Overview of Discrete Event Systems Opacity: models, validation, and quantification

Romain Jacob, Jean-Jacques Lesage, Jean-Marc Faure

Overview of Discrete Event Systems Opacity: models, validation, and quantification[☆]

Romain Jacob[a,*], Jean-Jacques Lesage[a,*], Jean-Marc Faure[b]

[a] *LURPA, ENS Cachan, Univ. Paris-Sud, Université Paris-Saclay, 94235 Cachan, France*
[b] *LURPA, ENS Cachan, Univ. Paris-Sud, Supmeca, Universit Paris-Saclay, 94235 Cachan, France*

Abstract

Over the last decade, opacity of discrete event systems (DES) has become a very fertile field of research. Driven by safety and privacy concerns in network communications and online services, much theoretical work has been conducted in order to design opaque systems. A system is opaque if an external observer in unable to infer a "secret" about the system behavior. This paper aims to review the most commonly used techniques of opacity validation for deterministic models and opacity quantification for probabilistic ones. Available complexity results are also provided. Finally, we review existing tools for opacity validation and current applications.

Keywords: Opacity, Discrete event systems, Validation, Verification, Enforcement, Quantification, Secrecy, Privacy, Security

1. Introduction

Online services and network communications have become ubiquitous over the past 30 years. This evolution in our everyday life brought along new preoccupations regarding security and privacy. Despite continuously releasing tons of information about everything we do and think, we still want some information to remain secret. Thus, a new problem has arisen in computer science, called *Information Flow*. It characterizes the (possibly illegal and indirect) transmission of secret data from a high level user to a low level one. Various information flow properties have been defined in the literature: anonymity, noninterference, secrecy, privacy, security, and opacity; e.g., refer to Schneider & Sidiropoulos (1996); Focardi & Gorrieri (1994); Hadj-Alouane et al. (2005); Bérard et al. (2015a); Alur et al. (2006).

In this paper, we focus on *opacity*. It is a general information flow property: anonymity and secrecy can be formulated as opacity problems. Opacity characterizes whether a given "secret" about a system behavior is hidden or not from an external observer, further called the intruder. It is assumed the intruder has full knowledge of the system's structure but only partial observability. Based on its observations, the intruder constructs an estimate of the system's behavior. The secret is said to be opaque if the intruder's estimate never reveals the system's secret. Specifically, the system is opaque if, for any secret behavior, there exists, at least, one other non-secret behavior that looks the same to the intruder.

Opacity is a rather recent field of research. It was first introduced in 2004 in computer science to analyze cryptographic protocols (Mazaré, 2004). It reached the discrete event systems (DES) community with the work of Bryans et al. (2005) which investigated opacity in systems modeled as Petri nets. Secrets were set as predicates over Petri net markings (i.e., states). In Bryans et al. (2008), previous work was extended by investigating opacity in labeled transition systems (LTS), in which secrets were defined as predicates over runs. More recently, Saboori and Hadjicostis investigated state-based opacity properties using finite-state automata (FSA) models (Saboori & Hadjicostis (2007); Saboori (2011)). Many researchers have considered the validation of opacity properties which spans from system's opacity verification (e.g., Badouel et al. (2007); Hadjicostis & Keroglou (2014); Saboori & Hadjicostis (2008b, 2009, 2010a, 2013)) to the synthesis of a controller/scheduler which assures opacity, either through supervision (Dubreil (2010); Dubreil et al. (2008); Saboori & Hadjicostis (2008a)) or enforcement (Falcone et al. (2014)). In a general way, opacity validation w.r.t. a given secret and intruder is a yes/no question for logical models (whether with deterministic or nondeterministic transition function). In the opposite, if one is interested in quantifying the risk of possible information leakage from the system, it implies usage of probabilistic models (e.g., Bérard et al. (2010, 2015a); Ibrahim et al. (2014); Keroglou & Hadjicostis (2013); Saboori & Hadjicostis (2010b,a)).

[☆] This paper is an extended version of one published in the proceedings of the 5^{th} IFAC International Workshop On Dependable Control of Discrete Systems (DCDS'15), May 26-28, 2015.

[*] Corresponding authors

Email addresses: rjacob@ens-cachan.fr (Romain Jacob), jean-jacques.lesage@ens-cachan.fr (Jean-Jacques Lesage), jean-marc.faure@ens-cachan.fr (Jean-Marc Faure)

From a practical point of view, these properties are of great interest for anyone aiming for more privacy, safety, or even secrecy, in communication protocols, complex networked systems, or even a simple software architecture. It is ever more common to have privacy-related specifications in both software and hardware design. Using opacity theory, one can formally verify whether or not these specifications are satisfied, or, at least, have a quantitative measure for the risk of violation.

This paper aims to provide a comprehensive and general review of opacity related work considering DES models, hence, we purposefully leave out too technical details. We assume the reader has a general knowledge of DES theory and practice and of classically related problems (i.e., formalism of finite automata and probabilistic automata, diagnosis, verification, supervisory control...). In case of need, please refer to Cassandras & Lafortune (2008) for more information on these problems. This paper is an extended version of a paper presented by the authors at the 5^{th} IFAC International Workshop On Dependable Control of Discrete Systems (DCDS'15)(Jacob et al., 2015).

After introducing relevant notations in Section 2, we synthesize different notions of opacity used in the literature in Section 3. Section 4 reviews validation methods of various opacity properties. Section 5 presents extensions to probabilistic models and Section 6 summarizes decidability and complexity of most approaches surveyed in this paper. Finally, applications of opacity in DES are presented in Section 7 and Section 8 suggests some perspectives for further research.

2. Preliminaries

Let E be an alphabet of events. E^* is the set of all finite strings composed of elements of E, including the empty string ε. A language $L \subseteq E^*$ is a set of finite-length strings of labels in E. For a string t, $|t|$ denotes the length of t. For a string ω, $\overline{\omega}$ denotes the prefix-closure of ω and is defined as $\overline{\omega} = \{t \in E^* | \exists s \in E^*, ts = \omega\}$. The post-string ω/s of ω after s is defined as $\omega/s = \{t \in E^*, st = \omega\}$.

A finite-state automaton $G = (X, E, f, X_0)$ is a 4-tuple composed of a finite set of states $X = \{0, 1, ..., N-1\}$, a finite set of events E, a partial state transition function $f : X \times E \to X$, and a set of initial states X_0. The function f is extended to the domain $X \times E^*$ in the usual manner. The language generated by the system G describes the system's behavior and is defined by $\mathcal{L}(G, X_0) = \{s \in E^* | \exists i \in X_0, f(i, s) \text{ is defined}\}$; it is prefix-closed by definition.

Note that in opacity problems, the initial state needs not to be known *a priori*, therefore, we have a *set* of initial states instead of a single initial state. We consider partially observable systems. The event set is partitioned into an observable set E_o and an unobservable set E_{uo}. Given a string $t \in E^*$, its observation is the output of the natural projection function $P : E^* \to E_o^*$, which is recursively defined as $P(te) = P(t)P(e)$ where $t \in E^*$ and $e \in E$. The projection of an event $P(e) = e$ if $e \in E_o$, while $P(e) = \varepsilon$ if $e \in E_{uo} \cup \{\varepsilon\}$. Finally, for a language $J \subseteq E^*$, the inverse projection is defined as $P^{-1}(J) = \{t \in E^* : P(t) \in J\}$.

3. Opacity of discrete event systems

In this section, we formalize different opacity properties of DES. In the general case, the intruder is assumed to have full knowledge of the system structure (plus eventually of the system's controller) but he/she has only partial observability over it. Opacity is parameterized by a secret predicate S and by the intruder's observation mapping P over the system's executions. A system is opaque w.r.t. S and P if, for any secret run in S, there is another run not in S which is observably equivalent.

In cases of DES models, the secret predicate S can be of two classes: a subset of executions (or parts of executions) or a subset of states. This classifies opacity properties into two families: *language-based opacity* and *state-based opacity*.

3.1. Language-based opacity – LBO

LBO has been formalized in different ways in the literature. It was first introduced in Badouel et al. (2007) and Dubreil et al. (2008). LBO (also referred to as trace-based opacity) is defined over a secret behavior described by a language $L_S \subseteq E^*$. The system is opaque w.r.t. L_S and the projection map P if no execution leads to an estimate that is completely inside the secret behavior. Alternatively, in Lin (2011), LBO is defined over two sublanguages of the system, $(L_1, L_2) \subseteq (\mathcal{L}(G, X_0))^2$. Sublanguage L_1 is opaque w.r.t. L_2 and an observation mapping θ if the intruder confuses every string in L_1 with some strings in L_2 under θ. In most recent papers considering LBO, the latter definition is used with the observation mapping θ being the natural projection mapping P.

Definition 1 (LBO – Strong Opacity). Given a system $G = (X, E, f, X_0)$, a projection P, a secret language $L_S \subseteq \mathcal{L}(G, X_0)$, and a non-secret language $L_{NS} \subseteq \mathcal{L}(G, X_0)$, G is language-based opaque if for every string $t \in L_S$, there exists another string $t' \in L_{NS}$ such that $P(t) = P(t')$.
Equivalently, $L_S \subseteq P^{-1}[P(L_{NS})]$.

The system is language-based opaque if for any string t in the secret language L_S, there exists, at least, one other string t' in the non-secret language L_{NS} with the same projection. Therefore, given the observation $s = P(t) = P(t')$, the intruder cannot conclude whether the secret string t or the non-secret string t' has occurred. Note that L_S and L_{NS} do not need to be prefix-closed in general, nor even regular.

Part of the literature refers to Definition 1 as *strong opacity*. In Lin (2011), a smoother opacity property is also introduced.

