Note on Opacity with powerful attackers

Loïc Hélouët [*] **Hervé Marchand** [*] **Laurie Ricker** [**]

[*] *INRIA Rennes (e-mail: surname.name@inria.fr).*
[**] *Mount Allison University, (e-mail: lricker@mta.ca)*

Abstract: This paper considers state-based opacity in a setting where attackers of a secret have additional observation capacities allowing them to know which inputs are allowed by a system. We show that regular opacity is decidable in this setting. We then address the question of controlling a system so that it becomes opaque, and solve this question by recasting the problem in a game setting.

1. INTRODUCTION

Language-based security properties, such as non-interference or opacity, are a way to guarantee that a system fulfills some privacy requirements. The original work of (Goguen and Meseguer, 1982) is motivated by security issues: given a set of confidential actions performed only by users with high credentials, other users should not be able to infer from their observation whether a "high" action has occurred or not. The formal definition of non-interference states that occurrences of high actions should not affect what other users can *see or do*. Implicitly, this supposes that users have *capacities* that allow them to check whether some actions are feasible, and can also be used to gain information about the current state of the system.

Non-interference is a limited class of properties: it only addresses questions of leakage from a high to a low level and, in the case of language-based interference, on the occurrence of high actions. However, the properties to protect are not always definable in terms of occurrences of actions from a given subset. A larger setting called *opacity* has been proposed (Bryans et al., 2005) to consider hiding more general properties of runs. This notion formalizes the absence of information flow or, more precisely, the impossibility for an attacker to infer the truth of a predicate ϕ (it could be the occurrence in the system of some particular sequences of events, or the fact that the system is in some particular secret configuration). If the truth of this predicate can be inferred by an attacker, based on its knowledge of the system and the information deduced from its observations, then there is an information flow and the secret is revealed.

This paper considers state-based opacity in a setting where attackers partially observe a system and, in addition to their observations, can query the system to discover the accepted inputs at the current state. In some sense, giving this additional power to attackers reintroduces the notion of capacities from (Goguen and Meseguer, 1982). We provide several different models of attackers, ranging from the usual attackers that can only observe a system and try to determine whether the system is currently in a sensitive secret state, to very powerful attackers that know at every instant the possible inputs accepted by the system. We then show how state-based opacity with powerful attackers can be checked. Lastly, we consider opacity enforcement: We bring back this question to the computation of a controller that forbids some inputs to the system to prevent an information flow. We show that opacity enforcement can be modeled as a game in which the controller tries to prevent an information flow, while the rest of the system may systematically perform the worst choices, allowing the attacker to gain information.

This paper is organized as follows: Section 2 introduces notations used throughout the paper, Section 3 gives several models of powerful attackers. Section 4 considers enforcement of opacity for passive attackers, and Section 5 opacity enforcement with active attackers, before conclusion. Proofs of theorems and propositions are provided in appendix.

2. PRELIMINARIES

Let Σ be an alphabet. A *word* $w = a_1 \ldots a_n$ is an element of Σ^*. We denote by ϵ the empty word. Given a subset $\Sigma' \subseteq \Sigma$ and a word $w = a_1 \ldots a_n \in \Sigma^*$, the projection of w onto Σ' is the word $P_{\Sigma'}(w)$ obtained by erasing from w all letters from $\Sigma \setminus \Sigma'$. Similarly, we denote by $P_{\Sigma'}^{-1}(\mu)$ the *inverse projection* of μ, i.e., $P_{\Sigma'}^{-1}(\mu) = \{w \in \Sigma^* \mid P_{\Sigma'}(w) = \mu\}$.

Definition 1. A *Labelled Transition System* G is a tuple (Q, q_o, Σ, δ) where Q is a set of states with a distinguished element q_o called *the initial state*, Σ is the set of *events* of G, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation.

The *runs* of G are sequences of transitions of the form $\rho = q_0 \xrightarrow{a_1} q_1 \ldots \xrightarrow{a_n} q_n$ such that $(q_i, a_{i+1}, q_{i+1}) \in \delta$ for every $i \in 0 \ldots n-1$. The word associated with a run ρ is the sequence of letters $w = a_1 \ldots a_n$ (we also say that w labels run ρ). We denote by $Run(G)$ the set of runs of G, and by $\mathcal{L}(G)$ the (regular) language of words that label runs of G. Furthermore, we assume that G is deterministic, i.e., $\forall q \in Q, \forall a \in \Sigma, |(q, a, q')| \leq 1$. Note that as G is deterministic, for every word $w \in \mathcal{L}(G)$ there is a unique run ρ labeled by w. Thus, the map $l : Run(G) \rightarrow \mathcal{L}(G)$ that associates runs with their labelings is a bijection.

We assume that Σ is partitioned between $\Sigma_?$, the set of input events, $\Sigma_!$, the set of output events and Σ_I the set of internal events. For a given state $q \in Q, \Gamma(q) = \{a \in \Sigma \mid \exists q', (q, a, q') \in \delta\}$ denotes the active event set at q. Similarly, $\Gamma_?(q)$ and $\Gamma_!(q)$ denotes the set of input (respectively output) events that are admissible at state q. Given a set of states $X \subseteq Q$ and a set of input events

$\Sigma' \subseteq \Sigma_?$, let $\Gamma_?(X, \Sigma') = \{q \in X \mid \Gamma_?(q) = \Sigma'\}$, i.e., the states of X that have exactly Σ' as active input event set.

2.1 General notion of opacity

We consider a setting where a system should not leak any secret information that might be contained in the actual run of the system. Such a secret can be depicted as a predicate ϕ over runs of the system, as proposed in (Bérard et al., 2015). In this paper, we will consider that the secret is represented as a subset of states $S \subseteq Q$. Intuitively, states of S can represent states in which a system is vulnerable to attacks, or states in which some confidential information is in use, and is not yet declassified. We denote by $Run_S(G)$ the set of runs of G ending in a state $q \in S$.

We consider that attackers are agents of the system, that are equipped with observation and capacities. They collect information online with executions of the system and try to infer *with certainty* whether the system's current state belongs to S or not.

Definition 2. (Attackers). An *attacker* of G is a map $\mathcal{A}$: $Run(G) \to ((\Sigma \cup \epsilon) \times O)^*$, where O is a finite alphabet of observed facts obtained through use of capacities. We further assume that attackers can be defined inductively, and from their final transitions: for a run $\rho = q_0 \ldots q_{n-1} \xrightarrow{a_n} q_n$, $\mathcal{A}(\rho) = \mathcal{A}(q_0 \ldots q_{n-1}).obs_{\mathcal{A}}(a_n, q_{n-1})$, where $obs_{\mathcal{A}} : \Sigma \times Q \to (\Sigma \times O) \cup \epsilon$ defines the observation $obs_{\mathcal{A}}(a, q)$ of the attacker when the system fires a from q.

The definition of attackers above is compositional, i.e., if $\rho = \rho_1.\rho_2$ then $\mathcal{A}(\rho) = \mathcal{A}(\rho_1).\mathcal{A}(\rho_2)$, and transition-based, i.e., for a particular transition (q, a, q'), the observation is of the form (a, o) regardless of its position in a run. Note, however, that an attacker is not necessarily aware of every move of the system as $obs_{\mathcal{A}}(a, q) = \epsilon$ for some actions and states, thus $|\mathcal{A}(\rho)| \le |\rho|$. Let μ be an observation of the attacker, then the set of runs compatible with μ w.r.t. $\mathcal{A}$ is the set $[\mu]_{\mathcal{A}} = \{\rho \in Run(G) \mid \mathcal{A}(\rho) = \mu\}$. Similarly, we define the *equivalence class* of $\mu \in \Sigma^*$ w.r.t. $\mathcal{A}$ as the set $[\![\mu]\!]_{\mathcal{A}} = \{w \in \Sigma^* \mid \exists \rho \in [\mu]_{\mathcal{A}}, l(\rho) = w\}$.

Definition 3. (Opacity). A secret S is *opaque* w.r.t. $\mathcal{A}$ iff $\forall \rho \in Run_S(G)$ there exists a run ρ' in $[\mathcal{A}(\rho)]_{\mathcal{A}}$ such that $\rho' \notin Run_S(G)$.

Intuitively, opacity says that an attacker is not able to infer that a run is currently in a secret state from its observation. Definition 2 allows us to decouple information observed from transitions fired during a run, and information acquired from visited states. Though this definition is very generic it encompasses many types of attackers, starting with the standard ones without capacities, and whose observations are limited to a subset of actions $\Sigma_o \subseteq \Sigma$. The role of an attacker is to maintain on-line with its observations, a *belief*, i.e., a set of states in which the system could be. Upon the arrival of a new observation, this belief can be updated as follows:

Definition 4. (Belief update). Let G be a system, $\mathcal{A}$ be an attacker, $X \subseteq Q$ be a belief, and (a, o) be an observation. We denote by $\delta_{\mathcal{A}}(X, (a, o))$ the set of states X' such that there exists a run $\rho_{x,x'}$ from a state $x \in X$ to a state of $x' \in X'$, and $\mathcal{A}(\rho_{x,x'}) = (a, o)$. Belief updating extends to words and we can define a function $\Delta_{\mathcal{A}}$ as: $\Delta_{\mathcal{A}}(X, \epsilon) = X$, $\Delta_{\mathcal{A}}(X, w.(a, o)) = \delta_{\mathcal{A}}(\Delta_{\mathcal{A}}(X, w), (a, o))$.

Definition 5. (Observers). Let $G = (Q, q_o, \Sigma, \delta)$. The observer of an attacker $\mathcal{A}$ is the deterministic automaton $Det_{\mathcal{A}}(G) = (\mathcal{X}, X_0, \Sigma_{\mathcal{A}}, \delta_{\mathcal{A}})$, with $\mathcal{X} = 2^Q$, $X_0 = \{q_0\}$ [1] and $\Sigma_{\mathcal{A}} = \Sigma_o \times O$.

The construction of $\delta_{\mathcal{A}}$ depends on the power given to the attacker. However, with the restrictions given to attackers, $Det_{\mathcal{A}}(G)$ is always a finite-state machine. We denote by $Tr_{\mathcal{A}}(G) = \mathcal{L}(Det_{\mathcal{A}}(G))$, the language generated by $Det_{\mathcal{A}}(G)$. The classical notion of opacity can be generalized to observers with capacities as follows:

Proposition 1. Secret S is *opaque* w.r.t. G and $\mathcal{A}$ iff $\forall \rho \in Run_S(G)$, $\Delta_{\mathcal{A}}(X_0, \mathcal{A}(\rho)) \not\subseteq S$.

