

## Concurrent secrets with quantified suspicion

Loïc Hélouët, Hervé Marchand, John Mullins

► **To cite this version:**

Loïc Hélouët, Hervé Marchand, John Mullins. Concurrent secrets with quantified suspicion. ACSD' 2018 - 18th International Conference on Application of Concurrency to System Design, Jun 2018, Bratislava, Slovakia. pp.1-15. hal-01757949

**HAL Id: hal-01757949**

**<https://hal.inria.fr/hal-01757949>**

Submitted on 4 Apr 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Concurrent secrets with quantified suspicion

Loïc Hélouët

INRIA Rennes, Campus de Beaulieu,  
35042 Rennes Cedex, France  
Email: loic.helouet@inria.fr

John Mullins

École Polytechnique de Montréal,  
P.O. Box 6079, Station Centre-ville  
Montréal (Québec) Canada, H3C 3A7  
Email: John.Mullins@polymtl.ca

Hervé Marchand

INRIA Rennes, Campus de Beaulieu,  
35042 Rennes Cedex, France  
Email: herve.marchand@inria.fr

**Abstract**—A system satisfies opacity if its secret behaviors cannot be detected by any user of the system. Opacity of distributed systems was originally set as a boolean predicate before being quantified as measures in a probabilistic setting. This paper considers a different quantitative approach that measures the efforts that a malicious user has to make to detect a secret. This effort is measured as a distance w.r.t a regular profile specifying a normal behavior. This leads to several notions of quantitative opacity. When attackers are passive that is, when they just observe the system, quantitative opacity is brought back to a language inclusion problem, and is PSPACE-complete. When attackers are active, that is, interact with the system in order to detect secret behaviors within a finite depth observation, quantitative opacity turns to be a two-player finite-state quantitative game of partial observation. A winning strategy for an attacker is a sequence of interactions with the system leading to a secret detection without exceeding some profile deviation measure threshold. In this active setting, the complexity of opacity is EXPTIME-complete.

## I. INTRODUCTION

Opacity of a system is a property stating that occurrences of runs from a subset  $S$  of runs of the system (the *secret*) can not be detected by malicious users. Opacity [4], [2] can be used to model several security requirements like anonymity and non-interference [7]. In the basic version of non-interference, actions of the system are divided into high (classified) actions and low (public) ones, and a system is non-interferent iff one can not infer from observation of low operations that high-level actions were performed meaning that occurrence of high actions cannot affect “*what an user can see or do*”. This implicitly means that users have, in addition to their standard behavior, *observation capacities*.

Non-interference is characterized as an equivalence between the system as it is observed by a low-level user and a ideally secure version of it where high-level actions and hence any information flow, are forbidden. This generic definition can be instantiated in many ways, by considering different modeling formalisms (automata, Petri nets, process algebra,...), and equivalences (language equivalence, bisimulation(s),...) representing the discriminating power of an attacker. (see [11] for a survey).

Opacity generalizes non-interference. The secrets to hide in a system are sets of runs that should remain indistinguishable from other behaviors. A system is considered as opaque if, as observed, one can not deduce that the current execution belongs to the secret. In the standard setting, violation of opacity is a

*passive* process: attackers only rely on their partial observation of runs of the system. Checking whether a system is opaque is a PSPACE-complete problem [5].

As such, opacity does not take in account information that can be gained by *active attackers*. Indeed, a system may face an attacker having the capability not only to observe the system but also to interact with him in order to eventually disambiguate observation and detect a secret. A second aspect usually ignored is the quantification of opacity : the more executions leaking information are costly for the attacker, the more secure is the system. In this paper we address both aspects. A first result of this paper is to consider *active opacity*, that is opacity in a setting where attackers of a system perform actions in order to collect information on secrets of the system. Performing actions in our setting means playing standard operations allowed by the system, but also using observation capacities to infer whether a sensible run is being performed. Checking opacity in an active context is a partial information reachability game, and is shown EXPTIME-complete.

We then address opacity in a quantitative framework, characterizing the efforts needed for an attacker to gain hidden information with a cost function. Within this setting, a system remains opaque if the cost needed to obtain information exceeds a certain threshold. This cost is measured as a distance of the attacker’s behavior with respect to a regular profile, modeling that deviations are caught by anomaly detection mechanisms. We use several types of distances, and show that quantitative and passive opacity remains PSPACE-complete, while quantitative and active opacity remains EXPTIME-complete.

Opacity with passive attackers has been addressed in a quantitative setting by [3]. They show several measures for opacity. Given a predicate  $\phi$  characterizing secret runs, a first measure quantifies opacity as the probability of a set of runs which observation suffice to claim the run satisfies  $\phi$ . A second measure considers observation classes (sets of runs with the same observation), and defines the *restrictive probabilistic opacity measure* as an harmonic mean (weighted by the probability of observations) of probability that  $\phi$  is false in a given observation class. Our setting differs from the setting of [3] in the sense that we do not measure secrecy as the probability to leak information to a passive attacker, but rather quantify the minimal efforts required by an active attacker to obtain information.

The paper is organized as follows: Section II introduces our model for distributed systems, and the definition of opacity. Section III recalls the standard notion of opacity usually found in the literature and its PSPACE-completeness, shows how to model active attackers with strategies, and proves that active opacity can be solved as a partial information game over an exponential size arena, and is EXPTIME-complete. Section IV introduces quantification in opacity questions, by measuring the distance between the expected behavior of an agent and its current behavior, and solves the opacity question with respect to a bound on this distance. Section V enhances this setting by discounting distances, first by defining a suspicion level that depends on evolution of the number of errors within a bounded window, and then, by averaging the number of anomalies along runs. The first window-based approach does not change the complexity classes of passive/active opacity, but deciding opacity for averaged measures is still an open problem.

## II. MODEL

Let  $\Sigma$  be an alphabet, and let  $\Sigma' \subseteq \Sigma$ . A word of  $\Sigma^*$  is a sequence of letters  $w = \sigma_1 \dots \sigma_n$ . We denote by  $w^{-1}$  the *mirror* of  $w$ , i.e.,  $w^{-1} = \sigma_n \dots \sigma_1$ . The *projection* of  $w$  on  $\Sigma' \subseteq \Sigma$  is defined by the morphism  $\pi_{\Sigma'} : \Sigma^* \rightarrow \Sigma'^*$  defined as  $\pi_{\Sigma'}(\epsilon) = \epsilon$ ,  $\pi_{\Sigma'}(a.w) = a.\pi_{\Sigma'}(w)$  if  $a \in \Sigma'$  and  $\pi_{\Sigma'}(a.w) = \pi_{\Sigma'}(w)$  otherwise. The *inverse projection* of  $w$  is the set of words which projection is  $w$ , and is defined as  $\pi_{\Sigma'}^{-1}(w) = \{w' \in \Sigma^* \mid \pi_{\Sigma'}(w') = w\}$ . For a pair of words  $w, w'$  defined over alphabets  $\Sigma$  and  $\Sigma'$ , the shuffle of  $w$  and  $w'$  is denoted by  $w \parallel w'$  and is defined as the set of words  $w \parallel w' = \{w'' \mid \pi_{\Sigma}(w'') = w \wedge \pi_{\Sigma'}(w'') = w'\}$ . The shuffle of two languages  $L_1, L_2$  is the set of words obtained as a shuffle of a words of  $L_1$  with a word of  $L_2$ .

*Definition 1:* A concurrent system  $\mathcal{S} = (\mathcal{A}, U)$  is composed of:

- A finite automaton  $\mathcal{A} = (\Sigma, Q, \longrightarrow, q_0, F)$
- A finite set of agents  $U = u_1, \dots, u_n$ , where each  $u_i$  is a tuple  $u_i = (\mathcal{A}_i, \mathcal{P}_i, \mathcal{S}_i, \Sigma_o^i)$ , where  $\mathcal{A}_i, \mathcal{P}_i, \mathcal{S}_i$  are automata and  $\Sigma_o^i$  an observation alphabet.

Agents behave according to their own logic, depicted by a finite automaton  $\mathcal{A}_i = (\Sigma_i, Q_i, \longrightarrow_i, q_0^i, F_i)$  over an action alphabet  $\Sigma_i$ . We consider that agents moves synchronize with the system when performing their actions. This allows modeling situations such as entering critical sections. We consider that in  $\mathcal{A}$  and in every  $\mathcal{A}_i$ , all states are accepting. This way, every sequence of steps of  $\mathcal{S}$  that conforms to transition relations is a behavior of  $\mathcal{S}$ . An agent  $u_i$  observes a subset of actions, defined as an observation alphabet  $\Sigma_o^i \subseteq \Sigma^1$ .

Every agent  $u_i$  possesses a secret, defined as a regular language  $\mathcal{L}(\mathcal{S}_i)$  recognized by automaton  $S_i = (\Sigma, Q_i^S, \longrightarrow_i^S, q_{0,i}^S, F_i^S)$ . All states of secret automata are *not* accepting, i.e. some behaviors of an agent  $u_i$  are secret, some are not. We equip every agent  $u_i$  with a profile  $\mathcal{P}_i = (\Sigma, Q_i^P, \delta_i^P, s_{0,i}^P, F_i^P)$ , that specifies its "normal" behavior. The profile of an agent is

<sup>1</sup>A particular case is  $\Sigma_o^i = \Sigma_i$ , meaning that agent  $u_i$  observes only what it is allowed to do.

prefix-closed. Hence,  $F_i^P = Q_i^P$ , and if  $w.a$  belongs to profile  $\mathcal{L}(\mathcal{P}_i)$  then  $w$  is also in user  $u_i$ 's profile. In profiles, we mainly want to consider actions of a particular agent. However, for convenience, we define profiles over alphabet  $\Sigma$ , and build them in such a way that  $\mathcal{L}(\mathcal{P}_i) = \mathcal{L}(\mathcal{P}_i) \parallel (\Sigma \setminus \Sigma_i)^*$ .

We assume that the secret  $\mathcal{S}_i$  of an user  $u_i$  can contain words from  $\Sigma^*$ , and not only words in  $\Sigma_i^*$ . This is justified by the fact that an user may want to hide some behavior that are sensible only if they occur after other agents actions ( $u_1$  plays  $b$  immediately after  $a$  was played by another agent). For consistency, we furthermore assume that  $\Sigma_i \subseteq \Sigma_o^i$ , i.e., an user observes at least its own actions. Two users may have common actions (i.e.,  $\Sigma_i \cap \Sigma_j \neq \emptyset$ ), which allows synchronizations among agents. We denote by  $\Sigma_U = \bigcup_{i \in U} \Sigma_i$  the possible actions of all users. Note that  $\Sigma_U \subseteq \Sigma$  as the system may have its own internal actions.

Intuitively, in a concurrent system,  $\mathcal{A}$  describes the actions that are feasible with respect to the current *global* state of the system (available resources, locks, access rights,...). The overall behavior of the system is a synchronized product of agents behaviors, intersected with  $\mathcal{L}(\mathcal{A})$ . Hence, within a concurrent system, agents perform moves that are allowed by their current state if they are feasible in the system. If two or more agents can perform a transition via the same action  $a$ , then all agents that can execute  $a$  move conjointly to the next state in their local automaton. More formally, a configuration of a concurrent system is a tuple  $C = (q, q_1, \dots, q_{|U|})$ , where  $q \in Q$  is a state of  $\mathcal{A}$  and each  $q_i \in Q_i$  is a *local state* of user  $u_i$ . The first component of a configuration  $C$  is denoted *state*( $C$ ). We consider that the system starts in an initial configuration  $C_0 = (q_0, q_0^1, \dots, q_0^{|U|})$ .

A *move* from a configuration  $C = (q, q_1, \dots, q_{|U|})$  to a configuration  $C' = (q', q_1', \dots, q_{|U|}')$  via action  $a$  is allowed

- if  $a \notin \Sigma_U$  and  $(q, a, q') \in \longrightarrow$ , or
- if  $a \in \Sigma_U$ ,  $(q, a, q') \in \longrightarrow$ , there exists at least one agent  $u_i$  such that  $(q_i, a, q_i') \in \longrightarrow_i$ , and for every  $q_j$  such that some transition labeled by  $a$  is firable from  $q_j$ ,  $(q_j, a, q_j') \in \longrightarrow_j$ .

The local state of agents that cannot execute  $a$  remains unchanged, i.e., if agent  $u_k$  is such that  $a \in \Sigma_k$  and  $(q_j, a, q_j') \notin \longrightarrow_j$ , then  $q_k = q_k'$ .

A *run* of  $\mathcal{S} = (\mathcal{A}, U)$  is a sequence of moves  $\rho = C_0 \xrightarrow{a_1} C_1 \dots C_k$ . Given a run  $\rho = C_0 \xrightarrow{a_1} C_1 \dots \xrightarrow{a_k} C_k$ , we denote by  $l(\rho) = a_1 \dots a_k$  its corresponding word. The set of run of  $\mathcal{S}$  is denoted by  $Runs(\mathcal{S})$ , while the language  $\mathcal{L}(\mathcal{S}) = l(Runs(\mathcal{S}))$  is the set of words labeling runs of  $\mathcal{S}$ . We denote by  $Conf(\mathcal{S})$  the configurations reached by  $\mathcal{S}$  starting from  $C_0$ . The size  $|\mathcal{S}|$  of  $\mathcal{S}$  is the size of its set of configurations. Given an automaton  $\mathcal{A}$ ,  $\mathcal{P}_i$ , or  $S_i$ , we denote by  $\delta(q, \mathcal{A}, a)$  (resp  $\delta(q, \mathcal{P}_i, a)$ ,  $\delta(q, S_i, a)$ ) the states that are successors of  $q$  by a transition labeled by  $a$ , i.e.  $\delta(q, \mathcal{A}, a) = \{q' \mid q \xrightarrow{a} q'\}$ . This relation extends to sets of states the obvious way, and to words, i.e.  $\delta(q, \mathcal{A}, w.a) = \delta(\delta(q, \mathcal{A}, w), \mathcal{A}, a)$  with  $\delta(q, \mathcal{A}, \epsilon) = \{q\}$ . Last, for a given sub-alphabet  $\Sigma' \subseteq \Sigma$  and a letter  $a \in \Sigma'$ , we define by  $\Delta_{\Sigma'}(q, \mathcal{A}, a)$  the set of states that are reachable

from  $q$  in  $\mathcal{A}$  by sequences of moves which observation is  $a$ . More formally,  $\Delta_{\Sigma'}(q, \mathcal{A}, a) = \{q' \mid \exists w \in (\Sigma \setminus \Sigma')^*, q' \in \delta(q, \mathcal{A}, w.a)\}$ .

