Timed Non-interference Under Partial Observability and Bounded Memory

Anthony Spriet, Didier Lime, Olivier H. Roux

Timed non-interference under Partial Observability and Bounded Memory*

Anthony Spriet, Didier Lime, Olivier H. Roux

Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes

Abstract. We investigate a timed non-interference property for security systems modeled as timed automata, in which a low-security level user should not be able to deduce the occurrence of some high-security level actions. We assume an attack model in which the malicious (low-level) user has the ability to partially observe and memorize the set of runs of the timed automaton modeling the system.
We first formalize a non-interference property that ensures the system security under such an attack model and we then prove the undecidability of that property when the attacker can have an arbitrarily big memory, i. e., when they are able to memorize sequences of previous observations, with time-stamps, of any length. We next assume bounded memory for the attacker and show that the property can then be decided in PSPACE for a subclass of timed automata ensuring finite duration between distinct observations.

Keywords: Non-interference, Timed Automata, Partial observability.

1 Introduction

Information leakage is a type of software vulnerability in which information is unintentionally accessible to unauthorized party, potentially aiding malicious attackers or leading to malfunction of systems. Various information flow properties have been defined in the literature such as anonymity, non-interference, secrecy, privacy, opacity, non-deducibility. These properties intersect with each other and formal comparisons have been made as in [BKMR08] between opacity, anonymity, (trace-based) non-interference and non-deducibility. We focus here on non-interference properties that could thus be written as opacity.

A system is said to be non-interferent [GM84] if the information available on the interface of the system is not sufficient to infer some classified internal information. That is to say, the information available in the low-level channel (not classified, observable by most users) does not contain any clues about the information only available on the high-level channel (classified information, only accessible to some or no users). Another way to put it is that a sequence of low-level inputs will always produce the same low-level outputs, regardless of what happens on high-level inputs. It has been shown that additional information that

* This work has been partially funded by ANR project ProMiS ANR-19-CE25-0015.

is not directly provided by the system can be used to make inferences on what happens on the high-level channel. In particular, a system may be non-interferent when considering sequences of low-level inputs and outputs but interferent when the dates of each observation of the different inputs and outputs are measured and memorized by a malicious observer [Koc96,FS00,BB07,KPJJ13].
In [BT03], Barbuti *et al* proposed two notions of timed non-interference in systems modeled by timed automata. One is a trace-based notion in which the attacker can observe the consecutive actions performed by the system and tries to guess if a hidden action has occurred. The trace-based approach has since been refined by taking simulation properties into account or by providing control synthesis algorithms [GMR07,BCLR15,GSB18]. The other approach is state-based: the attacker can observe the state of the system directly but does not see any action. If the observed state is not reachable using only low-level actions, the attacker is able to infer the use of a high-level action. The state-based approach has also been refined since by being extended to parametric timed automata [AK20]. Other notions of non-interference have also been proposed, a common framework involves ensuring that no information is leaked through the execution time of a timed automaton with final locations [ALMS22,WZ18,AETYM21].

Contributions We introduce a new state-based non interference property (POS-NNI) and discuss its relevance when it comes to modeling realistic attackers. We prove that the POS-NNI verification problem is undecidable when the attacker memory is infinite. We provide a subclass of timed automata for which this property is decidable for attackers with finite memory of any length and we prove that its verification is PSPACE-complete.

In section 2, we recall definitions used to formalize the timed non-interference property and present some basic constructions over timed automata used to model the systems and the information flow. In section 3, we introduce the POS-NNI property and we prove its undecidability when the attacker memory is infinite. Lastly, in section 4, we provide a subclass of timed automata for attackers with finite memory of any length for which the POS-NNI verification problem is PSPACE-complete.

2 Definitions

2.1 Timed automata

Timed automata [AD94] are one of the many formalism to model real-time systems, they consist in adding real-valued clocks to finite automata in order to capture timed behaviors accurately. We consider here *security automata*, in which the set of actions is partitioned into two subsets, modeling two users with different access privileges to information or commands.

Definition 1 (Clocks and valuations). *A clocks is a real-valued variable that evolves at rate* 1 *w.r.t. time. We define a set* $\mathbb{X} = \{x_1, x_2, ..., x_H\}$ *of clocks.*

A clock valuation is a function $\nu : \mathbb{X} \to \mathbb{R}_{\geq 0}$. *We denote by* $\mathbb{R}^{\mathbb{X}}_{\geq 0}$ *the set of valuations.*

Given $d \in \mathbb{R}_{\geq 0}, \nu + d$ *denotes the valuation s. t.* $(\nu + d)(x) = \nu(x) + d$. *Given* $R \subseteq \mathbb{X}$, *we define the reset of a valuation* ν, *denoted by* $[\nu]_R$ *s. t.* $[\nu]_R(x) = 0$ *if* $x \in R$, *and* $[\nu]_R(x) = \nu(x)$ *otherwise.*

Definition 2 (Clock Constraint). *A clock constraint* g *is a constraint over* $\mathbb{X}$ *defined by a conjunction of inequalities of the form* $x \bowtie d$ *with* $d \in \mathbb{N}$ *and* $\bowtie \in \{\leq, <, >, \geq\}$.

Given g, *we write* $\nu \models g$ *iff each inequality in* g *evaluates to true when each clock* x *is replaced with its value* $\nu(x)$.