Proposition 1 says that a system is opaque if an attacker is not able to infer that a run is currently in a secret state by computing the beliefs obtained by synchronizing its observer automaton with the currently observed run. The verification of opacity is then reduced to the question of reachability of $F = 2^S$ in $Det_{\mathcal{A}}(G)$. This verification is PSPACE-complete (Cassez et al., 2012) in the standard setting where attackers observe only a subset of the labels attached to transitions (defined as attacker $\mathcal{A}_1$ later in this section). As G is deterministic, Proposition 1 can be rephrased as a property of $Det_{\mathcal{A}}(G)$. Furthermore, PSPACE-completeness proof easily extends to the more general setting.

Corollary 1. (of Proposition 1). Secret S is *opaque* w.r.t. G and $\mathcal{A}$ iff $\forall \mu \in Tr_{\mathcal{A}}(G)$, $\Delta_{\mathcal{A}}(X_0, \mu) \not\subseteq S$.

Corollary 2. Opacity of a system w.r.t. an attacker $\mathcal{A}$ is PSPACE-complete.

2.2 Attacker 1: Standard observer

A standard setting in opacity is to assume that the attacker $\mathcal{A}_1$ observes the system through an interface (a subset of its actions). The attacker does not get information from the current state of the system, but is only aware of the last observable event triggered by the system. This can be modeled in our generic setting, by defining a single observed fact $\bullet$ (i.e., $O = \{\bullet\}$) and a function $obs_{\mathcal{A}_1}$:

$$obs_{\mathcal{A}_1}(a, q) = \begin{cases} \epsilon & \text{if } a \notin \Sigma_o, \\ (a, \bullet) & \text{otherwise.} \end{cases}$$

Given a sequence w of G, the observation of the attacker $\mathcal{A}_1$ is given by the natural projection $P_{\Sigma_o}(w)$. We denote by $Tr_{\mathcal{A}_1}(G) = P_{\Sigma_o}(\mathcal{L}(G))$ this set of observations. Given an observation $\mu \in Tr_{\mathcal{A}_1}(G)$, we denote by $[\![\mu]\!]_{\mathcal{A}_1} = P_{\Sigma_o}^{-1}(\mu) \cap \mathcal{L}(G) \cap \Sigma.\Sigma_o^*$ the set of sequences compatible with μ that end with an observable event.

Given an observation $\mu \in Tr_{\mathcal{A}_1}(G)$, we denote by $\Delta_o(X_0, \mu) = \{\delta(q_0, [\![\mu]\!]_{\mathcal{A}_1})\}$, the set of states in which the system can be after the observation of the trace μ, with $X_o = \{q_0\}$. Similarly, for any subset $X \in 2^Q$ and any observable event $a \in \Sigma_o$, $\Delta_o(X, a) = \{q' \in Q \mid \exists q \in X, w \in [\![a]\!]_{\mathcal{A}_1}, q' = \delta(q, w)\}$ denotes the set of states in which the system can be after the observation of a, knowing that the belief of the set of states was $X \in 2^Q$. The secret S is not opaque w.r.t. $\mathcal{A}_1$ whenever there exists $\mu \in Tr_{\mathcal{A}_1}(G)$ such that $\Delta_o(X_o, \mu) \subseteq S$ and the verification is reduced to a reachability problem as described in (Cassez et al., 2012).

---

[1] Thus, the attacker has perfect information of the initial state.

3. ATTACKERS WITH ADDITIONAL CAPACITIES

The attacker $\mathcal{A}_1$ is the standard notion of a passive observer that is only informed of a subset of system's actions. However, this model gives very little discriminating power to attackers. In this section, we propose several models of attackers that can collect additional information along runs of the system.

3.1 Attacker 2: an input-aware attacker

Let us first define an attacker $\mathcal{A}_2$ that is aware of all the inputs events it might executes at any time during a run of the system. We assume that the set of inputs that $\mathcal{A}_2$ can observe at any instant is a set $\Sigma_? \subseteq \Sigma_o$, and that this set of available inputs is returned instantaneously after each move as an answer γ, with $\gamma \subseteq \Sigma_?$. We still consider that some actions remain invisible to the attacker. Hence after the execution of each action a of the system:

- if $a \in \Sigma_o$, the attacker receives a pair (a, γ);
- if $a \in \Sigma_{uo}$, the attacker receives a pair (ϵ, γ).

Note that each time an event is triggered in the system, $\mathcal{A}_2$ is now aware of this occurrence, which can be seen as a "tick" for the attacker whenever the event is not observable. Within this setting, $O_2 = 2^{\Sigma_?}$ and the function $obs_{\mathcal{A}_2}$ is given by

$$obs_{\mathcal{A}_2}(a, q) = \begin{cases} (a, \Gamma_?(\delta(q, a))) & \text{if } a \in \Sigma_o, \\ (\epsilon, \Gamma_?(\delta(q, a))) & \text{otherwise.} \end{cases}$$

and the map $\mathcal{A}_2$ directly follows from the definition of $obs_{\mathcal{A}_2}$. Meanwhile, based on this new notion of an attacker, we redefine the various notions introduced in the previous sections in the classical framework . An *observation* of the attacker is now a sequence from $\left((\Sigma_o \cup \{\epsilon\}) \times 2^{\Sigma_?}\right)^*$.

The set of observations of $\mathcal{A}_2$ that can appear when observing runs of G is denoted by $Tr_{\mathcal{A}_2}(\mathcal{G})$. Given an observable trajectory $\mu \in Tr_{\mathcal{A}_2}(\mathcal{G})$, the set of compatible sequences is given by: $[\![\epsilon]\!]_{\mathcal{A}_2} = \{\epsilon\}$ and

$$[\![\mu.(a, \gamma)]\!]_{\mathcal{A}_2} = \begin{cases} \{w \in [\![\mu]\!]_{\mathcal{A}_2}.a \cap L(G) \mid q_0 \xrightarrow{w} q \wedge \Gamma_?(q) = \gamma\}, \\ \qquad \qquad \qquad \text{if } a \in \Sigma_o; \\ \{w \in [\![\mu]\!]_{\mathcal{A}_2}.\sigma \cap L(G) \mid \sigma \in \Sigma_{uo}, q_0 \xrightarrow{w} q \wedge \\ \qquad \qquad \Gamma_?(q) = \gamma\} \text{ if } a = \epsilon \end{cases}$$

Similar to the case of $\mathcal{A}_1$, the computation of the observer, starting from a subset of states X, can be defined inductively. Upon the arrival of a new observation (a, γ), the update of belief X by attacker $\mathcal{A}_2$ is defined by the function $\delta_{\mathcal{A}_2}(X, (a, \gamma))$:

- $\delta_{\mathcal{A}_2}(X, (\epsilon, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \mid \exists q \in X, a \in \Sigma_{uo}, (q, a, q') \in \delta\}$
- For every $a \in \Sigma_o, \delta_{\mathcal{A}_2}(X, (a, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \mid \exists q \in X, (q, a, q') \in \delta\}$

Then, for a sequence of observations, we can define:

- $\Delta_{\mathcal{A}_2}(X, \epsilon) = X$, and
- $\Delta_{\mathcal{A}_2}(X, \mu.(\sigma, \gamma)) = \delta_{\mathcal{A}_2}(\Delta_{\mathcal{A}_2}(X, \mu), (\sigma, \gamma))$ for every $\mu \in Tr_{\mathcal{A}_2}(G)$ and observation (σ, γ).

3.2 Attacker 3: input-aware attacker upon observation

We now model a type of attacker $\mathcal{A}_3$ who is aware of all allowed input events that are admissible only after an observable action $a \in \Sigma_o$ occurs. We still assume that inputs belong to $\Sigma_? \subseteq \Sigma_o$, and that the set of allowed inputs is revealed in a single shot. Unlike attacker $\mathcal{A}_2$, attacker $\mathcal{A}_3$ does not update its belief when the system performs an unobservable move. Within this setting, $O_3 = 2^{\Sigma_?}$ and the function $obs_{\mathcal{A}_3}$ is given by:

$$obs_{\mathcal{A}_3}(a, q) = \begin{cases} (a, \Gamma_?(\delta(q, a))) & \text{if } a \in \Sigma_o \\ \epsilon & \text{otherwise.} \end{cases}$$

The set of observations of $\mathcal{A}_3$ that can appear during runs of G is defined as $Tr_{\mathcal{A}_3}(G) \subseteq (\Sigma_o \times 2^{\Sigma_?})^*$. Each observation in $Tr_{\mathcal{A}_3}(G)$ is a sequence of the form $(a_1, \gamma_1)(a_2, \gamma_2)\ldots$, where each a_i is an observable event and $\gamma_i \subseteq \Sigma_?$ corresponds to the set of input events that are admissible after the observation of the event a_i in the state of the system reached after executing a_i. Given an observation $\mu \in Tr_{\mathcal{A}_3}(G)$, the set of compatible sequences is given by : $[\![\epsilon]\!]_{\mathcal{A}_3} = \{\epsilon\}$ and for $a \in \Sigma_o$:

$$[\![\mu.(a, \gamma)]\!]_{\mathcal{A}_3} = \{w \in [\![\mu]\!]_{\mathcal{A}_3}.a \cap \mathcal{L}(G) \mid q_0 \xrightarrow{w} q \wedge \Gamma_?(q) = \gamma\}$$

The refinement of a state estimation X by an observation (a, γ) is given by $\delta_{\mathcal{A}_3}(X, (a, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \in Q \mid \exists q \in X, \exists \rho = q \longrightarrow \ldots q', l(\rho) = a\}$. The state estimates of the attacker, starting from a set of states X, is recursively defined as follows:

- $\Delta_{\mathcal{A}_3}(X, \epsilon) = X$, and
- $\Delta_{\mathcal{A}_3}(X, \mu.(a, \gamma)) = \delta_{\mathcal{A}_3}(\Delta_{\mathcal{A}_3}(X, \mu), (a, \gamma))$, for $\mu \in Tr_{\mathcal{A}_3}(G)$ and $(a, \gamma) \in \Sigma_o \times 2^{\Sigma_?}$.