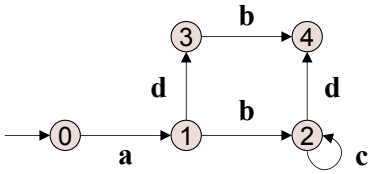


Figure 1: From Wu & Lafortune (2013) – Example 1 (LBO)

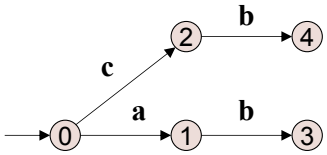


Figure 2: From Wu & Lafortune (2013) – The system G discussed in Example 2 (CSO) and 4 (IFO)

Definition 2 (LBO – Weak Opacity). Given a system $G = (X, E, f, X_0)$, a projection P, a secret language $L_S \subseteq \mathcal{L}(G, X_0)$, and a non-secret language $L_{NS} \subseteq \mathcal{L}(G, X_0)$, G is *weakly opaque* if for some string $t \in L_S$, there exists another string $t' \in L_{NS}$ such that $P(t) = P(t')$.
Equivalently, $L_S \cap P^{-1}[P(L_{NS})] \neq \emptyset$.

The system is weakly opaque if some strings in L_S are confused with some strings in L_{NS}. As a consequence, we can further define easily the property of *no opacity*.

Definition 3 (LBO – No opacity). L_S is *no opaque* w.r.t. L_{NS} and P if L_S is not weakly opaque w.r.t. L_{NS} and P.
Equivalently, $L_S \cap P^{-1}[P(L_{NS})] = \emptyset$.

Remark 1. It is shown in Ben-Kalefa & Lin (2009) that LBO properties are closed under union, but may not be closed under intersection. They further discuss how to modify languages to satisfy the strong, weak, and no opacity by investigating sublanguages and superlanguages.

Example 1. From Wu & Lafortune (2013) – Consider the system G in Fig. 1 with $E_o = \{a, b, c\}$.

It is language-based opaque when $L_S = \{abd\}$ and $L_{NS} = \{abcc^*d, adb\}$ because whenever the intruder sees $P(L_S) = \{ab\}$, it is not sure whether string abd or string adb has occurred.

However, this system is not language-based opaque if $L_S = \{abcd\}$ and $L_{NS} = \{adb, abd, abccc^*d\}$; no string in L_{NS} has the same projection as the secret string $abcd$.

Remark 2. In general, LBO refers to strong opacity in the literature, as in the rest of this paper.

3.2. State-based opacity – SBO

The state-based approach for opacity of DES was introduced in Bryans et al. (2005) for Petri nets models then extended to FSA in Saboori & Hadjicostis (2007). The state-based approach relates to the intruder ability to infer that the system is or has been in a given "secret" state or set of states. Depending on the nature of the secret set, different opacity properties have been defined.

3.2.1. Current-State Opacity – CSO

CSO was first introduced in Bryans et al. (2005) and called *final opacity* in the context of Petri nets. The definition was then adapted to LTS in Bryans et al. (2008), and further developed in finite state automata models in Saboori & Hadjicostis (2007). A system is CSO if the intruder can never infer, from its observations, whether the current state of the system is a secret state or not.

Definition 4 (Current-State Opacity). Given a system $G = (X, E, f, X_0)$, a projection P, a set of secret states $X_S \subseteq X$, and a set of non-secret states $X_{NS} \subseteq X$, G is current-state opaque if
$\forall i \in X_0$ and $\forall t \in \mathcal{L}(G, i)$ such that $f(i, t) \in X_S$,
$\exists j \in X_0$ and $\exists t' \in \mathcal{L}(G, j)$ such that
$f(j, t') \in X_{NS}$ and $P(t) = P(t')$.

The system is CSO if for every string t that leads to a secret state, there exists another string t' leading to a non-secret state whose projection is the same. As a result, the intruder can never assert with certainty that the system's current state belongs to X_S.

Remark 3. In Bryans et al. (2005), the property of *always-opacity* is also introduced. A system is always-opaque (or *total-opaque* in Bryans et al. (2008)) over a set of runs if it is CSO for any state visited during these runs. This is equivalent to consider a set of secret states which lies on a prefix-closed language.

Example 2. From Wu & Lafortune (2013) – Consider G in Fig. 2 and the sets of secret and non-secret states $X_S = \{3\}$ and $X_{NS} = X \backslash X_S$.

If $E_o = \{b\}$, then G is current-state opaque because the intruder is always confused between ab and cb when observing b; that is, the intruder cannot tell if the system is in state 3 or 4.

However, if $E_o = \{a, b\}$, CSO does not hold because the intruder is sure that the system is in state 3 when observing ab.

3.2.2. Initial-State Opacity – ISO

ISO property relates to the membership of the system's initial state within a set of secret states. The system is *initial-state opaque* if the observer is never sure whether the system's initial state was a secret state or not.

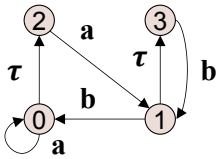


Figure 3: From Wu & Lafortune (2013) – The system G discussed in Example 3 (ISO)

Definition 5 (Initial-State Opacity). Given a system $G = (X, E, f, X_0)$, a projection P, a set of secret initial states $X_S \subseteq X_0$, and a set of non-secret initial states $X_{NS} \subseteq X_0$, G is initial-state opaque if
$\forall i \in X_S$ and $\forall t \in \mathcal{L}(G, i)$,
$\exists j \in X_{NS}$ and $\exists t' \in \mathcal{L}(G, j)$ such that $P(t) = P(t')$.

The system is ISO (or *initial-opaque* in Bryans et al. (2005)) if, for every string t that originates from a secret state i, there exists another string t' originating from a non-secret state j such that t and t' are observationally equivalent. Therefore, the intruder can never determine whether the system started from a secret state i or from a non-secret state j.

Example 3. From Wu & Lafortune (2013) – Consider G in Fig. 3 with $E_o = \{a, b\}$, $X_S = \{2\}$, and $X_{NS} = X \backslash X_S$. G is initial-state opaque because for every string t starting from state 2, there is another string $(\tau)t$ starting from state 0 that looks the same.

However, ISO does not hold if $X_S = \{0\}$. Whenever the intruder sees string aa, it is sure that the system originated from state 0; no other initial states can generate strings that look the same as aa.

Remark 4. There is one important difference to note between current-state and initial-state opacity in terms of monotony: initial state opacity exhibits a monotonic property (the set of possible initial states can only decrease as more observations become available), in contrast with current state opacity, for which there is no guarantee to obtain more relevant information over time.

Remark 5. Hadjicostis (2012) defines resolution of initial state w.r.t. a secret set of states S. It requires that when the system starts from a secret state, the observer will be able to eventually (i.e., after a finite sequence of events/observations) determine with certainty that the system's initial state lied within the set of secret states S. It is worth pointing out at this point that absence of resolution of initial state is necessary but not sufficient for ISO.

3.2.3. Initial-and-Final-State Opacity – IFO

In Wu & Lafortune (2013), the authors introduce initial-and-finite opacity. It is an extension of ISO and CSO which requires both the initial and final state to be hidden from the intruder. The only difference is that the secret is now defined over pairs of states (and not only states).

Definition 6 (Initial-and-Final-State Opacity). Given system $G = (X, E, f, X_0)$, projection P, set of secret state pairs $X_{SP} \subseteq X_0 \times X$, and set of non-secret state pairs $X_{NSP} \subseteq X_0 \times X$, G is initial-and-final-state opaque if
$\forall (x_0, x_f) \in X_{SP}$ and $\forall t \in \mathcal{L}(G, x_0)$ such that
$f(x_0, t) = x_f$,
$\exists (x_0', x_f') \in X_{NSP}$ and $\exists t' \in \mathcal{L}(G, X_0)$ such that
$f(x_0', t') = x_f'$ and $P(t) = P(t')$.

The system is initial-and-final-state opaque if for any string t that starts from x_0 and ends at x_f, with $(x_0, x_f) \in X_{SP}$, there exists another string t' starting from x_0' and ending at x_f', where $(x_0', x_f') \in X_{NSP}$, that has the same projection. Therefore, the intruder can never determine whether the initial-and-final state pair is a secret pair or a non-secret pair.

Remark 6. ISO and CSO are special cases of IFO.
To obtain an ISO problem from an IFO formulation, set $X_{SP} = X_S \times X$ and $X_{NSP} = X_{NS} \times X$.
Likewise, to obtain a CSO problem, set $X_{SP} = X_0 \times X_S$ and $X_{NSP} = X_0 \times X_{NS}$.

Example 4. From Wu & Lafortune (2013) – Consider again G in Fig. 3 and take $X_{SP} = \{(3, 1)\}$.

G is initial-and-final state opaque if the non-secret state pair set is $X_{NSP} = \{(1, 0), (1, 1), (1, 2), (1, 3)\}$.

However, initial-and-final-state opacity property does not hold if we take $X_{NS} = \{(0, 0)\}$ since $(0, 0)$ is the only state pair that corresponds to string aa; no other state pairs give strings that look the same as aa.

3.2.4. K-step opacity

Except for ISO, previously defined opacity properties do not consider the system behavior once it has exited a secret state. A more general problem would be to keep secret the fact the system *was* in a secret state a few steps ago. This property is called *K-step opacity* and was first introduced in Saboori & Hadjicostis (2007).

Definition 7 (K-step (weak) opacity).
Given a system $G = (X, E, f, X_0)$, a projection P, and a set of secret states $X_S \in X$, G is K-step (weakly) opaque w.r.t. X_S and P for $K \geq 0$ (or (X_S, P, K)-(weakly) opaque) if
$\forall i \in X_0$, $\forall t \in \mathcal{L}(G, i)$, and $\forall t' \in \overline{t}$ such that
$f(i, t') \in X_S$ and $|P(t)/P(t')| \leq K$,
$\exists j \in X_0$, $\exists s \in \mathcal{L}(G, j)$, and $\exists s' \in \overline{s}$, such that
$f(j, s') \in X_{NS}$, $P(s) = P(t)$ and $P(s') = P(t')$.

This definition can be reformulated as in Falcone et al. (2014). The system is (X_S, P, K)-opaque if for every execution t of G and for every secret execution t' prefix of t with an observable difference inferior to K, there exists two executions s and s' observationally equivalent respectively to t and t' such that s' is not a secret execution (i.e., which does not bring the system in a secret state).