### III. OPACITY FOR CONCURRENT SYSTEMS

The standard Boolean notion of opacity introduced by [4], [2] says that the secret of  $u_i$  in a concurrent system  $\mathcal{S}$  is opaque to  $u_j$  if, every secret run of  $u_i$  is equivalent with respect to  $u_j$ 's observation to a non-secret run. In other words,  $u_j$  cannot say with certainty that the currently executed run belongs to  $\mathcal{L}(S_i)$ .

Implicitly, opacity assumes that the specification of the system is known by all participants. In the setting of concurrent system with several agents and secrets, concurrent opacity can then be defined as follows:

*Definition 2 (Concurrent Opacity):* A concurrent system  $\mathcal{S}$  is *opaque* w.r.t.  $U$  (noted  $U$ -Opaque) if

$$\forall i \neq j, \forall w \in \mathcal{L}(S_i) \cap \mathcal{L}(\mathcal{S}), \pi_{\Sigma_o^j}^{-1}(\pi_{\Sigma_o^j}(w)) \cap \mathcal{L}(\mathcal{S}) \not\subseteq \mathcal{L}(S_i)$$

Clearly,  $U$ -opacity is violated if one can find a pair of users  $u_i, u_j$  and a run labeled by a word  $w \in \mathcal{L}(S_i) \cap \mathcal{L}(\mathcal{S})$  such that  $\pi_{\Sigma_o^j}^{-1}(\pi_{\Sigma_o^j}(w)) \cap \mathcal{L}(\mathcal{S}) \subseteq \mathcal{L}(S_i)$ , i.e. after playing  $w$ , there is no ambiguity for  $u_j$  on the fact that  $w$  is a run contained in  $u_i$ 's secret. Unsurprisingly, checking opacity can be brought back to a language inclusion question, and is hence PSPACE-complete. This property was already shown in [5] with a slightly different model (with a single agent  $j$  which behavior is  $\Sigma_j^*$  and a secret defined as a sub-language of the system  $\mathcal{A}$ ).

*Theorem 3 ([5]):* Deciding whether  $\mathcal{S}$  is  $U$ -opaque is PSPACE-complete.

**Proof:**[sketch] The proof of PSPACE-completeness consists in first showing that one can find a witness run in polynomial space. One can chose a pair of users  $u_i, u_j$  in logarithmic space with respect to the number of users, and then find a run after which  $u_j$  can estimate without error that  $u_i$  is in a secret state. Then, an exploration has to maintain  $u_j$ 's estimation of possible configuration of status of  $u_i$ 's secret with  $|\text{Conf}(\mathcal{S})| * |S_i|$  bits. It is also useless to consider runs of length greater than  $2^{|\text{Conf}(\mathcal{S})| * |S_i|}$ . So finding a witness is in NPSPACE and using Savitch's lemma [12] and closure of PSPACE by complementation, opacity is in PSPACE. Hardness comes from a reduction from universality question for regular languages. We refer interested readers to appendix for a complete proof.  $\square$

The standard notion of opacity considers accidental leakage of secret information to an honest user  $u_j$  that is passive, i.e. that does not behave in order to obtain this information. One can also consider an *active* setting, where a particular agent  $u_j$  behaves in order to obtain information on a secret  $S_i$ . In this setting, one can see opacity as a partial information reachability game, where player  $u_j$  tries to reach a state in which his estimation of  $S_i$ 's states is contained in  $F_i^S$ .

Following the definition of non-interference by Goguen & Messeguer [7], we also equip our agents with observation capacities. These capacities can be used to know the current status of resources of the system, but not to get directly

information on other agents states. We define a set of atomic propositions  $\Gamma$ , and assign observable propositions to each state of  $\mathcal{A}$  via a map  $O : Q \rightarrow 2^\Gamma$ . We next equip users with additional actions that consist in asking for the truth value of a particular proposition  $\gamma \in \Gamma$ . For each  $\gamma \in \Gamma$ , we define action  $a_\gamma$  that consists in checking the truth value of proposition  $\gamma$ , and define  $\Sigma_\Gamma = \{a_\gamma \mid \gamma \in \Gamma\}$ . We denote by  $a_\gamma(q)$  the truth value of proposition  $\gamma$  in state  $q$ , i.e.,  $a_\gamma(q) = tt$  if  $\gamma \in O(q)$  and  $ff$  otherwise. Given a set of states  $X = \{q_1, \dots, q_k\}$ , the refinement of  $X$  with assertion  $\gamma = v$  where  $v \in \{tt, ff\}$  is the set  $X_{\gamma=v} = \{q_i \in X \mid a_\gamma(q_i) = v\}$ . Refinement easily extends to a set of configurations  $CX \subseteq \text{Conf}(\mathcal{S})$  with  $CX_{\gamma=v} = \{C \in CX \mid a_\gamma(\text{state}(C)) = v\}$ .

We allow observation from any configuration for every user, hence a behavior of a concurrent system with active attackers shuffles behaviors from  $\mathcal{L}(\mathcal{S})$ , observation actions from  $\Sigma_\Gamma^*$  and the obtained answers. To simplify notations, we assume that a query and its answer are consecutive transitions. The set of queries of a particular agent  $u_j$  will be denoted by  $\Sigma_j^\Gamma$ .

Adding the capacity to observe states of a system forces to consider runs of  $\mathcal{S}$  containing queries followed by their answers instead of simply runs over  $\Sigma^*$ .

We will denote by  $\mathcal{S}^\Gamma$  the system  $\mathcal{S}$  executed in an active environment. Formally, a *run* of  $\mathcal{S}^\Gamma$  in an active setting is a sequence  $\rho = C_0 \xrightarrow{e_1}_{\mathcal{S}^\Gamma} C_1 \dots \xrightarrow{e_k}_{\mathcal{S}^\Gamma} C_k$  where  $C_0, \dots, C_k$  are usual configurations, each  $e_i$  is a letter from  $\Sigma \cup \Sigma_\Gamma \cup \{tt, ff\}$ , such that

- if  $e_k \notin \Sigma_\Gamma$  then  $C_{k+1} \in \delta(C_k, \mathcal{S}, e_k)$ .
- if  $e_k = a_\gamma \in \Sigma_\Gamma$ , then  $e_{k+1} = a_\gamma(q_{k-1})^2$ , and  $C_{k-1} = C_{k+1}$ . Intuitively, testing the value of a proposition does not change the current state of the system. Furthermore, playing action  $a_\gamma$  from  $C_{k-1}$  leaves the system in the same configuration, but remembering that an agent just made the query  $a_\gamma$ . We will write  $C_k = C_{k-1}(a_\gamma)$  to denote this situation. The semantics of  $\mathcal{S}^\Gamma$  can be easily obtained from that of  $\mathcal{S}$ . It can be defined as a new labeled transition system  $LTS^\Gamma(\mathcal{S}) = (\text{Conf}(\mathcal{S}^\Gamma), \xrightarrow{\mathcal{S}^\Gamma}, C_0)$  over alphabet  $\Sigma \cup \Sigma_\Gamma \cup \{tt, ff\}$  recognizing runs of  $\mathcal{S}^\Gamma$ . If  $LTS(\mathcal{S}) = (\text{Conf}(\mathcal{S}), \xrightarrow{\mathcal{S}})$  is an LTS defining runs of  $\mathcal{S}$ , then  $LTS(\mathcal{S}^\Gamma)$  can be built by adding a loop of the form  $C_k \xrightarrow{a_\gamma}_{\mathcal{S}^\Gamma} C_k(a_\gamma) \xrightarrow{a_\gamma(q_k)}_{\mathcal{S}^\Gamma} C_k$  from each configuration  $C_k$  in  $\text{Conf}(\mathcal{S})$ .

We denote by  $\text{Runs}(\mathcal{S}^\Gamma)$  the set of runs of system  $\mathcal{S}$  in an active setting with observation actions  $\Sigma_\Gamma$ . As usual,  $\rho$  is a secret run of agent  $u_i$  iff  $l(\rho)$  is recognized by automaton  $S_i$ . The *observation* of a run  $\rho$  by user  $u_j$  is a word  $l_j(\rho)$  obtained by projection of  $l(\rho)$  on  $\Sigma_j \cup \Sigma_j^\Gamma \cup \{tt, ff\}$ . Hence, an observation of user  $j$  is a word  $l_j(\rho) = \alpha_1 \dots \alpha_k$  where  $\alpha_{m+1} \in \{tt, ff\}$  if  $\alpha_m \in \Sigma_j^\Gamma$  ( $\alpha_m$  is a query followed by the corresponding answer).

<sup>2</sup>This entails that we assume that queries are faster than the rest of the system, i.e. not event can occur between a query and its answer. Hence we have  $\mathcal{L}(\mathcal{S}^\Gamma) \subseteq \mathcal{L}(\mathcal{S}) \parallel (\Sigma_\Gamma \cdot \{tt, ff\})^*$ . We could easily get rid of this hypothesis, by remembering in states of  $\mathcal{S}^\Gamma$  which query (if any) was sent by an user, and returning the answer at any moment.

Let  $w \in (\Sigma_j \cdot (\Sigma_j^\Gamma \cdot \{tt, ff\})^*)^*$ . We denote by  $l_j^{-1}(w)$  the set of runs of  $\mathcal{S}^\Gamma$  which observation by  $u_j$  is  $w$ . A malicious agent can only rely on his observation of  $\mathcal{S}$  to take the decisions that will provide him information on other users secret. Possible actions to achieve this goals are captured by the notion of *strategy*.

**Definition 4:** A *strategy* for an user  $u_j$  is a map  $\mu_j$  from  $Runs(\mathcal{S}^\Gamma)$  to  $\Sigma_j \cup \Sigma_j^\Gamma \cup \{\epsilon\}$ . We assume that strategies are observation based, that is if  $l_j(\rho) = l_j(\rho')$ , then  $\mu_j(\rho) = \mu_j(\rho')$ . A run  $\rho = C_0 \xrightarrow{e_1} C_1 \dots C_k$  conforms to strategy  $\mu_j$  iff,  $\forall i, \mu_j(l(C_0 \rightarrow \dots C_i)) \neq \epsilon$  implies  $e_{i+1} = \mu_j(l(C_0 \rightarrow \dots C_i))$  or  $e_{i+1} \notin \Sigma_j \cup \Sigma_j^\Gamma$ .

Intuitively, a strategy indicates to player  $u_j$  the next move to choose (either an action or an observation or nothing). Even if a particular action is advised, another player can play before  $u_j$  does. We will denote by  $Runs(\mathcal{S}, \mu_j)$  the runs of  $\mathcal{S}$  that conform to  $\mu_j$ . Let  $\mu_j$  be a strategy of  $u_j$  and  $\rho \in Runs(\mathcal{S}^\Gamma)$  be a run ending in a configuration  $C = (q, q_1, \dots, q_{|U|})$ , we now define the set of all possible configurations in which  $\mathcal{S}$  can be after observation  $l_j(\rho)$  under strategy  $\mu_j$ . It is inductively defined as follows:

- $\Delta_{\mu_j}(X, \mathcal{S}^\Gamma, \epsilon) = X$  for every set of configurations  $X$
- $\Delta_{\mu_j}(X, \mathcal{S}^\Gamma, w.e) = \begin{cases} \Delta_{\Sigma_o^j}(\Delta_{\mu_j}(X, \mathcal{S}^\Gamma, w), \mathcal{S}^\Gamma, e) & \text{if } e \in \Sigma_j \\ \Delta_{\mu_j}(X, \mathcal{S}^\Gamma, w) & \text{if } e = a_\gamma \in \Sigma_j^\Gamma, \\ (\Delta_{\mu_j}(X, \mathcal{S}^\Gamma, w))_{\setminus \gamma(q)} & \text{if } e \in \{tt, ff\} \\ \text{and } w = w'.a_\gamma \text{ for some } \gamma \in \Gamma \end{cases}$

Now,  $\Delta_{\mu_j}(\{C_0\}, \mathcal{S}^\Gamma, w)$  is the estimation of the possible set of reachable configurations that  $u_j$  can build after observing  $w$ . We can also define a set of plausible runs leading to observation  $w \in (\Sigma_o^j)^*$  by  $u_j$ . A run is *plausible* after  $w$  if its observation by  $u_j$  is  $w$ , and at every step of the run ending in some configuration  $C_k$  a test performed by  $u_j$  refine  $u_j$ 's estimation to a set of configuration that contain  $C_k$ . More formally, the set of plausible runs after  $w$  under strategy  $\mu_j$  is  $Pl_j(w) = \{\rho \in Runs(\mathcal{S}, \mu_j) \mid l_j(\rho) = w \wedge \rho \text{ is a run from } C_0 \text{ to a configuration } C \in \Delta_{\mu_j}(\{C_0\}, \mathcal{S}^\Gamma, w)\}$ .

We now redefine the notion of opacity in an active context. A strategy  $\mu_j$  of  $u_j$  to learn  $S_i$  is not efficient if despite the use of  $\mu_j$ , there is still a way to hide  $S_i$  for an arbitrary long time. In what follows, we assume that there is only one attacker of the system.