3.3 Attacker 4: an attacker informed of input changes

The last model of a passive attacker is informally defined as follows: we assume that the attacker $\mathcal{A}_4$ can continuously observe the possible inputs to the system, but is not aware of state changes. Hence, if two consecutive states q, q' in a run have the same inputs, and the move from q to q' produces no observable events, then $\mathcal{A}_4$ is not aware of any state change.

For $\mathcal{A}_4$, we have $O_4 = 2^{\Sigma_?}$ and $obs_{\mathcal{A}_4}$ is

$$obs_{\mathcal{A}_4}(a, q) = \begin{cases} (\epsilon, \Gamma_?(\delta(q, a))), & \text{if } a \in \Sigma_{uo} \wedge \Gamma_?(q) \neq \Gamma_?(\delta(q, a)); \\ \epsilon, & \text{if } a \in \Sigma_{uo} \wedge \Gamma_?(q) = \Gamma_?(\delta(q, a)); \\ (a, \Gamma_?(\delta(q, a)), & \text{if } a \in \Sigma_o. \end{cases}$$

The observations of $\mathcal{A}_4$ look like observations of $\mathcal{A}_2$, but with the difference that $\mathcal{A}_2$ obtains one pair (σ, γ) for each transition in a run. An observer $Det_{\mathcal{A}_4}(G)$, its transition relation $\delta_{\mathcal{A}_4}$ and $\Delta_{\mathcal{A}_4}$ can be built as for our previous attackers.

- $\delta_{\mathcal{A}_4}(X, (\epsilon, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \mid \exists \rho = q \xrightarrow{a_1} q_1 \ldots \xrightarrow{a_k} q_k \xrightarrow{a_{k+1}} q', l(\rho) = \epsilon \wedge \forall i \in 1..k, \Gamma_?(q) = \Gamma_?(q_i) \wedge \Gamma_?(q) \neq \Gamma_?(q')\}$
- For every $a \in \Sigma_o, \delta_{\mathcal{A}_4}(X, (a, \gamma)) = \Gamma_?(X', \gamma)$, where $X' = \{q' \mid \exists q \in X, (q, a, q') \in \delta\}$

Then, for a sequence of observations, we can define:

- $\Delta_{\mathcal{A}_4}(X, \epsilon) = X$, and
- $\Delta_{\mathcal{A}_4}(X, \mu.(\sigma, \gamma)) = \delta_{\mathcal{A}_4}(\Delta_{\mathcal{A}_4}(X, \mu), (\sigma, \gamma))$ for every $\mu \in Tr_{\mathcal{A}_4}(G)$ and an observation (σ, γ).

3.4 Attacker 5: an active attacker

We consider a slightly different attacker, namely one that is *active*. More precisely, attacker $\mathcal{A}_5$ learns whether an

input is active only by trying to play it. This attacker can observe actions from Σ_o, and also test whether an input is allowed at the current state by attempting to enter this input. If an input a is allowed at a state q, then the system moves to the next state $\delta(q, a)$. If the input is not allowed, then the system remains in state q and the attacker gets the information *not a*, indicating that the input was ineffective. We also consider that $\mathcal{A}_5$ can only test inputs to the system from the subset $\Sigma_{att} \subseteq \Sigma_? \cap \Sigma_o$.

To reflect the fact that an input is not fireable in a state we introduce a new alphabet $\Sigma_?^{not} = \{not\ \sigma | \sigma \in \Sigma_?\}$. In this setting, the alphabet of observed facts O_5 is given by $O_5 = \Sigma_o \cup \Sigma_?^{not}$, whereas the function $obs_{\mathcal{A}_5}$ is

$$obs_{\mathcal{A}_5}(a, q) = \begin{cases} \epsilon, & \text{if } a \in \Sigma_{uo} \\ (a, \epsilon), & \text{if } a \in \Sigma_o \setminus \Sigma_{att} \\ (a, a), & \text{if } a \in \Sigma_{att} \cap \Gamma_?(q) \\ (a, not\ a), & \text{if } a \in \Sigma_{att} \wedge a \notin \Gamma_?(q) \end{cases}$$

The set of observations is then defined as $Tr_{\mathcal{A}_5}(G) \subseteq (\Sigma_o \times (\Sigma_o \cup \Sigma_?^{not} \cup \epsilon))^*$, while the transition relation of $Det_{\mathcal{A}_5}(G)$ is:

$$\delta_{\mathcal{A}_5}(X, (a, \gamma)) = \begin{cases} \{q' \in Q \mid \exists \rho = q \longrightarrow \ldots q', l(\rho) = a, \\ \quad \text{if } a \in \Sigma_o \setminus \Sigma_{att} \wedge \gamma = \epsilon \\ \{q' \in Q \mid \exists q \in X, (q, a, q') \in \delta\}, \\ \quad \text{if } a \in \Sigma_{att} \wedge \gamma = a \\ \{q \in X \mid \nexists q', (q, a, q') \in \delta\}, \\ \quad \text{if } a \in \Sigma_{att} \wedge \gamma = not\ a \end{cases}$$

As it was for the other attackers, the transition relation of observer $Det_{\mathcal{A}_5}(G)$ extends to words with $\Delta_{\mathcal{A}_5}(X, \mu.(a, \gamma)) = \delta_{\mathcal{A}_5}(\Delta_{\mathcal{A}_5}(X, \mu), (a, \gamma))$.

3.5 Comparing attackers

We can now extend the results of Corollary 1 to all attackers:

Proposition 2. For every $\mathcal{A}_i$, $i \leq 5$, S is opaque w.r.t. G and $\mathcal{A}_i$ iff $\forall \mu \in Tr_{\mathcal{A}_i}(G)$, $\Delta_{\mathcal{A}_i}(X_0, \mu) \not\subseteq S$.

Let us now compare the discriminating power of attackers. We will say that an attacker $\mathcal{A}_i$ is *less discriminating* than an attacker $\mathcal{A}_j$, and write $\mathcal{A}_i \sqsubseteq \mathcal{A}_j$ (with fixed alphabets $\Sigma, \Sigma_o, \Sigma_{att} \Sigma_?$) iff for every automaton G and every set S of secret states, S is opaque w.r.t $\mathcal{A}_j$ implies that S is opaque w.r.t $\mathcal{A}_i$. Looking at the information collected by each type of attacker, we can state the following.

Proposition 3. We have the following relationships between attackers:

- $\mathcal{A}_1 \sqsubseteq \mathcal{A}_2$, $\mathcal{A}_1 \sqsubseteq \mathcal{A}_3$, $\mathcal{A}_1 \sqsubseteq \mathcal{A}_4$, and $\mathcal{A}_1 \sqsubseteq \mathcal{A}_5$
- $\mathcal{A}_3 \sqsubseteq \mathcal{A}_2$, $\mathcal{A}_4 \sqsubseteq \mathcal{A}_2$, and $\mathcal{A}_5 \sqsubseteq \mathcal{A}_2$

Obviously, $\mathcal{A}_1$ is the weakest attacker, as it only observes occurrences of actions in Σ_o, while all other observers have at least this information when an observable transition is executed. It might be surprising that $\mathcal{A}_5$, which is active, is less powerful that $\mathcal{A}_2$. Indeed, $\mathcal{A}_5$ can not always test *all* inputs of the system at the current state to get the same information as $\mathcal{A}_2$, since each test may force the system to move to another state.

4. ENFORCING OPACITY

If a system is not opaque, then one might try to control the system (or to restrict its behavior) so that the secret becomes opaque. This is achieved by dynamically forbidding or *disabling* certain events. Hence forth, we attach to the system a *controller* that is in charge of deciding which inputs are allowed to occur or *enabled* at each point of an execution. We assume that the controller can observe all the events of Σ, but cannot control all of them. Thus, Σ The alphabet of the system is hence partitioned into Σ_c the set of controllable events, and Σ_{uc}, the set of uncontrollable events. We assume that the controller has full knowledge of the system. Further, the controller are *belief-based*, i.e., they base their decisions on the finest state estimation that an attacker may have.

Definition 6. A *controller* is a function $\mathcal{C} : \mathcal{L}(G) \to 2^{\Sigma}$ which provides the set of actions that are enabled after a sequence $u \in \mathcal{L}(G)$, with the constraint that $\forall u \in \mathcal{L}(G), \mathcal{C}(u) \cap \Sigma_{uc} = \Gamma(\delta(q_o, u)) \cap \Sigma_{uc}$ (controllers cannot disable uncontrollable events). Given a state q, let $C_v(q)$ denote the set of valid control policies in q, i.e., all sets $c \subseteq \Gamma(q)$ that respect $c \cap \Sigma_{uc} = \Gamma(q) \cap \Sigma_{uc}$.

Intuitively, when a sequence u ends in a state q, then a controller for G can only use a policy in $C_v(q)$. A controller $\mathcal{C}$ can be encoded by an automaton $\mathcal{A}_{\mathcal{C}} = (Q_C, q_{0_C}, \delta_C, \Sigma, \lambda_C)$, where Q_C is a not-necessarily finite set of states, δ_C is a deterministic transition relation, and $\lambda_C : Q_C \to 2^{\Sigma}$ associates each state with a set of enabled events.

Definition 7. Let G be an automaton, and let $\mathcal{C} : \mathcal{L}(G) \to 2^{\Sigma}$ be a controller, described by its automaton $\mathcal{A}_{\mathcal{C}}$. Then $G/\mathcal{C} = (Q_{G/\mathcal{C}}, (q_0, q_{0_C}), \Sigma, \delta_{G/\mathcal{C}})$ is the automaton obtained by taking the synchronous product of G with $\mathcal{A}_{\mathcal{C}}$, inductively built as follows.

- the initial state is (q_0, q_{0_C})
- if $(q, q_C) \in Q_{G/\mathcal{C}}$, $\delta(q, a) = q'$, $a \in \lambda_C(q_C)$ and $\delta(q_C, a) = q'_C$, then $((q, q_C), a, (q', q'_C)) \in \delta_{G/\mathcal{C}}$, and (q', q'_C) belongs to $Q_{G/\mathcal{C}}$

One can immediately notice that $\mathcal{L}(G/\mathcal{C}) \subseteq \mathcal{L}(G)$.