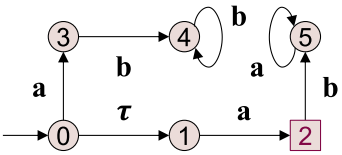


Figure 4: From Falcone & Marchand (2013) – System discussed in Example 5.

Example 5. From Falcone & Marchand (2013) – Consider G in Fig. 4 with $E_o = \{a, b\}$, $X_S = \{2\}$, and $X_{NS} = X \backslash X_S$. Secret state is shown as red square.

G is $(X_S, P, 1)$-opaque, but it is not $(X_S, P, 2)$-opaque, as only $(\tau)aba$ is a compatible execution with the observation aba. Hence, after the second a has occurred, the intruder can deduce that the system was in state 2 two steps before.

Remark 7. K-step opacity is a direct extension of CSO. CSO is equivalent to 0-step opacity (Saboori & Hadjicostis (2007)).

In Falcone et al. (2014), Definition 7 is referred to as *K-step weak opacity*. The property of *K-step strong opacity* holds if the system is K-step weakly opaque and there exists a trace of the system (observably equivalent to the actual execution) which does not cross any secret state over the last K steps. This can be formalized with the following definition.

Definition 8 (K-step strong opacity).
Given a system $G = (X, E, f, X_0)$, a projection P, and a set of secret states $X_S \in X$, G is K-step strongly opaque w.r.t. X_S and P for $K \geq 0$ (or (X_S, P, K)-strongly opaque) if
$\forall i \in X_0$ and $\forall t \in \mathcal{L}(G, i)$,
$\exists j \in X_0$ and $\exists s \in \mathcal{L}(G, j)$ such that
$P(s) = P(t)$ and
$\forall s' \in \overline{s}$, $|P(s)/P(s')| \leq K$
$\Rightarrow \exists j' \in X_0$ such that $f(j', s') \in X_{NS}$.

Example 6. From Falcone & Marchand (2013) – Consider G in Fig. 5 with $E_o = \{a, b, c\}$, $X_S = \{2, 7\}$, and $X_{NS} = X \backslash X_S$. Secret states are shown as red square.

G is (X_S, P, K)-weakly opaque for any $K \in \mathbb{N}$. The intuition is that we will always have confusion between pairs of states $(5, 2)$, $(6, 3)$, and $(7, 4)$, such that the intruder will never know with absolute certainty that the system was in state 2 or 7 at a given point in time.

It also holds that the system is $(X_S, P, 1)$-strongly opaque. However, it is not $(X_S, P, 2)$-strongly opaque since after observing aba, we know that the system is either in state 7 (which is a secret state) or in state 4, which implies it was in state 2 (the other secret state) two steps ago. Ultimately, we know for sure that the system was in a secret state at most 2 steps ago.

Remark 8. In general, K-step opacity refers to weak opacity in the literature, as in the rest of this paper.

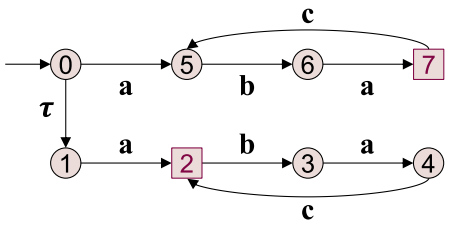


Figure 5: From Falcone & Marchand (2013) – System discussed in Example 6.

3.2.5. Infinite-step opacity

In Saboori & Hadjicostis (2009) and Saboori (2011), K-step opacity has been further extended to infinite-step opacity.

Definition 9 (Infinite-step opacity). Given a system $G = (X, E, f, X_0)$, a projection P, and a set of secret states $X_S \in X$, G is infinite-step opaque w.r.t. X_S and P, or (X_S, P, ∞)-opaque, if
$\forall i \in X_0$, $\forall t \in \mathcal{L}(G, i)$, and $\forall t' \in \overline{t}$ such that
$f(i, t') \in X_S$,
$\exists j \in X_0$, $\exists s \in \mathcal{L}(G, j)$, and $\exists s' \in \overline{s}$, such that
$f(j, s') \in X_{NS}$, $P(s) = P(t)$, and $P(s') = P(t')$.

A system is infinite-step opaque if, for every execution of the system, after having observed an arbitrarily long sequence of events, the intruder cannot infer that the system was in a secret state at some point (at any step back in the execution).

3.3. Transformations between different opacity properties

The aforementioned opacity properties have strong connections between each other. Several works have addressed the translation between them.

Saboori & Hadjicostis (2008a) adapts the language-based definition to ISO in order to apply supervisory control methods (refer to Section 4.2). On the contrary, Cassez et al. (2009, 2012) describes transformations from LBO to CSO. In Wu & Lafortune (2013), the authors extend these works and provide a full transformation mapping between LBO, CSO, ISO, and IFO.

In addition, we already mentioned that K-step opacity is an extension of CSO. CSO is equivalent to 0-step opacity.

Finally, in Saboori (2011), a language-based translation of K-step opacity is suggested: *trace-based K-step opacity*. It is a special case of K-step opacity which, to the best of our knowledge, has never been used or considered in any other work. It is mentioned here for the sake of completeness.

3.4. Distributed opacity

Even though most opacity-related studies account for a single intruder only, a few of them consider distributed notions of opacity. Hence, Badouel et al. (2007) consider multiple intruders, each of them having its own observation mapping and secret of interest. The system is said to

be *concurrently opaque* if all secrets are safe. A different notion, called *joint opacity* is presented in Wu & Lafortune (2013) and Wu (2014). In this setting, several intruders collaborate through a coordinator in order to discover the same secret. Finally, Paoli & Lin (2012) consider decentralized framework with and without coordination among agents and formalize definitions of decentralized opacity. It is shown to be an extension of co-observability, used in traditional supervisory control (Ramadge & Wonham, 1989).

3.5. Infinite DES models

Up to a few years ago, opacity-related studies only considered finite-state DES models. There are recent works addressing extensions to infinite-state. CSO and diagnosability verification are investigated for infinite-state DES modeled by pushdown automata in Kobayashi & Hiraishi (2013) (therein called pushdown systems), as well as in Chédor et al. (2014) and Chédor (2014), in the more general setting of recursive tile systems.

Extension to timed DES has also been considered, but it has been shown in Cassez (2009) that even for a very restrictive class of Timed Automata, opacity is already undecidable for a problem in dense-time. However, considering not dense-time domains (e.g., $\mathbb{N}$) may render the opacity problem tractable.

3.6. Relation with other DES and information flow properties

We mentioned in Section 1 that opacity is a very general information flow property. Relations between several of these properties can be easily drawn.

Easiest is *secrecy*, in which the system predicate is *secret* if the predicate and its complement are simultaneously opaque (Badouel et al. (2007)). It is also referred to as *symmetrical opacity* (Bérard et al. (2015b)). It was shown that *anonymity* (Bryans et al., 2008; Bérard & Mullins, 2014) and some *non-interference* problems (Cassez et al., 2007; Bryans et al., 2008; Benattar et al., 2015; Bérard et al., 2015c) may be reduced to opacity by using suitable observation functions and depending on the type of secret under consideration. The equivalence between opacity and *intransitive non-interference* is proven in Mullins & Yeddes (2013). Lin (2011) also establishes links between opacity, anonymity, and secrecy and shows that observability, diagnosability, and detectability can be reformulated as opacity as well.

More generally, opacity is a problem closely related to diagnosis (Sampath et al., 1996; Zaytoon & Lafortune, 2013): for opacity to hold, the secret should not be diagnosable from the viewpoint of the intruder. Some instances of opacity problem can be formulated as diagnosis ones (e.g., resolution of ISO from Hadjicostis (2012)). As a result, several opacity-related works try to bridge with the huge amount of work done in the diagnosis community; e.g., Dubreil et al. (2009); Dubreil (2010); Kobayashi & Hiraishi (2013); Chédor et al. (2014).

4. Ensuring opacity

Traditional opacity formulations from the literature were presented in Section 3. The questions are now the following: How does one know that a given system G is opaque w.r.t. a secret and the information available to intruders? Furthermore, if it is not, what can be done to make it opaque? These questions have been continuously addressed and this section aims to synthesize the approaches available in the literature.

There are three main approaches to ensure opacity properties of DES:

1. *Verification*, which roughly consists in model-checking opacity properties;
2. *Supervisory control* theory (SCT), which restricts the system's behavior in order to preserve the secret;
3. *Enforcement*, which inputs observable events of the systems and outputs (possibly) modified information to observers, such that the secret is preserved.

The key difference to note between SCT and enforcement is that SCT constrains the system behavior (by restraining its output) by means of a supervisor while enforcement allows the system free-behavior but post processes all its output. For more details about these approaches and their pros-and-cons, one can refer Falcone et al. (2014). The three mechanisms are illustrated by Fig. 6.

4.1. Verification of opacity properties

As mentioned in Introduction, opacity is a rather recent field of research. Verification of opacity relates directly to the general problem of verification of DES, which has been extensively studied and is well-known. It was shown in Cassez et al. (2012) that opacity verification is equivalent to the universality problem (i.e., whether or not the system admits all possible words constructed on its alphabet). The specific task to perform is to encode the opacity property of interest (refer to Section 3) such that classical model-checking approaches and tools can be used. However, such opacity encoding might not be trivial, like for K-step opacity for instance. Verification of K-step opacity was tackled in Saboori & Hadjicostis (2011b) by use of two types of K-delay state estimators. It is also developed in Falcone et al. (2014).

Remark 9. We mentioned in Section 3.6 that opacity can be related to diagnosability. Dubreil et al. (2009) investigate the use of techniques from diagnosis of DES (Zaytoon & Lafortune (2013)) to detect and predict the flow of secret information and construct a monitor that allows an administrator to detect it.

