Preprints of the 2012 20th Mediterranean Conference on Control & Automation (MED), Barcelona, Spain, July 3-6, 2012

A Unifying Decision-Making Framework to Study Secrecy in Discrete Event Systems

Ahmed Khoumsi and Lucien Ouedraogo

***Abstract*— This paper deals with *secrecy*, i.e. how an information-flow property of system can be kept secret from observers. Recently, a discrete-event system based model has been developed to study secrecy. With such a model, secrecy preservation of a property is modeled as the impossibility for observers to determine whether executed event sequences belong or not to a given language. In the present paper, we adopt this model of secrecy and study it by using a *Unifying Decision-Making Framework* (UDMF) which has been recently developed. UDMF is a generic decision-making framework for discrete-event systems, which has been shown to generalize supervisory control, diagnosis and prognosis of discrete-event systems, which are thus particular instances of UDMF. Following the same idea, we express preservation of secrecy as a particular case of UDMF. We take advantage of results of UDMF to obtain results in secrecy preservation.**

I. INTRODUCTION

We study secrecy, i.e. the problem of how to keep an information-flow property of system secret from observers. The authors of [1] have developed a model where secrecy preservation is expressed as follows: a discrete-event system (DES) we call *plant* is observed partially by observers, and secrecy preservation of a property (on information-flow) is modeled as the impossibility for observers to determine whether event sequences executed by the plant satisfy a given property. The authors of [1] show that their model is general enough to capture several standard information-flow properties, such as Noninterference [2] or Perfect Security Property [3].

The authors of [4] adopt the model of [1] and define secrecy of a property $\mathcal{P}$ as follows: impossibility for observers of the plant to determine with certainty when $\mathcal{P}$ is satisfied and when $\mathcal{P}$ is violated. Let $\mathcal{L}_{\mathcal{P}}$ be the language where $\mathcal{P}$ is satisfied, and λ be an event sequence executed by the plant. Secrecy of $\mathcal{P}$ is then formulated as follows: impossibility for observers to determine when $\lambda \in \mathcal{L}_{\mathcal{P}}$ and when $\lambda \notin \mathcal{L}_{\mathcal{P}}$. This definition of secrecy is more general than in [5], [6], [7] who consider only the first point (i.e. $\lambda \in \mathcal{L}_{\mathcal{P}}$). The authors of [4] study the case of a centralized architecture where a single observer tries to break secrecy, i.e. tries to determine when $\lambda \in \mathcal{L}_{\mathcal{P}}$ and when $\lambda \notin \mathcal{L}_{\mathcal{P}}$. The authors of [4] study also *weak-secrecy*, which corresponds to secrecy in the case where the observer does not know the plant behavior.

We study secrecy and weak-secrecy by using a *Unifying Decision-Making Framework* (UDMF) which has been recently developed in [8]. The author of [8] shows that supervisory control, diagnosis and prognosis of DES can be studied as particular instances of UDMF. Now, we go further by showing that also secrecy can be studied as a particular case of UDMF. UDMF enables us to synthesize *automatically* the conditions of secrecy and weak-secrecy, i.e. the conditions to the non-existence of observers that can break secrecy and weak-secrecy. The synthesized conditions are equivalent to those obtained in [4], but their formulations are simpler than in [4]. This simplicity permits us to verify condition of secrecy more efficiently than in [4].

The remainder of the paper is organized as follows: In Section II, UDMF does not consider a specific decision-maker (e.g., supervisor, diagnoser, prognoser, breaker of secrecy); it rather considers a *generic* decision-maker which targets to respect given properties. We characterize and categorize the types of properties that must be respected by decision-makers in UDMF. Then, we specify secrecy (and weak-secrecy) preservation as a particular decision-making problem of UDMF.

In Section III, UDMF does not consider a specific architecture; it rather considers a *generic* architecture qualified as *well-formed*. The latter is characterized as any architecture for which we can identify a condition to the existence of decision-makers satisfying the properties mentioned in Section II. When applying this framework to secrecy, we obtain a generic condition of secrecy preservation for any well-formed architecture.

In Section IV, we specialize the generic architecture of Section III to the case of a *centralized* architecture.

In Section V, we show how the study of [4] can be realized as a particular instance of the centralized architecture of Section IV. The latter permits us to synthesize automatically the conditions of secrecy and weak-secrecy which are equivalent to those of [4]. But our formulation of secrecy permits us to develop a more efficient algorithm for verifying secrecy than in [4].

Section VI concludes and proposes some future work.

Let us present some preliminaries and notations. Σ is a set of events, also called *alphabet*. A *finite* sequence of events of Σ is called a *trace*. Σ^* denotes the set of all traces, including the empty trace. A set of traces $K \subseteq \Sigma^*$ is called a *language*. A trace μ is said *prefix* of a trace λ, if there exists a trace ν such that $\lambda = \mu\nu$. $\overline{K}$ denotes the set of all prefixes of traces of a language K. K is said prefix-closed if $\overline{K} = K$. For two languages K and L, the language $K \setminus L$ contains every trace λ such that $\lambda \in K$ and $\lambda \notin L$. In all the paper, we consider a plant whose behavior is modeled by a finite state automaton

A. Khoumsi is with the Department of Electrical and Computer Engineering, University of Sherbrooke, Sherbrooke, QC, J1L1T8 Canada Ahmed.Khoumsi@UShebrooke.ca

L. Ouedraogo is a postdoctoral fellow at the Department of Electrical and Computer Engineering, Iowa State University, Ames, IA, USA

978-1-4673-2529-5/12/$31.00 ©2012 IEEE

$G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the finite set of states, Σ is the alphabet, δ is a transition function $Q \times \Sigma \to Q$, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the (finite) set of marked states. Let $\mathcal{L}$ and $\mathcal{L}_m$ be the generated and marked languages of G, respectively, i.e., $\mathcal{L}$ (resp. $\mathcal{L}_m$) contains the traces of G leading from q_0 to any state of Q (resp. Q_m). For a set E, $|E|$ is the cardinality of E.