**Definition 5 (Opacity with active observation strategy):** A secret  $S_i$  is opaque for any observation strategy to user  $u_j$  in a system  $\mathcal{S}$  iff  $\nexists \mu_j$  and a bound  $K \in \mathbb{N}$ , such that  $\forall \rho \in Runs(\mathcal{S}, \mu_j)$ ,  $\rho$  has a prefix  $\rho_1$  of size  $\leq K$ ,  $l(Pl(\rho_1)) \subseteq \mathcal{L}(S_i)$ . A system  $\mathcal{S}$  is *opaque* for any observation strategy iff  $\forall i \neq j$ , secret  $S_i$  is opaque for any observation strategy of  $u_j$ .

Let us comment differences between passive (def. 2) and active opacity (def. 5). A system that is not U-opaque *may* leak information while a system that not opaque with active observation strategy *cannot avoid* leaking information if  $u_j$  implements an adequate strategy. U-opaque systems are not necessarily opaque with strategies, as active tests give additional

information that can disambiguate state estimation. However, if a system is U-opaque, then strategies that do not use disambiguation capacities do not leak secrets. Note also that a non-U-opaque system may leak information in more runs under an adequate strategy. Conversely, a non-opaque system can be opaque in an active setting, as the system can delay leakage of information for an arbitrary long time. Based on the definition of active opacity, we can state the following result:

**Theorem 6:** Given a system  $\mathcal{S} = (\mathcal{A}, U)$  with  $n$  agents and a set secrets  $S_1, \dots, S_n$ , observation alphabets  $\Sigma_o^1, \dots, \Sigma_o^n$  and observation capacities  $\Sigma_1^\Gamma, \dots, \Sigma_n^\Gamma$ , deciding whether  $\mathcal{S}$  is opaque with active observation strategies is EXPTIME-complete.

**Proof:**[sketch] An active attacker  $u_j$  can claim that the system is executing a run  $\rho$  that is secret for  $u_i$  iff it can claim with certainty that  $\rho$  is recognized by  $S_i$ . This can be achieved by maintaining an estimation of the system's current configuration, together with an estimation of  $S_i$ 's possible states. We build an arena with nodes of the form  $n = (b, C, s, ES)$  contains a player's name  $b$  (0 or 1): intuitively, 0 nodes are nodes where all agents but  $u_j$  can play, and 1 nodes are nodes where only agent  $u_j$  plays. Nodes also contain the current configuration  $C$  of  $\mathcal{S}$ , the current state  $s$  of  $S_i$ , an estimation  $ES$  of possible configurations of the system with secret's current state by  $u_j$ ,  $ES_j = \{(C_1, s_1), \dots, (C_k, s_k)\}$ . The attacker starts with an initial estimation  $ES_0 = \{(C_0, q_{0,i}^S)\}$ . Then, at each occurrence of an observable move, the state estimation is updated as follows : given a letter  $a \in \Sigma_o^j$ , for every pair  $(C_k, s_k)$ , we compute the set of pairs  $(C'_k, s'_k)$  such that there exists a runs from  $C_k$  to  $C'_k$ , that is labeled by a word  $w$  that is accepted from  $s_k$  and leads to  $s'_k$  in  $S_i$  and such that  $l_j(w) = a$ . The new estimation is the union of all pairs computed this way.

Moves in this arena represent actions of player  $u_j$  (from nodes where  $b = 1$  and actions from the rest of the system (see appendix for details). Obviously, this arena is of exponential size w.r.t. the size of configurations of  $\mathcal{S}$ .

A node  $n = (b, C, s, ES)$  is not secret if  $s \notin F_i^S$ , and secret otherwise. A node is *ambiguous* if there exists  $(C_p, s_p)$  and  $(C_m, s_m)$  in  $ES$  such that  $s_p \in F_i^S$  is secret and  $s_m \notin F_i^S$ . If the restriction of  $ES$  to its second components is contained in  $F_i^S$ ,  $n$  leaks secret  $S_i$ . The set of winning nodes in the arena is the set of nodes that leak  $S_i$ . Player  $u_j$  can take decisions only from its state estimation, and wins the game if it can reach a node in the winning set. This game is hence a partial information reachability game. Usually, solving such games requires computing an exponentially larger arena containing players beliefs, and then apply polynomial procedures for a perfect information reachability game. Here, as nodes already contain beliefs, there is no exponential blowup, and checking active opacity is hence in EXPTIME.

For the hardness part, we use a reduction from the problem of language emptiness for alternating automata to an active opacity problem. (see appendix for details)  $\square$

Moving from opacity to active opacity changes the complexity class from PSPACE-complete to EXPTIME-complete. This is due to the game-like nature of active opacity. However,

using observation capacities does not influence complexity: even if an agent  $u_j$  has no capacity, the arena built to verify opacity of  $S_i$  w.r.t.  $u_j$  is of exponential size, and the reduction from alternating automata used to prove hardness does not assume that observation capacities are used.

#### IV. OPACITY WITH THRESHOLD DISTANCES TO PROFILES

So far, we have considered passive opacity, i.e. whether a secret can be leaked during normal use of a system, and active opacity, i.e. whether an attacker can force secret leakage with an appropriate strategy and with the use of capacities. In this setting, the behavior of agents is not constrained by any security mechanism. This means that attackers can perform illegal actions with respect to their profile without being discovered, as long as they are feasible in the system.

We extend this setting to systems where agents behaviors are monitored by anomaly detection mechanisms, that can raise alarms when an user's behavior seems abnormal. Very often, abnormal behaviors are defined as difference between observed actions and a model of normality, that can be a discrete event model, a stochastic model,.... These models or *profiles* can be imposed *a priori* or learnt from former executions. This allows for the definition of profiled opacity, i.e. whether users that behave according to predetermined profile can learn a secret, and active profiled opacity, i.e. a setting where attackers can perform additional actions to refine their knowledge of the system's state and force secret leakage in a finite amount of time without leaving their normal profile.

One can assume that the behavior of an honest user  $u_j$  is a distributed system is predictable, and specified by his profile  $\mathcal{P}_j$ . The definitions of opacity (def. 2) and active opacity (def. 5) do not consider these profiles, i.e. agents are allowed to perform legally any action allowed by the system to obtain information. In our opinion, there is a need for a distinction between what is feasible in a system, and what is considered as normal. For instance, changing access rights of one of his file by an agent should always be legal, but changing access rights too many times within a few seconds should be considered as an anomaly. In what follows, we will assume that honest users behave according to their predetermined regular profile, and that deviating from this profile could be an active attempt to break the system's security. Yet, even if an user is honest, he might still have possibilities to obtain information about other user's secret. This situation is captured by the following definition of opacity wrt a profile.

*Definition 7:* A system  $\mathcal{S} = (\mathcal{A}, U)$  is opaque w.r.t. profiles  $\mathcal{P}_1, \dots, \mathcal{P}_n$  if  $\forall i \neq j, \forall w \in \mathcal{L}(S_i) \cap \mathcal{L}(S)$ ,

$$w \in \mathcal{L}(\mathcal{P}_j) \Rightarrow \pi_{\Sigma_o^j}^{-1}(\pi_{\Sigma_o^j}(w)) \cap \mathcal{L}(S) \not\subseteq \mathcal{L}(S_i)$$

Intuitively, a system is opaque w.r.t profiles of its users if it does not leak information when users stay within their profiles. If this is not the case, i.e. when  $w \notin \mathcal{L}(\mathcal{P}_j)$ , then one can assume that an anomaly detection mechanism that compares users action with their profiles can raise an alarm. Definition 7 can be rewritten as  $\forall i \neq j, \forall w \in \mathcal{L}(S_i) \cap \mathcal{L}(\mathcal{P}_j) \cap$

$\mathcal{L}(S), \pi_{\Sigma_o^j}^{-1}(\pi_{\Sigma_o^j}(w)) \cap \mathcal{L}(S) \not\subseteq \mathcal{L}(S_i)$  Hence, *PSPACE*-completeness of opacity in Theorem 3 extends to opacity with profiles: it suffices to find witness runs in  $\mathcal{L}(S) \cap \mathcal{L}(S_i) \cap \mathcal{L}(\mathcal{P}_j)$ .

*Corollary 8:* Deciding whether a system  $\mathcal{S}$  is opaque w.r.t. a set of profiles  $\mathcal{P}_1, \dots, \mathcal{P}_n$  is *PSPACE* complete.

If a system is U-opaque, then it is opaque w.r.t its agents profiles. Using profiles does not change the nature nor complexity of opacity question. Indeed, opacity w.r.t. a profile mainly consists in considering regular behaviors in  $\mathcal{L}(\mathcal{P}_j)$  instead of  $\mathcal{L}(A_j)$ . In the rest of the paper, we will however use profiles to *measure how much users deviate from their expected behavior* and quantify opacity accordingly.

One can similarly define a notion of active opacity w.r.t. profiles, by imposing that choices performed by an attacker are actions that does not force him to leave his profile. This can again be encoded as a game. This slight adaptation of definition 5 does not change the complexity class of the opacity question (as it suffices to remember in each node of the arena a state of the profile of the attacker). Hence active opacity with profiles is still a partial information reachability game, and is also EXPTIME-complete. Passive opacity (profiled or not) holds iff certain inclusion properties are satisfied by the modeled system, and active opacity holds if an active attacker has no strategy to win a partial information reachability game. Now, providing an answer to these opacity questions returns a simple boolean information on information leakage. It is interesting to quantify the notions of profiled and active opacity for several reasons. First of all, profiles can be seen as approximations of standard behaviors: deviation w.r.t. a standard profile can be due to errors in the approximation, that should not penalize honest users. Second, leaving a profile should not always be considered as an alarming situation: if profiles are learned behaviors of users, one can expect that from time to time, with very low frequency, the observed behavior of a user differs from what was expected. An alarm should not be raised as soon as an unexpected event occurs. Hence, considering that users shall behave exactly as depicted in their profile is a too strict requirement. A sensible usage of profiles is rather to impose that users stay *close* to their prescribed profile. The first step to extend profiled and active opacity to a quantitative setting is hence to define what "close" means.

*Definition 9:* Let  $u, v$  be two words of  $\Sigma^*$ . An *edit operation* applied to word  $u$  consists in inserting a letter  $a \in \Sigma$  in  $u$  at some position  $i$ , deleting a letter  $a$  from  $u$  at position  $i$ , or substituting a letter  $a$  for another letter  $b$  in  $u$  at position  $i$ .

Let  $OPs(\Sigma)$  denote the set of edit operations on  $\Sigma$ , and  $\omega(\cdot)$  be a cost function assigning a weight to each operation in  $OPs(\Sigma)$ . The *edit distance*  $d(u, v)$  between  $u$  and  $v$  is the minimal sum of costs of operations needed to transform  $u$  in  $v$ .

Several edit distances exist, the most known ones are

- the Hamming distance  $ham((u, v))$ , that assumes that  $OPs(\Sigma)$  contains only substitutions, and counts the number of substitutions needed to obtain  $u$  from  $v$  ( $u, v$  are supposed of equal lengths).

- the Levenshtein distance  $lev((u, v))$  is defined as the distance obtained when  $\omega(\cdot)$  assigns a unit to every operation (insertion, substitution, deletion). One can notice that  $lev((u, v))$  is equal to  $lev((v, u))$ , and that  $max(|u|, |v|) \geq lev((u, v)) \geq ||u| - |v||$ .

For a particular distance  $d(\cdot)$  among words, the *distance* between a word  $u \in \Sigma^*$  and a language  $R \subseteq \Sigma^*$  is denoted  $d(u, R)$  and is defined as  $d(u, R) = \min\{d(u, v) \mid v \in R\}$ .

We can now quantify opacity. An expected secure setting is that no secret is leaked when users have behaviors that are within or close enough from their expected profile. In other words, when the observed behavior of agents  $u_1, \dots, u_k$  resemble the behavior of their profiles  $\mathcal{P}_1, \dots, \mathcal{P}_k$ , no leakage should occur. Resemblance of  $u_i$ 's behavior in a run  $\rho$  labeled by  $w$  can be defined as the property  $d(w, \mathcal{L}(\mathcal{P}_i)) \leq K$  for some chosen notion of distance  $d(\cdot)$  and some threshold  $K$  fixed by the system designers. In what follows, we will use the Hamming and Levenshtein distances as a proximity measures w.r.t. profiles. However, we believe that this notion of opacity can be extended to many other distances. We are now ready to propose a quantified notion of opacity.

**Definition 10 (threshold profiled opacity):** A system  $\mathcal{S}$  is opaque wrt profiles  $\mathcal{P}_1, \dots, \mathcal{P}_n$  with tolerance  $K$  for a distance  $d$  iff  $\forall i \neq j, \forall w \in \mathcal{L}(S_i) \cap \mathcal{L}(\mathcal{S})$ ,

$$d(w, \mathcal{L}(\mathcal{P}_j)) \leq K \Rightarrow \pi_{\Sigma_o}^{-1}(\pi_{\Sigma_o^j}(w)) \cap \mathcal{L}(\mathcal{S}) \not\subseteq \mathcal{L}(S_i)$$

Threshold profiled opacity is again a passive opacity. In some sense, it provides a measure of how much anomaly detection mechanisms comparing users behaviors with their profiles are able to detect passive leakage. Consider the following situation: the system  $\mathcal{S}$  is opaque w.r.t. profiles  $\mathcal{P}_1, \dots, \mathcal{P}_n$  with threshold  $K + 1$  but not with threshold  $K$ . Then it means there exists a run of the system with  $K + 1$  anomalies of some user  $u_j$  w.r.t. profile  $\mathcal{P}_j$ , but no run with  $K$  anomalies. If anomaly detection mechanisms are set to forbid execution of runs with more than  $K$  anomalies, then the system remains opaque.

We can also extend the active opacity with thresholds. Let us denote by  $Strat_j^K$  the set of strategies that forbid actions leaving a profile  $\mathcal{P}_j$  if the behavior of the concerned user  $u_j$  is already at distance  $K$  from  $\mathcal{P}_j$  (the distance can refer to any distance, e.g., Hamming or Levenshtein).