A question that immediately arises: how is the knowledge of an attacker is affected by control? We assume that attackers do not know precisely the control applied to the system. Nevertheless, this does not mean that control does not affect observation. For instance, consider a situation where the current state is q, $\Gamma(q) = \{?a, ?b\}$, but the controller disables $?a$ at q. We consider that attackers get information on the actual set of available actions, i.e., when some control is in use. An attacker will receive information of the form $(e, \{?b\})$ when the system enters state q, where e is either ϵ or an action name. Let X be the current state estimation of the attacker. It can be refined by the observation, but recall that some actions that are not feasible from current state might actually be forbidden by a controller. Hence we define $Refact(X, \gamma) = X \setminus \{q \in Q \mid \exists e \in \gamma, \nexists q', q \xrightarrow{e} q'\}$. This refinement of state estimation removes from X all states from which actions from γ are not fireable.

We can now define the control and synthesis problem:

Control Problem: *Given a system G, a secret S, an attacker $\mathcal{A}$ that observes Σ_o, and a controllable alphabet $\Sigma_c \subseteq \Sigma$, is there a controller $\mathcal{C}$ such that S is opaque for $\mathcal{A}$ in $G/\mathcal{C}$?*

Synthesis Problem: *Given a system G, an attacker $\mathcal{A}$, and a secret S such that a controller with controllable alphabet $\Sigma_c \subseteq \Sigma$ exists to keep S opaque for $\mathcal{A}$, synthesize a map $\mathcal{C} : \mathcal{L}(G) \rightarrow 2^\Sigma$ such that S is opaque for $\mathcal{A}$ in $G/\mathcal{C}$.*

In the rest of this section we will show that for passive attackers $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ and $\mathcal{A}_4$, controllers are indeed strategies in finite co-reachability games. Within these games, positional strategies are sufficient to win, yielding finiteness of controllers.

Definition 8. A two-player arena with partial information is a tuple $H = (V, v_0, \mathcal{M}, \longrightarrow, \mathcal{O}_0, \mathcal{O}_1)$, where V is a set of positions, partitioned into $V_0 \uplus V_1$ (depicting player 0 and player 1's nodes, respectively); v_0 is the initial node of the arena; $\longrightarrow \subseteq V \times \mathcal{M} \times V$ depicts moves in the arena; and $\mathcal{O}_i : V \rightarrow \Psi_i, i \in \{0,1\}$ are observation functions. A two player co-reachability game is given by an arena and a winning condition $Win \subseteq V$. Player 0 wins the game if it can prevent reaching a position in Win.

A *play* in an arena is either a finite sequence $\pi = v_0.v_1.v_2 \ldots v_k$ or an infinite one $\pi = v_0.v_1.v_2 \ldots v_k \ldots$. We denote by $\pi(j)$ the jth vertex of play π. A play π is *winning* for player 1 (and losing for player 0) with the co-reachability condition Win if there exists j such that $\pi(j) \in Win$. A *strategy* for player i is a map $\sigma_i : V*.V_i \rightarrow 2^{Act_i}$, where Act_i is the set of moves of player i. In particular, we require that

- for every play $\pi = v_0.v_1 \ldots v_k$ with $v_k \in V_i$, $\sigma_i(\pi) \subseteq Act(v_k)$, where $Act(v_k)$ is the set of legal moves from node v_k in the arena.
- for every pair of plays π, π' such that $\mathcal{O}_i(\pi) = \mathcal{O}_i(\pi')$, $\sigma_i(\pi) = \sigma_i(\pi')$ (i.e., strategies are consistent with observations).

A play is *consistent* with strategy σ_i iff for for every $\pi(j) \in V_i$, $\pi(j+1) \in \sigma_i(\pi(0) \ldots \pi(j))$. Player i *wins* in position v iff it has a strategy σ_i such that for every strategy σ_{1-i} of player $1-i$, all plays starting from v and consistent with strategies σ_i, σ_{1-i} are winning for player i.

We will say that a two-player game is a game of perfect information iff $\forall v \in V, O_1(v) = O_2(v) = \{v\}$. Otherwise, the games are called games with partial information. It is known that perfect information games with (co)reachability, Büchi, or parity objectives are determined and need only positional strategies (Grädel and Thomas, 1998). It is known that players have a winning strategy in a partial information game iff they have a belief-based strategy (Chatterjee and Doyen, 2010). Consequently, to decide whether a game is winning for player i, it is sufficient to build an arena containing the representation of the knowledge that each player has of the possible current state of the system. Then, solving a full-information game on this arena allows for the computation of belief-based strategies.

We are now ready to define a game that allows us to solve the opacity control and synthesis problem for each type of attacker. We begin with attackers of type $\mathcal{A}_2$.

4.1 A controller for attackers of type $\mathcal{A}_2$

In this game, the controller is player 0, the rest of the system is the player 1, and the attacker of the system

is used to determine a winning condition on states. It might seem strange *a priori* to consider the system as an opponent to the controller, and not to the attacker. The reason for this is that attackers are passive. Hence they cannot guide the system to play the best actions leading to information leakage. However, to ensure that a secret is preserved, one can assume that the system, even if not malevolent, will always play the worst choice with respect to secrecy.

Let G be a system over alphabet Σ, $S \subseteq Q$ be a set of secret states, q_0 be an initial state, Σ_o be the set of events observed by attacker $\mathcal{A}_2$, and Σ_c be a set of controllable events. The arena for this opacity control and synthesis problem is defined as a tuple $H_{G,S,\mathcal{A}_2} = (\mathcal{V}, v_0, \mathcal{M}, \longrightarrow_H, \mathcal{O}_0, \mathcal{O}_1)$ where:

- $\mathcal{V} = \mathcal{V}_0 \cup \mathcal{V}_1$ is a set of nodes. We have $\mathcal{V}_1 = Q \times 2^Q \times 2^\Sigma$. This set of nodes denotes positions of player 1 (the system). $\mathcal{V}_1$ contains nodes of the form (q, X, Σ') where q is the current state of G, X is the attacker's estimation of the set of possible states according to its observations, and $\Sigma' \subseteq \Sigma$ is the last choice of the controller, i.e., a set of actions that the controller allows from state q. Similarly, $\mathcal{V}_0 = Q \times 2^X \times \Sigma$ is the set of positions of player 0 (i.e., the controller). Every node $(q, X, a) \in \mathcal{V}_0$ represents the next current state q and the next possible state estimation X after performing action a. The initial node of the game is node $v_0 = ((q_0, \{q_0\}), \epsilon) \in \mathcal{V}_0$, i.e., the system starts from its initial node, the attacker has a perfect knowledge of it, and it is then up to the controller to decide which actions are allowed from the initial state. In this node, ϵ symbolizes the fact that no action has yet been performed by the system.
- $\mathcal{M} = \mathcal{M}_0 \cup \mathcal{M}_1$ is the alphabet of the game, where $\mathcal{M}_0 = \Sigma \times 2^\Sigma$ represents the moves of the controller, which makes a choice between the admissible control policies, and $\mathcal{M}_1 = \Sigma$ represents the moves of the system.
- $\mathcal{O}_0(v) = \mathcal{O}_1(v) = v$, i.e., both players have perfect knowledge of games positions.
- Moves of the arena are a subset $\rightarrow_H \subseteq (\mathcal{V}_1 \times \mathcal{M} \times \mathcal{V}_0) \cup (\mathcal{V}_0 \times \mathcal{M} \times \mathcal{V}_1)$ and are built as follows:
  i) for every $c_i \in C_v(q_o)$, $(v_0, c_i, (q_0, \{q_0\}), c_i))$ is a move of $\rightarrow_H$. Note that $(q_0, \{q_0\}), c_i) \in \mathcal{V}_1$.
  ii) If $((q, X), c)$ is a node of $\mathcal{V}_1$, and $a \in c$ is an action allowed by the controller, then
  $((q, X), c), ((\delta(q, a), X), a) \in \rightarrow_H$.
  iii) Given a state $((q, X), a) \in \mathcal{V}_0$, the node $((q, X), a), (a, c), (q, X', c))$ in $\rightarrow_H$ for all $c \in C_v(q)$. Let $\gamma = c \cap \Gamma_?(\delta(q, a))$, and $X_{O,a} = \Delta_o(X, a)$ (Defined as $\Delta_{\mathcal{A}_1}$). Then we have $X' = RefAct(X_{0,a}, \gamma)$, i.e., the attacker's knowledge is updated, and includes states that are reachable from X when observing a, and from which one can execute at least inputs in γ. Note that as the attacker is not aware of c_i, X' is the only deduction that an attacker of type $\mathcal{A}_2$ can do. (See Fig. 1)

Fig. 1 shows the representation of several possible types of moves in arena H. Nodes of the controller (player 0) are represented as circles, and nodes of player 1 as rectangles. In state (q, X, c), the system decides to fire action $a \in \Sigma \cap c$. The system then moves to a position $(\delta(q, a), X, a)$
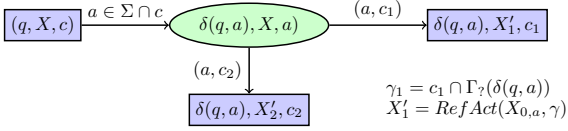
Fig. 1. A part of the arena for the opacity game

meaning that the system will move to position $\delta(q,a)$. From this node, the controller has to decide which actions will be fireable from the next state $\delta(q,a)$, which will impact what is observable by the attacker. In Fig. 1, the controller has two possible choices, namely c_1 and c_2. After a choice of the controller the move is performed and the attacker is informed of the move. Following the definition of an attacker of type $\mathcal{A}_2$, the observation produced is of the form (e, γ), where $e = \epsilon$ if the action played by the system is unobservable, and $e = a$ if the performed action is observable. Similarly, when control c_i is chosen, the set of possible inputs that attacker $\mathcal{A}_2$ can observe is $\gamma = \Gamma_?(\delta(q,a)) \cap c_i$. The state estimate of the attacker is updated accordingly, i.e., upon the choice of control c_i the state estimate becomes $X' = RefAct(\Delta_o(X,a), \gamma)$.