Definition 3 (ϵ-Timed Automatons (ϵ-TA)). *An* ϵ*-TA* $\mathcal{A}$ *is a tuple* $\mathcal{A} = (\Sigma \cup \{\epsilon\}, L, l_0, \mathbb{X}, I, E)$ *where:* Σ *is a finite set of actions,* ϵ *is the unique "silent action", L is a finite set of locations,* l_0 *is the initial location,* $\mathbb{X}$ *is a finite set of clocks, I assigns to every* $l \in L$ *a clock constraint* $I(l)$ *called* invariant, *E is a finite set of edges* $e = (l, g, a, R, l')$ *where* $l, l' \in L$ *are respectively the source and target locations,* $a \in \Sigma \cup \{\epsilon\}, R \subseteq \mathbb{X}$ *is a set of clocks to be reset, and* g *is a clock constraint.*

Definition 4 (Semantics of timed automata). *Given an* ϵ*-TA* $\mathcal{A} = (\Sigma \cup \{\epsilon\}, L, l_0, \mathbb{X}, I, E)$, *the semantics of* $\mathcal{A}$ *is given by the timed transition system* $(S, s_0, \to)$, *with:*

- $S = \{(l, \nu) \in L \times \mathbb{R}^{\mathbb{X}}_{\geq 0} \mid \nu \models I(l)\}$
- $s_0 = (l_0, \boldsymbol{0})$ *with* $\boldsymbol{0}$ *the valuation of clocks with all clocks equal to 0.*
- $\to$ *consists of the discrete and continuous delay transitions:*
  - *discrete transitions:* $(l, \nu) \xrightarrow{a} (l', \nu')$ *with* $(l, \nu), (l', \nu') \in S$ *and there exists* $e = (l, g, a, R, l') \in E$ *such that* $\nu' = [\nu]_R$ *and* $\nu \models g$
  - *delay transitions:* $(l, \nu) \xrightarrow{d} (l, \nu + d)$ *with* $d \in \mathbb{R}_{\geq 0}$ *if* $\forall d' \in [0, d], (l, \nu + d') \in S$

Definition 5 (Run). *Let* $\mathcal{A} = (\Sigma \cup \{\epsilon\}, L, l_0, \mathbb{X}, I, E)$ *be a* ϵ*-TA and* $(S, s_0, \to)$ *its semantics. A (finite) run* ρ *of* $\mathcal{A}$ *is a finite sequence* $\rho = s_0 e_0 s_1 e_1 \cdots s_n$ *with* $\forall i, s_i \in S$ *and* $(s_i, e_i, s_{i+1}) \in \to$, *which we will also write* $\rho = s_0 \xrightarrow{e_0} s_1 \xrightarrow{e_1} \cdots \xrightarrow{e_{n-1}} s_n$.

A run ρ *can always be put in the form:*$\rho = s_0 \xrightarrow{d_1} s_1 \xrightarrow{a_2}\xrightarrow{d_2} s_2 \xrightarrow{a_2} \ldots s_{n-1} \xrightarrow{a_n}\xrightarrow{d_n} s_n$ *where delays and actions strictly alternate. we have omitted some state names for brevity.*

We write $\Psi(\mathcal{A})$ the set of all runs of $\mathcal{A}$. We say that a state s is reachable in $\mathcal{A}$ if there exists a run $\rho \in \Psi(\mathcal{A})$ s.t. $s \in \rho$. We write $Q^{\mathcal{A}}$ set of all the reachable states in $\mathcal{A}$

2.2 Timed language

Sequences of actions generated by a TA, together with their dates, describe the behavior of the automaton.

Definition 6 (Timed word and trace). *A timed word over the alphabet Σ is a finite sequence $w = (a_0, t_0)(a_1, t_1)...(a_n, t_n)$ so that $\forall i \geq 0, a_i \in \Sigma, t_i \in \mathbb{R}_{\geq 0}$.*

Given a run $\rho = s_0 \xrightarrow{d_1} s_1 \xrightarrow{a_2}\xrightarrow{d_2} s_2 \xrightarrow{a_2} ...s_{n-1} \xrightarrow{a_n}\xrightarrow{d_n} s_n$, its trace *is the timed word $(a_1, d_1)(a_2, d_1 + d_2) \cdots (a_n, \sum_{i=1}^{n-1} d_i)$*

We denote $\mathcal{U}(\Sigma)$ the set of all timed words over the alphabet Σ

Definition 7 (Timed Language). *For a given ϵ-TA $\mathcal{A}$, its timed language $\mathcal{L}(\mathcal{A})$ is the set of the traces of all its runs.*

2.3 Security Timed Automata

Definition 8 (Security Timed Automaton). *A Security Timed Automaton $\mathcal{A} = (\Sigma \cup \{\epsilon\}, L, l_0, \mathbb{X}, I, E)$ is an ϵ-TA whose set of (visible) actions Σ is partitioned in two subsets Σ_{low} and Σ_{high} such that $\Sigma_{\mathsf{low}} \cap \Sigma_{\mathsf{high}} = \emptyset$ and $\Sigma_{\mathsf{low}} \cup \Sigma_{\mathsf{high}} = \Sigma$.*

In order to compare high-level behaviors and low-level behaviors by removing all the high-level information we define restricted timed automata as follows. Fig. 1 shows a security TA $\mathcal{A}$ and its restriction $\mathcal{A}_{\setminus \Sigma_{\mathsf{high}}}$.

