the L view v_L. Nondeterminism will be used to represent the universal behaviour of player 2, and also to guess certain aspects of game state transitions. The role of the H strategy π in the encoding will be to perform certain checking operations. We make use of the sets $K(v_H, \pi, v_L)$ to represent states of G. H will observe all actions of H and L in the game, so H is always aware of the game state. However, there remains some uncertainty in H's knowledge of the state of the system $M(G)$ - we use this to represent the positions of the n plates in the game as a set of n states of the system $M(G)$.

5 Synchronous Bisimulation-based Notions

In this section we establish the complexity of `RES`, Theorem 2(3). We first note:

Lemma 4. *1. The largest synchronous unwinding relation is transitive.*
2. If all states in M are reachable then the largest synchronous unwinding relation $\sim$ is an equivalence relation.
3. A system satisfies `RES` *iff its restriction to its reachable states satisfies* `RES`*.*

It is not hard to show that finding the largest synchronous unwinding relation (if any exists) on the reachable fragment of a machine $\langle S, A, s_0, \rightarrow, O, obs\rangle$ can be done in polynomial time. The following algorithm, which works in $\mathcal{O}(|S|^3 \times |{\rightarrow}|)$ time, resembles the algorithm for calculating the relational coarsest partition by Kanellakis and Smolka [KS83].

Algorithm 1 *Let $S_o = \{s \in S \mid obs_L(s) = o\}$, and the initial partion $P_0 = \{S_o \mid o \in O\} \cup p_{bad}$ with $p_{bad} = \emptyset$. We repeat the following for all $i \geq 0$ until termination. Try every $p_1, p_2 \in P_i$ in the following transformation rules:*

1. *If there exist $s \in p_1$, $a_1, a_2 \in A_H$ and $a_3 \in A_L$ such that (1) there exist $s \xrightarrow{(a_1,a_3)} t_1$ and $t_1 \in p_2$, and (2) $s \xrightarrow{(a_2,a_3)} t_2$ implies $t_2 \notin p_2$, then $P_{i+1} = P_i \setminus \{p_1, p_{bad}\} \cup \{p_1 \setminus \{s\}, p_{bad} \cup \{s\}\}$.*
2. *If there exist $a_1 \in A_H$ and $a_2 \in A_L$, and we can split p_1 into nonempty sets p_{11} and p_{12} such that (1) for all $s_1 \in p_{11}$, $\{t' \in S \mid s_1 \xrightarrow{(a_1,a_2)} t'\} \cap p_2 \neq \emptyset$ and (2) for all $s_2 \in p_{12}$, $\{t' \in S \mid s_2 \xrightarrow{(a_1,a_2)} t'\} \cap p_2 = \emptyset$, then we let $P_{i+1} = P_i \setminus \{p_1\} \cup \{p_{11}, p_{12}\}$.*

When neither transformation rule applies, $P_{i+1} = P_i$, and we return P_i.

The above algorithm produces the coarsest partition over S according to the definition of synchronous unwinding, which yields a relation that is not necessarily reflexive. If (s_0, s_0) is within that relation then the system is in `RES`. In practice, whether or not (s_0, s_0) is still within a 'good' partition can be checked on-the-fly: note that once (s_0, s_0) is moved into p_{bad}, the algorithm can immediately return false, indicating that the system is not in `RES`.

6 Related Work

In asynchronous machines the verification complexities of NDI and NDS are both PSPACE-complete, and RES (based on asynchronous unwinding) is in polynomial time [FG95b,FG96,vdMZ07]. Interestingly, PSPACE is also the complexity result for verifying Mantel's BSP conditions [Man00a] on asynchronous finite state systems. For (asynchronous) push-down systems, the verification problem is undecidable [DHK$^+$08].

A number of works have defined notions of security for synchronous or timed systems, but fewer complexity results are known. Köpf and Basin [KB06] define a notion similar to RES and show it is PTIME decidable. Similar definitions are also used in the literature on language-based security [Aga00,VS97].

Focardi et al [FGM00] define a spectrum of definitions related to ours in a timed process algebraic setting, and state a decidability result for one of them, close to our notion tNDS. However, this result concerns an approximation to the notion tBNDC that is their real target, and they do not give a complexity result. Beauquier and Lanotte defined *covert channels* in timed systems with *tick* transitions by using strategies [BL06]. They prove that the problem of the existence of a covert channel in such systems is decidable. However, their definition of covert channel requires that H and L have strategies to enforce a system into sets of runs with projections into disjoint sets of L views. Intuitively, the induced definition on *free of covert channels* turns out to be a weaker notion than `NDS`.