In the sequel of this section, we choose to further present solely control and enforcement approaches, for which specific opacity-related methods have been developed. For more details about verification of LBO, refer to Lin (2011);

(a) Verification (b) Supervision (c) Enforcement

Figure 6: The main three approaches for ensuring opacity

for SBO, refer to Falcone et al. (2014) and Hadjicostis & Keroglou (2014). General results about verification of DES can be found in Cassandras & Lafortune (2008).

4.2. Supervisory control theory – SCT

The potential use of SCT for opacity validation of DES is rather obvious. Several works present construction of minimally restrictive opacity-enforcing supervisory controllers; e.g., Takai & Oka (2008); Takai & Kumar (2009); Saboori & Hadjicostis (2008a); Saboori (2011); Saboori & Hadjicostis (2012); Ben-Kalefa & Lin (2011). It is shown that optimal control always exists for strong-opacity (Dubreil et al., 2010; Dubreil, 2010).

In these approaches, the intruder is generally assumed to have full knowledge of the supervisor's structure in addition to the system's. Moreover, the set of events the intruder can observe is fixed.

The applicability of SCT depends on the hypothesis made on the system's model. Given E_I, E_O and E_C being respectively, the set of events observable by the intruder, the set of events observable by the supervisor, and the set of controllable events, SCT can be directly applied in the following cases (Dubreil (2010)):

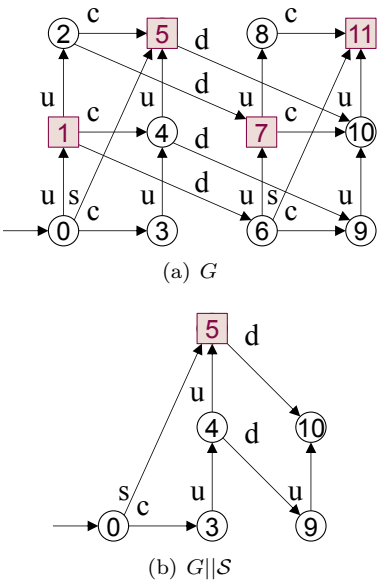
1. $E_C \subseteq E_O \subseteq E_I$;
2. $E_I \subseteq E_C \subseteq E_O$.

Furthermore, to deal with the following two cases, slight extensions of SCT have been suggested in Dubreil (2010) and Dubreil et al. (2010)

3. $E_C \subseteq E_I \subseteq E_O$;
4. $E_I \subseteq E_O$ and $E_C \subseteq E_O$
   but without E_C and E_I being comparable.

Example 7. From Dubreil et al. (2010). Let G be the transition system depicted in Fig. 7(a), with $X = \{0, ..11\}$, $X_0 = \{0\}$, $X_S = \{1, 5, 7, 11\}$ and $E = \{u, d, c, s\}$. Let f be the transition function defined according to the arcs of the figure. Secret states are shown as red square.

G is a representation of all sequences of possible moves of an agent in a three story building with a south wing and a north wing, both equipped with lifts and both connected by a corridor at each floor. Moreover, there is a staircase that leads from the first floor in the south wing to the third floor in the north wing. The agent starts from the first floor in the south wing. He can walk up the stairs



(a) G

(b) $G||\mathcal{S}$

Figure 7: (a) G: Nominal model of the system; (b) $G||\mathcal{S}$: Minimally restrictive supervisor ensuring opacity of G regarding the agent being at a secret floor.

(s) or walk through the corridors (c) from south to north without any control. The lifts can be used several times one floor upwards (u) and at most once one floor downwards (d) altogether. The moves of the lifts are controllable. Thus $E_C = \{u, d\}$. The secret is that the agent is either at the second floor in the south wing or at the third floor in the north wing. The adversary may gather the exact subsequence of moves in $E_I = \{u, c, s\}$ from sensors, but he cannot observe the downwards moves of the lifts. Furthermore, all events are observable to the supervisor, i.e., $E_O = E$. Hence, this example falls into the fourth of the aforementioned cases.

The derivation of the minimally restrictive supervisor ensuring the secret will not be disclosed is performed by a sequential derivation of *condensed state estimators* and their *losing configurations*. In a nutshell, a configuration is combination of the true system state (the upper line) and the best estimate the intruder can make of the system state (the lower line). A losing configuration is such that the intruder's estimate belongs to the set of secret states. One can track controllable actions backward from losing configurations on acyclic paths of the condensed estimator and disable the last controllable transition on each losing path. The next condensed state estimator is then derived,

taking into account the newly disabled transitions. Once we get to a condensed state estimator without any losing configuration, we have reached the minimally restrictive supervisor. Refer to Dubreil et al. (2010) for more details.

Fig. 8(a), (b), and (c) show the subsequent condensed state estimators needed to solve this example. losing configurations are boxed red, the controllable transitions to disabled in the next step are in dashed lines. Fig. 7(b) is the resulting supervisor.

One can notice that one secret state only remains in the controlled system. This means others secret states cannot be opaque to the intruder. If the agent aims to go and stay in state 1, 7 or 11, it will be inferred by the intruder. This is an example of ensuring *current-state opacity by supervisory control*. Note that this system is not 1-opaque (refer to Def. 7).

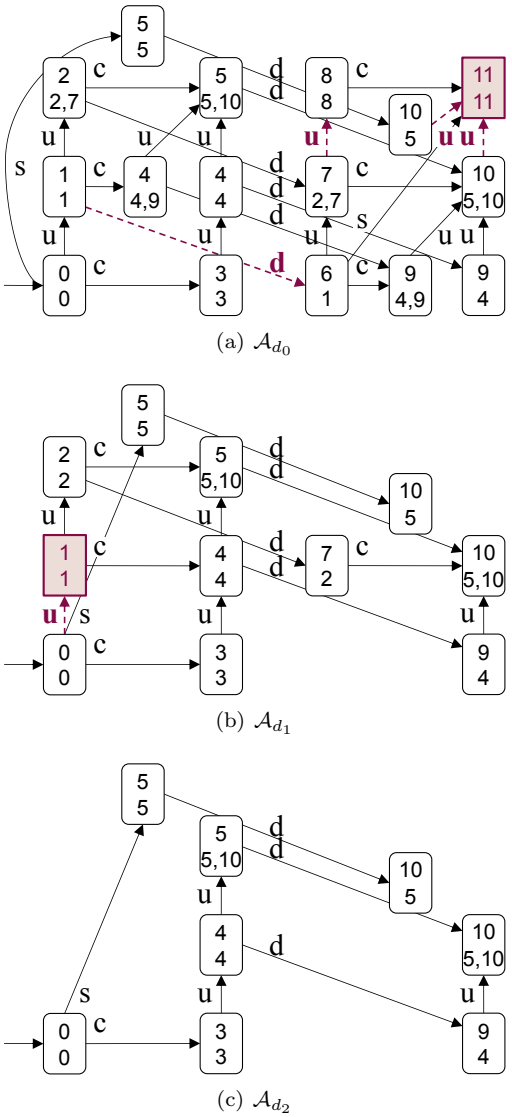
In Badouel et al. (2007), the authors solved the problem of concurrent secrecy (Section 3.4) using SCT. Sufficient conditions to compute an optimal supervisor preserving all secrets are provided, assuming that the supervisor has complete knowledge of the system and full control over it.

The work of Ben-Kalefa & Lin (2011) considers the verification of both strong and weak LBO. It shows that the solution to the Strong-Opacity Control Problem (SOCP) exists and is unique if all controllable events are observable. However solutions for the Weak-Opacity Control Problem (WOCP) does not exist. This means that if a system is not weakly opaque w.r.t. a given secret language, there exists no controllable and observable sublanguage which can assure weak opacity.

In Darondeau et al. (2014), the authors lift the opacity enforcing control problem using SCT from a single finite transition systems to families of finite transition systems specified by modal transition systems (Larsen (1990)). The objective is to ensure opacity of a secret predicate on all LTS derived from a given modal transition system.

Using SCT is naturally more suited to language-based notions of opacity. However, the verification of initial state opacity has been addressed in Saboori & Hadjicostis (2008a) by means of reformulation of ISO into LBO, under regular SCT hypothesis (cases (1) and (2)). Similar work was performed in Saboori & Hadjicostis (2012) for infinite-step opacity even though it cannot be so easily translated to LBO. It is shown that the approach for ISO can be extended by using a finite bank of supervisors and ensure infinite-step opacity in a minimally restrictive way.

Remark 10. In a nutshell, supervisory control resumes to find the supremal sublanguage that ensures opacity. In Ben-Kalefa & Lin (2009), the authors further investigate language composition and show that opacity properties (with secrets being languages) are closed under union, but may not be closed under intersection. They also demonstrate the following results:



(a) $\mathcal{A}_{d_0}$

(b) $\mathcal{A}_{d_1}$

(c) $\mathcal{A}_{d_2}$

Figure 8: The three condensed state estimator; (a) $\mathcal{A}_{d_0}$: first step, one losing configuration $\{11, 11\}$, 4 transitions to disabled; (b) $\mathcal{A}_{d_1}$: second step, one losing configuration $\{1, 1\}$, 1 transition to disabled; (c) $\mathcal{A}_{d_2}$: third step, no more losing configuration, minimally restrictive supervisor is reached.

(i) the supremal strongly opaque sublanguage exists and is unique;

(ii) the minimal strongly opaque superlanguage exists but may not be unique;

(iii) the minimal weakly opaque superlanguage exists but may not be unique;

(iv) the supremal not opaque sublanguage exists and is unique.

4.3. Enforcement of opacity properties

Opacity enforcement at run-time was introduced in Schneider (2000) and recently surveyed in Falcone et al. (2014). Enforcement does not restrict the system behavior anymore. Instead, it "hides" some of the system's output events whenever it is necessary. It is a non-intrusive approach compared to supervision. There are three main methods used for opacity enforcement:

1. Deleting occurrences of events from the output;
2. Adding events to the output;
3. Delaying the output.

4.3.1. Deletion of events

Considering a trace observed by the intruder, it may happen that the observation of the next event discloses the secret. A simple idea is to hide the occurrence of this event from observation at run-time (and possibly only this single occurrence) to avoid information flow.