Definition 9 (Restricted timed Automata). *Let $\mathcal{A} = (\Sigma, L, l_0, \mathbb{X}, I, E)$ be a TA and $\Gamma \subseteq \Sigma$. We define the Γ-restriction TA $\mathcal{A}_{\setminus \Gamma} = (\Sigma \setminus \Gamma, L, l_0, \mathbb{X}, I, E_{\setminus \Gamma})$ where $(l, g, a, R, l') \in E_{\setminus \Gamma}$ iff $a \in (\Sigma \setminus \Gamma)$ and $(l, g, a, R, l') \in E$.*

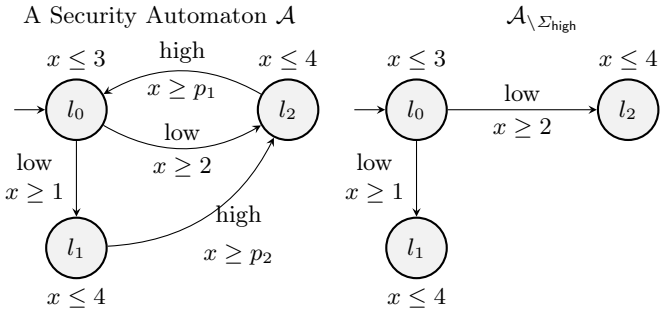


Fig. 1: A security TA $\mathcal{A}$ and $\mathcal{A}_{\setminus \Sigma_{\mathsf{high}}}$

3 State-based non-interference properties

3.1 Already existing state-based non-interference properties

To our knowledge, the first proposal of a state-based non-interference property for timed automata comes from Barbuti and Tesei, [BT03] and has later been refined by Gardey *et al* [GMR07]. To define a state-based property of non-interference we can picture a memoryless attacker who is able to fully observe the system states (both location and clocks valuation) and tries to guess if a high-level action has occurred. Because they are memoryless, they have to make their guess based on a single observation of the system, which intuitively corresponds to the property called *St-NNI* (State non-interference) by Gardey *et al* [GMR07]: a security automaton $\mathcal{A}$ satisfies St-NNI iff $Q^{\mathcal{A}} = Q^{\mathcal{A}_{\setminus \Sigma_{\mathsf{high}}}}$

3.2 A more realistic property

As we described St-NNI can be seen as a model for a really powerful attacker, in the sense that they can fully observe the system states, but with the minimal deductive abilities due to their memorylessness. If we were to give memory to the attacker, in the sense that the attacker would have access to all their previous observations, while keeping their ability to observe the state directly: we would end up with an unrealistically high standard for security. We therefore propose an attacker that has memory and is able to continuously observe the system but is only able to observe part of the information, in the sense that they cannot observe systems clocks and that some locations are indistinguishable to them. Formally, we do so by attributing a value to each location, the attacker being able to observe this value and associate it with a time-stamp.

Definition 10 (Observation functions). *Let Ω be a finite set with $|\Omega| \leq |L|$. We call the elements of Ω* observations. *An* observation function *is a function $\mathcal{O} : L \to \Omega$. If $\mathcal{O}(l) = \mathcal{O}(l')$ (resp: $\mathcal{O}(l) \neq \mathcal{O}(l')$) we say that l and l' are indistinguishable (resp: distinguishable).*

We now use the observation to define the runs from the attacker point of view.

Definition 11 (Observed runs and events). *Given a run $\rho = (l_0, \mathbf{0}) \xrightarrow{d_1} (l_1, \nu_1) \xrightarrow{a_2}\xrightarrow{d_2} (l_2, \nu_2) \dots (l_{n-1}, \nu_{n-1}) \xrightarrow{a_n}\xrightarrow{d_n} (l_n, \nu_n)$ and an observation function $\mathcal{O}$. We define the projecting operator $\mathcal{O}$ as:*

$\mathcal{O}(\rho) = (\mathcal{O}(l_{i_0}), 0)(\mathcal{O}(l_{i_1}), \sum_{k=1}^{i_1-1} d_k) \dots (\mathcal{O}(l_{i_{m-1}}), \sum_{k=1}^{i_m-1} d_k)(\mathcal{O}(l_{i_m}), \sum_{k=1}^{i_m} d_k)$, *where the sequence $(i_0, i_1, ..., i_m)$ is defined as follows:*

- $i_0 = 0$
- *for $1 \leq k < m$, i_k is the smallest index $j > i_{k-1}$ such that $\mathcal{O}(l_{i_{k-1}}) \neq \mathcal{O}(l_j)$*
- $i_m = n$

We call $\mathcal{O}(\rho)$ an observed run of $\mathcal{A}$ (with respect to $\mathcal{O}$). We note $\mathcal{O}(\mathcal{A})$ the set of observed runs of $\mathcal{A}$.

*When the observed run $\mathcal{O}(\rho)$ of a run ρ contains an element (a, t) we say that an **observed event** occured at time t in the run.*

To put it simply, because attackers are able to continuously observe the system, they are therefore able to memorize the absolute date of the events (when a discrete transitions between distinguishable locations occurs). The last element of the observed run is there to model the time elapsed since the last event, by construction $\mathcal{O}(l_{i_m}) = \mathcal{O}(l_{i_{m-1}})$. If an observed run is possible both with and without high-level actions, an attacker cannot deduce anything from this observed run. Therefore, ensuring that no observed run is possible only for high-level users guarantees security.