**Definition 11 (active profiled Opacity):** A system  $\mathcal{S}$  is opaque w.r.t. profiles  $\mathcal{P}_1, \dots, \mathcal{P}_n$  with tolerance  $K$  iff  $\forall i \neq j, \nexists \mu_j \in Strat_j^K$  such that it is unavoidable for  $u_j$  to reach a correct state estimation  $X \subseteq F_i^S$  in all runs of  $Runs(\mathcal{S}, \mu_j)$ .

Informally, definition 10 says that a system is opaque if no attacker  $u_j$  of the system have a strategy that leaks a secret  $S_i$  and costs less than  $K$  units to reach this leakage. Again, we can propose a game version for this problem, where attacker  $u_j$  is not only passive, but also has to play his best actions in order to learn  $u_i$ 's secret.

A player  $u_j$  can attack  $u_i$ 's secret iff it has a strategy  $\mu_j$  to force a word  $w \in \mathcal{L}(S_i)$  that conforms to  $\mu_j$ , such that  $d(w, \mathcal{L}(\mathcal{P}_j)) \leq K$  and  $\pi_{\Sigma_o}^{-1}(\pi_{\Sigma_o^j}(w)) \cap \mathcal{L}(\mathcal{S}) \subseteq \mathcal{L}(S_i)$ . This can be seen as a partial information game between  $u_j$  and the rest of the system, where the exact state of each agent is

partially known to others. The system wins if it can stay forever in states where  $u_j$ 's estimates does not allow to know that the secret automaton  $S_i$  is in one of its accepting states. The arena is built in such a way that  $u_j$  stops playing differently from its profile as soon as it reaches penalty  $K$ . This is again a partial information reachability game and that is decidable on *finite* arenas [6]. Fortunately, we can show (in lemma 12 below) that the information to add to nodes with respect to the games designed for active opacity (in theorem 6) is finite.

**Lemma 12:** For a given automaton  $\mathcal{G}$ , one can compute an automaton  $\mathcal{G}^K$  that recognizes words at distance at most  $K$  of  $\mathcal{L}(\mathcal{G})$ , where the distance is either the Hamming or Levenshtein distance.

**Proof:** Let us first consider the Hamming distance. For an automaton  $\mathcal{G}_R = (Q_R, \rightarrow_R, q_{0R}, F_R)$ , we can design an automaton  $\mathcal{G}_{ham}^K = (Q^K, \rightarrow^K, q_0^K, F^K)$  that recognizes words at a distance at most  $K$  from the reference language  $\mathcal{L}(\mathcal{G})$ . We have  $Q^K = Q_R \times \{0..K\}$ ,  $F^K = Q \times \{0..K\}$ , and  $q_0^K = (q_0, 0)$ . Last, we give the transition function: we have  $((q, i), a, (q', i)) \in \rightarrow^K$  iff  $(q, a, q') \in \rightarrow_R$ , and  $((q, i), a, (q', i + 1)) \in \rightarrow^K$  if  $(q, a, q') \notin \rightarrow_R$  and  $i + 1 \leq K$ , and there exists  $b \neq a$  such that  $(q, b, q') \in \rightarrow_R$ . This way,  $\mathcal{G}_{ham}^K$  recognizes sequences of letters that end on a state  $(q_f, i)$  such that  $q_f$  is an accepting state of  $\mathcal{G}_R$ , and  $i \leq K$ . One can easily show that for any accepting path in  $\mathcal{G}_{ham}^K$  ending on state  $(q_f, i)$  recognizing word  $w$ , there exists a path in  $\mathcal{G}_R$  of identical length recognizing a word  $w'$  that is at hamming distance at most  $K$  of  $w$ . Similarly, let us consider any accepting path  $\rho = q_{0R} \xrightarrow{a_1} q_1 \dots \xrightarrow{a_n} q_f$  of  $\mathcal{G}_R$ . Then, every path of the form  $\rho^k = (q_{0R}, 0) \dots \xrightarrow{a_{i_1}} (q_{i_1}, 1) \dots (q_{i_{k-1}}, k - 1) \xrightarrow{a_{i_k}} (q_{i_k}, k) \dots \xrightarrow{a_n} (q_f, i)$  such that  $i \leq K$  and for every  $(q_{i_{j-1}}, j - 1) \xrightarrow{a_{i_j}} (q_{i_j}, j)$ ,  $a_{i_j}$  is not allowed in state  $q_{i_j}$  is a path that recognizes a word at distance  $i$  of a word in  $R$  and is also a word of  $\mathcal{G}_R^K$ . One can show by induction on the length of paths that the set of all paths recognizing words at distance at most  $k$  can be obtained by random insertion of at most  $k$  such letter changes in each path of  $\mathcal{G}_R$ . The size of  $\mathcal{G}_{ham}^K$  is exactly  $|\mathcal{G}_R| \times K$ .

Let us now consider the Levenshtein distance. Similarly to the Hamming distance, we can compute an automaton  $\mathcal{G}_{Lev}^K$  that recognizes words at distance at most  $K$  from  $\mathcal{L}(\mathcal{G})$ . Namely,  $\mathcal{G}_{Lev}^K = (Q_{lev}, \rightarrow_{lev}, q_{0,Lev}, F_{lev})$  where  $Q_{lev} = Q \times \{0..K\}$ ,  $q_{0,lev} = (q_0, 0)$ ,  $F_{lev} = F \times \{0..K\}$ . Last the transition relation is defined as  $((q, i), a, (q', i)) \in \rightarrow_{lev}$  if  $(q, a, q') \in \rightarrow$ ,  $((q, i), a, (q, i + 1)) \in \rightarrow_{lev}$  if  $\nexists q', (q, a, q') \in \rightarrow$  (this transition simulates insertion of letter  $a$  in a word),  $((q, i), a, (q', i + 1)) \in \rightarrow_{lev}$  if  $\exists (q, b, q') \in \rightarrow$  with  $b \neq a$  (this transition simulates substitution of a character),  $((q, i), \epsilon, (q', i + 1)) \in \rightarrow_{lev}$  if  $\exists (q, a, q') \in \rightarrow$  (this last move simulates deletion of a character from a word in  $\mathcal{L}(\mathcal{G})$ ).

One can notice that this automaton contains  $\epsilon$  transition, but after and  $\epsilon$ -closure, one obtains an automaton without epsilon that recognizes all words at distance at most  $K$  from  $\mathcal{L}(\mathcal{G})$ . The proof of correctness of the construction follows the same lines as for the Hamming distance, with the particularity that

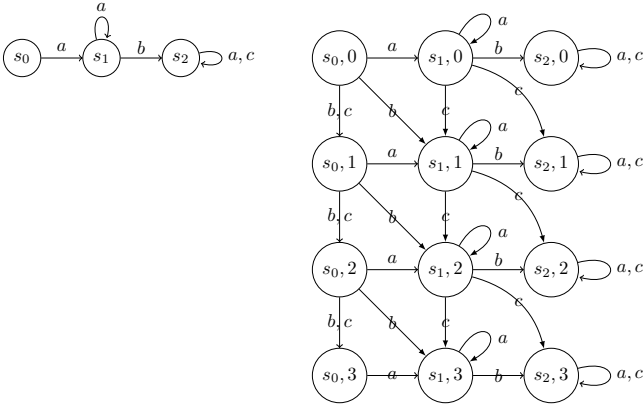


Fig. 1: An automaton  $\mathcal{G}$  and the automaton  $\mathcal{G}_{Ham}^3$  that recognizes words at Hamming distance  $\leq 3$  of  $\mathcal{L}(\mathcal{G})$ .

one can randomly insert transitions in paths, by playing letters that are not accepted from a state, leaving the system in the same state, and simply increasing the number of differences. Notice that if a word  $w$  is recognized by  $\mathcal{G}_{Lev}^K$  with a path ending in a state  $(q, i) \in F_{Lev}$ , this does not mean that the Levenshtein distance from  $\mathcal{L}(\mathcal{G})$  is  $i$ , as  $w$  can be recognized by another path ending in a state  $(q', j) \in F_{Lev}$  with  $j < i$ .  $\square$

One can notice that the automata built in the proof of lemma 12 are of size in  $O(K \cdot |\mathcal{G}|)$ , even after  $\epsilon$ -closure. Figure 1 represents an automaton  $\mathcal{G}$  that recognizes the prefix closure of  $a.a^*.b.(a+c)^*$ , and the automaton  $\mathcal{G}_{Ham}^3$ .

**Theorem 13:** Deciding threshold opacity for the Hamming and Levenshtein distance is PSPACE complete.

**Proof:** First of all, one can remark that, for a distance  $d(\cdot)$ , a system  $\mathcal{S}$  is not opaque if there exists a pair of users  $u_i, u_j$  and a word  $w$  in  $\mathcal{L}(\mathcal{S}) \cap \mathcal{L}(S_i)$  such that  $d(w, \mathcal{L}(\mathcal{P}_j)) \leq K$ , and  $\pi_{\Sigma_o}^{-1}(\pi_{\Sigma_o^j}(w)) \cap \mathcal{L}(\mathcal{S}) \subseteq \mathcal{L}(S_i)$ . As already explained in the proof of theorem 3,  $w$  belongs to  $\mathcal{L}(S_i)$  if a state  $q_w$  reached by  $S_i$  after reading  $w$  belongs to  $F_i^S$ . Still referring to the proof of Theorem 3, one can maintain online when reading letters of  $w$  the set  $reach_j(w)$  of possible configurations and states of  $S_i$  that are reached by a run which observation is the same as  $\pi_{\Sigma_o^j}(w)$ .

One can also notice that  $lev(w, \mathcal{L}(\mathcal{P}_j)) \leq K$  iff  $w$  is recognized by  $\mathcal{P}_{j,lev}^K$ , the automaton that accepts words at Levenshtein distance at most  $K$  from a word in  $\mathcal{P}_j$ . Again, checking online whether  $w$  is recognized by  $\mathcal{P}_{j,lev}^K$  consists in maintaining a set of states that can be reached by  $\mathcal{P}_{j,lev}^K$  when reading  $w$ . We can denote by  $reach_{j,lev}^K(w)$  this set of states. When no letter is read yet,  $reach_{j,lev}^K(\epsilon) = \{q_0^K\}$ , and if  $lev(w, \mathcal{L}(\mathcal{P}_j)) > K$ , we have  $reach_{j,lev}^K(w) = \emptyset$ , meaning that the sequence of actions played by user  $u_j$  have left the profile. We can maintain similarly a set of states  $reach_{j,Ham}^K(w)$  for the Hamming distance. In what follows, we will simply use  $reach_j^K(w)$  to denote a state estimation using Levenstein or Hamming distance.

Hence, non-opacity can be rephrased as existence of a run, labeled by a word  $w$  such that  $reach_j(w) \subseteq F_i^S$  and  $reach_{j,Ham}^K(w) \neq \emptyset$ . The contents of  $reach_j(w)$  and  $reach_{j,Ham}^K(w)$  after reading a word  $w$  can be recalled with a vector of

$h = |\mathcal{S}| + |\mathcal{P}_j^K|$  bits. Following the same arguments as in Theorem 3, it is also useless to consider runs of size greater than  $2^h$ . One can hence non-deterministically explore the whole set of states reached by  $reach_j(w)$  and  $reach_{j,Ham}^K(w)$  during any run of  $\mathcal{S}$  by remembering  $h$  bits and a counter which value is smaller or equal to  $2^h$ , and can hence be encoded with at most  $h$  bits. So, finding a witness for non-opacity is in NP-SPACE, and by Savitch's theorem and closure by complementation of PSPACE, opacity with a threshold  $K$  is in PSPACE.

For the hardness part, it suffices to remark that profiled opacity is exactly threshold profiled opacity with  $K = 0$ .  $\square$

**Theorem 14:** Deciding active profiled opacity for the Hamming and Levenshtein distance is EXPTIME-complete.

**Proof:**[sketch] Let us first consider the Hamming distance. One can build an arena for a pair of agents  $u_i, u_j$  as for the proof of theorem 6. This arena is made of nodes of the form  $(b, C, s, spjk, ES, d)$  that contain: a bit  $b$  indicating if it is  $u_j$  turn to play and choose the next move,  $C$  the current configuration of  $\mathcal{S}$ ,  $s$  the current state of  $S_i$ , the estimation of  $ES$  of possible pairs  $(C, s)$  of current configuration and current state of the secret by player  $u_j$ , and  $spjk$  a set of states of the automaton  $\mathcal{P}_{j,ham}^K$  that recognizes words that are at Hamming distance at most  $K$  from  $\mathcal{P}_j$ . In addition to this information, a node contains the distance  $d$  of currently played sequence w.r.t. profile  $\mathcal{P}_j$ . This distance can be easily computed: if all states of  $\mathcal{P}_{j,ham}^K$  memorized in  $spjk$  are pairs of state and distance, i.e.,  $spjk = \{(q_1, i_1), (q_2, i_2), \dots, (q_k, i_k)\}$  then  $d = \min\{i_1, \dots, i_k\}$ . User  $u_j$  (the attacker) has partial knowledge of the current state of the system (i.e. a configuration of  $\mathcal{S}$  and of the state of  $S_i$ ), perfect knowledge of  $d$ . User  $j$  wins if it can reach a node in which his estimation of the current state of secret  $S_i$  is contained in  $F_i^S$  (a non-ambiguous and secret node), without exceeding threshold  $K$ . The rest of the system wins if it can prevent player  $u_j$  to reach a non-ambiguous and secret node of the arena. We distinguish a particular node  $\perp$  reached as soon as the distance w.r.t. profile  $\mathcal{P}_j$  is greater than  $K$ . We consider this node as ambiguous, and every action from it gets back to  $\perp$ . Hence, after reaching  $\perp$ , player  $u_j$  has no chance to learn  $S_i$  anymore. The moves from a node to another are the same as in the proof for theorem 6, with additional moves from any node of the form  $n = (1, q, s, spjk, ES, d)$  to  $\perp$  using action  $a$  is the cost of using  $a$  from  $n$  exceeds  $K$ . We add an equivalence relation  $\sim$ , such that  $n = (b, q, s, spjk, ES, d) \sim n' = (b', q', s', spjk', ES', d')$  iff  $b = b'$ ,  $spjk = spjk'$ ,  $d = d'$ , and  $ES = ES'$ . Obviously,  $u_j$  has a strategy to violate  $u_i$ 's secret without exceeding distance  $K$  w.r.t. its profile  $\mathcal{P}_j$  iff there is a strategy to reach  $Win = \{(b, q, s, spjk, ES, d) \mid ES \subseteq F_i^S\}$  for player  $u_j$  with partial information that does not differentiate states in the equivalence classes of  $\sim$ .