II. DECISION-MAKING IN UDMF

A. Basic Characteristics of a Decision-Maker

Compared to UDMF in [8], we will omit elements which are not indispensable for the study of secrecy. Besides, improvements are added to UDMF so that it supports adequately the study of secrecy.

One or several so-called *decisional elements* are associated to the plant, and decisions are made for each decisional element. For example, in secrecy, a decisional element is a property which must be kept secret. We will consider a *single* decisional element; in the presence of several decisional elements, the same study must be made for each of them. We consider a decision-maker that takes decisions for traces executed by the plant, and we denote by $\mathrm{Dec}(\lambda)$ the decision taken after the execution of a trace λ. We consider that only the decisions taken on the traces of a given regular language $\mathcal{S} \subseteq \mathcal{L}$ are relevant. (Note that this does not forbid to have $\mathcal{S} \subseteq \mathcal{L}_m$.) $\mathcal{S}$ is not necessarily known by the decision-maker. $\mathcal{S}$ depends on the type of decision (e.g., supervisory control, diagnosis, prognosis, breaking secrecy). For example, we will see that $\mathcal{S} = \mathcal{L}_m$ in the case of secrecy.

Definition 2.1: A decision-maker is assumed to take its decisions among two possible values, noted $1, 0$. It may also choose to take *no* decision, which is noted ϕ. Formally, a decision-maker is a map $\mathrm{Dec} : \mathcal{S} \mapsto \{1, 0, \phi\}$.

The requirements targeted by a decision-maker consist of *a generic decisional requirement* (GDR) and *specific decisional requirements* (SDR) which are defined in the following two sections, respectively.

B. Generic Decisional Requirement (GDR)

To a decisional element, are associated two languages $\mathcal{Y}$ and $\mathcal{Z}$, such that $\mathcal{Y} \cup \mathcal{Z} \subseteq \Sigma^*$ and $\mathcal{Y} \cap \mathcal{Z} = \emptyset$. The ideal objective is that the decision-maker takes a decision 1 (resp. 0) when the plant has executed a trace $\lambda \in \mathcal{Y}$ (resp. $\lambda \in \mathcal{Z}$). $\mathcal{Y}$ and $\mathcal{Z}$ are assumed known by any decision-maker. Since $\mathcal{Y}$ and $\mathcal{Z}$ are not necessarily sub-languages of $\mathcal{S}$, while only the decisions for $\lambda \in \mathcal{S}$ are relevant, we can consider $\lambda \in \mathcal{S} \cap \mathcal{Y}$ and $\lambda \in \mathcal{S} \cap \mathcal{Z}$ instead of $\lambda \in \mathcal{Y}$ and $\lambda \in \mathcal{Z}$. If the decision-maker has only a *partial observation* of the plant, it may be unsure of the executed trace and hence unable to satisfy the ideal objective. In this case, the minimal requirement, called *Generic Decisional Requirement* (GDR), is that the decision-maker decides ϕ when it is uncertain to satisfy the ideal objective. In other words, it must not decide 0 (resp. 1) when the plant has executed a trace $\lambda \in \mathcal{S} \cap \mathcal{Y}$ (resp. $\lambda \in \mathcal{S} \cap \mathcal{Z}$). Formally:

$$\text{GDR: } \left\{ \begin{array}{l} \lambda \in \mathcal{S} \cap \mathcal{Y} \Rightarrow \mathrm{Dec}(\lambda) \neq 0 \\ \lambda \in \mathcal{S} \cap \mathcal{Z} \Rightarrow \mathrm{Dec}(\lambda) \neq 1 \end{array} \right\} \quad (1)$$

C. Specific Decisional Requirements (SDR)

GDR permits to never decide (i.e., always decide ϕ). But it may be required that a decision-maker takes decisions $\mathrm{Dec}(\lambda) \neq \phi$ in specific circumstances where $\lambda \in \mathcal{S} \cap (\mathcal{Y} \cup \mathcal{Z})$. For that purpose, we define the notion of *Specific Decisional Requirement* $[\alpha \hookrightarrow \beta]$, which is a formal expression specifying that β is required under a situation specified by α. We consider α and β that have the following forms, where λ is a symbol or an expression representing a trace:

- $\boldsymbol{\alpha}$ is an expression representing a situation in $\mathcal{S} \cap (\mathcal{Y} \cup \mathcal{Z})$. It may contain none, one or more expressions of the following forms: $\mathbf{Dec(\lambda) \neq 1}$, $\mathbf{Dec(\lambda) \neq 0}$, $\mathbf{Dec(\lambda) = \phi}$.
- $\boldsymbol{\beta}$ is an expression that has one of the following forms: $\mathbf{Dec(\lambda) = 1}$, $\mathbf{Dec(\lambda) = 0}$, $\mathbf{Dec(\lambda) \neq \phi}$.