Example 1 (A run and its associated observed run). Given a run $\rho = (l_0, \mathbf{0}) \xrightarrow{d_1,a_1} (l_1, \nu_1) \xrightarrow{d_2,a_2} (l_2, \nu_2) \xrightarrow{d_3,a_3} (l_3, \nu_3) \xrightarrow{d_4,a_4} (l_4, \nu_4) \xrightarrow{d_5,a_5} (l_5, \nu_5) \xrightarrow{d_6} (l_5, \nu_6)$ with $\mathcal{O}(l_0) = \mathcal{O}(l_1) = \mathcal{O}(l_3) = \mathcal{O}(l_4) = o_1, \mathcal{O}(l_2) = o_2$ and $\mathcal{O}(l_5) = o_3$. We have $\mathcal{O}(\rho) = (o_1, 0)(o_2, d_1 + d_2)(o_1, d_1 + d_2 + d_3)(o_3, \sum_{k=1}^{5} d_k)(o_3, \sum_{k=1}^{6} d_k)$

We say that observed events occurred at time $0, d_1, \sum_{k=1}^{2} d_k, \sum_{k=1}^{4} d_k$ and $\sum_{k=1}^{5} d_k$. No event occurred at $\sum_{k=1}^{6} d_k$, the last term is to account for the time elapsed since the last event.

Definition 12 (Partial Observability State-based non-interference (POS-NNI)). *A security automaton $\mathcal{A}$ satisfies POS-NNI with respect to the observation function $\mathcal{O}$ if and only if $\mathcal{O}(\mathcal{A}) = \mathcal{O}(\mathcal{A}_{\setminus \Sigma_{\mathsf{high}}})$*

We now show that verifying POS-NNI is undecidable by reducing to it the problem of timed language universality, which is known to be undecidable [AD94]. We do so in the following way: we first define a transformation $\mathcal{T}$ on TAs that allows us to define an observation function so that there is a one to one correspondence between the observed run set of $\mathcal{T}(\mathcal{A})$ and the timed language of $\mathcal{A}$. Then for any TA $\mathcal{A}$, we define an augmented security automaton that is equal to $\mathcal{T}(\mathcal{A})$ when restricted to low-level actions (therefore having the same language) but has the universal language when not restricted. The augmented automaton therefore satisfies POS-NNI with respect to our observation function if and only if the timed language of $\mathcal{A}$ is the universal language.

Definition 13 (Transformation $\mathcal{T}$). *Given a TA $\mathcal{A} = (\Sigma, L, l_0, \mathbb{X}, I, E)$, we define $\mathcal{T}(\mathcal{A}) = (\Sigma, L_{\mathcal{T}(\mathcal{A})}, l_{0_{\mathcal{T}(\mathcal{A})}}, \mathbb{X}, I_{\mathcal{T}(\mathcal{A})}, E_{\mathcal{T}(\mathcal{A})})$ with:*

- $l_{0_{\mathcal{T}}} = (l_0, \epsilon, 0)$
- $L_{\mathcal{T}} = \{L\} \times \{\Sigma \cup \{\epsilon\}\} \times \{0; 1\}$
- $\forall l \in L, \forall a \in \Sigma, \forall i \in \{0, 1\}, (l, a, i) \in L_{\mathcal{T}(\mathcal{A})} \wedge I_{\mathcal{T}(\mathcal{A})}((l, a, i)) = I(l)$
- $E_{\mathcal{T}(\mathcal{A})}$ *is the smallest set such that* $\forall e = (l, g, a, R, l') \in E$:
  - $\forall b \in (\Sigma \cup \{\epsilon\}), ((l, b, 0), g, a, R, (l', a, 1)) \in E_{\mathcal{T}(\mathcal{A})}$
  - $\forall b \in (\Sigma \cup \{\epsilon\}), ((l, b, 1), g, a, R, (l', a, 0)) \in E_{\mathcal{T}(\mathcal{A})}$

The idea of transformation $\mathcal{T}$ is to preserve the language while augmenting the set of locations by splitting it into triplets (l, σ, b). When using a discrete transition labeled σ, the automaton $\mathcal{T}(\mathcal{A})$ necessarily reaches a location labeled σ as well. The parameter b is a boolean value ensuring the absence of self-loop transitions in $\mathcal{T}(\mathcal{A})$, any discrete transition available from a locality marked with

0 would reach a locality marked with 1 and vice-versa. This way, we are able to define an observation function on $\mathcal{T}(\mathcal{A})$ so that every transition of a run is associated with an observed event and every observation is associated with a unique transition label.

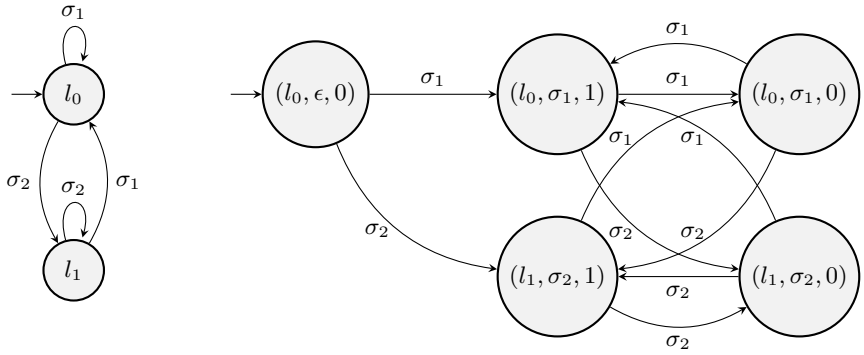


Fig. 2: A timed automaton and its transformation with $\mathcal{T}$ (clock constraints are omitted)

Proposition 1. *Given a TA $\mathcal{A}$ defined over the alphabet Σ and its transformation $\mathcal{T}(\mathcal{A})$, the following properties holds:*

i) $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{T}(\mathcal{A}))$
ii) $\forall((l_s, \sigma_s, j_s), g, a, R, (l_t, \sigma_t, j_t)) \in E_{\mathcal{T}(\mathcal{A})}, a = \sigma_t$
iii) $\forall((l_s, \sigma_s, j_s), g, a, R, (l_t, \sigma_t, j_t)), \in E_{\mathcal{T}(\mathcal{A})}, j_s \neq j_t$
iv) There is no self-loop edge in $E_{\mathcal{T}(\mathcal{A})}$