This is a partial information reachability game over an arena of size in  $O(2 \cdot |Conf(\mathcal{S})| \cdot |S_i| \cdot 2^{|Conf(\mathcal{S})| \cdot |S_i| \cdot K \cdot |\mathcal{P}_j|})$ , that is exponential in the size of  $\mathcal{S}$  and of the secret  $S_i$  and profile  $\mathcal{P}_j$ . This setting is a partial information reachability game over an arena of exponential size. As in the Boolean setting, the nodes of the arena already contain a representation of the beliefs



that are usually computed to solve such games, and hence transforming this partial information reachability game into a perfect information game does not yield an exponential blowup. Hence, solving this reachability game is in EXPTIME.

The hardness part is straightforward: the emptiness problem for alternating automaton used for the proof of theorem 6 can be recast in a profiled and quantified setting by setting each profile  $\mathcal{P}_i$  to an automaton that recognizes  $(\Sigma_i^\Gamma)^*$  (i.e., users have the right to do anything they want as far as they always remain at distance 0 from their profile).  $\square$

## V. DISCOUNTING ANOMALIES

Threshold opacity is a first step to improve the standard Boolean setting. However, this form of opacity supposes that anomaly detection mechanisms memorize all suspicious moves of users and never revises their opinion that a move was unusual. This approach can be too restrictive. In what follows, we propose several solutions to discount anomalies. We first start by counting the number of substitutions in a bounded suffix with respect to the profile of an attacker. A suspicion score is computed depending on the number of differences within the suffix. This suspicion score increases if the number of errors in the considered suffix is above a maximal threshold, and it is decreased as soon as this number of differences falls below a minimal threshold. As in former sections, this allows for the definition of passive and active notions of opacity, that are respectively PSPACE-complete and EXPTIME-complete. We then consider the *mean* number of discrepancies w.r.t. the profile as a discounted Hamming distance.

### A. A Regular discounted suspicion measure

Let  $u \in \Sigma^K \cdot \Sigma^*$  and let  $v \in \Sigma^*$ . We denote by  $d^K(u, v)$  the distance between the last  $K$  letters of word  $u$  and any suffix of  $v$ , i.e.  $d^K(u, v) = \min\{d(u_{[|u|-K, |u|]}, v') \mid v' \text{ is a suffix of } v\}$ . Given a regular language  $R$  we define  $d^K(u, R) = \min\{d^K(u, v) \mid v \in R\}$

**Lemma 15:** Let  $R$  be a regular language. For a fixed  $K \in \mathbb{N}$ , and for every  $k \in [0..K]$ , one can compute an automaton  $\mathcal{C}_k$  that recognizes words which suffixes of length  $K$  are at Hamming distance  $k$  from a suffix of a word of  $R$ .

We now define a cost model, that penalizes users that get too far from their profile, and decreases this penalty when getting back closer to a normal behavior. For a profile  $\mathcal{P}_j$  and fixed values  $\alpha, \beta \leq K$  we define a suspicion function  $\Omega_j$  for words in  $\Sigma^*$  inductively:

$$\Omega_j(w) = 0 \text{ if } |w| \leq K$$

$$\Omega_j(a.w.b) = \begin{cases} \Omega_j(a.w) + 1 & \text{if } d^K(w.b, \mathcal{P}_j) \geq \beta \\ \max(\Omega_j(a.w) - 1, 0) & \text{if } d^K(w.b, \mathcal{P}_j) \leq \alpha \end{cases}$$

As an example, let us take as profile  $\mathcal{P}_j$  the automaton  $\mathcal{G}$  of Figure 1. Let us fix a suffix size of  $K = 5$ , an upper bound  $\beta = 2$  and a lower bound  $\alpha = 1$ . Suppose that a word  $w = a.a.a.c.b.b.a.c.b.a.a$  is read. When reading this word, one observes successively seven suffixes  $w_1, \dots, w_7$  of size 5. Suffixes  $w_1$  and  $w_7$  are at distance 1 from a suffix of  $\mathcal{G}$ , other suffixes at distance 2. Graphics if figure 2 shows the

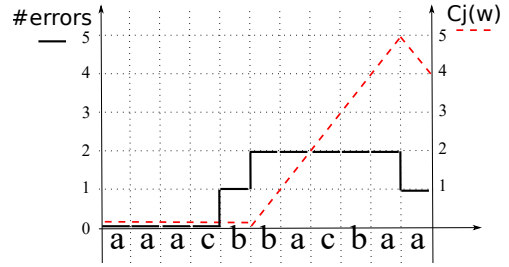


Fig. 2: Evolution of suspicion wrt profile of Figure 1 when reading word  $w = a.a.a.c.b.b.a.c.b.a.a$ .

distance  $d^K(w_{[i, i+5]}, \mathcal{P}_j)$  at each letter of  $w$  (plain line), and the evolution of the suspicion function (dashed line).

One can easily define a notion of passive opacity with respect to a suspicion threshold  $T$ . Again, verifying this property supposes finding a witness run of the system that leaks information without exceeding suspicion threshold, which can be done in PSPACE (assuming that  $T$  is smaller than  $2^{|\text{Conf}|}$ ). As for profiled opacity, we can define  $\text{Strat}^T$  the set of strategies of an user that never exceed suspicion level  $T$ . This immediately gives us the following definitions and results.

**Definition 16:** Let  $K \in \mathbb{N}$  be a suffix size,  $\alpha, \beta \leq K$  and  $T \in \mathbb{N}$  be a suspicion threshold.  $\mathcal{S}$  is opaque with suspicion threshold  $T$  iff  $\forall i \neq j, \forall w \in \mathcal{L}(\mathcal{S}_i) \cap \mathcal{L}(\mathcal{S})$ ,  $\Omega_j(w) < T$  implies  $\pi_{\Sigma_0}^{-1}(\pi_{\Sigma_0}(w)) \cap \mathcal{L}(\mathcal{S}) \not\subseteq \mathcal{L}(\mathcal{S}_i)$ .

**Theorem 17:** Opacity with suspicion threshold for the Hamming distance is PSPACE-complete.

**Definition 18:** Let  $K \in \mathbb{N}$  be a suffix size,  $\alpha, \beta \leq K$  and  $T \in \mathbb{N}$ .  $\mathcal{S}$  is actively opaque with suspicion threshold  $T$  iff  $\forall i \neq j$  there exists no strategy  $\mu_j \in \text{Start}^T$  such that it is unavoidable for  $u_j$  to reach a correct state estimation  $X \subseteq F_i^S$  in all runs of  $\text{Runs}(\mathcal{S}, \mu_j)$ .

**Theorem 19:** Active opacity with suspicion threshold for the Hamming distance is EXPTIME-complete.

**Proof:** We build an arena that contains nodes of the form  $n = (b, C, ES, EC_0, \dots, EC_k, sus)$ .  $C$  is the actual current configuration of  $\mathcal{S}^\Gamma$ ,  $ES$  is the set of pairs  $(C, s)$  of configuration and secret sates in which  $\mathcal{S}^\Gamma$  could be according to the actions observed by  $u_j$  and according to the belief refinements actions performed by  $u_j$ . Sets  $EC_1 \dots EC_k$  remembers sets of states of cost automata  $\mathcal{C}_0, \dots, \mathcal{C}_K$ . Each  $EC_i$  memorizes the states in which  $\mathcal{C}_i$  could be after reading the current word. If  $EC_i$  contains a final state, then the  $K$  last letters of the sequence of actions executed so far contain exactly  $i$  differences. Note that only one of these sets can contain an accepting state. Suspicion  $sus$  is a suspicion score between 0 and  $T$ . When reading a new letter, denoting by  $p$  the number of discrepancies of the  $K$  last letters wrt profiles, one can update the suspicion score using the definition of  $C_j$  above, depending on whether  $p \in [0, \alpha]$ ,  $p \in [\alpha, \beta]$  or  $p \in [\beta, K]$ .

The winning condition in this game is the set  $\text{Win} = \{(b, C, ES, EC_0, \dots, EC_k, sus) \mid ES \subseteq \text{Conf}(\mathcal{S}) \times F_i^S\}$ . We partition the set of nodes into  $V_0 = \{(b, C, ES, EC_0, \dots, EC_k, sus) \mid b = 0\}$  and  $V_1 = \{(b, C, ES, EC_0, \dots, EC_k, sus) \mid b = 1\}$ . We de-

fine moves from  $(b, C, ES, EC_0, \dots, EC_k, sus)$  to  $(1 - b, C, ES, EC_0, \dots, EC_k, sus)$  symbolizing the fact that it is user  $u_j$ 's turn to perform an action. There is a move from  $n = (b, C, ES, EC_0, \dots, EC_k, sus)$  to  $n' = (b', C', ES, EC'_0, \dots, EC'_k, sus')$  if there is a transition  $(C, a, C')$  in  $\mathcal{S}^\Gamma$  performed by an user  $u_i \neq u_j$ , and  $a$  is not observable by  $u_j$ . There is a move from  $n = (b, C, ES, EC_0, \dots, EC_k, sus)$  to  $n' = (b', C', ES', EC'_0, \dots, EC'_k, sus)$  if there is a transition  $(C, a, C')$  in  $\mathcal{S}^\Gamma$  performed by an user  $u_i \neq u_j$  and  $a$  is observable by  $u_j$ . We have  $ES' = \Delta_{\Sigma_0^j}(ES, \mathcal{S}^\Gamma, a)$ . Suspicion and discrepancies observation (sets  $EC_i$ ) remain unchanged as this move does not represent an action played by  $u_j$ .

There is a move from  $n = (b, C, ES, EC_0, \dots, EC_k, sus)$  to  $n' = (1 - b, C', ES', EC'_0, \dots, EC'_k, sus)$  if  $b = 1$  and there is a transition  $(q, a, q')$  in  $\mathcal{S}^\Gamma$  performed by user  $u_j$  from the current configuration. Set  $ES$  is updated as before  $ES' = \Delta_{\Sigma_0^j}(ES, \mathcal{S}^\Gamma, a)$  and sets  $EC_i$  are updated according to transition relation  $\delta_i^{su}$  of automaton  $\mathcal{C}_i$ , i.e.  $EC'_i = \delta_i^{su}(EC_i, a)$ . Similarly,  $sus'$  is the new suspicion value obtained after reading  $a$ . Last, there is a move from  $n = (b, C, ES, EC_0, \dots, EC_k, sus)$  to  $n' = (b, C, ES', EC'_0, \dots, EC'_k, sus)$ , if there is a sequence of moves  $(C, a, C(a_\gamma)), (C(a_\gamma), a_\gamma(q), C)$  in  $\mathcal{S}^\Gamma$ ,  $ES' = ES/a_\gamma(q)$ , and  $EC'_i$ 's and  $sus'$  are computed as in the former case.

As for the proofs of theorems 6 and 14, opacity can be brought back to a reachability game of partial information, and no exponential blowup occurs to solve it. For the hardness, there is a reduction from active profiled opacity. Indeed, active profiled opacity can be expressed as a suspicion threshold opacity, by setting  $\alpha = \beta = K = 0$ , to disallow attackers to leave their profile.  $\square$

## B. Discounted Opacity : an open problem

A frequent interpretation of discounting is that weights or penalties attached to a decision should decrease progressively over time, or according to the length of runs. This is captured by averaging contribution of individual moves.

*Definition 20:* The *discounted Hamming distance* between a word  $u$  and a language  $R$  is the value  $\hat{d}(u, R) = \frac{ham(u, R)}{|u|}$

This distance measures the average number of substitutions in a word  $u$  with respect to the closest word in  $R$ . The next quantitative definition considers a system as opaque if an active attacker can not obtain a secret while maintaining a mean number of differences w.r.t. its expected behavior below a certain threshold. Let  $\lambda \in \mathbb{Q}$  be a positive rational value. We denote by  $\widehat{Strat}^\lambda(R)$  the set of strategies that does not allow an action  $a$  after a run  $\rho$  labeled by a sequence of actions  $w$  if  $\hat{d}(w.a, R) > \lambda$ .

*Definition 21 (Discounted active Opacity):* A system  $\mathcal{S}$  is opaque wrt profiles  $\mathcal{P}_1, \dots, \mathcal{P}_n$  with discounted tolerance  $\lambda$  iff  $\forall i \neq j, \nexists \mu_j \in \widehat{Strat}^\lambda(\mathcal{P}_j)$ , strategy of agent  $u_j$  such that it is unavoidable for  $u_j$  to reach a correct state estimation  $X \subseteq F_S^i$  in all runs of  $Runs(\mathcal{S}, \mu_j)$ .