Let SDR denote a set of specific decisional requirements targeted by a decision-maker. Depending on the context, the term SDR can also denote a single specific decisional requirement. It has been shown in [8] that the above forms of SDR are sufficient to study supervisory control, diagnosis and prognosis of DES. In Section II-D, we show that these forms are also sufficient to specify the objective of secrecy preservation in DES.

The equation below is an example of SDR taken from [8] for the detection of the absence of faults. $\mathbf{N}$ is the set of positive integers, $\mathcal{H}$ the set of unfaulty traces, and $\nu < \mu$ means that ν is a prefix of μ. This is an SDR $[\alpha \hookrightarrow \beta]$, where α is the expression "$\exists \ell \cdots \mathrm{Diag}(\lambda\nu) \neq 0$)" at the left hand side of $\Rightarrow$, and β is $\mathrm{Diag}(\lambda\mu) = 0$. This SDR requires that if the diagnoser remains unsure ($\mathrm{Diag}(\lambda\nu) \neq 0$) during a sufficiently long portion of unfaulty execution (of length ℓ), then the diagnoser must be able to determine that the execution is unfaulty ($\mathrm{Diag}(\lambda\mu) = 0$). In other words, the diagnoser must not remain unsure about the non-occurrence of faults during an arbitrarily long unfaulty execution.

$$\text{SDR: } \left\{ \begin{array}{l} \exists \ell \in \mathbf{N} \text{ s.t. } \forall \lambda\mu \in \mathcal{H} \text{ s.t. } |\mu| = \ell : \\ (\forall \nu < \mu, \mathrm{Diag}(\lambda\nu) \neq 0) \Rightarrow (\mathrm{Diag}(\lambda\mu) = 0) \end{array} \right\}$$

D. Application to Secrecy

Let us show how GDR and SDR permit to define the problem of secrecy preservation. In secrecy, the decision-maker is an observer of the plant who tries to break secrecy in the following sense: determine whether any trace $\lambda \in \mathcal{L}_m$ executed by the plant belongs or not to a given language $\mathcal{K} \subseteq \Sigma^*$. It issues the decision (or verdict) 1 (resp. 0) to mean that $\lambda \in \mathcal{K}$ (resp. $\lambda \notin \mathcal{K}$). The verdict ϕ is issued when the decision-maker is unsure whether λ belongs or not to $\mathcal{K}$. To make the link with the generic case, we have $\mathcal{Y} = \mathcal{K}$, $\mathcal{Z} = \Sigma^* \setminus \mathcal{K}$, and $\mathcal{S} = \mathcal{L}_m$ (we are interested uniquely by decisions taken for $\lambda \in \mathcal{L}_m$). Therefore, GDR of (1) becomes:

$$\text{GDR: } \left\{ \begin{array}{l} \forall \lambda \in \mathcal{L}_m \cap \mathcal{K}, \mathrm{Dec}(\lambda) \neq 0 \\ \forall \lambda \in \mathcal{L}_m \setminus \mathcal{K}, \mathrm{Dec}(\lambda) \neq 1 \end{array} \right\} \quad (2)$$

The objective of the decision-maker is the opposite of secrecy preservation, that is, it targets to break secrecy at least once. In other terms, there must exist at least one executed trace for which the decision-maker issues a decision 1 or 0 (i.e., $\neq \phi$). Formally, we obtain the following SDR:

$$\text{SDR: } \left\{ \ \exists \lambda \in \mathcal{L}_m, \mathrm{Dec}(\lambda) \neq \phi \ \right\} \quad (3)$$

Secrecy preservation corresponds to the nonexistence of a decision-maker satisfying (2,3).

Eq. (3) is an example of SDR $[\alpha \hookrightarrow \beta]$ where α does not contain "$\mathrm{Dec}(\lambda)$". In Section II-C, we have presented an example of $[\alpha \hookrightarrow \beta]$ where α contains $\mathrm{Dec}(\lambda)$.

III. WELL FORMED ARCHITECTURES IN UDMF

In this section, we develop a generic decision-making framework based on a notion of *well-formed architecture*. This framework is doubly generic because: (1) it does not consider an architecture in particular (e.g., centralized, decentralized); and (2) it does not consider a particular decision-making (e.g., control, diagnosis, prognosis, breaking secrecy).

A. Generic Decision-Making Architecture

Figure 1 is a generic representation of decision-making, where functions $\mathcal{O}$ and $\mathcal{D}$ are defined by:

- $\mathcal{O}$: After execution of a trace λ, $\mathcal{O}(\lambda)$ represents λ as it has been observed.
- $\mathcal{D}$: This function synthesizes a decision in $\{1, 0, \phi\}$ from $\mathcal{O}(\lambda)$ and the languages $\mathcal{S} \cap \mathcal{Y}$ and $\mathcal{S} \cap \mathcal{Z}$. Hence, the decision is noted $\mathrm{Dec}(\lambda) = \mathcal{D}(\mathcal{O}(\lambda), \mathcal{S} \cap \mathcal{Y}, \mathcal{S} \cap \mathcal{Z})$. Actually, this is true only if $\mathcal{S}$ is known by the decision-maker. If on the contrary $\mathcal{S}$ is unknown by the decision-maker, the decision depends on $(\mathcal{Y}, \mathcal{Z})$ instead of $(\mathcal{S} \cap \mathcal{Y}, \mathcal{S} \cap \mathcal{Z})$, and hence is noted $\mathrm{Dec}(\lambda) = \mathcal{D}(\mathcal{O}(\lambda), \mathcal{Y}, \mathcal{Z})$.