Proof. i) First we recall that by definition of $\mathcal{T}$, $\forall l' = (l, a, i) \in L_{\mathcal{T}(\mathcal{A})}, I_{\mathcal{T}(\mathcal{A})}(l') = I(l)$. Then we notice that $\forall e = (l_s, g, a, R, l_t) \in E, \exists e' = ((l_s, \sigma_s, j_s), g, a, R, (l_t, a, j_t)) \in E_{\mathcal{T}(\mathcal{A})}$ for any pair (σ_s, j_s) and for some j_t, which implies $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{T}(\mathcal{A}))$. Conversely, $\forall e = ((l_s, \sigma_s, j_s), g, a, R, (l_t, a, j_t)) \in E_{\mathcal{T}(\mathcal{A})}, \exists e' = (l_s, g, a, R, l_t) \in E$ which implies $\mathcal{L}(\mathcal{T}(\mathcal{A})) \subseteq \mathcal{L}(\mathcal{A})$.

ii) By definition of $\mathcal{T}$, there is no edge of the form $((l_s, \sigma_s, j_s), g, a, R, (l_t, \sigma_t, j_t))$ with $a \neq \sigma_t$ in the set $E_{\mathcal{T}(\mathcal{A})}$. In other words, each location of $\mathcal{T}(\mathcal{A})$ is associated with a unique letter of Σ and can only be reached using an edge labeled with this letter.

iii) By definition of $E_{\mathcal{T}(\mathcal{A})}$, there is no edge of the form $((l_s, \sigma_s, 0), g, \sigma_t, R, (l_t, \sigma_t, 0))$ or $((l_s, \sigma_s, 1), g, \sigma_t, R, (l_t, \sigma_t, 1))$.

iv) Is a direct consequence of (iii). □

Theorem 1. *The verification problem associated with POS-NNI is undecidable.*

Proof (sketch). Given a TA $\mathcal{A}$ defined over the alphabet Σ, we define a trivially universal automaton $\mathcal{U} = (\Sigma, l_u, l_u, \emptyset, I(l_u) = true, (l_u, \emptyset, \Sigma, \emptyset, l_u))$. We then

define a security TA $\mathcal{A}^*$ that is the union of $\mathcal{T}(\mathcal{A})$ and $\mathcal{T}(\mathcal{U})$ with only one high-level edge going from the initial location of $\mathcal{T}(\mathcal{A})$ to the initial location of $\mathcal{T}(\mathcal{U})$. To this TA we associate an observation function so that all locations are distinguishable from every other except the two initial locations of $\mathcal{T}(\mathcal{A})$ and $\mathcal{T}(\mathcal{U})$. This way, the high-level location does not appear in the set of observed runs (no event occurs when using it). By choosing the initial location of $\mathcal{T}(\mathcal{A})$ to be the initial location of $\mathcal{A}^*$ we show that $\mathcal{A}^*$ has the language of $\mathcal{U}$ when not restricted but the language of $\mathcal{A}$ when restricted to low-level actions. Using the absence of self-loops in $\mathcal{T}(\mathcal{A})$ and $\mathcal{T}(\mathcal{U})$ (Proposition 1), we show that any low-level action (actions of Σ) corresponds to an observed event, it allows us to provide a bijection between the set of projected runs and the language, showing the reduction of POS-NNI to language universality. □

4 A decidable sub-problem

We now consider that the attacker has a bounded memory, which seems not unreasonable.

Such an attacker could choose between various policies to store and discard information. As a first approach we choose to model an attacker who uses a first-in/first-out policy, only keeping the most recent information and discarding the oldest one each time new information is available to them. This can be modeled by taking the suffixes of observed runs (and relabeling with new time-stamps).

Definition 14 (Observed runs with memory k).
Given an observation function $\mathcal{O}$, a run ρ and its observed run of $\mathcal{O}(\rho) = (o_1, t_1 = 0)(o_2, t_2)\dots(o_n, t_n)(o_n, t_{n+1})$, the observed run of memory size k associated with ρ, noted $\mathcal{O}_k(\rho)$, is the sequence defined by:

if $n \leq k$**:** $\mathcal{O}_k(\rho) = \mathcal{O}(\rho)$
if $n > k$**:** $\mathcal{O}_k(\rho) = (o_{n-k}, 0)(o_{n-k+1}, t_{n-k+1} - t_{n-k})\dots(o_n, t_n - t_{n-k})(o_n, t_{n+1} - t_{n-k})$

We note $\mathcal{O}_k(\mathcal{A})$ the set of observed runs with memory k for every run of $\mathcal{A}$

Similar to observed runs (with infinite memory), observed runs with memory k contain $k+1$ elements as one is added to keep track of the time elapsed since the last event.

Example 2 (An observed run and its associated observed run with memory 3). Let us go back to example 1, given a run ρ and its associated projected run: $\mathcal{O}(\rho) = (o_1, 0)(o_2, d_1 + d_2)(o_1, d_1 + d_2 + d_3)(o_3, \sum_{k=1}^{5} d_k)(o_3, \sum_{k=1}^{6} d_k)$ We have $\mathcal{O}_3(\rho) = (o_2, 0)(o_1, d_3)(o_3, \sum_{k=3}^{5} d_k)(o_3, \sum_{k=3}^{6} d_k)$.