Main work achieving this is synthesized in Cassez et al. (2009) and Cassez et al. (2012). In this approach, the enforcer is a device called a mask. This mask restricts the observable outputs of the system either in a static or dynamic fashion. The latter case allows the mask to adapt to the intruder observation mapping (assumed to be dynamic) at each execution step.

Example 8. From Cassez et al. (2012). Consider the automaton G of Fig. 9, where the set of secret states is $X_S = \{2, 5\}$. If $E_O = E = \{a, b\}$, the system is not opaque (e.g., b^*ab leads in the set of secret states). If either $E_O = \{a\}$ or $E_O = \{b\}$, then it becomes opaque. Thus, one can define static sets of observable events, where at least one event will have to be permanently unobservable. This is a valid but very restrictive control. One could hide fewer events, the observable behavior of the system would be more important, and the control would be less restrictive. Thus, one should try to reduce as much as possible the hiding of events. On this particular example, we can be more efficient by using a *dynamic mask* that will render unobservable an event only when necessary. In this example, after observing b^*, the intruder knows that the system is still in the initial state. However, if a subsequent a follows, then the intruder should not be able to observe b as this particular b would revel the system is in a secret state. We can then design a dynamic events hider as follows: at the beginning, everything is observable; when an a occurs, the mask hides any subsequent b occurrence and permits only the observation of a. Once an a has been observed, the mask releases its hiding by letting both a and b be observable again. Hence, event deletion is minimal.

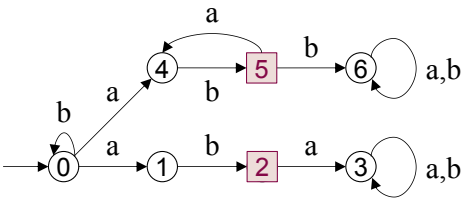


Figure 9: From Cassez et al. (2012) – Automaton G for Example 8

In Zhang et al. (2014) and Zhang et al. (2015), the authors introduced the Maximum Information Release Problem which aims to restrict as few occurrences of output events as possible. They consider both strong and weak opacity. This work is very similar to the enforcement by means of a mask. The main difference comes from the initial definition of opacity used. They use the language inclusion definition from Lin (2011), while Cassez et al. (2012) considers a state-based approach. This allows the Maximum Information Release Problem to adapt more easily to weak opacity, but the two methods are essentially the same.

4.3.2. Addition of events

Deleting events from the output was still considered as intrusive by some researchers. Even if the internal behavior of the system is no longer restricted (as it is with SCT), its actual output is.

To cope with this problem, Wu and Lafortune derived a method which artificially adds outputs to the set of observed events at run-time. This approach is called *insertion functions* (refer to Wu & Lafortune (2014); Wu (2014)). An insertion function is a monitoring interface at the system's output that changes it by inserting additional ("fake") occurrences of observable events.

Remark 11. These two approaches were suggested in Ligatti et al. (2005), which proposed an enforcement mechanism called edit-automata. This mechanism featured the idea of "suppressing" and "inserting" actions in the current execution of a system but without direct application to information flow and opacity.

4.3.3. Delay of events

The last approach to enforce opacity properties is to delay emissions of one or several events which would have disclosed the secret, up to the point where the disclosure is of no interest anymore, or the system reaches a state in which opacity holds again. This method allows the full system behavior as well, but can only apply to secrets for which time duration is of concern.

This approach has been presented in Saboori & Hadjicostis (2007) and applied to K-step (weak) opacity in Saboori & Hadjicostis (2011b). It was later extended in Falcone et al. (2014) to K-step strong opacity.

5. Quantifying opacity

We presented in Section 3 the main formulations of opacity properties which have been considered in the literature. With these definitions, even decidable problems (refer to Section 6) only provide a yes/no answer to the system's opacity. Supervisory control (Section 4.2) and enforcement (Section 4.3) can manage to turn a non-opaque system into an opaque one.

However, this only accounts for logical models, with deterministic transition function, which is known to be a strong limitation in practice. Thus, researchers extended some definitions and tried to quantify opacity in a probabilistic setting. That is, how can one evaluate the possible information leakage of a system w.r.t. a given secret? Hence, for a given system's execution, we do not ask if there exists an observably equivalent execution, but how many there are, with a probabilistic measure taking into account the likelihood of such executions.

The reader should note that there is no absolute consensus on the interpretation of primitives. Depending on the authors and the problem considered, the model, the type of secret, and the meaning of probabilities can all vary to some extend. We attempt to formulate thereafter the problem statements as clearly as possible.

5.1. Quantification of language-based opacity

Initial work on quantification of opacity properties was presented in Lakhnech & Mazaré (2005) and reviewed in Bryans et al. (2011). It provides quantitative measures of LBO in a probabilistic setting but it is limited to purely probabilistic models, based on labeled Markov chains.

In Bérard et al. (2015b), two dual notions of probabilistic opacity are introduced:

(i) *Liberal probabilistic opacity (LPO)* measures the probability for an intruder observing a random execution of the system to be able to gain information he can be sure about. This definition provides a measure of how insecure the system is. $LPO = 0 \Leftrightarrow LBO$. Hence, computation of LPO is irrelevant for opaque systems.

(ii) *Restrictive probabilistic opacity (RPO)* measures the level of certitude in the information acquired by an intruder observing the system. $RPO = 0$ means the system is never opaque, whichever the running execution. Hence, computation of RPO express "how opaque" an opaque system is, which is irrelevant for non-opaque systems.

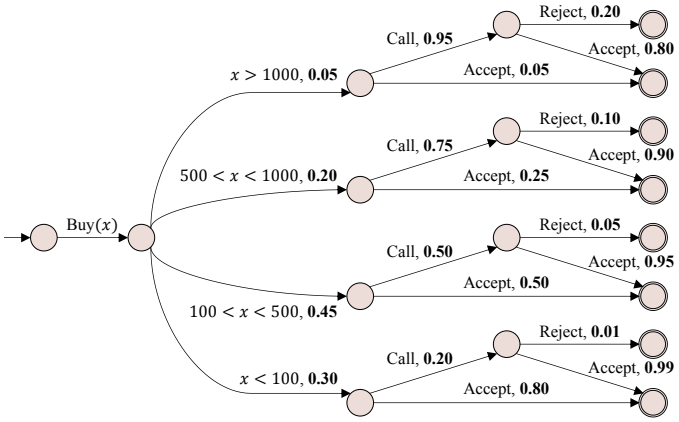


Figure 10: From Bérard et al. (2015b) – The Debit Card system of Example 9 – ,

Example 9. From Bérard et al. (2015a) – Consider a Debit Card system in a store. When a card is inserted, an amount of money x to be debited is entered, and the client enters his/her pin number (all this being gathered under the action $Buy(x)$). The amount of the transaction is given probabilistically as an abstraction of the statistics of such transactions. Provided the pin is correct, the system can either directly allow the transaction, or interrogate the client's bank for solvency. In order to balance the cost associated with this verification (bandwidth, server computation, etc.) with the loss induced if an insolvent client was debited, the decision to interrogate the bank's servers is taken probabilistically according to the amount of the transaction. When interrogated, the bank can reject the transaction with a certain probability or accept it. This system is represented by the automaton of Fig. 10.

Let assume the intruder can only observe whether or not the bank is called. This can be achieved, for example, by measuring the time taken for the transaction to be accepted (it takes longer when the bank is called). Suppose the intruder wants to know if the transaction was worth more than 500, say euros. This is described by the opaque language $L_S = E^*(\text{"}x > 1000\text{"} \textit{ or } \text{"}500 < x < 1000\text{"})E^*$.

This system is of course opaque, as there is always a chance of the bank being called (or not) whatever the transaction amount. It follows $LPO = 0$. However, if the intruder sees a call, there are rather high chances that the transaction was worth more than 500. RPO evaluates the level of confidence in this information. In this case, simple probabilistic calculi return $RPO \approx 0.718$. Refer to Bérard et al. (2015b) for more details on the computation procedure.

This work was extended in Bérard et al. (2015a) to Markov decision processes with infinite executions. Quantification is performed through the computation of a *probabilistic disclosure (PD)*, which is the probabilistic measure that a run disclosing the secret has been executed. Several problems are addressed:

(i) Value: What is the PD of the system?

(ii) General disclosure: Is PD bigger than a threshold?

(iii) Limit disclosure: Is $PD = 1$?

(iv) Almost-sure disclosure: does there exists a scheduler such that $PD = 1$?

(v) Almost-sure opacity: Is $PD = 0$?

Future extensions to this work would include the investigation of disclosure before some given delay, either as a number of steps in the spirit of Saboori & Hadjicostis (2011b) or Saboori (2011), or for probabilistic timed systems with an explicit time bound. However this last perspective is seriously hindered by the undecidability of verification for dense time DES models (Cassez, 2009).

5.2. Quantification of state-based opacity

Saboori first investigated the extension of state-based opacity properties to probabilistic models. Three probabilistic properties are introduced in Saboori & Hadjicostis (2010b,a, 2014)

(i) *Step-based almost current-state opacity* considers the *a priori* probability of violating current state opacity following any sequence of events of length K. It requires this probability to lie below a threshold for all possible lengths $k = (0, 1, \dots K)$. It is the extension of K-step opacity.
As for LBO, step-based almost current-state opacity aims to quantify the probability of the secret to be disclosed, which is only relevant for non-opaque systems.

(ii) *Almost current-state opacity* is equivalent to step-based almost current-state opacity with no consideration regarding the length of the sequence of events, i.e., it considers the *a priori* probability of violating CSO following any sequence of events. It requires this probability to lie below a threshold. It is the extension of infinite-step opacity.
Similarly, it is relevant only for non-opaque systems.

(iii) *Probabilistic current-state opacity* holds if the maximum increase in the conditional probability that the system's current state lies in the set of secret states (conditioned on a sequence of observations) compared to the case when no observation is available (prior probability) is bounded.
As for RPO, probabilistic current-state opacity is only relevant for opaque systems. Otherwise, the probability of being in a secret state reaches 1 eventually.