Intuitively, $\mathcal{O}$ specifies the observed information, and $\mathcal{D}$ specifies how the observed information is used to take decisions.

In Def. 2.1, a decision-maker is defined as a function that generates a decision $\mathrm{Dec}(\lambda)$ from an executed trace λ. Let us now define the structure of a decision-maker and a decision-making architecture:

Definition 3.1: The structure of a decision-maker is defined by fixed $\mathcal{O}$ and $\mathcal{D}$. A decision-making architecture represents a set of decision-makers associated to a fixed $\mathcal{O}$ while $\mathcal{D}$ may vary within a given structure.

$(\mathcal{O}, \mathcal{D})$ will be illustrated in Section IV for a centralized architecture.

Definition 3.2: A *strong* (resp. *weak*) architecture denotes an architecture where $\mathcal{S}$ is known (resp. unknown) to its decision-makers. A *strong* (resp. *weak*) decision-maker denotes a decision-maker of a strong (resp. weak) architecture. More precisely, a strong (resp. weak) decision-maker takes decisions $\mathrm{Dec}(\lambda) = \mathcal{D}(\mathcal{O}(\lambda), \mathcal{S} \cap \mathcal{Y}, \mathcal{S} \cap \mathcal{Z})$ (resp. $\mathrm{Dec}(\lambda) = \mathcal{D}(\mathcal{O}(\lambda), \mathcal{Y}, \mathcal{Z})$).

The results of the following Sections III-B and III-C hold for strong and weak architectures

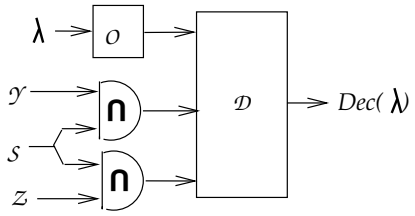

Fig. 1. Generic decision-making architecture

B. Characterization of Well-Formed Architectures

For an architecture Ψ, we denote by DM_Ψ any decision-maker of Ψ. Consider the "existence question": *Does there exist a* DM_Ψ *satisfying given GDR and SDR?* Our objective in Section III is to characterize a *well-formed architecture*, as an architecture for which we can answer that existence question. The well-formed characterization is based on GDR and does not depend on SDR. More precisely, SDR influences the *answer* (but *not* the *capacity to answer*) to the existence question.

Definition 3.3: Consider a pair of languages $(\mathcal{U}, \mathcal{V})$ such that $\mathcal{U} \subseteq \mathcal{Y}$ and $\mathcal{V} \subseteq \mathcal{Z}$. An architecture Ψ is said *well-formed* w.r.t $\langle \mathcal{U}, \mathcal{V} \rangle$ if the next points (a) and (b) are satisfied:

(a) If a DM_Ψ satisfies GDR, then it satisfies (4):

$$\forall \lambda \in \mathcal{S} \cap (\mathcal{Y} \cup \mathcal{Z}) : \left\{ \begin{array}{l} (\mathrm{Dec}(\lambda) = 1) \Rightarrow (\lambda \in \mathcal{U}) \\ (\mathrm{Dec}(\lambda) = 0) \Rightarrow (\lambda \in \mathcal{V}) \end{array} \right\} \quad (4)$$

(b) There exists DM_Ψ satisfying GDR and (5):

$$\left\{ \begin{array}{l} (\lambda \in \mathcal{U}) \Rightarrow (\mathrm{Dec}(\lambda) = 1) \\ (\lambda \in \mathcal{V}) \Rightarrow (\mathrm{Dec}(\lambda) = 0) \end{array} \right\} \quad (5)$$

Intuitively, when Ψ is well-formed architecture w.r.t $\langle \mathcal{U}, \mathcal{V} \rangle$, all DM_Ψ satisfying GDR always decide ϕ outside $\mathcal{U} \cup \mathcal{V}$, and there exists DM_Ψ satisfying GDR that never decides ϕ inside $\mathcal{U} \cup \mathcal{V}$.

Definition 3.4: When Ψ is a well-formed architecture w.r.t $\langle \mathcal{U}, \mathcal{V} \rangle$, the language $\mathcal{U}$ (resp. $\mathcal{V}$) is called positive (resp. negative) decision domain of Ψ.

C. Existence of Solutions in a Well-Formed Architecture

Definition 3.5: $(\mathcal{Y}, \mathcal{Z})$ is said *tractable* w.r.t $\langle \mathcal{U}; \mathcal{V}; \mathrm{SDR} \rangle$ if, after the following replacements in every decisional requirement $[\alpha \hookrightarrow \beta]$ of SDR, the obtained expressions are satisfied.

- **In α**: every "$\mathrm{Dec}(\lambda) \neq 1$" is replaced by "$\lambda \notin \mathcal{U}$",
  every "$\mathrm{Dec}(\lambda) \neq 0$" is replaced by "$\lambda \notin \mathcal{V}$",
  every "$\mathrm{Dec}(\lambda) = \phi$" is replaced by "$\lambda \notin (\mathcal{U} \cup \mathcal{V})$".
- **In β**: every "$\mathrm{Dec}(\lambda) = 1$" is replaced by "$\lambda \in \mathcal{U}$",
  every "$\mathrm{Dec}(\lambda) = 0$" is replaced by "$\lambda \in \mathcal{V}$",
  every "$\mathrm{Dec}(\lambda) \neq \phi$" is replaced by "$\lambda \in (\mathcal{U} \cup \mathcal{V})$".