The objective of the controller is to avoid states of the form (q, X, c), where $X \subseteq S$, i.e., player 1 wins if it can force the system to one of these states. This is a classic turn-based co-reachability game, where the controller needs to choose valid control policies that avoid positions where the attacker's knowledge is contained in S. It is well-known that perfect information (co-)reachability games are positional and determined, and can be solved in linear time by computing an attractor for the set of bad states that player 0 wants to avoid. The construction of this attractor is described in appendix 6.1. It remains to show that a controller for opacity exists iff there is a wining strategy in this reachability game.

Theorem 1. Let G be an automaton, S be a secret, and $\mathcal{A}_2$ be an attacker. Secret S can be made opaque by some controller $\mathcal{C}$ iff player 0 has a winning strategy in the co-reachability game over arena $H_{G,S,\mathcal{A}_2}$ with winning condition $Win = Q \times 2^S \times 2^\Sigma$.

Note that belief-based controllers are not the only possible controllers. Consider the automaton of Fig. 2, where $\Sigma = \{a, u, c\}$ and $\Sigma_o = \{a, c\}$. To avoid reaching secret state S, a controller can forbid action a when the difference between the number of as and cs is equal to k. Such a controller never computes the state estimation of an attacker, but yet is sufficient to guarantee opacity.
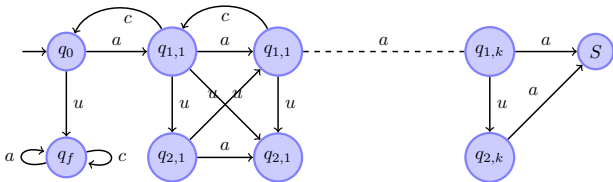

Fig. 2. An automaton G that admits simple controllers

Corollary 3. For an alphabet of fixed size, enforcement of opacity for an attacker of type $\mathcal{A}_2$ is in EXPTIME.

As already mentioned, positional strategies for two player games with perfect information are sufficient. It is hence sufficient to design strategies that associate a decision to

each node of the arena. Recall however that our games are played on an arena that represents attacker's knowledge. This gives us the following corollary.

Corollary 4. Finite state (belief-based) controllers are sufficient to ensure opacity. Opacity controllers are of size in $O(|G|.2^{|G|})$. There exist automata of size in $O(n)$ in which opacity can be ensured only by controllers of size $O(n.2^n)$.

4.2 A controller for attackers of type $\mathcal{A}_3$ and type $\mathcal{A}_4$

The technique used in Theorem 1 for attackers of type $\mathcal{A}_2$ can be easily adapted for attackers of type $\mathcal{A}_3$ and $\mathcal{A}_4$, and the opacity enforcement game remains a perfect information reachability game. The construction of the arena is very similar: the major difference resides in how the attacker's state estimate is updated. With attacker $\mathcal{A}_3$, for instance, a move from node $((q, X), c) \in \mathcal{V}_1$ with action a leads to a node $((\delta(q,a), X, a)$, as was the case for attacker $\mathcal{A}_2$. Moves from nodes of the controller (nodes of $\mathcal{V}_0$) of the form (q, X, a) lead to nodes of the form (q, X, c) if a is unobservable, and to nodes of the form (q, X', c), where $X' = Refact(X_o, c \cap \Gamma_?(\delta(q,a)))$ is an update of the knowledge of $\mathcal{A}_3$, with $X_o = \{q' \in \Delta_o(X, a)\}$.
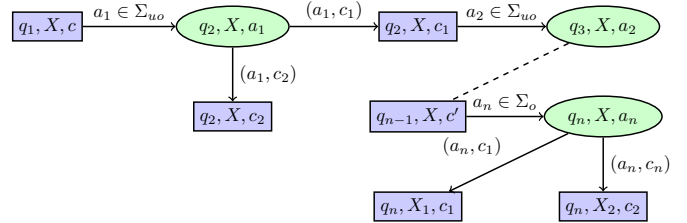

Fig. 3. Part of the arena for attacker $\mathcal{A}_3$

For attacker $\mathcal{A}_4$, the knowledge update is performed as soon at the input set changes. From a node of the form (q, X, c) in $\mathcal{V}_1$ the system can choose an action a allowed by the control c, and move to a state $(\delta(q,a), X, a, q)$. Notice that the state that is left is remembered. From a node (q', X, a, q) of $\mathcal{V}_0$ when choosing a control c, one moves to node (q', X, c) if $\Gamma(q) = \Gamma(q')$ and a is unobservable, and to node (q', X', c) otherwise, where X' is computed as for attackers of type $\mathcal{A}_2$ and $\mathcal{A}_3$. Opacity enforcement games remain unchanged on this slightly modified arena.

5. ENFORCEMENT WITH ACTIVE ATTACKERS

With an active attacker $\mathcal{A}_5$, the game setting changes: since attackers are active, they can have a strategy to drive the system into configurations leaking the secret S. In the previous games designed to solve opacity for passive attackers, the partial observation of such an attacker is only used to designate which nodes of the arena are losing states for the controller. Observation functions for this kind of game are perfect information functions. Active attacker $\mathcal{A}_5$, on the other hand, can take decisions, and is now one of the players in the arena, although it has only a partial information on the actual state of G. We assume that the attacker is one of the processes of the system, and owns a subset of actions $\Sigma_{att} \subseteq \Sigma_o$ that are performed only by him. These inputs can be performed/tested by the attacker immediately after each observable action. Yet, in this setting, the rest of the system can still perform actions that help $\mathcal{A}_5$ acquire further information. We will hence distinguish nodes where the system plays, and nodes where

the attacker chooses to input something to the system. This setting forces us to consider a new type of game, namely, three-player games, as introduced in (Chatterjee and Doyen, 2014).

Definition 9. A *three-player game* is a tuple $\mathcal{H} = (Q, q_0, \delta, O_{i,i \in 1..3}, obs_{i,i \in 1..3})$, where Q is a set of nodes, q_0 an initial node, and $\delta : Q \times A_1 \times A_2 \times A_3 \to Q$ is a deterministic function that gives the successor state $\delta(q, a_1, a_2?a_3)$ of q when players $1, 2, 3$ concurrently play actions a_1, a_2, a_3. The game is *turn-based* if for every state q and every triple in $A_1 \times A_2 \times A_3$ the next state depends only on the action chosen by one of the players. For every player, $O_i \subseteq 2^Q$ is a partition of the set of states, and $obs_i : Q \to O_i$ the function that assigns to q the unique observation of player i that contains q. A winning reachability condition is a subset of states $Win \subseteq Q$. An observation-based strategy is a map $\sigma_i : Q \times O_i^* \to Q$. Given a game arena $\mathcal{H}$ and a winning condition Win, the three-player reachability problem is to decide if $\exists \sigma_1, \forall \sigma_2, \exists \sigma_3$ such that the outcome of play that satisfies strategies $\sigma_1, \sigma_2, \sigma_3$ visits Win at least once.

We define an arena $\mathcal{H}_G^{active} = (V, n_0, \delta, O_i, obs_i)$ as follows:

The set of nodes V partitioned into $V = V_1 \cup V_2 \cup V_3$, where V_1 represents positions of the attacker $\mathcal{A}_5$, V_2 positions of the controller, and V_3 positions of the system.

$V_1 = \{(q, X, c, att) \mid X \in 2^Q, C \in 2^\Sigma\}$
$V_2 = \{(q, X, a, q') \mid q, q' \in Q, X \in 2^Q, a \in \Sigma\}$
$V_3 = \{(q, X, c, sys) \mid X \in 2^Q, C \in 2^\Sigma\}$

We set $v_0 = (q_0, \{q_0\}, \epsilon, q_0) \in V_2$, i.e., the controller starts (as no action is played yet, the third component of the node is simply ϵ). The transition relation δ is then computed as follows: the game allows only moves from V_1 to V_2, from V_3 to V_2 and V_1 and from V_2 to V_3. It is turn based, i.e., for every node $n \in V_i$, $\delta(n, a_1, a_2, a_3)$ only depends on a_i.

There exists a move from $(q, X, c, sys) \in V_3$ to $(q, X, c, att) \in V_1$ if $\Gamma_?(q) \cap c \neq \emptyset$ [2] . This move models the situation where the system does not perform an action, some inputs are feasible, and the system lets the attacker test them.

There exists a move from $(q, X, c, sys) \in V_3$ to $(q, X, a, \delta(q, a)) \in V_2$ if $a \in c$. This move models the system's decision to perform an action. Right after this decision, upon reaching the destination state, the controller has to take a control decision for inputs in Σ_c. Similarly to games with attackers of type $\mathcal{A}_2$ and $\mathcal{A}_3$, knowledge of $\mathcal{A}_5$ is not updated by this move, which allows the controller to choose the next control decision.

There exists a move from $(q, X, c, att) \in V_1$ to $(q, X, a, \delta(q, a)) \in V_2$ if $a \in c \cap \Gamma_?(q)$. This move models input to the system performed by the attacker, and the consequences of this input (i.e., a state change).

There exists a move from $(q, X, c, att) \in V_1$ to $(q, X', c, sys) \in V_2$ if $a \notin c \cap \Gamma_?(q)$, with $X' = X \setminus \{q \in X \mid a \in \Gamma_?(q)\}$ if $a \in \Sigma_{uc}$, and $X' = X$, otherwise. Let us detail the computation of X'. Node (q, X, c, att) is an attacker node, i.e., from this node, an attacker may want to test an input to the system to refine its estimation of the system's state.

[2] If we consider that $\Gamma_?(q) \subseteq \Sigma_o$. Otherwise, we should write $\Gamma_?(q) \cap c \cap \Sigma_o \neq \emptyset$

If this input $?a$ fails, then it is either because this input is not feasible from the current state, or because it was disabled by the controller. If the input is controllable, an attacker cannot distinguish between a state q_f such that $?a \notin \Gamma_?(q_f)$ and a state q_a such that $?a \in \Gamma_?(q_a)$ but $?a \notin c$. If the attacker estimates that the system can be in state X before testing $?a$, and subsequently learns that this input is not feasible, then its knowledge remains unchanged. On the other hand, if the input does not succeed, and $?a$ is not controllable, then X can be refined, since the current state of the system is necessarily a state q with $?a \notin \Gamma(q)$.