This gives rise to an updated version of the non-interference property adapted to attackers with finite memory.

Definition 15 (Partial Observability State-based non-interference with finite memory of size k (k-POS-NNI)). *A security automaton $\mathcal{A}$ satisfies k-POS-NNI with respect to the observation $\mathcal{O}$ if and only if $\mathcal{O}_k(\mathcal{A}) = \mathcal{O}_k(\mathcal{A}_{\setminus \Sigma_{\text{high}}})$*

Remark 1. If $k = 0$, any run ρ ending in a location l would have the same observed run with memory 0: $\mathcal{O}_0(\rho) = (\mathcal{O}(l), 0)$. It is therefore clear that 0-POS-NNI is equivalent to the reachability of locations, which is decidable [AD94].

If $k = 1$, observed runs of memory k are reduced to a measurement of the time elapsed since the last observed event of the run. It follows that 1-POS-NNI is decidable as well, since the (possibly infinite) maximal time that can be elapsed in a set of indistinguishable locations is computable (see, e. g., Section 4.2).

4.1 First approach to 2-POS-NNI

One intuitive way to solve 2-POS-NNI is to add two attacker clocks x_{old} and x_{new} to the system to model the time elapsed since the last two observed events and to synchronize on transitions associated with observable events the security automaton with another automaton modeling the last sequence of observations of the attacker. As the last two events occur, the attacker clocks are successively reset to zero.

An automaton with two observations A and B, and a single clock x (left) and its associated attacker automaton with 2 clocks x_{old} and x_{new} (right) are given in Fig. 3. Location l_A and $l_{A'}$ are observed as A and location l_B is observed as B.

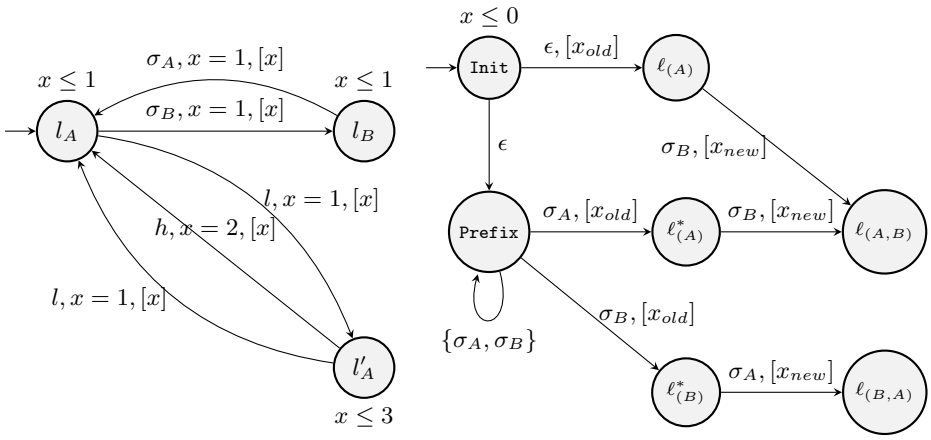


Fig. 3: A TA and a TA modeling the attacker memory ([] denote clock resets)

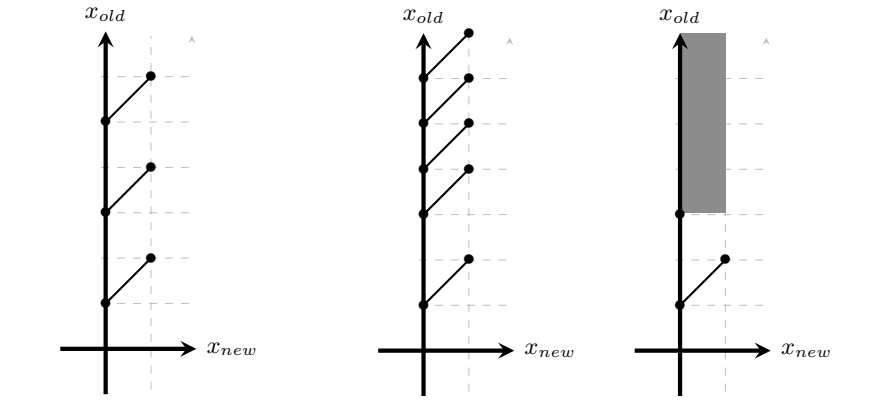
The automata shown in Fig. 3 illustrate how 2-POS-NNI can be reduced to a state reachability problem. For instance, having $(\ell_{(A,B)}; x_{old} = 2.5, x_{new} = 0.5)$ reachable on the attacker automaton is equivalent to the existence of a run so that at some point 2 time units elapse in l_A and l'_A, followed by 0.5 time unit elapsed in l_B. Conversely, for any run with n observed events ($n > 2$) there exists one run of the attacker automaton that loops $n - 2$ times on location `Prefix`

before reaching $\ell^*_{(A)}$ followed by $\ell_{(A,B)}$ (or $\ell^*_{(B)}$ followed by $\ell_{(B,A)}$), resetting the attacker clocks along the way. Location $\ell_{(A)}$ is meant to account for runs with two or less events by "skipping" the `Prefix` cycle. However, in a run with $n \geq 2$ events, the corresponding run on the attacker automaton could cycle n times or $n-1$ times on location `Prefix` so that it ends in location `Prefix`, $\ell^*_{(A)}$ or $\ell^*_{(B)}$ instead of $\ell_{(A,B)}$ or $\ell_{(B,A)}$. Therefore, clock valuations on these locations do not correspond to any observed runs.