Remark 1: The replacements of Def. 3.5 are consistent only if every $[\alpha \hookrightarrow \beta]$ is specified in $\mathcal{S} \cap (\mathcal{Y} \cup \mathcal{Z})$.

We have the following generic existence theorem:

Theorem 3.1: For any well-formed architecture Ψ w.r.t $\langle \mathcal{U}, \mathcal{V} \rangle$, there exists a DM_Ψ satisfying GDR and SDR, *if and only if* $(\mathcal{Y}, \mathcal{Z})$ is tractable w.r.t $\langle \mathcal{U}; \mathcal{V}; \mathrm{SDR} \rangle$.

By our framework, we have transformed the problem of constructing and proving a theorem of existence, if any, into a problem of constructing a well-form architecture Ψ, i.e.: *identifying the decision domains $\mathcal{U}$ and $\mathcal{V}$ of Ψ*. Once we have constructed a well-formed architecture, we have that the existence condition (i.e., tractability) is *synthesized automatically* from SDR and $(\mathcal{U}, \mathcal{V})$ by using Def. 3.5. Also, it is worth noting that for a well-formed architecture and assuming that $(\mathcal{Y}, \mathcal{Z})$ is tractable, *any* DM_Ψ satisfying GDR and (5) can be used, *independently of* SDR. The satisfaction of SDR will be guaranteed by the tractability of $(\mathcal{Y}, \mathcal{Z})$.

D. Application to Secrecy

As we have seen in Section II-D, the problem of secrecy preservation is expressed by taking $\mathcal{S} = \mathcal{L}_m$, $\mathcal{Y} = \mathcal{K}$, $\mathcal{Z} = \Sigma^* \setminus \mathcal{K}$, GDR of (2) and SDR of (3), and secrecy preservation corresponds to the nonexistence of a decision-maker satisfying (2,3). Therefore, for a well-formed architecture Ψ, Theorem 3.1 implies that tractability of Def. 3.5 is equivalent to the possibility to break secrecy with Ψ. Hence, the negation of tractability is equivalent to the preservation of secrecy with Ψ. Let us thus determine the generic formulation of secrecy preservation. From (3) and Def. 3.5, tractability can be formulated by: $\exists \lambda \in \mathcal{L}_m$ such that $\lambda \in (\mathcal{U} \cup \mathcal{V})$. This is equivalent to state that $(\mathcal{U} \cup \mathcal{V}) \neq \emptyset$. Therefore, secrecy preservation is equivalent to $(\mathcal{U} \cup \mathcal{V}) = \emptyset$, i.e. $\mathcal{U} = \emptyset$ and $\mathcal{V} = \emptyset$. We have therefore the following result:

Proposition 3.1: Secrecy preservation with a well-formed Ψ is equivalent to $\mathcal{U} = \emptyset$ and $\mathcal{V} = \emptyset$, where $\mathcal{U}$ and $\mathcal{V}$ are the decision domains of Ψ.

IV. CENTRALIZED DECISION-MAKING IN UDMF

The alphabet Σ is partitioned into an observable alphabet Σ_o and an unobservable alphabet Σ_{uo}. We have $\Sigma_o \cup \Sigma_{uo} = \Sigma$ and $\Sigma_o \cap \Sigma_{uo} = \emptyset$. We consider the natural projection $\mathrm{P} : \Sigma \mapsto \Sigma_o$. For $S \subseteq \Sigma^*$, $\mathrm{P}(S) = \{\mathrm{P}(\lambda) \in \Sigma_o^* \,|\, \lambda \in S\}$. P^{-1} denotes the inverse projection, that is, for $S \subseteq \Sigma_o^*$, $\mathrm{P}^{-1}(S) = \{\lambda \in \Sigma^* \,|\, \mathrm{P}(\lambda) \in S\}$.

The structure of a decision-maker and a decision-making architecture was defined generically in Section III-A by using two functions $(\mathcal{O}, \mathcal{D})$. This definition is specialized for a centralized architecture as follows:

Definition 4.1: A *centralized decision-maker* (or Cent-DM) w.r.t P is a decision-maker whose structure is defined by $\mathcal{O}(\lambda) = \mathrm{P}(\lambda)$ and some map $\mathcal{D} : \mathrm{P}(\mathcal{S}) \mapsto \{1, 0, \phi\}$. The *centralized architecture* (or Cent-Arch) w.r.t P is the set of Cent-DMs w.r.t P.

In the sequel, the expression "w.r.t P" will be omitted.

Definition 4.2: A *strong* Cent-DM is defined as a given map $\mathcal{D} : \mathrm{P}(\mathcal{S}) \mapsto \{1, 0, \phi\}$ that depends on $\mathcal{S} \cap \mathcal{Y}$ and $\mathcal{S} \cap \mathcal{Z}$. More precisely, for a trace $\lambda \in \mathcal{S}$, the computation of the decision $\mathrm{Dec}(\lambda) = \mathcal{D}(\mathrm{P}(\lambda))$ depends on whether $\mathrm{P}(\lambda)$ belongs to $\mathrm{P}(\mathcal{S} \cap \mathcal{Y})$ or $\mathrm{P}(\mathcal{S} \cap \mathcal{Z})$. The *strong* Cent-Arch is defined as the set of strong Cent-DMs. *Weak* Cent-DM and *weak* Cent-Arch are defined like their strong counterparts, except that their decisions depends on $(\mathcal{Y}, \mathcal{Z})$ instead of $(\mathcal{S} \cap \mathcal{Y}, \mathcal{S} \cap \mathcal{Z})$.