Example 10. From Saboori & Hadjicostis (2014) – Consider the probabilistic finite automaton from Fig. 11 with $E_0 = E = \{\alpha, \beta, \gamma\}$. Assume $X_S = \{4\}$ and the initial probability distribution is $\pi_0 = [1, 0, 0, 0, 0]'$ (i.e., the system starts in state 0.

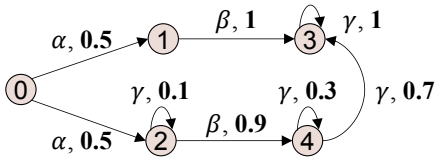


Figure 11: Example 10 – From Saboori & Hadjicostis (2014)

The set of words disclosing the secret is referred to as $L_C = \alpha\gamma\gamma^*\beta\gamma^*$ (first γ is necessary to make sure the system is in the lower branch). The system is step-based almost current-state opaque with respect to a threshold θ if, for any $k > 0$,

$$Pr^k = \sum_{t \in L_C, |t|=k} Pr(t) < \theta$$

There are no words in L_C of length less than 3.

$Pr^3 = Pr(\alpha\gamma\beta) = 0.045$ and

$Pr^4 = Pr(t) = Pr(\alpha\gamma\gamma\beta) + Pr(\alpha\gamma\beta\gamma) = 0.018$.

It is not hard to see in this case that Pr^k decreases with k which implies that this system is step-based almost current-state opaque for any $\theta > 0.045$.

The set of words disclosing the secret *for the first time* is referred to as $L_C^{\mathcal{P}} = \alpha\gamma\gamma^*\beta$ (i.e., no prefix of one of such words reveals the secret). The system is almost current-state opaque with respect to a threshold θ if

$$Pr^\infty = \sum_{t \in L_C^{\mathcal{P}}} Pr(t) < \theta$$

In this case, $Pr^\infty = \sum_{n=0}^{\infty} = 0.5 \times 0.1 \times (0.1)^n \times 0.9 = 0.05$, which implies that this system is almost current-state opaque for any $\theta > 0.05$.

Finally, assume now that $X_S = \{3\}$ and the initial probability distribution is $\pi_0 = [0.2, 0.2, 0.2, 0.2, 0.2]'$. The system is probabilistic current-state opaque with respect to a threshold θ if

$$\forall t \in E_O^*, \; ||\pi_t(X_S)|| - ||\pi_0(X_S)|| \leq \theta$$

where $\pi_t(X_S)$ denotes the probability of being in a secret state after observing word t and $||.||$ is vector 1-norm. We are interested in ensuring the confidence of being in the secret state is never higher than 0.75, that is, we want $0.75 - 0.2 = 0.55-$probabilistic current-state opacity. This does not hold, as after observing the sequence $\alpha\beta\gamma$, the probability distribution vector $\pi_{\alpha\beta\gamma} = [0, 0, 0, 0.79, 0.21]'$, and $0.79 - 0.2 = 0.59 > 0.55$.

These definitions were extended to ISO in Keroglou & Hadjicostis (2013) for systems modeled as probabilistic finite automata:

(i) *Step-based almost initial state opacity* captures the *a priori* probability that the system will generate behavior that violates initial state opacity after a certain number of events.

(ii) *Almost initial-state opacity* captures the *a priori* probability that the system will eventually generate behavior that violates initial state opacity.

Finally, Ibrahim et al. (2014) extended step-based almost current-state opacity from Saboori & Hadjicostis (2010a). Instead of the disclosure probability being below a threshold at each time step, it considers the probability of revealing the secret over the set of all behaviors. Two properties are introduced:

(i) *S_τ-Secrecy* (stochastic-secrecy) holds if the probability of secret disclosure is always below τ.
Secrecy $\Leftrightarrow$ S_0-secrecy.

(ii) *I-S-Secrecy* (increasing stochastic-secrecy) hold if whatever the threshold, there exists a size n of execution length beyond which every trace has a disclosure probability below the threshold.

6. Decidability and Complexity of opacity properties

Opacity is a very general property. As a result, many opacity problems are undecidable. This was demonstrated in Bryans et al. (2008) by reducing opacity verification to the reachability problem for Turing machines. It remains undecidable for general finite labeled transition systems if you do not restrict the class of observation function. Even when decidable, opacity problems are computationally complex to solve in general. This section synthesizes decidability and complexity results demonstrated in the literature.

Note that LBO, ISO, CSO, and IFO– referred to as general opacity problems – have been proven to be reducible into one another in polynomial time (Wu & Lafortune (2013), Chédor et al. (2014)). Therefore, these problems have same decidability and complexity (since their complexity is, at least, polynomial).

We propose in Table 1 to 3 a general overview of decidability and complexity results published up to date in the literature. Several problems have been addressed by different approaches (e.g., initial-state opacity), which results in different order of complexity. When appropriate, we only kept the best (i.e., the smaller) order with the associated reference.

- Table 1 synthesizes decidability and complexity results of general opacity problems w.r.t the system's model and the observation mapping. Static observers are constrained to a fixed a priori interpretation of (un)observable events. Dynamic observers have different capabilities depending on previous events. Orwellian observers can also re-interpret past unobservable events on the base of subsequent observation. The first two are special cases of the latter.

- Table 2 gathers results from opacity quantification approaches.

- Finally, more specific complexity results are presented in Table 3.

7. Applications and related issues

Most opacity properties and validation strategies have been applied and evaluated in the literature. One reference case study is known as the *Dinning cryptographers* problem, introduced by Chaum (1988); see e.g., Lakhnech & Mazaré (2005); Bérard et al. (2010); Wu & Lafortune (2013). It illustrates properties of ISO and CSO. Another ISO application is presented in Saboori & Hadjicostis (2008b), related to *encryption using pseudo-random generators*. The same work also presents the problem of *sensor network coverage* for vehicle tracking (also detailed in Saboori & Hadjicostis (2011a)). Similar problems have been considered in Dubreil et al. (2010), more precisely, the guidance of semi-autonomous agents traveling through finite networks, with the objective of preventing current positions from being known to adversaries that receive partial information from sensors (see Example 7). *Opacity Issues in Games with Imperfect Information* is another application considered in Maubert et al. (2011). It exhibits relevant opacity verification problems, which noticeably generalizes approaches considered in the literature for opacity analysis in DES.

We mentioned in Introduction that opacity theory applies naturally in privacy-enhancing problems such as those we face nowadays in communication protocols design. In Saboori & Hadjicostis (2010b), the authors present a motivational example of the use of probabilistic opacity methods to evaluate the well-known anonymity protocol *Crowds* for the world-wide-web, initially presented in Reiter & Rubin (1998). More recently, Wu and Lafortune addressed the issue of *Ensuring Privacy in Location-Based Services* in Wu et al. (2014) and Wu (2014), using opacity enforcement techniques. To the best of our knowledge, this is the closest it gets to real-life applications so far.

Indeed, most of the current literature on opacity remains mainly theoretical. Nevertheless, there have been a few successful implementations. There are briefly introduced in the following subsection.

Tools and implementation

Saboori used the UMDES library (UMDES, 2009) to implement his verification method for infinite-step opacity, as described in Saboori & Hadjicostis (2011b) and Saboori (2011). UMDES is a library of C routines developed at the University of Michigan for studying DES modeled by finite automata.

Falcone developed a specific toolbox named TAKOS: a Java Toolbox for the Analysis of K-Opacity of Systems (TAKOS (2010)) to implement the K-step opacity enforcement method presented in Falcone et al. (2014) using delays.

Table 1: Decidability and complexity results for general opacity problems and regular languages

System model	Observation mapping	Decidability	Complexity	Reference
Petri Nets	Static	Undecidable	–	Bryans et al. (2008)
Finite labeled transition system	Static	Decidable	PSPACE-complete	Cassez et al. (2012)
	Dynamic	Decidable	PSPACE-complete	
	Orwellian	Decidable	PSPACE-complete	Bérard & Mullins (2014)
Timed Automata (dense-time)	Static	Undecidable		Cassez (2009)
Pushdown automata (PDA)	Static			Kobayashi & Hiraishi (2013)
General case		Undecidable	–	
If $X \backslash X_S$ is a visible PDA		Decidable	2-EXPTIME	
If $X \backslash X_S$ and X_S are visible PDA		Decidable	1-EXPTIME	
Recursive tile systems (RTS)	Static	Undecidable	–	Chédor et al. (2014)
Weighted RTS ($\mathcal{C}$wRTS)		Decidable	2-EXPTIME	

Table 2: Decidability and complexity results for quantified opacity problems

Problem	Decidability		Complexity	Reference
with controller observability being	Perfect/Partial		Perfect/Partial	Bérard et al. (2015a)
Value, General disclosure, Limit disclosure	Decidable	Undecidable	Polynomial	–
Almost-sure disclosure, Almost-sure opacity	Decidable	Decidable for ω-regular secrets	Polynomial	EXPTIME
(Step-based) Almost current-state opacity	Decidable		PSPACE-hard	
Probabilistic current-state opacity opacity	Undecidable		–	Saboori & Hadjicostis (2014)
(Step-based) Almost initial-state opacity	Decidable		–	Keroglou & Hadjicostis (2013)
S_τ-Secrecy/I-S-Secrecy	Decidable		–	Ibrahim et al. (2014)

Table 3: Other complexity results

Problem	Complexity	Order	Reference
Current-state opacity	PSPACE-complete	$O(2^N)$	Saboori (2011)
K-Step weak opacity	NP-hard	$O((\lvert E_{obs}\rvert + 1)^K \times 2^N)$	
Infinite-Step opacity	PSPACE-hard	–	
K-Step strong opacity	NP-hard	$O((\lvert E_{obs}\rvert + 1)^K \times 2^N)$	Falcone et al. (2014)
Initial-state opacity	PSPACE-complete	$O(2^N)$	Wu & Lafortune (2013)
Resolution of initial-state	Polynomial	–	Hadjicostis & Keroglou (2014)
LBO Strong-opacity	PSPACE-complete	–	Lin (2011)
LBO Weak-opacity	Polynomial	–	Zhang et al. (2012)
Static mask synthesis	PSPACE-complete	–	Cassez et al. (2012)
Dynamic mask systhesis	EXPTIME lower bound	–	

Finally, in Klai et al. (2014), a symbolic observation graph-based opacity checker has been implemented in C++ using a binary decision diagram package called BuDDy (BuDDy (1998)). Results are compared with the Takos toolbox on the also well known *Dinning philosophers* problem.