This method can be generalized to any observation function: when one of the last two observable events occurs, we add the new observation to the sequence memorized by the attacker. Previous events are accounted for by cycling on location `Prefix`.

However, the region graph (or zone-based graphs) does not faithfully represent all reachable states of a given automaton because it relies for finiteness on the use of some maximal constant above which the property that if some valuation in a region is reachable, then all valuations in the region are, does not hold. And it may well be that interfering states have some clock values above the maximal constant found in clock constraints of the automaton.

For instance, for the automaton introduced in Fig. 3, we get the possible clock valuations associated with the location $\ell_{(A,B)}$ (clock x is omitted) depicted in Figure 4.



(a) Reachable clock valuations[1] using only low-level actions

(b) Reachable clock valuations[1] with all actions

(c) Reachable regions[2] with all actions

Fig. 4: Reachable states and Reachable regions location $\ell_{(A,B)}$ of Fig. 3

[1] Regions without c-approximation, i.e. each point represent a reachable state of location $\ell_{(A,B)}$ associated with given values of x_{new} and x_{old}

[2] On this figure, regions are represented according to the original definition proposed by Alur & Dill ([AD94]), that is with c-approximation, c being the largest synthaxic value used to define the automaton

We see that the interfering behaviors occur at the earliest when $x_{old} = 4$, after the maximal constant, which is 3. Here this could therefore be solved by taking 4 instead of 3 as the constant for the definition of regions. However we can define automata with more complex behaviors. For instance, changing the 2 with a 3 and every 1s by 2s in the automaton of Fig. 3 would change the required maximal constant of regions to 6 instead of 4. By adding more locations to the automata the maximal constant can be pushed even further from the largest constant of the syntax. We found no trivial relation between the size of the automaton and the necessary constant to ensure the validity of this approach.

Remark 2. Note that this example also shows that the claims of [BT03,GMR07] that n-state-non-interference and St-NNI are decidable in the general case by computing the set of reachable states of the considered automata do not hold, unless the clocks are bounded.

This issue only gets worse when we extend this method to any memory size or when the security automaton uses more clocks. This kind of example highlights the need for automata with bounded time between events (i.e. bounded attacker clocks) to decide k-POS-NNI.

4.2 Finite Duration Observation Automata (FDO automata)

In this part, we provide a class of automata for which k-POS-NNI is decidable. It consists of automata for which the time elapsed between two consecutive observed events of any run is bounded.

Definition 16. *An automaton $\mathcal{A}$ is a Finite Duration Observation Automaton (FDO Automaton) with respect to an observation $\mathcal{O}$*[1] *if and only if $\exists M \in \mathbb{R}_{\geq 0}$ such that $\forall \rho \in \Psi(\mathcal{A})$ with $\mathcal{O}(\rho) = (o_0, t_0 = 0)(o_1, t_1) \cdots (o_n, t_n), \forall 0 \leq k < n$, we have $t_{k+1} - t_k \leq M$.*

We call M the observation bound *of $\mathcal{A}$.*

By focusing on systems in which time cannot elapse indefinitely without any event occurring we get systems with a finite set of unapproximated regions reachable on the attacker automaton as introduced above. Moreover, using a different construct we will show in the next section that these regions only contain clock valuations smaller than a specific bound, ensuring the equivalence between reachable states (each corresponding to an observed runs of finite memory) and the region graph. But first we provide a few results about FDO automata.

Proposition 2. *Given an automaton $\mathcal{A}$ associated with an observation function $\mathcal{O}$. The membership of $\mathcal{A}$ in the set of FDO automata with respect to $\mathcal{O}$ is decidable.*

Proof (sketch). Given a TA $\mathcal{A}$ with a set of clocks $\mathbb{X}$ and an observation function $\mathcal{O}$, we define the TA $\mathcal{A}_{\mathcal{O}}$ as $\mathcal{A}$ but with an extra clock y and a self-loop edge on

[1] We also say the automaton is FDO w.r.t. $\mathcal{O}$.

every location to reset this clock. We then compute the region graph $\mathcal{RG}$ of $\mathcal{A}_{\mathcal{O}}$ using standard techniques. We define $\mathcal{RG}^-$ by removing every discrete transition between two distinguishable locations from $\mathcal{RG}$. Let R be a region of $\mathcal{RG}^-$ in which $y = 0$. We call *unit cycle* a path in $\mathcal{RG}^-$ starting in R and ending in a region R' with $y \geq 1$ and all other clocks satisfying the same constraints as in R. Given any observation bound M, if there is a *unit cycle* in the region graph there is a run executing the *unit cycle* at least $M + 1$ time, showing that the maximal elapsed time between events is not bounded by M, hence $\mathcal{A}$ is not FDO w.r.t. $\mathcal{O}$. We now suppose that $\mathcal{A}$ does not contain any *unit cycle* yet is not FDO w.r.t $\mathcal{O}$. Because $\mathcal{A}$ is not FDO, for any $M \in \mathbb{R}_{\geq 0}$ there exists $\rho \in \Psi(\mathcal{A})$ s.t. $\mathcal{O}_1(\rho) = (o, 0)(o, t > M)$. Let us call s the state reached when the event leading to observation o occurs. Because of the absence of *unit cycles*, neither s nor any state contained in the same region as s is ever reached in ρ after one time unit has elapsed unless an observed event occurred. Because we can take $M > 1$, it follows that at least one state s' contained in a different region than s has to be reached at some point in ρ after s'. Because the same argument applies to s' by taking $M > 2$. Inductively, it follows that after $M+1$ time unit elapsed, at least M regions of the state-space are not reachable unless a event occurred. Because the amount of regions (defined with c-approx) is finite, it follows that $\mathcal{A}$ is FDO w.r.t. $\mathcal{O}$ with an observation bound less than the number of regions, which is a contradiction. This proves that $\mathcal{A}$ is FDO w.r.t. $\mathcal{O}$ if and only if it has no *unit cycle* in its region graph, which is decidable. □