The controller allows initial moves from node n_0 to every node of the form (q_0, c, sys) where $\Gamma(q_0) \cap \Sigma_{uc} \subseteq c \subseteq \Gamma(q_0)$. All moves from a control node of the form $n = (q, X, a, q') \in V_2$, with $a \notin \Sigma_o$ are moves to nodes of the system of the form $(q', X, c, sys) \in V_3$, with $c \in 2^\Sigma$. Notice that the knowledge of the attacker is not updated. This models the fact that unobservable moves do not bring information to the attacker. All moves from a control node of the form $n = (q, X, a, q')$, with $a \in \Sigma_o$ are moves of the form $(q', X', c, sys) \in S_3$, with $c \in 2^\Sigma$, and $X' = \Delta_o(X, a)$. The knowledge of the attacker is updated only according to the observation of action a, whereas for attacker $\mathcal{A}_5$, observation of allowed inputs is performed actively, i.e., via a test made by the attacker.

We can now define the observation functions: The controller (player 2) has perfect knowledge of the system's state, so we define $O_2(n) = n$ for every node of the system. The system (player 3) also has perfect information on the current node of the system, so $O_3(n) = n$. The attacker only has partial information, i.e., if $n = (q, X, x, sys)$, then $O_2(n) = X$. Thus, the attacker cannot differentiate states containing the same state estimation. Lastly, the winning condition is $Win_S = \{n \mid O_2(n) \subseteq S\}$.
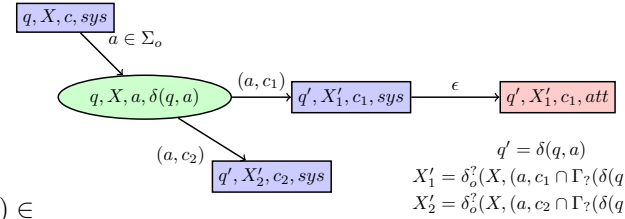
Fig. 4. An arena for the opacity game with an active attacker $\mathcal{A}_5$. States of the attacker are represented as red rectangles, states of the controller as green circles, and states of the system as blue rectangles.

Theorem 2. Let G be an automaton and S be a secret. The opacity of S w.r.t. an active attacker $\mathcal{A}_5$ can be enforced by a belief-based controller iff player 2 has a winning strategy σ_2 against all strategies σ_1, σ_3 of players 1 and 3 in the three-player reachability game over arena H_G^{active} with winning condition Win_S.

In their most general form, three-player games of partial information are undecidable (Reif, 1979), but (Chatterjee and Doyen, 2014) showed that three-player games are decidable when player 1 is less informed than player 2. This is the case in our setting, as the attacker only has partial information on the current location in the arena, while players 2 and 3 have perfect knowledge of it. More precisely, (Chatterjee and Doyen, 2014) show that three-player games are 2-EXPTIME complete, in general. The

main idea of their proof is to show a reduction from a three-player game to an exponentially larger partial information two-player game with the same objective. As partial information two-player games can be solved in EXPTIME, this yields the 2-EXPTIME upper bound for complexity. A remark in this paper is particularly interesting: if player 2 has perfect information on the game, then the decision procedure is only in EXPTIME. We can even avoid an additional exponential blowup, as explained in the following corollary.

Corollary 5. Given an automaton G, a set of secret states S and an active attacker $\mathcal{A}_5$, deciding existence of a controller ensuring opacity of G wrt active attacker $\mathcal{A}_5$ can be solved in EXPTIME.

6. CONCLUSION

We have proposed a unified setting to address opacity with passive and active attackers. For all attackers, opacity enforcement can be brought back to a game: a system can be controlled to enforce opacity if and only if a player representing a controller added to the system has a winning strategy.

Related work : Opacity can be seen as the opposite of diagnosis problems. In diagnosis (Sampath et al., 1995), the question of interest is whether fault occurrences can be detected with partial observation of a running system. This question boils down to checking that there exists no pair of observationally-equivalent runs, one of which is faulty, the other one that is not. Ensuring diagnosability is a desirable property of a system. The goal of opacity, on the other hand, is to prevent an attacker from gaining certainty about properties of executed runs by making such runs observationally equivalent to a run that does not possess the property of interest. From an algorithmic point of view, techniques to ensure diagnosability consist of verifying the absence of pairs of runs, which can be done in polynomial time on the self-product of the specification. A contrario, to verify opacity we need to check properties of a whole equivalence class for traces, and not only a pair of runs. This requires modeling the complete knowledge of an attacker, and immediately yields an exponential blowup.

Opacity has been addressed in several settings since the work of (Bryans et al., 2005) and (Badouel et al., 2007). Our approach does not only consist of checking whether opacity holds, but also provides the means to control a system in such a way that observed traces are not sufficient to decide whether the current state of the system is a secret one. Another enforcement technique was proposed by (Cassez et al., 2012). This approach consists of dynamically changing the observation of an attacker using *masks*, that hide a subset of transition labels that would otherwise be seen by the attacker. In this setting, the behavior of the system remains unchanged, but control dynamically changes the way the system is observed. Our approach differs, as we forbid some behaviors. In fact, the approaches are orthogonal: one can easily design a system that is opaque with the help of masks, and not with control (e.g., when all events are uncontrollable). Conversely, one can easily design a system where masks are useless but where control can make a system opaque. This occurs rapidly when masks cannot reduce the set of observable events.

Future work: This work can be extended in several ways: first of all, it should be easy to consider enforcement of opacity for regular properties. Indeed, as soon as runs satisfying a property P are recognized by an automaton $\mathcal{A}_P$, enforcing opacity of runs of G satisfying P simply means enforcing state-based opacity in $G \times \mathcal{A}_P$. Other open questions deal with the power of attackers. So far, we have considered that the attacker is alone, and can observe and interact with the system. An interesting question is whether this setting can be extended to a coalition of players that run distributed attacks. We have also considered that controllers of the system had full knowledge of the current state of the system. This allows us to recast opacity control in a game setting with perfect observation for the controller. Now, if the controller is a particular component of the system, and observes only a subset of actions performed and has only partial information on the current global state of the system, the setting for enforcement questions changes. For passive attackers, the enforcement games become two-player games *with partial information*, which are exponentially harder to solve than perfect information games. In an active setting, opacity enforcement becomes a three-player game with partial information. Such games are usually undecidable (Peterson and Reif, 1979), but there is still a chance that that opacity enforcement problems can be recast in a decidable setting of (Chatterjee and Doyen, 2014).

REFERENCES

Badouel, E., Bednarczyk, M.A., Borzyszkowski, A.M., Caillaud, B., and Darondeau, P. (2007). Concurrent secrets. *Discrete Event Dynamic Systems*, 17(4), 425–446.

Bérard, B., Mullins, J., and Sassolas, M. (2015). Quantifying opacity. *Mathematical Structures in Computer Science*, 25(2), 361–403.

Bertrand, N., Fabre, E., Haar, S., Haddad, S., and Hélouët, L. (2014). Active diagnosis for probabilistic systems. In *Proc. of FOSSACS 2014*, volume 8412 of *LNCS*, 29–42.

Bryans, J., Koutny, M., Mazaré, L., and Ryan, P.Y.A. (2005). Opacity generalised to transition systems. In *Formal Aspects in Security and Trust, Third International Workshop, FAST 2005, Newcastle upon Tyne, UK, July 18-19, 2005, Revised Selected Papers*, 81–95.

Cassez, F., Dubreil, J., and Marchand, H. (2012). Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40(1), 88–115.

Chatterjee, K. and Doyen, L. (2010). The complexity of partial-observation parity games. In *Proc. of LPAR'17*, volume 6397 of *LNCS*, 1–14.

Chatterjee, K. and Doyen, L. (2014). Games with a weak adversary. In *Proc. of ICALP 2014, Part II*, volume 8573 of *LNCS*, 110–121.

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

Grädel, E. and Thomas, W. (1998). *Automata Logics, and Infinite Games : A Guide to Current Research*, volume 2500 of *LNCS*.

Haar, S., Haddad, S., Melliti, T., and Schwoon, S. (2017). Optimal constructions for active diagnosis. *J. Comput. Syst. Sci.*, 83(1), 101–120.

Peterson, G.L. and Reif, J.H. (1979). Multiple-person alternation. In *FOCS'79*, 348–363. IEEE.

Reif, J. (1979). Universal games of incomplete information. In *STOC*, ACM Press.

Sampath, M., Sengupta, R., Lafortune, S., Sinaamohideen, K., and Teneketzis, D. (1995). Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9), 1555–1575.

APPENDIX

6.1 Construction of an attractor

We want to avoid leaking nodes of the arena, i.e., the nodes in which the belief of the attacker is contained in S. Initially, $Bad_0 = Win = \{(q, X, c) \in \mathcal{V}_1 \mid X \subseteq S\}$ and $Bad_0^c = \emptyset$. We then iterate the following two fix-points up to stabilization:

$$Bad_{i+1}^c = Bad_i^c \cup \{((q, X), a) \in \mathcal{V}_0 \mid a \in \Sigma \cup \epsilon,$$
$$\forall c \in C_V(q), \exists ((q', X'), c) \in Bad_i \text{ and}$$
$$(((q, X), a), (a, c), ((q', X'), c)) \in \to_H \}$$

$$Bad_{i+1} = Bad_i \cup \{((q, X), c) \in \mathcal{V}_1 \mid \exists a \in \Sigma,$$
$$(((q, X), c), a, (\delta(q, a), X), a) \in \to_H$$
$$\text{with } (\delta(q, a), X), a) \in Bad_i^c \}$$

We denote by Bad and Bad^c the results of these fix-point computations. If $v_0 \in Bad^c$, then there exists a path from v_0 to Bad_0. In particular, let $\pi = n_0..n_k$ be one of the longest acyclic paths from v_0 to Bad_0. One can remark that k is smaller than the number of steps needed to achieve a fix-point, and that $v_0 \in Bad_k$ but $v_0 \in Bad_{k-1}$. This means, that every choice from v_0 leads to a node in $Bad_{k-1}^C \cup Bad_{k-1}$. Thus, no strategy of player 0 can avoid reaching $Bad_{k-1}^C \cup Bad_{k-1}$, i.e., there is no control policy that avoids getting closer to Bad_0. One can extend this reasoning to the next k steps (at most), showing that if $v_0 \in Bad^c$ there is no winning strategy for the controller to avoid Win.