The following Eqs.(6) and (7) are examples of strong and weak Cent-DMs respectively:

$$\forall \lambda \in \mathcal{S}, \mathrm{Dec}(\lambda) = \mathcal{D}(\mathrm{P}(\lambda)) = \left\{ \begin{array}{ll} 1, & \text{if } \mathrm{P}(\lambda) \in \mathrm{P}(\mathcal{S} \cap \mathcal{Y}) \setminus \mathrm{P}(\mathcal{S} \cap \mathcal{Z}) \\ 0, & \text{if } \mathrm{P}(\lambda) \in \mathrm{P}(\mathcal{S} \cap \mathcal{Z}) \setminus \mathrm{P}(\mathcal{S} \cap \mathcal{Y}) \\ \phi, & \text{otherwise} \end{array} \right\} \quad (6)$$

$$\forall \lambda \in \mathcal{S}, \mathrm{Dec}(\lambda) = \mathcal{D}(\mathrm{P}(\lambda)) = \left\{ \begin{array}{ll} 1, & \text{if } \mathrm{P}(\lambda) \in \mathrm{P}(\mathcal{Y}) \setminus \mathrm{P}(\mathcal{Z}) \\ 0, & \text{if } \mathrm{P}(\lambda) \in \mathrm{P}(\mathcal{Z}) \setminus \mathrm{P}(\mathcal{Y}) \\ \phi, & \text{otherwise} \end{array} \right\} \quad (7)$$

Notation 4.1: Consider the following languages $\mathcal{U}_s^{cent} = (\mathcal{S} \cap \mathcal{Y}) \backslash \mathrm{P}^{-1}\mathrm{P}(\mathcal{S} \cap \mathcal{Z})$, $\mathcal{V}_s^{cent} = (\mathcal{S} \cap \mathcal{Z}) \backslash \mathrm{P}^{-1}\mathrm{P}(\mathcal{S} \cap \mathcal{Y})$, $\mathcal{U}_w^{cent} = (\mathcal{S} \cap \mathcal{Y}) \backslash \mathrm{P}^{-1}\mathrm{P}(\mathcal{Z})$, $\mathcal{V}_w^{cent} = (\mathcal{S} \cap \mathcal{Z}) \backslash \mathrm{P}^{-1}\mathrm{P}(\mathcal{Y})$.

Proposition 4.1: The strong Cent-Arch is well-formed w.r.t $\langle \mathcal{U}_s^{cent}, \mathcal{V}_s^{cent} \rangle$, for *any* $(\mathcal{S}, \mathcal{Y}, \mathcal{Z})$ and SDR. The weak Cent-Arch is well-formed w.r.t $\langle \mathcal{U}_w^{cent}, \mathcal{V}_w^{cent} \rangle$, for *any* $(\mathcal{S}, \mathcal{Y}, \mathcal{Z})$ and SDR.

With the strong (resp. weak) Cent-Arch, the generic tractability (Def. 3.5) becomes the following strong (weak) Cent-Tractability:

Definition 4.3: $(\mathcal{Y}, \mathcal{Z})$ is said to be *strongly (resp. weakly) Cent-Tractable* if all the obtained expressions are satisfied after the replacements of Def. 3.5 in every requirement $[\alpha \hookrightarrow \beta]$ of SDR, using $\mathcal{U} = \mathcal{U}_s^{cent}$ and $\mathcal{V} = \mathcal{V}_s^{cent}$ (resp., $\mathcal{U} = \mathcal{U}_w^{cent}$ and $\mathcal{V} = \mathcal{V}_w^{cent}$).

From Theorem 3.1 and Proposition 4.1, we deduce:

Theorem 4.1: There exists a strong (resp. weak) Cent-DM satisfying GDR and SDR, *if and only if* $(\mathcal{Y}, \mathcal{Z})$ is strongly (resp. weakly) Cent-Tractable.

V. SECRECY PRESERVATION USING THE CENTRALIZED ARCHITECTURE

Let us consider separately the strong and weak Cent-ArchS. For each one, we will use the result (shown in Section III-D) that secrecy preservation with a well-formed architecture Ψ is equivalent to the emptiness of the decision domains $\mathcal{U}$ and $\mathcal{V}$ of Ψ.

A. Secrecy Using the Strong Centralized Architecture

In secrecy with the strong Cent-Arch, an observer who knows $\mathcal{L}_m$ tries to break secrecy by using equation (6). In secrecy, we obtain: $\mathcal{U}_s^{cent} = (\mathcal{L}_m \cap \mathcal{K}) \backslash \mathrm{P}^{-1}\mathrm{P}(\mathcal{L}_m \setminus \mathcal{K})$ and $\mathcal{V}_s^{cent} = (\mathcal{L}_m \setminus \mathcal{K}) \backslash \mathrm{P}^{-1}\mathrm{P}(\mathcal{L}_m \cap \mathcal{K})$. Secrecy preservation with the strong Cent-Arch is equivalent to $\mathcal{U}_s^{cent} = \emptyset$ and $\mathcal{V}_s^{cent} = \emptyset$. $\mathcal{U}_s^{cent} = \emptyset$ is equivalent to $(\mathcal{L}_m \cap \mathcal{K}) \subseteq \mathrm{P}^{-1}\mathrm{P}(\mathcal{L}_m \setminus \mathcal{K})$, i.e., $\mathrm{P}(\mathcal{L}_m \cap \mathcal{K}) \subseteq \mathrm{P}(\mathcal{L}_m \setminus \mathcal{K})$. And $\mathcal{V}_s^{cent} = \emptyset$ is equivalent to $\mathrm{P}(\mathcal{L}_m \setminus \mathcal{K}) \subseteq \mathrm{P}(\mathcal{L}_m \cap \mathcal{K})$. The conjunction of these two inclusions is equivalent to $\mathrm{P}(\mathcal{L}_m \cap \mathcal{K}) = \mathrm{P}(\mathcal{L}_m \setminus \mathcal{K})$, which leads to:

Proposition 5.1: With the strong Cent-Arch, secrecy is preserved if and only if $\mathrm{P}(\mathcal{L}_m \cap \mathcal{K}) = \mathrm{P}(\mathcal{L}_m \setminus \mathcal{K})$.