A system is opaque in a discounted active setting iff one can find a strategy for  $u_j$  to reach a state estimation that reveals the secret  $S_i$  while maintaining a discounted distance wrt  $\mathcal{P}_j$  smaller than  $\lambda$ . At first sight, this setting resembles discounted games with partial information, already considered in [13]. It was shown that finding optimal strategies for such mean payoff games is in  $NP \cap co-NP$ . The general setting for mean payoff games is that average costs are values of nodes in an arena, i.e. the minimal average reward along infinite runs that one can achieve with a strategy starting from that node. As a consequence, values of nodes are mainly values on connected components of an arena, and costs of moves leading from a component to another have no impact. In our setting, the game is not a value minimization over infinite run, but rather a co-reachability game, in which at any moment in a run, one shall not exceed a mean number of unexpected moves.

For a fixed pair of users  $u_i, u_j$ , we can design an arena with nodes of the usual form  $n = (b, C, ES, l, su)$  in which  $b$  indicates whether it is  $u_j$ 's turn to play,  $C$  is the current configuration of the system,  $ES$  the estimation of the current configuration and of the current state of secret  $S_i$  reached,  $l$  is the number of moves played so far, and  $su$  the number of moves that differ from what was expected in  $\mathcal{P}_j$ . As before, the winning states for  $u_j$  are the states where all couples in state estimation refer to an accepting state of  $S_i$ . In this arena, player  $u_j$  loses if it can never reach a winning node, or if it plays an illegal move from a node  $n = (b, C, ES, l, su)$  such that  $\frac{su+1}{l+1} > \lambda$ . One can immediately notice that defined this way, our arena is not finite anymore.

Consider the arena used in theorem 6, i.e. composed of nodes of the form  $n = (b, C, ES)$  that only build estimations of the attacker. Obviously, when ignoring mean number of discrepancies, one can decide whether the winning set of nodes is reachable from the initial node under some strategy in polynomial time (wrt the size of the arena). The decision algorithm builds an attractor for the winning set (see for instance [8] for details), but can also be used to find short paths under an adequate strategy to reach  $Win$  (without considering mean number of discrepancies). If one of these paths keeps the mean number of discrepancies lower or equal to  $\lambda$  at each step, then obviously, this is a witness for non-opacity. However, if no such path exists, there might still be a way to play longer runs that decrease the mean number of discrepancies before moving to a position that requires less steps to reach the winning set.

We can show an additional sufficient condition : Let  $\rho = n_0.n_1 \dots n_w$  be a path of the arena in theorem 6 (without length nor mean number of discrepancies recall) from  $n_0$  to a winning node  $n_w$ . Let  $d_i$  denote the number of discrepancies with respect to profile  $\mathcal{P}_j$  at step  $i$ . Let  $n_i$  be a node of  $\rho$  such that  $\frac{d_i}{i} \leq \lambda$  and  $\frac{d_{i+1}}{i+1} > \lambda$ . We say that  $u_j$  can enforce a decreasing loop  $\beta = n_j.n_{j+1} \dots n_j$  at node  $n_j$  if  $\beta$  is a cycle that  $u_j$  can enforce with an appropriate strategy, and if the mean number of discrepancies is smaller in  $\rho_\beta = n_0 \dots n_j.\beta$  than in  $n_0 \dots n_j$ , and the mean cost of any prefix of  $\beta$  is smaller than  $\lambda$ . A consequence is that the mean cost  $M_\beta$  of cycle  $\beta$  is smaller than  $\lambda$ . We then have a sufficient condition:

*Proposition 22:* Let  $\rho$  be a winning path in an arena built to check active opacity for users  $u_i, u_j$  such that  $\frac{d_i}{i} > \lambda$  for some  $i \leq |\rho|$ . If there exists a node  $n_b$  in  $\rho$  such that  $\frac{d_k}{k} \leq \lambda$  for every  $k \leq b$  and  $u_j$  can enforce a decreasing loop at  $n_b$ , then  $u_j$  has a strategy to learn  $S_i$  without exceeding mean number of discrepancies  $\lambda$ .

**Proof:** The winning path is of the form  $\rho = n_0.n_1 \dots n_b.n_{b+1} \dots n_w$ . Let  $d_b$  be the number of discrepancies in  $n_0.n_1 \dots n_b$  and  $\lambda_b = \frac{d_b}{b}$ . Player  $u_j$  can choose any integer value  $B$  and enforce path  $\rho_B = n_0.n_1 \dots n_b.\beta^B$ . The mean number of discrepancies in  $\rho_B$  is equal to  $\frac{d_b + B \cdot d_\beta}{i + B \cdot |\beta|}$ , i.e. as  $B$  increases, this number tends towards  $M_\beta$ . Similarly, if  $B$  is large enough, playing any prefix of  $n_{b+1} \dots n_w$  to reach the winning set does not increase enough the mean number of discrepancies to exceed  $\lambda$ . A lower bound for  $B$  such that  $\lambda$  is never exceeded in  $n_0 \dots n_b.\beta^B.n_{b+1} \dots n_w$  can be easily computed. Hence, if one can find a path in a simple arena withouts mean discrepancy counts, and a decreasing loop in this path, then  $u_j$  has a strategy to learn  $S_i$  without exceeding threshold  $\lambda$ .  $\square$

## VI. CONCLUSION

We have shown several ways to quantify opacity with passive and active attackers. In all cases, checking passive opacity can be brought back to a language inclusion question, and is hence PSPACE-complete. In active settings, opacity violation is brought back to existence of strategies in reachability games over arenas which nodes represent beliefs of agents, and is EXPTIME-complete.

Suspicion can be discounted or not. Non-discounted suspicions simply counts the number of anomalies w.r.t. a profile, and raises an alarm when a maximal number  $K$  of anomalies is exceeded. We have shown that when anomalies are substitutions, deletions and insertions of actions, words with less than  $K$  anomalies w.r.t. the considered profile (words at Hamming or Levenshtein distance  $\leq K$ ) are recognized by automata of linear size. This allows to define active and passive profiled opacity, with the same PSPACE/EXPTIME-complete complexities. A crux in the proofs is that words at distance lower than  $K$  of a profile are recognized by automata. A natural extension of this work is to see how regular characterization generalizes to other distances.

Discounting the number of anomalies is a key issue to avoid constantly raising false alarms. It is reasonable to consider that the contribution to suspicion raised by each anomaly should decrease over time. The first solution proposed in this paper computes a suspicion score depending on the number of discrepancies found during the last actions of an agent. When differences are only substitutions, one can use finite automata to maintain online the number of differences. This allows to enhance the arenas used in the active profiled setting without changing the complexity class of the problem (checking regular discounted suspicion remains EXPTIME-complete). Again, we would like to see if other distances (eg the Levenstein distance) and suspicion scores can be regular, which would allow for the defition of new opacity measures.

Discounted suspicion weights discrepancies between the expected and actual behavior of an agent according to run length. This suspicion measure can be seen as a quantitative game, where the objective is to reach a state leaking information without exceeding an *average* distance of  $\lambda \in \mathbb{Q}$ . In our setting, the mean payoff has to be compared to a threshold at every step. This constraint can be recast as a reachability property for timed automata with one stopwatch and linear diagonal constraints on clock values. We do not know yet if this question is decidable but we provide a sufficient condition for discounted opacity violation.

In the models we proposed, discounting is performed according to runs length. However, it seems natural to consider discrepancies that have occurred during the last  $\Delta$  seconds, rather than This requires in particular considering timed systems and they timed runs. It is not sure that adding timing to our setting preserves decidability, as opacity definitions rely a lot on languages inclusion, which are usually undecidable for timed automata [1]. If time is only used to measure durations elapsed between actions of an attacker, then we might be able to recast the quantitative opacity questions in a decidable timed setting, using decidability results for timed automata with one clock [9] or event-clock timed automata.

## REFERENCES

- [1] R. Alur and D.L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
- [2] E. Badouel, M.A. Bednarczyk, A.M. Borzyszkowski, B. Caillaud, and P. Darondeau. Concurrent secrets. *Discrete Event Dynamic Systems*, 17(4):425–446, 2007.
- [3] B. Bérard, J. Mullins, and M. Sassolas. Quantifying opacity. *Mathematical Structures in Computer Science*, 25(2):361–403, 2015.
- [4] J. Bryans, M. Koutny, L. Mazaré, and P.Y.A. Ryan. Opacity generalised to transition systems. In *FAST 2005*, volume 3866 of *LNCS*, pages 81–95, 2006.
- [5] F. Cassez, J. Dubreil, and H. Marchand. Synthesis of opaque systems with static and dynamic masks. *Formal Methods in System Design*, 40(1):88–115, 2012.
- [6] K. Chatterjee and L. Doyen. The complexity of partial-observation parity games. In *LPAR'10*, volume 6397 of *LNCS*, pages 1–14, 2010.
- [7] J.A. Goguen and J. Meseguer. Security policies and security models. In IEEE Computer Society Press, editor, *Proc of IEEE Symposium on Security and Privacy*, pages 11–20, April 1982.
- [8] E. Grädel and W. Thomas. *Automata Logics, and Infinite Games : A Guide to Current Research*, volume 2500 of *LNCS*. 1998.
- [9] J. Ouaknine and J. Worrell. On the language inclusion problem for timed automata: Closing a decidability gap. In *LICS 2004*, pages 54–63, 2004.
- [10] J.H. Reif. Universal games of incomplete information. In *STOC*, ACM Press, 1979.
- [11] A. Sabelfeld and A.C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.
- [12] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.
- [13] U. Zwick and M.S. Paterson. The complexity of mean payoff games. In *International Computing and Combinatorics Conference*, volume 959 of *LNCS*, pages 1–10, 1995.

## PROOF OF THEOREM 3

**Proof:** Let us first prove that  $U$ -opacity is in PSPACE. A system is not opaque if one can find a pair of users  $u_i, u_j$ , and a run  $w$  of  $\mathcal{S}$  such that  $w \in \mathcal{L}(S_i)$  and  $\pi_{\Sigma_o}^{-1}(\pi_{\Sigma_o^j}(w)) \cap \mathcal{L}(\mathcal{S}) \subseteq \mathcal{L}(S_i)$ . One can non-deterministically choose a pair of users  $u_i, u_j$  in space logarithmic in  $n$ , and check that  $i \neq j$  in logarithmic space. To decide whether a run of  $\mathcal{S}$  belongs to  $S_i$ , it is sufficient to know the set of states reached by  $S_i$  after recognizing  $w$ . A word  $w$  belongs to  $\mathcal{L}(S_i)$  is the state  $q_w$  reached by  $S_i$  after reading  $w$  belongs to  $F_i^S$ . Now, observe that an user  $u_j$  does not have access to  $w$ , but can only observe  $\pi_{\Sigma_o^j}(w)$ , and may hence believe that the run actually played is any run with identical observation, i.e. any run of  $\pi_{\Sigma_o^j}^{-1}(\pi_{\Sigma_o^j}(w)) \cap \mathcal{L}(\mathcal{S})$ .

Let  $\rho$  be a run of  $\mathcal{S}$ , one can build online the set of states  $reach_j(w)$  that are reached by a run which observation is the same as  $\pi_{\Sigma_o^j}(w)$ . We have  $reach_j(\epsilon) = \{q \in Q_i^S \mid \exists w, q_{0,i}^S \xrightarrow{w} q \wedge \pi_{\Sigma_o^j}(w) = \epsilon\}$  and  $reach_j(w.a) = \{q \in Q_i^S \mid \exists q' \in reach_j(w), \exists w', q' \xrightarrow{w'} q \wedge \pi_{\Sigma_o^j}(w') = a\}$ . Obviously, a word  $w$  witnesses a secret leakage from  $S_i$  to  $u_j$  if  $reach_j(w) \subseteq F_i^S$ . To play a run of  $\mathcal{S}$ , it is hence sufficient to remember a configuration of  $\mathcal{S}$  and a subset of states of  $S_i$ . Let  $q_\rho$  denote the pair  $(q, X)$  reached after playing run  $\rho$ .

Now we can show that witness runs with at most  $K_1 = |\text{Conf}| \cdot 2^{|\mathcal{S}_i|}$  letters observable by  $u_j$  suffice. Let us assume that there exists a witness  $\rho$  of size  $\geq K_1$ . Then,  $\rho$  can be partitioned into  $\rho = \rho_1 \cdot \rho_2 \cdot \rho_3$  such that  $q_{\rho_1} = q_{\rho_1 \cdot \rho_2}$ . Hence,  $\rho_1 \cdot \rho_3$  is also a run that witness a leakage of secret  $S_i$  to  $u_j$ , but of smaller size.

Hence one can find a witness of secret leakage by a non-deterministic exploration of size at most  $|\text{Conf}| \cdot 2^{|\mathcal{S}_i|}$ . To find such run, one only needs to remember a configuration of  $\mathcal{S}$  (which can be done with  $\log(|\mathcal{S}|)$  bits, all states of  $reach_j(\rho)$  for the current run  $\rho$  followed in  $\mathcal{S}$ , which can be done with  $|\mathcal{S}_i|$  bits of information, and an integer of size at most  $K_1$ , which requires  $\log|\mathcal{S}| \cdot |\mathcal{S}_i|$  bits. Finding a witness can hence be done in NPSPACE, and by Savitch's lemma it is in PSPACE. As PSPACE is closed by complement, deciding opacity of a system is in PSPACE.