8. Conclusions and open problems

Over the past ten years, opacity applied to DES has been broadly studied. Almost all opacity problems proven decidable have a known complexity. Future trends are oriented toward infinite-state discrete event models, eventually coupled with probabilistic transition functions.

Some ongoing work tackles the verification of state-based opacity for some classes of Petri nets.

We already mentioned the similarities between opacity and diagnosis. There has been a quite decent amount of work related to prognosis (or predictability), which does not try to detect a fault but to predict that a fault will eventually happen in the future. It could be interesting to consider these approaches for the enforcement of opacity properties.

Moreover, in order to broaden the fields of applications, one could consider opacity validation from another perspective. Starting from a fully observable system and a given secret, which events one should "hide" in order to ensure opacity? This approach could provide a pragmatic methodology for people interested in designing opaque systems. Very recent work (O'Kane & Shell (2015)) is a first attempt in this direction. It models both the information we need the system to reveal and those we want to be opaque as lower and upper bound filters. It shows that determining whether it is possible to satisfy both the distinguishability and indistinguishability constraints is NP-hard, along with simulation results from their implementation.

As we are moving at high speed toward permanent connectedness, big data, user profiling and such, efficient tools to control the information we are disclosing and those to be kept private are becoming of paramount importance. Opacity is part of the answer. We believe it is now time to use this knowledge to handle the actual security and privacy problems we now face in our everyday life.

References

Alur, R., Černý, P., & Zdancewic, S. (2006). Preserving secrecy under refinement. In *Automata, Languages and Programming* (pp. 107–118). Springer. doi:10.1007/11787006_10.

Badouel, E., Bednarczyk, M., Borzyszkowski, A., Caillaud, B., & Darondeau, P. (2007). Concurrent secrets. *Discrete Event Dynamic Systems*, *17*, 425–446. doi:10.1007/s10626-007-0020-5.

Ben-Kalefa, M., & Lin, F. (2009). Opaque superlanguages and sublanguages in discrete event systems. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on* (pp. 199–204). IEEE. doi:10.1109/CDC.2009.5400704.

Ben-Kalefa, M., & Lin, F. (2011). Supervisory control for opacity of discrete event systems. In *Communication, Control, and Computing (Allerton), 2011 49th Annual Allerton Conference on* (pp. 1113–1119). IEEE. doi:10.1109/Allerton.2011.6120292.

Benattar, G., Cassez, F., Lime, D., & Roux, O. H. (2015). Control and synthesis of non-interferent timed systems. *International Journal of Control*, *88*. doi:10.1080/00207179.2014.944356.

Bérard, B., Chatterjee, K., & Sznajder, N. (2015a). Probabilistic opacity for markov decision processes. *Information Processing Letters*, *115*, 52–59. doi:10.1016/j.ipl.2014.09.001.

Bérard, B., & Mullins, J. (2014). Verification of information flow properties under rational observation. *CoRR*, *abs/1409.0871*. URL: http://arxiv.org/abs/1409.0871.

Bérard, B., Mullins, J., & Sassolas, M. (2010). Quantifying opacity. In *Quantitative Evaluation of Systems (QEST), 2010 Seventh International Conference on the* (pp. 263–272). IEEE. URL: http://arxiv.org/pdf/1301.6799.pdf.

Bérard, B., Mullins, J., & Sassolas, M. (2015b). Quantifying opacity. *Mathematical Structures in Computer Science*, *25*, 361–403. URL: http://arxiv.org/pdf/1301.6799.pdf. doi:10.1017/S0960129513000637.

Bérard, B., Mullins, J. et al. (2015c). Non-interference in partial order models. In *ACSD 2015*. IEEE.

Bryans, J. W., Koutny, M., Mazaré, L., & Ryan, P. Y. (2008). Opacity generalised to transition systems. *International Journal of Information Security*, *7*, 421–435. doi:10.1007/s10207-008-0058-x.

Bryans, J. W., Koutny, M., & Mu, C. (2011). *Towards quantitative analysis of opacity*. Technical Report No. CS-TR-1304 Newcastle University: Computing Science.

Bryans, J. W., Koutny, M., & Ryan, P. Y. (2005). Modelling opacity using petri nets. *Electronic Notes in Theoretical Computer Science*, *121*, 101–115. doi:doi:10.1016/j.entcs.2004.10.010.

BuDDy (1998). Buddy. online. URL: http://vlsicad.eecs.umich.edu/BK/Slots/cache/www.itu.dk/research/buddy/.

Cassandras, C. G., & Lafortune, S. (2008). *Introduction to discrete event systems*. Springer Science & Business Media.

Cassez, F. (2009). The dark side of timed opacity. In J. Park, H.-H. Chen, M. Atiquzzaman, C. Lee, T.-h. Kim, & S.-S. Yeo (Eds.), *Advances in Information Security and Assurance* (pp. 21–30). Springer Berlin Heidelberg volume 5576 of *Lecture Notes in Computer Science*. URL: http://dx.doi.org/10.1007/978-3-642-02617-1_3. doi:10.1007/978-3-642-02617-1_3.

Cassez, F., Dubreil, J., & Marchand, H. (2009). Dynamic observers for the synthesis of opaque systems. In *Automated Technology for Verification and Analysis* (pp. 352–367). Springer. doi:10.1007/978-3-642-04761-9_26.

Cassez, F., Dubreil, J., & Marchand, H. (2012). Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, *40*, 88–115. doi:10.1007/s10703-012-0141-9.

Cassez, F., Mullins, J., & Roux, O. H. (2007). Synthesis of non-interferent systems. In *4th Int. Conf. on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS'07)* (pp. 307–321). Saint Petersburg, Russia: Springer volume 1. URL: https://hal.inria.fr/inria-00363029.

Chaum, D. (1988). The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, *1*, 65–75. doi:10.1007/BF00206326.

Chédor, S. (2014). *Diagnosis, opacity and conformance testing for recursive tile systems*. Theses Université Rennes 1. URL: https://tel.archives-ouvertes.fr/tel-00980800.

Chédor, S., Morvan, C., Pinchinat, S., Marchand, H. et al. (2014). Diagnosis and opacity problems for infinite state systems modeled by recursive tile systems. *Discrete Event Dynamic Systems*, . doi:10.1007/s10626-014-0197-3.

Darondeau, P., Marchand, H., & Ricker, L. (2014). Enforcing opacity of regular predicates on modal transition systems. *Discrete Event Dynamic Systems*, (pp. 1–20). doi:10.1007/s10626-014-0193-7.

Dubreil, J. (2010). *Monitoring and Supervisory Control for Opacity Properties*. English. tel-00461306 Ph.D. dissertation, Software Engineering, Universit Rennes 1. URL: https://tel.archives-ouvertes.fr/tel-00461306/.

Dubreil, J., Darondeau, P., & Marchand, H. (2008). Opacity enforcing control synthesis. In *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on* (pp. 28–35). IEEE. doi:10.1109/WODES.2008.4605918.

Dubreil, J., Darondeau, P., & Marchand, H. (2010). Supervisory control for opacity. *Automatic Control, IEEE Transactions on*, *55*, 1089–1100. doi:10.1109/TAC.2010.2042008.

Dubreil, J., Jéron, T., Marchand, H. et al. (2009). Monitoring confidentiality by diagnosis techniques. In *European Control Conference* (pp. 2584–2589).

Falcone, Y., & Marchand, H. (2013). Runtime enforcement of k-step opacity. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on* (pp. 7271–7278). IEEE. doi:10.1109/CDC.2013.6761043.

Falcone, Y., Marchand, H. et al. (2014). Enforcement and validation (at runtime) of various notions of opacity. *Discrete Event Dynamic Systems*, (p. pp.42). doi:10.1007/s10626-014-0196-4.

Focardi, R., & Gorrieri, R. (1994). A taxonomy of trace-based security properties for ccs. In *Computer Security Foundations Workshop VII, 1994. CSFW 7. Proceedings* (pp. 126–136). IEEE. doi:10.1109/CSFW.1994.315941.

Hadj-Alouane, N. B., Lafrance, S., Lin, F., Mullins, J., & Yeddes, M. M. (2005). On the verification of intransitive noninterference in mulitlevel security. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *35*, 948–958. doi:10.1109/TSMCB.2005.847749.

Hadjicostis, C., & Keroglou, C. (2014). Opacity formulations and verification in discrete event systems. In *Emerging Technology and Factory Automation (ETFA), 2014 IEEE* (pp. 1–12). IEEE. doi:10.1109/ETFA.2014.7005032.

Hadjicostis, C. N. (2012). Resolution of initial-state in security applications of des. In *Control & Automation (MED), 2012 20th Mediterranean Conference on* (pp. 794–799). IEEE. doi:10.1109/MED.2012.6265735.

Ibrahim, M., Chen, J., & Kumar, R. (2014). Secrecy in stochastic discrete event systems. In *Networking, Sensing and Control (ICNSC), 2014 IEEE 11th International Conference on* (pp. 48–53). IEEE. doi:10.1109/ICNSC.2014.6819598.

Jacob, R., Lesage, J.-J., & Faure, J.-M. (2015). Opacity of discrete event systems: models, validation and quantification. In *DCDS15* (pp. 174–181). Cancun, Mexico. URL: https://hal.archives-ouvertes.fr/hal-01139890. doi:10.1016/j.ifacol.2015.06.490.

Keroglou, C., & Hadjicostis, C. N. (2013). Initial state opacity in stochastic des. In *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on* (pp. 1–8). IEEE. doi:10.1109/ETFA.2013.6648005.