B. Secrecy Using the Weak Centralized Architecture

In secrecy with the weak Cent-Arch, an observer who does not know $\mathcal{L}_m$ tries to break secrecy by using equation (7). In secrecy, we obtain: $\mathcal{U}_w^{cent} = (\mathcal{L}_m \cap \mathcal{K}) \backslash \mathrm{P}^{-1}\mathrm{P}(\Sigma^* \setminus \mathcal{K})$ and $\mathcal{V}_w^{cent} = (\mathcal{L}_m \setminus \mathcal{K}) \backslash \mathrm{P}^{-1}\mathrm{P}(\mathcal{K})$. Secrecy preservation with the weak Cent-Arch is equivalent to $\mathcal{U}_w^{cent} = \emptyset$ and $\mathcal{V}_w^{cent} = \emptyset$. We obtain:

Proposition 5.2: With the weak Cent-Arch, secrecy is preserved if and only if $\mathrm{P}(\mathcal{L}_m \cap \mathcal{K}) \subseteq \mathrm{P}(\Sigma^* \setminus \mathcal{K})$ and $\mathrm{P}(\mathcal{L}_m \setminus \mathcal{K}) \subseteq \mathrm{P}(\mathcal{K})$.

C. Verifying Secrecy

The authors of [4] have defined the notions of secrecy and weak-secrecy of $\mathcal{K}$ in their Definition 1. It can be easily proved that: secrecy of $\mathcal{K}$ in [4] is equivalent to the condition of Prop. 5.1, i.e., $\mathrm{P}(\mathcal{L}_m \cap \mathcal{K}) = \mathrm{P}(\mathcal{L}_m \setminus \mathcal{K})$; and weak-secrecy of $\mathcal{K}$ in [4] is equivalent to the condition of Prop. 5.2, i.e., $\mathrm{P}(\mathcal{L}_m \cap \mathcal{K}) \subseteq \mathrm{P}(\Sigma^* \setminus \mathcal{K})$ and $\mathrm{P}(\mathcal{L}_m \setminus \mathcal{K}) \subseteq \mathrm{P}(\mathcal{K})$. Therefore, secrecy (resp. weak secrecy) of [4] corresponds to our secrecy with the strong (resp. weak) Cent-Arch. With our formulation $\mathrm{P}(\mathcal{L}_m \cap \mathcal{K}) = \mathrm{P}(\mathcal{L}_m \setminus \mathcal{K})$, we will show that secrecy of $\mathcal{K}$ can be verified more efficiently than in [4]. As for the verification of weak-secrecy of $\mathcal{K}$, we have not found a more efficient verification method than that given in [4]. Let us thus present our method for verifying secrecy of $\mathcal{K}$. We consider the automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$ modeling the plant; its generated and marked languages are $\mathcal{L}$ and $\mathcal{L}_m$ respectively. We consider an automaton $R = (T, \Sigma, \alpha, t_0, T_m)$ which is a trim acceptor of $\mathcal{K}$; its generated and marked languages are $\overline{\mathcal{K}}$ and $\mathcal{K}$ respectively. We construct the automaton RR by adding to R a new "absorbing" state D which is reached by every trace of $\Sigma^* \setminus \overline{\mathcal{K}}$. The construction of RR consists simply in adding to each state $x \in T$ of R the transitions that execute the events of Σ which are not accepted by R in x. All the added transitions lead to state D. Besides, state D contains selfloops of all events of Σ. Then, we construct $G \otimes RR$, the synchronized product of G and RR. The states of $G \otimes RR$ are defined in the form $(x, y) \in Q \times (T \cup \{D\})$. In $G \otimes RR$, the traces of $\mathcal{L}_m \cap \mathcal{K}$ lead to states (x, y) such that $x \in Q_m$ and $y \in T_m$, while the traces of $\mathcal{L}_m \setminus \mathcal{K}$ lead to states (x, y) such that $x \in Q_m$ and $y \in T \cup \{D\} \setminus T_m$. Then, we construct $\mathrm{P}(G \otimes RR)$, the projection of $G \otimes RR$ in the observable alphabet Σ_o. Every state of $\mathrm{P}(G \otimes RR)$ is defined as a set of pairs $(x, y) \in Q \times (T \cup \{D\})$.

Proposition 5.3: The (necessary and sufficient) condition of secrecy: $\mathrm{P}(\mathcal{L}_m \cap \mathcal{K}) = \mathrm{P}(\mathcal{L}_m \setminus \mathcal{K})$ is satisfied if and only if, in every reachable state z of $\mathrm{P}(G \otimes RR)$:
$(\exists (x, y) \in (Q_m \times T_m) \cap z) \Leftrightarrow$
$(\exists (u, v) \in (Q_m \times (T \cup \{D\} \setminus T_m)) \cap z)$.