7 Conclusion

We remarked above that nondeducibility-based notions of security may have the disadvantage that they do not readily support a compositional approach to secure systems development, motivating the introduction of unwinding-based definitions of security. The complexity results of the present paper can be interpeted as lending further support to the value of unwinding-based definitions. We have found that the two nondeducibility notions we have considered, while both decidable, are intractable. On the other hand, the unwinding-based notion of synchronous restrictiveness has tractable complexity. This makes this definition a more appropriate basis for automated verification of security. Even if the desired security property is nondeducibility on inputs or nondeducibility on strategies, it is sufficient to verify that a system satisfies synchronous restrictiveness, since this is a stronger notion of security. It remains to be seen whether there is a significant number of practical systems that are secure according to the nondeducibility-based notions, but for which there does not exist a synchronous unwinding. If so, then an alternate methodology needs to be applied for the verification of security for such systems.

References

[Aga00] J. Agat. Transforming out timing leaks. In *POPL*, pages 40–53, 2000.

[BFPR03] A. Bossi, R. Focardi, C. Piazza, and S. Rossi. Bisimulation and unwinding for verifying possibilistic security properties. In *Proc. Int. Conf. on Verication, Model Checking, and Abstract Interpretation*, pages 223–237, 2003.

[BL06] D. Beauquier and R. Lanotte. Hiding information in multi level security systems. In *Proc. 4th Int. Workshop on Formal Aspect in Security and Trust (LNCS 4691)*, pages 250–269, 2006.

[DHK+08] D. D'Souza, R. Holla, J. Kulkarni, R. K. Ramesh, and B. Sprick. On the decidability of model-checking information flow properties. In *Proc. Int. Conf. on Information Systems Security*, pages 26–40, 2008.

[FG95a] R. Focardi and R. Gorrieri. A classification of security properties for process algebras. In *Journal of Computer Security*, 1, pages 5–33. IOS Press, 1995.

[FG95b] R. Focardi and R. Gorrieri. A classification of security properties for process algebras. In *Journal of Computer Security*, 1, pages 5–33. IOS Press, 1995.

[FG96] R. Focardi and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. Technical Report UBLCS-96-14, Università di Bologna, August 1996.

[FGM00] R. Focardi, R. Gorrieri, and F. Martinelli. Information flow analysis in a discrete-time process algebra. In *CSFW*, pages 170–184, 2000.

[GM84] J. A. Goguen and J. Meseguer. Unwinding and inference control. In *Proc. IEEE Symp. on Security and Privacy*, page 75, 1984.

[KB06] B. Köpf and D. A. Basin. Timing-sensitive information flow analysis for synchronous systems. In *ESORICS*, volume 4189 of *Lecture Notes in Computer Science*, pages 243–262. Springer, 2006.

[Koc96] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KS83] P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. In *Proc. 2nd Annual ACM Symp. on Principles of Distributed Computing*, pages 228–240, New York, NY, 1983.

[Man00a] H. Mantel. Possiblistic definitions of security – an assembly kit. In *Proc. Computer Security Foundations Workshop*, pages 185–199, 2000.

[Man00b] H. Mantel. Unwinding security properties. In *Proc. European Symp. on Research in Computer Security (LNCS 1895)*, pages 238–254, 2000.

[McC88] D. McCullough. Noninterference and the composability of security properties. In *Proc. IEEE Symp. on Security and Privacy*, pages 177–186, 1988.

[Rei84] J. H. Reif. The complexity of two-player games of incomplete information. *Journal of Computer and System Science*, 29(2):274–301, 1984.

[Rus92] J. Rushby. Noninterference, transitivity, and channel-control security policies. Technical report, SRI international, Dec 1992.

[SC79] L. J. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. *SIAM Journal of Computing*, 8(2):151–174, 1979.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time (preliminary report). In *Proc. ACM symp. on Theory of computing*, pages 1–9, 1973.

[vdMZ06] R. van der Meyden and C. Zhang. A comparison of semantic models for noninterference. In *Proc. 4th Int. Workshop on Formal Aspect in Security and Trust (LNCS 4691)*, pages 235–249, 2006.

[vdMZ07] R. van der Meyden and C. Zhang. Algorithmic verification on noninterference properties. In *ENTCS*, volume 168, pages 61–75, 2007.

[vdMZ08] R. van der Meyden and C. Zhang. Information flow in systems with schedulers. In *Proc. Computer Security Foundation Symp.*, pages 301–312, June 2008.

[VS97] D. M. Volpano and G. Smith. A type-based approach to program security. In *TAPSOFT*, volume 1214 of *Lecture Notes in Computer Science*, pages 607–621. Springer, 1997.

[WJ90] J. T. Wittbold and D. M. Johnson. Information flow in nondeterministic systems. In *Proc. IEEE Symp. on Security and Privacy*, pages 144–161, 1990.