Let us now consider the hardness part. We will reduce the non-universality of any regular language to an opacity problem. As universality is in PSPACE, non-universality is also in PSPACE. The language of an automaton  $\mathcal{B}$  defined over an alphabet  $\Sigma$  is not universal iff  $\mathcal{L}(\mathcal{B}) \neq \Sigma^*$ , or equivalently if  $\Sigma^* \not\subseteq \mathcal{L}(\mathcal{B})$ . For any automaton  $\mathcal{B}$ , one can design a system  $\mathcal{S}_{\mathcal{B}}$  with two users  $u_1, u_2$  such that  $\mathcal{S}_1 = \mathcal{B}$ ,  $\mathcal{L}(\mathcal{S}_2) = a \cdot \Sigma^*$  for some letter  $a$ ,  $\mathcal{A}$  accepts all actions, i.e. is such that  $\mathcal{L}(\mathcal{A}) = \Sigma^*$ ,  $\Sigma_o^2 = \Sigma_o^1 = \emptyset$ . Clearly, for every run of  $\mathcal{S}$ ,  $u_1$  observes  $\epsilon$ , and hence leakage can not occur from  $u_2$  to  $u_1$  (one cannot know whether a letter and in particular  $a$  was played). So the considered system is opaque iff  $\forall w \in \mathcal{L}(\mathcal{S}_1) \cap \mathcal{L}(\mathcal{S}), \pi_{\Sigma_o^2}^{-1}(\pi_{\Sigma_o^2}(w)) \not\subseteq \mathcal{L}(\mathcal{S}_1)$ . However, as  $\Sigma_o^2 = \emptyset$ , for every  $w$ ,  $\pi_{\Sigma_o^2}^{-1}(\pi_{\Sigma_o^2}(w)) = \Sigma^*$ . That is, the system is opaque iff  $\Sigma^* \not\subseteq \mathcal{L}(\mathcal{B})$ .  $\square$

**Proof:** An active attacker  $u_j$  can claim that the system is executing a run  $\rho$  that is secret for  $u_i$  iff it can claim with certainty that  $\rho$  is recognized by  $S_i$ . This can be achieved by maintaining an estimation of the system's current configuration, together with an estimation of  $S_i$ 's possible states. We build an arena with nodes  $N_0 \cup N_1$ . Each node of the form  $n = (b, C, s, ES)$  contains :

- a player's name  $b$  (0 or 1). Intuitively, 0 nodes are nodes where all agents but  $u_j$  can play, and 1 nodes are nodes where only agent  $u_j$  plays.
- the current configuration  $C$  of  $\mathcal{S}$
- the current state  $s$  of  $S_i$
- an estimation  $ES$  of the system's configuration and secret's current state by  $u_j$ ,  $ES_j = \{(C_1, s_1), \dots, (C_k, s_k)\}$

We write  $C \xrightarrow{\sigma} C'$  iff there exists a sequence of transitions of  $\mathcal{S}$  which observation by  $u_j$  is  $\sigma$ , and  $s \xrightarrow{\sigma}_i^S s'$  if there is such a sequence from  $s$  to  $s'$  in  $S_i$ . Then we define moves among nodes as a relation  $\delta^G \subseteq N_0 \cup N_1 \times N_0 \cup N_1$ .

- $(n, n') \in \delta^G$  if  $n$  and  $n'$  differ only w.r.t. their player's name
- $(n, n') \in \delta^G$  if  $n = (0, C, s, ES)$ ,  $n' = (1, C', s', ES')$  and there exists  $\sigma \in (\Sigma \setminus \Sigma_j) \cap \Sigma_o^j$  such that  $C \xrightarrow{\sigma} C'$ ,  $s \xrightarrow{\sigma}_i^S s'$  and  $ES'$  is th set of pairs  $(C_m, s_m)$  such that there exits a pair  $(C_p, s_p)$  in  $ES$ , and a sequence  $\rho$  of transitions from  $C_p$  to  $C_m$ , labeled by a word  $w$  such that  $\Pi_j(w) = \sigma$ , and one can move in  $S_i$  from  $s_p$  to  $s_m$  by reading  $w$ . Note that this set of sequences needs not be finite, but one can find in  $O(|\text{Conf}|)$  the set of possible pairs that are accessible while reading  $\sigma$ .
- $(n, n') \in \delta^G$  if  $n = (1, C, s, ES)$ ,  $n' = (1, C', s', ES')$  and there exists  $\sigma \in \Sigma_j$ , a transition  $C \xrightarrow{\sigma} C'$  in  $\mathcal{S}$ , a transition  $(s, \sigma, s') \in \rightarrow_i^S$  and  $ES'$  is the set of pairs of the form  $(C'_m, s'_m)$  such that there exists  $(C_m, s_m) \in ES$   $(C_m, \sigma, C'_m) \in \rightarrow$  and  $(s_m, \sigma, s'_m) \rightarrow_i^S$ .
- $(n, n') \in \delta^G$   $n = (1, C, s, ES)$ ,  $n' = (1, C, s, ES')$  if there exists  $\gamma \in \Sigma_j^\Gamma$  such that  $ES'$  is the refinement of  $ES$  by  $a_\gamma(\text{state}(C))$ . We assume that checking the status of a proposition does not affect the secrets of other users.

We says that a node  $n = (b, C, s, ES)$  is not secret if  $s \notin F_i^S$ , and say that  $n$  is secret otherwise. We say that a node is *ambiguous* if there exists  $(C_p, s_p)$  and  $(C_m, s_m)$  in  $ES$  such that  $s_p$  is secret and  $s_m$  is not. If the restriction of  $ES$  to it second components is contained in  $F_i^S$ , we says that  $n$  *leaks* secret  $S_i$ .

We equip the arena with an equivalence relation  $\sim \subseteq N_0 \times N_0 \cup N_1 \times N_1$ , such that  $n = (b, C, s, ES) \sim n' = (b', C', s', ES')$  iff  $b = b' = 1$  and  $ES = ES'$ . Intuitively,  $n \equiv n'$  if and only if they are nodes of agent  $u_j$ , and  $u_j$  cannot distinguish  $n$  from  $n'$  using the knowledge it has on executions leading to  $n$  and to  $n'$ .

Clearly, secret  $S_i$  is not opaque to agent  $u_j$  in  $\mathcal{S}$  iff there exists a strategy to make a leaking node accessible. This can be encoded as a partial information reachability game  $G =$

$(N_0 \uplus N_1, \delta^G, \equiv, Win)$ , where  $Win$  is the set of all leaking nodes. In these games, the strategy must be the same for every node in the same class of  $\equiv$  (i.e. where  $u_j$  has the same state estimation). Usually, partial information games are solved at the cost of an exponential blowup, but we can show that in our case, complexity is better. First, let us compute the maximal size of the arena. A node is of the form  $n = (b, C, s, ES)$ , hence the size of the arena  $|G|$  is in  $O(2 \cdot |Conf| \cdot |\mathcal{S}_i| \cdot 2^{|Conf| \cdot |S_i|})$  (and it can be built in time  $O(|Conf| \cdot |G|)$ ). Partial information reachability games are known to be EXPTIME-complete [10]. Note here that only one player is blind, but this does not change the overall complexity, as recalled by [6]. However, solving games of partial information consists in computing a "belief" arena  $G^B$  that explicitly represent players beliefs (a partial information on a state is transformed into a full knowledge of a belief), and then solve the complete information game on arena  $G^B$ . This usually yields an exponential blowup. In our case, this blowup is not needed, and the belief that would be computed to solve a partial information game simply duplicates the state estimation that already appears in the partial information arena. Hence, deciding opacity with active observation strategies can be done with  $|U|^2$  opacity tests (one for each pair of users) of exponential complexity, in only in EXPTIME.

Let us now prove the hardness of opacity with active attackers. We reduce the problem of emptiness of alternating automata to an opacity question. An alternating automaton is a tuple  $\mathcal{A}_{alt} = (Q, \Sigma, \delta, s_0, F)$  where  $Q$  contains two distinct subsets of states  $Q_{\forall}, Q_{\exists}$ .  $Q_{\forall}$  is a set of universal states,  $Q_{\exists}$  is a set of existential states,  $\Sigma$  is an alphabet,  $\delta \subseteq (Q_{\forall} \cup Q_{\exists}) \times \Sigma \times (Q_{\forall} \cup Q_{\exists})$  is a transition relation,  $s$  is the initial state and  $F$  is a set of accepting states. A *run* of  $\mathcal{A}_{alt}$  over a word  $w \in \Sigma^*$  is an acyclic graph  $G_{\mathcal{A}_{alt}, w} = (N, \rightarrow)$  where nodes in  $N$  are elements of  $Q \times \{1 \dots |w|\}$ . Edges in the graph connect nodes from a level  $i$  to a level  $i+1$ . The root of the graph is  $(s, 1)$ . Every node of the form  $(q, i)$  such that  $q \in Q_{\exists}$  has a single successor  $(q', i+1)$  such that  $q' \in \delta(q, w_i)$  where  $w_i$  is the  $i^{th}$  letter of  $w$ . For every node of the form  $(q, i)$  such that  $q \in Q_{\forall}$ , and for every  $q'$  such that  $q' \in \delta(q, w_i)$ ,  $((q, i), (q', i+1))$  is an edge. A run is complete if all its nodes with index in  $1..|w| - 1$  have a successor. It is accepting if all paths of the graph end in a node in  $F \times \{|w|\}$ . Notice that due to non-deterministic choice of a successor for existential states, there can be several runs of  $\mathcal{A}_{alt}$  for a word  $w$ . The emptiness problem asks whether there exists a word  $w \in \Sigma^*$  that has an accepting run. We will consider, without loss of generality that alternating automata are complete, i.e. all letters are accepted from any state. If there is no transition of the form  $(q, a, q')$  from a state  $q$ , one can nevertheless create a transition to a non-accepting absorbing state while preserving the language recognized by the alternating automaton.

Let us now show that the emptiness problem for alternating automata can be recast in an active opacity question. We will design three automata  $\mathcal{A}, \mathcal{A}_1, \mathcal{A}_2$ . The automata  $\mathcal{A}_1$  and  $\mathcal{A}_2$  are agents. Agent 1 performs actions from universal states and agent 2 chooses the next letter to recognize and performs actions from existential states. The automaton  $\mathcal{A}$  serves as a

communication medium between agents, indicates to  $\mathcal{A}_2$  the next letter to recognize, and synchronizes agents 1 and 2 when switching the current state of the alternating automaton from an existential state to an universal state or conversely.

We define  $\mathcal{A} = (Q_s, \rightarrow_s, \Sigma_s)$  with  $\Sigma_s = \{(end, 2 \rightarrow A); (end, A \rightarrow 1)\} \cup \Sigma \times \{2 \rightarrow A, A \rightarrow 1\} \times (Q_{\exists} \cup U) \times \{1 \rightarrow A, A \rightarrow 2, 2 \rightarrow A, A \rightarrow 2\}$ . To help readers, the general shape of automaton  $\mathcal{A}$  is given in Figure 3.

States of  $\mathcal{A}$  are of the form  $U, (U, \sigma), W, dU, dq_i, wq_i$  for every state in  $Q$ , and  $Eq_i$  for every existential state  $q_i \in Q_{\exists}$ . The initial state of  $\mathcal{A}$  is state  $U$  if  $s_0$  is an universal state, or  $s_0$  if  $s_0$  is existential. State  $U$  has  $|\Sigma|$  outgoing transitions of the form  $(U, \langle \sigma, 2 \rightarrow A \rangle, (U, \sigma))$ , indicating that the next letter to recognize is  $\sigma$ . It also has a transition of the form  $(U, \langle end, 2 \rightarrow A \rangle, end_1)$  indicating that  $\mathcal{A}_2$  has decided to test whether  $\mathcal{A}_1$  is in a secret state (i.e. simulates an accepting state of  $\mathcal{A}_{alt}$ ). There is a single transition  $(end_1, \langle end, A \rightarrow 2 \rangle, end_2)$  from state  $end_1$ , and a single transition  $(end_2, \langle Ackend, A \rightarrow 1 \rangle, end_3)$  indicating to  $\mathcal{A}_2$  that  $\mathcal{A}_1$  has acknowledged end of word recognition.

There is a transition  $((U, \sigma), \langle \sigma, A \rightarrow 1 \rangle, (W, \sigma))$  for any state  $(U, \sigma)$ , indicating to  $\mathcal{A}_1$  that the next letter to recognize from its current universal state is  $\sigma$ . In state  $W$ ,  $\mathcal{A}$  is waiting for an universal move from  $\mathcal{A}_1$ . Then from  $W$ ,  $\mathcal{A}$  can receive the information that  $\mathcal{A}_1$  has moved to an universal state, which is symbolized by a pair of transitions  $(W, \langle \sigma, U, 1 \rightarrow A \rangle, dU)$  and  $(dU, \langle again, A \rightarrow 2 \rangle, U)$ .

There is a transition  $(W, \langle \sigma, q_i, 1 \rightarrow A \rangle, dq_i)$  for every existential state  $q_i \in Q_{\exists}$ , followed by a transition  $(dq_i, \langle \sigma, q_i, A \rightarrow 2 \rangle, Eq_i)$ , indicating to  $\mathcal{A}_2$  that the system has moved to recognition of a letter from an existential state  $q_i$ .

There is a transition  $(Eq_i, \langle \sigma, 2 \rightarrow A \rangle, (Eq_i, \sigma))$  from every state  $Eq_i$  with  $q_i \in Q_{\exists}$  and every  $\sigma \in \Sigma$  to indicate that the next letter to recognize is  $\sigma$ . Then, there is a transition  $((Eq_i, \sigma), \langle \sigma, q_j, 2 \rightarrow A \rangle, (Wq_j, \sigma))$  for every existential move  $(q_i, \sigma, q_j) \in \delta$ . From every state  $(Wq_j, \sigma)$ , there is a transition of the form  $((Wq_j, \sigma), \langle \sigma, q_j, A \rightarrow 1 \rangle, (dq_j, \sigma))$  to inform  $\mathcal{A}_1$  of  $\mathcal{A}_2$ 's move. Then, from  $(Dq_j, \sigma)$  if  $q_j \in Q_{\exists}$ , there is a transition of the form  $((Dq_j, \sigma), \langle again, A \rightarrow 1 \rangle, Eq_j)$  and if  $q_j \in Q_{\forall}$ , a transition of the form  $((dq_j, \sigma), \langle again, A \rightarrow 1 \rangle, U)$ , indicating to  $\mathcal{A}_1$  that the simulation of the current transition recognizing a letter is complete, and from which state the rest of the simulation will resume.