Since $|\mathrm{P}(G \otimes RR)| = O(2^{|G||R|})$, it follows from Proposition 5.3 that the complexity of verifying secrecy is of the order of $O(2^{|G||R|})$. Hence our verification method is more efficient than the method proposed in [4] whose complexity is of the order of $O(|G||R|2^{|G||R|})$. The reason of this improvement is that in [4], the formulation of secrecy depends on the languages $\mathcal{L}_m \cap \mathcal{K}$ and $\mathcal{L}_m \setminus \mathcal{K}$ and on their projections $\mathrm{P}(\mathcal{L}_m \cap \mathcal{K})$ and $\mathrm{P}(\mathcal{L}_m \setminus \mathcal{K})$, while our formulation of secrecy depends uniquely on the projections.

D. Example of Secrecy Verification

To facilitate the comparison with [4], we will use Example 1 of [4] to illustrate our verification method. Figures 2(a) and 2(b) represent automata G and R, where $\Sigma = \{a, b, c\}$ and $\Sigma_o = \{a, b\}$, i.e., a and b are observable and c is unobservable. In each automaton, the initial state is identified by a small incoming arrow, and marked states are identified by double-circles. The augmented automaton RR, the synchronized product $G \otimes RR$ and its projection $\mathrm{P}(G \otimes RR)$ are shown in Figures 2(c), 2(d) and 2(e), respectively. G has a single marked state q_1, and R has a single marked state t_1. In this example, the condition of secrecy of Proposition 5.3 becomes: in every reachable state z of $\mathrm{P}(G \otimes RR)$, z contains a pair (q_1, t_1) if and only z contains a pair (q_1, t_0) or (q_1, D). This condition is satisfied in $\mathrm{P}(G \otimes RR)$ of Figure 2(e), because there is a single state that contains (q_1, t_1) and (q_1, D), while the two other states contain none of $(q_1, t_1), (q_1, t_0), (q_1, D)$. Therefore, for this example, secrecy with the strong Cent-Arch is preserved.

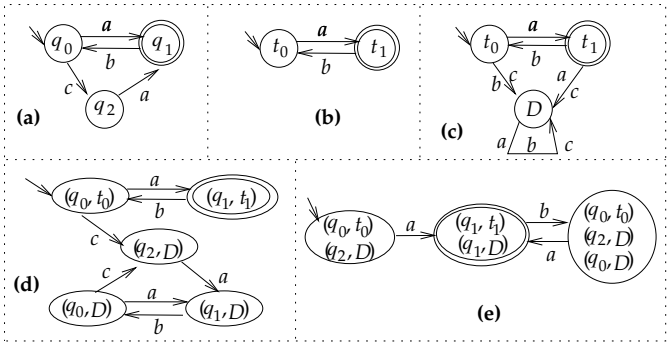

Fig. 2. Example of Verification of Secrecy with a Centralized Architecture: (a) G; (b) R; (c) RR; (d) $G \otimes RR$; (e) $\mathrm{P}(G \otimes RR)$.

VI. CONCLUSION

We have presented a Unifying Decision-Making Framework (UDMF) and shown how it can be used to study secrecy with a centralized architecture. We have adopted the definition of secrecy developed in [1], [4] where secrecy preservation is modeled as the impossibility to determine whether executed traces satisfy a given property. UDMF permitted us to determine a generic condition of secrecy preservation for any architecture characterized as well-formed. We have studied the centralized architecture and shown how it can be used for secrecy. In this context, we obtain the secrecy studied in [4], but we propose a more efficient algorithm for verifying secrecy than in [4].

A near future work is to study secrecy preservation by using decentralized architectures with the objective to improve secrecy preservation.

REFERENCES

[1] R. Alur, P. Cerny, and S. Zdancewic, "Preserving Secrecy Under Refinement," in *Intern. Colloquium on Automata, Languages and Programming, LNCS 4052*, Venice, Italy, 2006, pp. 107–118.

[2] A. Sabelfeld and A. Myers, "Language-Based Information-Flow Security," *IEEE Journal on Selected Areas in Communication*, vol. 21, no. 1, pp. 1–15, 2003.
[3] A. Zakinthinos and E. Lee, "A General Theory of Security Properties," in *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 1997, pp. 94–102.
[4] S. Takai and R. Kumar, "Verification and Synthesis for Secrecy in Discrete Event Systems," in *American Control Conference (ACC)*, St. Louis, MO, USA, June 2009.
[5] E. Badouel, M. Bednarczyk, A. Borzyszkowski, B. Caillaud, and P. Darondeau, "Concurrent Secrets," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 17, no. 4, pp. 425–446, 2007.
[6] J. Dubreil, P. Darondeau, and H. Marchand, "Opacity Enforcing Control Synthesis," in *9th Intern. Workshop on Discrete Event Systems (WODES)*, Göteborg, Sweden, May 2008, pp. 28–35.
[7] S. Takai and Y. Oka, "A Formula for the Supremal Controllable and Opaque Sublanguage Arising in Supervisory Control," *SICE Journal of Control, Measurement, and System Integration*, vol. 1, no. 4, pp. 307–311, 2008.
[8] A. Khoumsi, "A Unifying Decision-Making Framework in Discrete-Event Systems: Application to Centralized and Decentralized Control, Diagnosis and Prognosis," in *10th Intern. Workshop on Discrete Event Systems (WODES)*, Berlin, Germany, August 2010.