Let us now suppose that $v_0 \notin Bad_0$. Then we can allow, in all nodes of the controller, any control that leads to a node of the system in $\mathcal{V} \setminus Bad$, i.e., there exists at least one positional strategy to avoid leakage.

6.2 Proofs of theorems, propositions and corollaries

Proof of Proposition 1

Proof : To find witnesses of non-opacity, one can randomly select a run ρ of G. Every time ρ reaches a secret state, verify that the set of runs of G that are compatible with $\mathcal{A}(\rho)$ contain at least one run that does not end in a secret state. Obviously, this algorithm does not terminate. However, note that to check if a run meets the opacity condition, then it is sufficient to consider only its last state q, and the set X of last states of runs in $[\mathcal{A}(\rho)]$. Opacity is violated if $[\mathcal{A}(\rho)] \subseteq S$. Then, when a new transition (q, a, q') is chosen, it sufficient to compare q' (if $q' \in S$ and $\delta_{\mathcal{A}}(X, \mathcal{A}(a, q'))$. Note that states of $Det_{\mathcal{A}}(G)$ maintain $[\mathcal{A}(\rho)]$, which completes the proof. $\square$

Proof of Corollary 2

Proof : As the opacity problem has been recast as a reachability question for an automaton of size in $2^{|G|}$, one can perform a non-deterministic exploration of this automaton with memory linear in $|G|$, hence $PSPACE$ membership. For the hardness, one can reuse the results of (Cassez et al., 2012), showing that opacity with observers that simply read projections of runs on Σ_o is PSPACE-complete. $\square$

Proof of Theorem 1

Proof : Assume that player 0 has a strategy σ_0 in arena $H_{G,S,\mathcal{A}_2}$ that forbids reaching Win. All runs of the arena following strategy σ_0 avoid nodes leaking the secret S, i.e., nodes of the form (q, X, c) where $X \subseteq S$.

Let $\mathcal{C}_{\sigma_0}$ be the following controller: we have $q_{0c_{\sigma_0}} = n_0$, $Q_{\mathcal{C}_{\sigma_0}} = (Q \times 2^Q \times 2^\Sigma) \setminus Bad$ and
$$\delta_{\mathcal{C}\sigma_0} = \{((q, X, c), a, (q', X', c')) \mid q' = \delta(q, a) \wedge$$
$$(q', X', c') \in \sigma_0(q, X, a)\},$$
$\lambda_{C\sigma}(n) = c$ for every node of the form $n = (q, X, c)$. Now suppose $G/\mathcal{C}$ is not opaque, i.e., there exists a run $\rho = (q_0, X_0) \xrightarrow{a_1} (q_1, X_1) \ldots \xrightarrow{a_k} (q_k, X_k)$ of $G/\mathcal{C}$ such that $q_k \in S$ (ρ ends in a secret state), and every equivalent run ρ' ends in a state of S too. Then there exists a play $\pi_\rho = v_0.v_{0,c_0} \ldots v_{k-1,c}.v_{k-1}$ of the arena such that every $v_{i,c_i} = (q_{i-1}, X_{i-1}, c_i)$ is a node of player 1, every v_i is a node of player 0 of the form (q_i, X_i, a_i). Notice that X_k is contained in S. As ρ is a run of $G/\mathcal{C}_{\sigma_0}$, every transition $(q_{i-1}, X_{i-1}) \xrightarrow{a_i} (q_i, X_i)$ satisfies $a_i \in \lambda(q_i, X_i)$. However, as $X_k \subseteq S$, there exist an index $j \in 1..k$ such that for any set of controls $c_j, \ldots c_k$, $(q_j, X_j, c_j) \ldots (q_k, X_k, c_k) \subseteq Bad$. In particular the control $c_{j-1} = \lambda(q_{j-1}, X_{j-1})$ used at step $j - 1$ does not prevent leaking of the secret. This means that strategy σ_0 from a predecessor of node (q, X, c_{j-1}) fails to avoid Bad and is hence not a winning strategy (contradiction).

Let us now suppose that there exists a belief-based controller $\mathcal{C} = (Q_{\mathcal{C}}, q_{o,\mathcal{C}}, \Sigma, \delta_{\mathcal{C}}, \lambda_{\mathcal{C}})$, whose transitions are consistent with the state estimation of an attacker, such that $G/\mathcal{C}$ is opaque. Recall that the allowed inputs are published in every state reached after an observable event. We can design a strategy $\sigma_{\mathcal{C}}$ for player 0: we set $\sigma_{\mathcal{C}}(q, X, a) = (q', X', c')$, where $q' = \delta(q, a)$ and $(q', X') = \delta_{\mathcal{C}}((q, X), a)$, and $c = \lambda(q', X')$. Note that as we require beliefs of the controller to be consistent with the attacker's knowledge, we have $X' = RefAct(\Delta_o(X, a), \lambda(q', X') \cap \Gamma_?(q'))$.

As $G/\mathcal{C}$ is opaque, for every run ρ of $G/\mathcal{C}$ ending in a secret state $s \in S$, the observation of run ρ does not allow the attacker to build an estimation that contains only states in S and no state $s' \notin S$ reachable after an equivalent run ρ' of $G/\mathcal{C}$.

Every run of $G/\mathcal{C}$ is of the form $\rho = (q_0, \{q_0\}) \xrightarrow{a_1} (q_1, X_1) \xrightarrow{a_2} \ldots$, and is such that $\forall i, a_i \in \lambda(q_{i-1}, X_{i-1})$, and for every X_i, $S \cap X_i \neq \emptyset \Rightarrow X_i \setminus S \neq \emptyset$

Now, consider any run of the arena $\pi = (q_0, \{q_0\}, \epsilon) \longrightarrow (q_0, \{q_0\}, c) \longrightarrow (q_1, X_0, b_1) \longrightarrow (q_1, X_1, c_1) \ldots (q_k, X_k, c_k)$, that conforms to $\sigma_{\mathcal{C}}$. Since $\mathcal{C}$ is belief-based, for every (q_i, X_i, c_i) in run π, $c_i = \lambda(q_i, X_i)$. We have $\sigma_{\mathcal{C}}(q_0, \{q_0\}, \epsilon) = \lambda(q_0, \{q_0\})$, and $(q_0, \{q_0\}, \epsilon) \notin Bad_0$. We can prove that for any length n, $(q_n, X_n, c_n) \notin Bad^c$: indeed, for all (q_i, X_i), if $a \in \lambda(q_i, X_i)$ then $\delta((q_i, X_i), a) \notin Bad_0^c$ as this move is allowed in the controller. By induction on the length of π, we can show that $\delta((q_i, X_i), a_i) \notin Bad_0$.

So, for every $n = (q', X_i, a_i) \in \sigma_{\mathcal{C}}(q_i, X_i, c_i)$ and $n' = (q', X_{i+1}, c_{i+1})$ that are reachable after an action a_i allowed by c_i, $n' \notin Bad_0$. Hence, $\sigma_{\mathcal{C}}$ is a strategy to avoid Bad_0, i.e., a winning strategy for player 0. □

Proof of Corollary 3

Proof : The arena built is of size in $O(|Q|.2^{|Q|} \times 2^{|\Sigma|})$, i.e., exponential in the size of the original automaton. As co-reachability games with perfect information can be solved in linear time (in the size of the arena), this immediately gives us an exponential algorithm to check whether opacity can be enforced. □

Proof of Corollary 4

Proof : In the arena built in Theorem 1, all possible sets of state estimations of the attacker are built. There are at most $2^{|Q|}$ such estimations. It is well-known that for perfect information games with reachability, Büchi or parity winning conditions, if a strategy exists, then a positional strategy exists. Hence, positional strategies in $H_{G,S,\mathcal{A}_2}$ are sufficient to take the correct decisions that avoid positions in which an attacker has a state estimation fully contained in S. We have shown in Theorem 1 how to build a controller $\mathcal{C}_\sigma$ from a strategy σ. The states of these controllers contain the state estimation built by the attackers. So, belief-based controllers are sufficient to avoid secret leakage. Furthermore, these controllers need at most $|G|.2^G$ states to avoid secret leakage. Let us now show that some automata and secrets need controllers of exponential size.
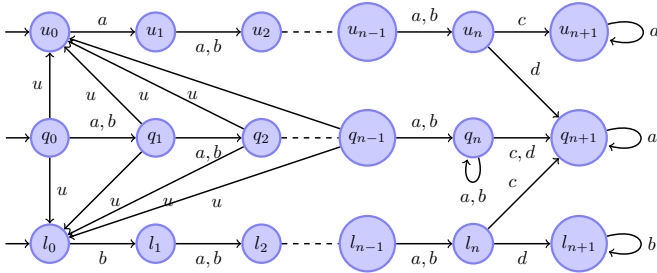

Fig. 5. An automaton for which attackers and controllers are of exponential sizes

The example of Fig. 5 is an automaton of size $3.n+6$. This example is borrowed from Haar et al. (2017), where it is used to demonstrate that active diagnosis has solutions that require observers of exponential sizes. Observable actions in this example are a, b, action u is unobservable, and only action c, d are controllable. The secret state is q_{n+1}. Let us consider how state estimation of an attacker evolves. The initial belief is $B_0 = \{q_0\}$. If an a is observed, then the belief can be updated to $B_{1,a} = \{q_1, u_1\}$, and if b is observed, then the belief is updated to $B_{1,b} = \{q_1, l_1\}$. Beliefs grow in this way. After n observations, beliefs are of the form $B_{n,w} = \{q_n, x_1 \ldots x_n\}$, where $x_i = u_i$ if the $n-i+1^{th}$ letter was an a and $x_i = l_i$ otherwise. Now, from a belief $B_{n,w}$, if an a or a b is observed, the belief becomes $B_{n,w.a} = B_{n,w.b} = \{q_n, x'_2 \ldots x'_n\}$ where $x'_i = u_i$ if $x_i = u_{i-1}$ and l_i otherwise. That is, beliefs encode the sequence of observations. Hence, after n observations, beliefs decrease and are updated accordingly with occurrences of as and bs. After $n-1$

additional observations of a, b starting from belief $B_{n,w}$, the remaining belief is of the form $B_{2n-1,w.w'} = \{q_n, u_n\}$ (from which letting action c occur represents no danger) or $B_{2n-1,w.w'} = \{q_n, l_n\}$ (from which letting action d occur represents no danger). One additional observation of a brings the belief in $B_{2n} = \{q_n\}$. From any belief of the form $B_{n+k,w.w'} = \{q_n, \ldots x_n\}$ the observation of c discards the possibility of being in any state other that q_n, u_n or l_n, and hence gives to the attacker the knowledge $B_{n+k,w.w'.c} = \{q_{n+1}, x_{n+1}\}$, where $x_{n+1} = u_{n+1}$ if $x_n = u_n$ in $B_{n+k,w.w'}$ and l_{n+1} otherwise. From this example, one can easily verify that the set of beliefs that can be computed in our example is of size $2.2^n + 3$. Note that the arena and the controllers computed from strategies in this arena have to correctly maintain all beliefs of attackers, as otherwise strategies can take the wrong decision. As a conclusion, for some automata whose size is in $O(n)$, the arena and the associated controller can be in $O(2^n)$. □