Klai, K., Hamdi, N., & Hadj-Alouane, N. B. (2014). An on-the-fly approach for the verification of opacity in critical systems. In *WETICE Conference (WETICE), 2014 IEEE 23rd International* (pp. 345–350). IEEE. doi:10.1109/WETICE.2014.84.

Kobayashi, K., & Hiraishi, K. (2013). Verification of opacity and diagnosability for pushdown systems. *Journal of Applied Mathematics*, *2013*. doi:10.1155/2013/654059.

Lakhnech, Y., & Mazaré, L. (2005). Probabilistic opacity for a passive adversary and its application to chaum's voting scheme. *IACR Cryptology ePrint Archive*, *2005*, 98. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.219.5335&rep=rep1&type=pdf.

Larsen, K. G. (1990). Modal specifications. In *Automatic Verification Methods for Finite State Systems* (pp. 232–246). Springer. doi:10.1007/3-540-52148-8_19.

Ligatti, J., Bauer, L., & Walker, D. (2005). Edit automata: Enforcement mechanisms for run-time security policies. *International Journal of Information Security*, *4*, 2–16. doi:10.1007/s10207-004-0046-8.

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, *47*, 496–503. doi:10.1016/j.automatica.2011.01.002.

Maubert, B., Pinchinat, S., & Bozzelli, L. (2011). Opacity issues in games with imperfect information. *arXiv preprint arXiv:1106.1233*, . doi:10.4204/EPTCS.54.7.

Mazaré, L. (2004). Using unification for opacity properties. In *Proceedings of WITS* (pp. 165–176). volume 4. URL: http://www-verimag.imag.fr/TR/TR-2004-24.pdf.

Mullins, J., & Yeddes, M. (2013). Opacity with orwellian observers and intransitive non-interference. *arXiv preprint arXiv:1312.6426*, . URL: http://arxiv.org/abs/1312.6426.

O'Kane, J. M., & Shell, D. A. (2015). Automatic design of discreet discrete filters. In *Proc. IEEE International Conference on Robotics and Automation*. URL: http://robotics.cs.tamu.edu/dshell/papers/icra2015discreet.pdf.

Paoli, A., & Lin, F. (2012). Decentralized opacity of discrete event systems. In *American Control Conference (ACC), 2012* (pp. 6083–6088). IEEE. doi:10.1109/ACC.2012.6315028.

Ramadge, P. J., & Wonham, W. M. (1989). The control of discrete event systems. *Proceedings of the IEEE*, *77*, 81–98. doi:10.1109/5.21072.

Reiter, M. K., & Rubin, A. D. (1998). Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security (TISSEC)*, *1*, 66–92. doi:10.1145/290163.290168.

Saboori, A. (2011). *Verification and enforcement of state-based notions of opacity in discrete event systems*. Ph.D. thesis University of Illinois at Urbana-Champaign. URL: http://hdl.handle.net/2142/18226.

Saboori, A., & Hadjicostis, C. N. (2007). Notions of security and opacity in discrete event systems. In *Decision and Control, 2007 46th IEEE Conference on* (pp. 5056–5061). IEEE. doi:10.1109/CDC.2007.4434515.

Saboori, A., & Hadjicostis, C. N. (2008a). Opacity-enforcing supervisory strategies for secure discrete event systems. In *CDC* (pp. 889–894). doi:10.1109/CDC.2008.4738646.

Saboori, A., & Hadjicostis, C. N. (2008b). Verification of initial-state opacity in security applications of des. In *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on* (pp. 328–333). IEEE. doi:10.1109/WODES.2008.4605967.

Saboori, A., & Hadjicostis, C. N. (2009). Verification of infinite-step opacity and analysis of its complexity. In *Dependable control of discrete systems* (pp. 46–51). volume 2.

Saboori, A., & Hadjicostis, C. N. (2010a). Opacity verification in stochastic discrete event systems. In *Decision and Control (CDC), 2010 49th IEEE Conference on* (pp. 6759–6764). IEEE. doi:10.1109/CDC.2010.5717580.

Saboori, A., & Hadjicostis, C. N. (2010b). Probabilistic current-state opacity is undecidable. In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems–MTNS*. volume 5. URL: http://fwn06.housing.rug.nl/mtns2010/Papers/084_478.pdf.

Saboori, A., & Hadjicostis, C. N. (2011a). Coverage analysis of mobile agent trajectory via state-based opacity formulations. *Control Engineering Practice*, *19*, 967–977. doi:10.1016/j.conengprac.2010.12.003.

Saboori, A., & Hadjicostis, C. N. (2011b). Verification of k-step opacity and analysis of its complexity. *Automation Science and Engineering, IEEE Transactions on*, *8*, 549–559. doi:10.1109/TASE.2011.2106775.

Saboori, A., & Hadjicostis, C. N. (2012). Opacity-enforcing supervisory strategies via state estimator constructions. *Automatic Control, IEEE Transactions on*, *57*, 1155–1165. doi:10.1109/TAC.2011.2170453.

Saboori, A., & Hadjicostis, C. N. (2013). Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, *246*, 115–132. doi:10.1016/j.ins.2013.05.033.

Saboori, A., & Hadjicostis, C. N. (2014). Current-state opacity formulations in probabilistic finite automata. *Automatic Control, IEEE Transactions on*, *59*, 120–133. doi:10.1109/TAC.2013.2279914.

Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. C. (1996). Failure diagnosis using discrete-event models. *Control Systems Technology, IEEE Transactions on*, *4*, 105–124. doi:10.1109/87.486338.

Schneider, F. B. (2000). Enforceable security policies. *ACM Trans-*

actions on Information and System Security (TISSEC), *3*, 30–50. doi:10.1145/353323.353382.

Schneider, S., & Sidiropoulos, A. (1996). Csp and anonymity. In *Computer SecurityESORICS 96* (pp. 198–218). Springer. doi:10.1007/3-540-61770-1_38.

Takai, S., & Kumar, R. (2009). Verification and synthesis for secrecy in discrete-event systems. In *American Control Conference, 2009. ACC'09.* (pp. 4741–4746). IEEE. doi:10.1109/ACC.2009.5160162.

Takai, S., & Oka, Y. (2008). A formula for the supremal controllable and opaque sublanguage arising in supervisory control. *SICE Journal of Control, Measurement, and System Integration*, *1*, 307–311. doi:10.9746/jcmsi.1.307.

Takos (2010). Takos: A java toolbox for analyzing the k-opacity of systems. online. URL: http://toolboxopacity.gforge.inria.fr/.

Umdes (2009). Umdes-lib. online. URL: http://www.eecs.umich.edu/umdes/toolboxes.html software library.

Wu, Y.-C. (2014). *Verification and Enforcement of Opacity Security Properties in Discrete Event Systems*. Ph.D. thesis University of Michigan. URL: http://hdl.handle.net/2027.42/108905.

Wu, Y.-C., & Lafortune, S. (2013). Comparative analysis of related notions of opacity in centralized and coordinated architectures. *Discrete Event Dynamic Systems*, *23*, 307–339. doi:10.1007/s10626-012-0145-z.

Wu, Y.-C., & Lafortune, S. (2014). Synthesis of insertion functions for enforcement of opacity security properties. *Automatica*, *50*, 1336–1348. doi:10.1016/j.automatica.2014.02.038.

Wu, Y.-C., Sankararaman, K. A., & Lafortune, S. (2014). Ensuring privacy in location-based services: An approach based on opacity enforcement. In *Discrete Event Systems* (pp. 33–38). volume 12. doi:10.3182/20140514-3-FR-4046.00008.

Zaytoon, J., & Lafortune, S. (2013). Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, *37*, 308–320. doi:10.1016/j.arcontrol.2013.09.009.

Zhang, B., Shu, S., & Lin, F. (2012). Polynomial algorithms to check opacity in discrete event systems. In *Control and Decision Conference (CCDC), 2012 24th Chinese* (pp. 763 – 769). IEEE. doi:10.1109/CCDC.2012.6244117.

Zhang, B., Shu, S., & Lin, F. (2014). Maximum information release while ensuring opacity in discrete event systems. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on* (pp. 3285–3290). IEEE. doi:10.1109/ICRA.2014.6907331.

Zhang, B., Shu, S., & Lin, F. (2015). Maximum information release while ensuring opacity in discrete event systems. *Automation Science and Engineering, IEEE Transactions on*, (pp. 1067–1079). doi:10.1109/TASE.2014.2379623.

Romain Jacob was a final year Master student at ENS Cachan, France at the time of writing. After one year as visiting scholar at the University of California Berkeley, he completed his M.Eng. degree in Industrial Automation and Control from ENS Cachan (2015). He is now with the ETH Zürich where he is pursuing his Ph.D. on wireless sensor networks architectures and control. His research interests span from control synthesis and artificial intelligence to communication protocol, modeling and simulation.

Jean-Jacques Lesage received the Ph.D. degree from the Ecole Centrale de Paris and the "Habilitation diriger des recherches" from the University Nancy 1 in 1989 and 1994 respectively. He is currently Professor of Automatic Control at the Ecole Normale Suprieure de Cachan, France. His research interests are in the field of formal methods and models for synthesis, analysis and diagnosis of Discrete Event Systems (DES), and applications to manufacturing systems, network automated systems, energy production, and ambient assisted living.

Jean-Marc Faure received the Ph.D. degree from Ecole Centrale de Paris in 1991. He is currently Professor of Automatic Control and Automation Engineering at the Institut Superieur de Mecanique de Paris and researcher at Ecole Normale Superieure de Cachan, France. His research fields are modeling, synthesis and analysis of Discrete Event Systems (DES) with special focus on formal verification and conformance test methods to improve dependability of critical systems. J.-M. Faure is member of the IEEE and Associate Editor of the Journal T-ASE since 2012. He is chair of the steering committee of the IFAC workshop series "Dependable Control of Discrete Systems" and has served in many committees of IFAC and IEEE conferences.