Let us now detail the construction of  $\mathcal{A}_2$ . A description of all its transition is given in Figure 4. This automaton has one universal state  $U$ , a state  $W$ , states of the form  $(U, \sigma)$ , a pair of states  $Eq_i$  and  $Wq_i$  and a state  $(Eq_i, \sigma)$  for every  $\sigma \in \Sigma$  and every  $q_i \in Q_{\exists}$ . Last,  $\mathcal{A}_1$  has two states  $End_1$  and  $End_2$ .

There is a transition  $(U, \langle \sigma, 2 \rightarrow A \rangle, (U, \sigma))$  from  $U$  for every  $\sigma \in \Sigma$ , symbolizing the choice of letter  $\sigma$  as the next letter to recognize when the system simulates an universal state. Note that  $\mathcal{A}_2$  needs not know which universal state is currently simulated. Then, there is also a transition  $((U, \sigma), \langle again, U \rangle, U)$  returning to  $U$  symbolizing the end of a transition of the alternating

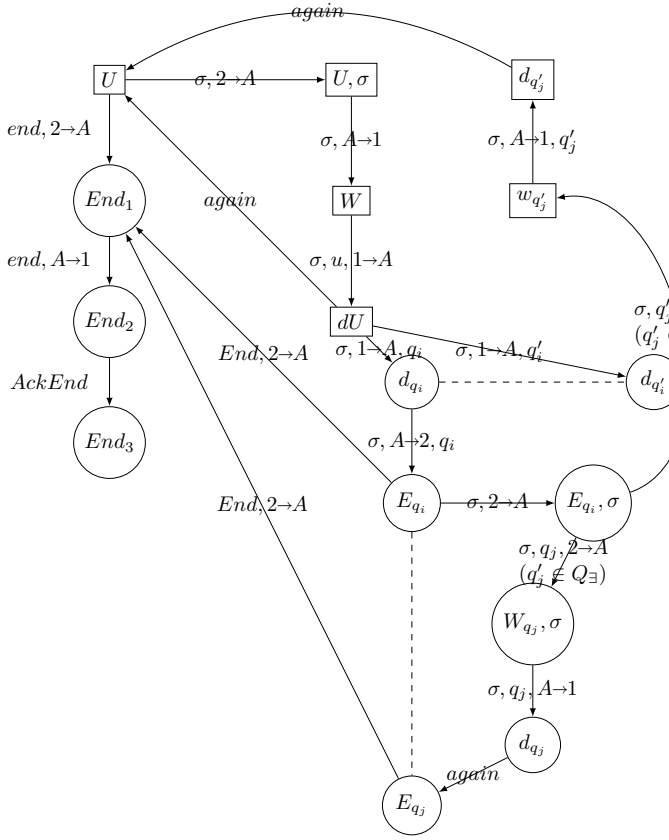


Fig. 3: Automaton  $\mathcal{A}$  in the proof of theorem 6.

automata that returns to an universal state (hence owned by  $\mathcal{A}_2$ ). From every state  $(U, \sigma)$  there is a transition  $((U, \sigma), \text{again}, U)$  and a transition  $((U, \sigma), \langle \sigma, q_i, A \rightarrow 2 \rangle, E_{q_i})$  for every existential state  $q_i$  that has an universal predecessor  $q$  with  $(q, \sigma, q_i) \in \delta$ . From a state  $E_{q_i}$  and for every  $\sigma \in \Sigma$ , there is a transition  $(E_{q_i}, \langle \sigma, 2 \rightarrow A \rangle, (E_{q_i}, \sigma))$  symbolizing the choice to recognize  $\sigma$  as the next letter. Then, from every state  $(E_{q_i}, \sigma)$  for every transition of the form  $(q_i, \sigma, q_j) \in \delta$  where  $q_j$  is existential, there is a transition  $((E_{q_i}, \sigma), \langle \sigma, q_j, 2 \rightarrow A \rangle, W_{q_j})$ . For every transition of the form  $(q_i, \sigma, q_j) \in \delta$  where  $q_j$  is universal, there is a transition  $((E_{q_i}, \sigma), \langle \sigma, q_j, 2 \rightarrow A \rangle, W)$ . Last, transitions  $((W_{q_j}, \sigma), \text{again}, E_{q_j})$  and  $(W, \text{again}, U)$  complete simulation of recognition of the current letter.

Last,  $\mathcal{A}_2$  has a transition  $(U, \langle \text{end}, 2 \rightarrow A \rangle, \text{End}_1)$ , a transition  $(E_{q_i}, \langle \text{end}, 2 \rightarrow A \rangle, \text{End}_1)$  for every existential state  $q_i \in Q_{\exists}$  and a transition  $(\text{end}_1, \text{ackend}, \text{End}_2)$ , symbolizing the decision to end recognition of a word.

Let us detail the construction of  $\mathcal{A}_1$ . The general shape of this automaton is described in Figure 5. This automaton has two states of the form  $U_{q_i}, (U_{q_i}, \sigma)$  per universal state and for each  $\sigma \in \Sigma$ . Similarly  $\mathcal{A}_1$  has a state  $E_{q_i}, (E_{q_i}, \sigma)$  per existential state and for each  $\sigma \in \Sigma$ . From state  $U_{q_i}$  there is a transition  $(U_{q_i}, \langle \sigma, A \rightarrow 1 \rangle, (U_{q_i}, \sigma))$  to acknowledge the decision to recognize  $\sigma$ .

From state  $(U_{q_i}, \sigma)$  there exists two types of transitions. For every universal state  $q_j$  such that  $(q_i, \sigma, q_j) \in \delta$ ,

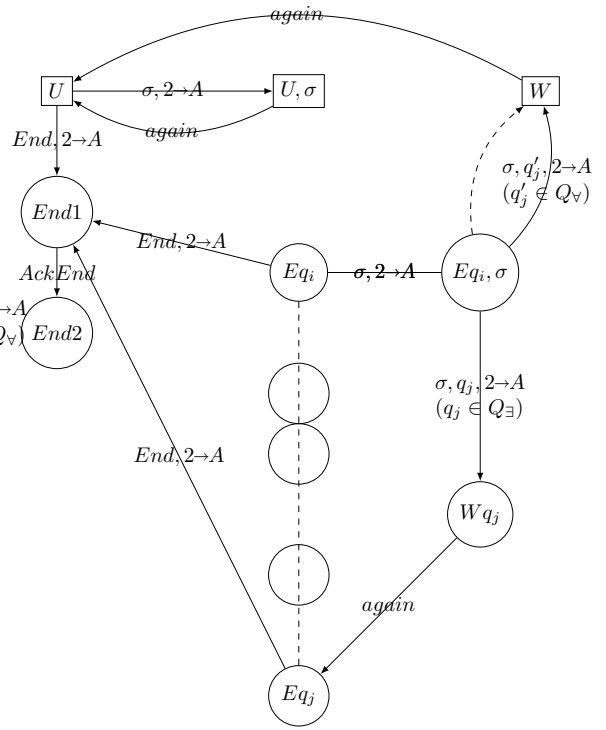


Fig. 4: Automaton  $\mathcal{A}_2$  in the proof of theorem 6, simulating existential moves .

there is a transition  $((U_{q_i}, \sigma), \langle \sigma, U, 1 \rightarrow A \rangle, U_{q_j})$ , symbolizing a move to universal state  $q_j$ . For every existential state  $q_j$  such that  $(q_i, \sigma, q_j) \in \delta$ , there is a transition  $((U_{q_i}, \sigma), \langle \sigma, q_j, 1 \rightarrow A \rangle, E_{q_j})$ .

Similarly, from a state  $E_{q_i}$ , there exists a transition  $(E_{q_i}, \langle \sigma, A \rightarrow 1 \rangle, (E_{q_i}, \sigma))$  indicating to  $\mathcal{A}_1$  the letter chosen by  $\mathcal{A}_2$ . From state  $(E_{q_i}, \sigma)$ , there is a transition  $((E_{q_i}, \sigma), \langle \sigma, q_j, A \rightarrow 1 \rangle, E_{q_j})$  for every existential state  $q_j$  such that  $(q_i, \sigma, q_j) \in \delta$ . There is also a transition  $((E_{q_i}, \sigma), \langle \sigma, U, 1 \rightarrow A \rangle, U_{q_j})$  for every universal state  $q_j$  such that  $(q_i, \sigma, q_j) \in \delta$ . Notice that the universal state reached is not detailed when  $\mathcal{A}_1$  sends the confirmation of a move to  $\mathcal{A}$ .

The remaining transitions are transitions of the form  $(E_{q_i}, \langle \text{end}, A \rightarrow 1 \rangle, S)$  and  $(U_{q_i}, \langle \text{end}, A \rightarrow 1 \rangle, \text{Sec})$  for every accepting state  $q_i \in F$ . We also create transitions of the form  $(E_{q_i}, \langle \text{end}, A \rightarrow 1 \rangle, \overline{\text{Sec}})$  and  $(U_{q_i}, \langle \text{end}, A \rightarrow 1 \rangle, \overline{\text{Sec}})$  for states that are not accepting. Reaching  $\overline{\text{Sec}}$  indicates the failure to recognize a word chosen by  $\mathcal{A}_1$  along a path in which universal moves were played by  $\mathcal{A}_1$  and existential moves by  $\mathcal{A}_2$ .

We define a agent  $v_1$ 's secret  $S_1$  as the automaton that recognizes all words that allow  $\mathcal{A}_1$  to reach state  $\text{Sec}$ .

Now, we can prove that if a word  $w$  is accepted by  $\mathcal{A}_{\text{alt}}$  then the strategy in which  $\mathcal{A}_2$  chooses letter  $w_i$  at its  $i^{\text{th}}$  passage through a letter choice state ( $U$  or  $E_{q_i}$ ), existential transitions appearing in the accepting run of  $\mathcal{A}_{\text{alt}}$ , and then transition  $\langle \text{end}, 2 \rightarrow A \rangle$  at the  $i + 1^{\text{th}}$  choice, is a strategy to force

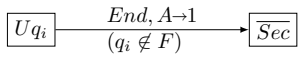
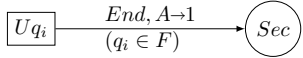
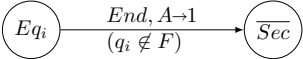
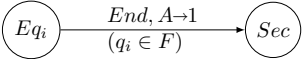
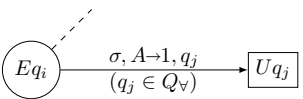
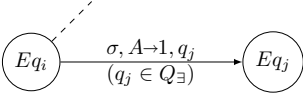
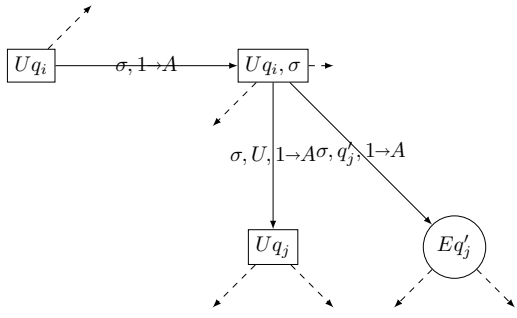


Fig. 5: Automaton  $\mathcal{A}_1$  in the proof of theorem 6, simulating Universal moves .

$\mathcal{A}_1$  to reach the secret state. Conversely, one can associate to every run of  $\mathcal{A}, \mathcal{A}_1, \mathcal{A}_2$ , a word  $w$  that is read, and a path in some run that is used to recognize  $w$ . If  $\mathcal{A}_2$  has a strategy to force  $\mathcal{A}_1$  secret leakage, then all path following this strategy lead to a winning configuration. As a consequence, there is a choice of existential moves such that all states simulated along a run of the alternating automaton with these existential moves end in accepting state. Hence,  $\mathcal{L}(\mathcal{A}_{alt})$  is empty iff the system composed of  $\mathcal{A}, \mathcal{A}_1, \mathcal{A}_2$  is opaque. Now, the system built to simulate  $\mathcal{A}_{alt}$  is of polynomial size in  $|\mathcal{A}_{alt}|$ , so there is a polynomial size reduction from the emptiness problem for alternating automata to the active opacity question, and active opacity is EXPTIME-complete.  $\square$

**Proof:** One can first recall that for the Hamming and Levenshtein distances, we have  $d(u, v) = d(u^{-1}, v^{-1})$ , where  $u^{-1}$  is the mirror of  $u$ . Similarly, we have  $d^K(u, R) = d(u_{[1, K]}^{-1}, R^{-1})$ . Let  $\mathcal{G}_R = (\Sigma, Q, q_0, \delta, F)$  be the automaton recognizing language  $R$ . We can build an automaton  $\mathcal{C}_k$  that recognizes words of length at least  $K$ , which suffixes of length  $K$  are at hamming distance at most  $k$  of suffixes of length  $K$  of words in  $R$ . We define  $\mathcal{C}_k = (\Sigma, Q_k^{suf}, q_{0, k}^{suf}, \delta_k^{suf}, F_k^{suf})$ . This automaton can be computed as follows : first build  $\mathcal{G}_R^{-1}$ , the automaton that recognizes mirrors of suffixes of  $R$ . This can be easily done by setting as initial states the final states of  $R$ , and then reversing the transition relation. Then by adding a  $K$ -bounded counter to states of  $\mathcal{G}_R^{-1}$ , and setting as accepting states states of the form  $(q, K)$ , we obtain an automaton  $\mathcal{B}^{-1}$  that recognizes mirrors of suffixes of  $R$  of length  $K$ . Then, for every  $k \in [0..K]$ , we can compute  $\mathcal{B}^k$ , the automaton that recognizes mirrors of words of length  $K$  that are at distance  $k$  from words in  $\mathcal{B}^{-1}$ , by adding another counter to states that counts substitutions, and which final states are of the form  $(q, K, k)$ . Then we can build (by sequential composition of automata for instance) the automaton  $\mathcal{C}_k$  that reads any word in  $\Sigma^*$  and then recognizes a word in  $(\mathcal{B}^k)^{-1}$ .  $\square$