Proof of Theorem 2

Proof :

We need to consider the additional observation capacities of active attackers when composing a controller with an automaton. Given a deterministic automaton G and a (belief-based) controller C, we define the active product as the automaton $G/C^{active} = (Q_{/C}, \delta_{/C}, \ldots)$ where

- $(q_0, q0_C) \in Q_{/C}$
- for every $a \in \Sigma \setminus \Sigma_{Att}$ (Actions without attacker's inputs), for every state $(q, q_c) \in Q_{/C}$ where $q_c = (q, X)$, if $q' = \delta(q, a)$ and $a \in \lambda_C(q_c)$, then $(q', (q', \Delta(X, a)) \in Q_C$ and $((q, q_c), a, (q', q'_c)) \in \delta_{/C}$ Make sure correct Delta is used.
- for every state $(q, q_c) \in Q_{/C}$ where $q_c = (q, X)$, for every attacker input $?x$ that is not in $\Gamma(q) \cap \lambda(q_c)$, $(q, q'_C) \in Q_{/C}$, with $q'_C = (q, RefAct(X, \{?x\})$ and $((q, q_c), \epsilon, (q', q'_c)) \in \delta_{/C}$.

As in the proof of theorem 1, for a given strategy σ_2, we will design a controller C_{σ_2} as an automaton $(Q_{C_{\sigma_2}}, q0_{C_{\sigma_2}}, \lambda_{C_{\sigma_2}})$. States of C_{σ_2} are of the form (q, X, c) where q is a state of G, X is a belief of the attacker, and $c \subseteq 2^\Sigma$ a control. We define $\delta_{C_{\sigma_2}}$ as the set of transitions of the form $((q, X, c), a, (q', X', c'))$, such that there exists a path in $\mathcal{H}_G{}^{active}$ that complies with σ_2 from a node $n = (q, X, c, p)$ to another node $n' = (q', X', c', p')$, in which a is the only observable action appearing along this path, and $p, p' \in \{Att, sys\}$. Note that as this path contains a single observable action, it necessarily includes a decision of the controller in the arena. Such paths can be for instance of the form $\pi_1.\pi_c.\pi_2$ where:

$$\pi_1 = (q, X, c, sys) \xrightarrow{\epsilon} (q, X, c, att) \xrightarrow{u_1}$$
$$(q, X_1, c, sys) \xrightarrow{\epsilon} \ldots (q, X, c, att) \overset{u_k}{(} q, X_k, c, sys)$$
$$\pi_C = \longrightarrow (q, X_k, c, sys) \xrightarrow{a} (q, X_k, a, \delta(q, a))$$
$$\xrightarrow{(a,c')} (q', X_{k+1}, c', sys')$$
$$\pi_2 = (q', X_{k+1}, c', sys') \xrightarrow{\epsilon} (q', X_{k+1}, c', att) \xrightarrow{u_{k+1}}$$
$$\ldots (q', X_{k+2}, c', att) \xrightarrow{u_{k'}} (q, X_{k'}, c, sys)$$

with $(q', X_k, c', sys) \in \sigma_2(q, X_k, a, \delta(q, a))$, and each u_i is an attacker's input that is not feasible in current state or due to control. Similarly, path starting or ending on

attacker nodes follow the same scheme, and one also has to consider path in which an input of the attacker results in state change, i.e.,. where $\pi_C = (q, X_k, c, att) \xrightarrow{?x} (q, X_k, ?x, \delta(q, ?x)) \xrightarrow{(?x,c')} (q', X_{k+1}, c', sys')$. We obviously set $q0_{C_{\sigma_2}} = (q_0, \{q_0\}, \sigma'_2(q, \{q_0\}))$, and $\lambda_{C_{\sigma_2}}(q, X, c) = c$. Notice that this controller is non-deterministic, but that the control affected to every state depends only on the position reached immediately after an observable action.

Let us now show that if a strategy σ_2 is a winning strategy, then no state that leaks secret S is reachable in $G/\mathcal{C}_{\sigma_2}$.

Suppose that there exists a path $\rho = (q_0, q0_{C_{\sigma_2}}) \xrightarrow{a_1} \dots \xrightarrow{a_k} (q_k, q_{c_k})$ in $G/\mathcal{C}_{\sigma_2}$ where $q_{c_k} = (q_k, X_k, c)$ with $X_k \subseteq S$. Every step $(q_i, q_{c_i}) \xrightarrow{a_k} (q_{i+1}, q_{c_{i+1}})$ can be mapped to a path $\rho_{1,i}.\rho_{C,i}.\rho_{2,i}$ that contains a single action that is an observable action or an allowed input of the attacker in the arena. Notice also that this path conforms to strategy σ_2. As such steps are allowed for every $i \in 1..k$, this means that there exists a path in the arena that is compatible with σ_2 and nevertheless reaches a node (q, X_k, c, p) where $p \in \{sys, Att\}$. To win, players 1 And 3 simply have to remember a prefix of path ρ_{bad}, and play their next transition in this path. Hence, there exists strategies σ_1, σ_3 that allow reaching a position where state estimates are contained in S. So, σ_2 is not a winning strategy for the controller (contradiction). So, if σ_2 is a winning strategy for player 2 in $\mathcal{H}_G{}^{active}$, then $\mathcal{C}_{\sigma_2}$ is a controller guaranteeing opacity of S in G.

Let us now assume that there exists a non-deterministic controller $C = (Q_C, \delta_C, q0_C, \lambda_C)$ for G. For every word w in $\mathcal{L}(G/C)$, there exists several paths from $(q_0, q0_C)$ to states from set $R_w = \{(q_i, q_{c,i})\}_{i \in 1..k}$. All states in R_w are labeled by the same control $\lambda(R_w)$. Let us assume that C is a controller that enforces opacity of S w.r.t. attacker $\mathcal{A}_5$, i.e.,. for every run ρ of G/C, $\Delta_{\mathcal{A}_5}(\{q_0\}, \mathcal{A}_5(\rho)) \nsubseteq S$.

Let us define the positional strategy σ_C for player 2 in the three-player arena as follows: For every run $\pi = v_0 \dots \xrightarrow{a_1} \dots \xrightarrow{a_k} v_k$ in the arena ending on a node (q, X, a_k, q'), and labeled by a sequence of actions $a_1 \dots a_k$, we let $\sigma_C(\pi) = \lambda(Q_{C,\pi})$, where $Q_{C,\pi} = \{(q, X) \mid (q_0, q0_C) \xrightarrow{a_1 \dots a_k} (q, X)$ is the set of states reached in C after reading labels in π. Let us now assume that there exists a run π in the arena that is compatible with σ_C but allows to reach Win_S. π is of the form

$$\pi = (q_0, \{q, 0\}, \epsilon) \xrightarrow{c_0} (q_0, c_0, \{q_0\}, sys) \dots$$
$$\xrightarrow{a} (q_i, X_i, a, q'_i) \xrightarrow{c_i} (q'_i, X'_i, c_i, sys) \dots$$
$$\dots \longrightarrow (q_k, X_k, c_k, p)$$

where p is either the system or the active attacker. One can show by induction that for every node (q_i, X_i, a_i, q'_i) there exists a path in G/C from its initial node to (q_i, X_i, a, q'_i) labeled by the sequence of actions $a_1 \dots a_i$ appearing in $\pi_{[0..i]}$ (the restriction of π to its first i steps). This is in particular true for the first control node before $n_k = (q_k, X_k, c_k, p)$. Let us call $n_j = (q_j, X_j, a_j, q'_j)$ this node. As there is no control node between n_j and n_k, then all moves between n_j and n_k either give control to the attacker or refine the attacker's knowledge by playing refused inputs. Hence there exists a sequence of ϵ moves from $(q_j, (q_j, X_j))$

to $(q_k, (q_k, X_k))$ in G/C. (Contradiction). So, σ_C is a strategy for the three-player game. $\square$

Proof of Corollary 5

Proof : We can reuse the complexity results from Chatterjee and Doyen (2014): A three-player game with partial information where player 1 is less informed than player 2 is usually in $2 - EXPTIME$. It is solved by first collapsing players 1 and 3 to obtain an exponentially larger game with two players and partial information. Two-player games of partial information are usually solved in exponential time by computing an exponentially larger arena that represents beliefs of players, and on which reachability games can be solved in linear time. This yields the doubly exponential complexity of reachability questions in three-player games. One can however notice an interesting property of models mentioned in Chatterjee and Doyen (2014): if player 2 has perfect information, then there is no exponential blowup when collapsing players 1 and 3. From a three-player reachability game over arena $\mathcal{H}_G{}^{active}$, one can hence compute a partial observation two-player game on an arena $\mathcal{H}'_G{}^{active}$ of size in $O(|\mathcal{H}_G{}^{active}|)$. Every strategy on $\mathcal{H}'_G{}^{active}$ has a counterpart in $\mathcal{H}_G{}^{active}$. Last, to solve this game, one just have to compute beliefs of players. As player two has perfect information, it remains to consider beliefs of player 1 when visiting nodes of the arena. One can immediately notice that the belief of player 1 is exactly the state estimation contained in positions of $\mathcal{H}_G{}^{active}$. This situation was already noticed for active diagnosis questions in Bertrand et al. (2014). Hence no additional exponential blowup is needed to solve the two-player partial information reachability game over $\mathcal{H}'_G{}^{active}$ and we obtain that checking existence of a controller for opacity of S in G can be done in $EXPTIME$. $\square$