Corollary 1. *Given an observation $\mathcal{O}$ associated to the automaton $\mathcal{A}$ with x clocks, c the largest integer used in clock constraints and L the size of the set of locations. If $\mathcal{A}$ is FDO w.r.t. $\mathcal{O}$, then its observation bound M verifies the relation:*

$$M \leq x!(2c + 2)^x L$$

Proof. As previously demonstrated, $\mathcal{A}$ is FDO w.r.t. $\mathcal{O}$ iff it has no *unit cycle*. Furthermore, we showed that the observation bound of an FDO automaton is bounded by its number of region. Using the bound on the number of regions given in [AD94], the relation follows. □

Corollary 2. *Given an observation $\mathcal{O}$ associated to the automaton $\mathcal{A}$. If $\mathcal{A}$ is FDO w.r.t. $\mathcal{O}$, then its observation bound M can be computed.*

Proof. By adding a clock y to $\mathcal{A}$ (with x clocks, L locations and c its largest constant used in clock constraints) and replacing any transition of the form (l, g, a, R, l') with $\mathcal{O}(l) \neq \mathcal{O}(l')$ by $(l, g, a, R \cup \{y\}, l')$, the cumulated time elapsed in a run since the last observed event always corresponds to the value of y. Thanks to the previous corollary we know that defining regions using constant $(x!(2c + 2)^x L + 1)$ instead of c will provide a region-graph of $\mathcal{A}$ that does not approximate the value of y. The maximal value of y can therefore be computed by taking the smallest value M so that no region in the latter region graph contains valuations of y greater than M. □

4.3 Deciding k-POS-NNI for FDO automata

In this section we generalize and formalize the construction proposed in Section 4.1.

Given a sequence u, we note $u(i)$ the i-th element of u, $u[a,b]$ the subset of u ranging from index a to index b, both $u(a)$ and $u(b)$ being included.

Given a security automaton $\mathcal{A} = (\Sigma_{\mathsf{low}} \cup \Sigma_{\mathsf{high}}, L, l_0, \mathbb{X}, I, E)$ associated with observation $\mathcal{O} : L \mapsto \Omega$ and an attacker memory k. We define Ω^j the set of sequences of observations from Ω of length j.

We define the attacker automaton $\mathcal{B}_k = (\{\epsilon\} \cup \Sigma_\Omega, L_{\mathcal{B}_k}, \texttt{Init}, Y, I_{\mathcal{B}_k}, E_{\mathcal{B}_k})$:

- $\Sigma_\Omega = \{\sigma_o \,|\, o \in \Omega\}$
- $L_{\mathcal{B}_k} = \{\texttt{Init}\} \cup \{\texttt{Prefix}\} \cup \{\ell^*_\omega \,|\, \omega \in \Omega^j, j \in [\![1, k-1]\!]\} \cup \{\ell_\omega \,|\, \omega \in \Omega^j, j \in [\![1, k]\!]\}$
- $I_{\mathcal{B}_k}(\texttt{Init}) = (Y = \mathbf{0}), I_{\mathcal{B}_k}(l \neq \texttt{Init}) = \texttt{true}$
- $Y = (y_i)_{i \in [\![1,k]\!]}$, where the y_i are some new clocks;
- $E_{\mathcal{B}_k}$ is defined as follows:
  - **(a) initialization** $(\texttt{Init}, \emptyset, \epsilon, [y_1], \ell_{\mathcal{O}(l_0)})$
  - **(a) event** $\forall j \in [\![2, k]\!], \forall \omega \in \Omega^j, (\ell_{\omega[1,j-1]}, \emptyset, \sigma_{\omega[j]}, y_j, \ell_\omega) \in E_{\mathcal{B}_k}$
  - **(b) initialization** $(\texttt{Init}, \emptyset, \epsilon, \emptyset, \texttt{Prefix})$
  - **(b) prefix event** $(\texttt{Prefix}, \emptyset, \Sigma_\Omega, \emptyset, \texttt{Prefix})$
  - **(b) first event** $\forall o \in \Omega, (\texttt{Prefix}, \emptyset, \sigma_o, [y_1], \ell^*_o) \in E_{\mathcal{B}_k}$
  - **(b) intermediary event** $\forall j \in [\![2, k-1]\!], \forall \omega \in \Omega^j, (\ell^*_{\omega[1,j-1]}, \emptyset, \sigma_{\omega[j]}, y_j, \ell^*_\omega) \in E_{\mathcal{B}_k}$
  - **(b) last event** $\forall \omega \in \Omega^k, (\ell^*_{\omega[1,k-1]}, \emptyset, \sigma_{\omega[k]}, [y_k], \ell_\omega) \in E_{\mathcal{B}_k}$

$\mathcal{B}_k$ is simply the generalization of the attacker automaton proposed in Example 3. In the example, the **(a) initialization** transition is the one going from location `Init` to location $\ell_{(A)}$, it is used for runs with k events or less (again, counting the initialization of the observed system as an event); the **(b) initialization** transition is the one going from `Init` to location `Prefix`, it is used for runs with more than k events; the transitions going from `Prefix` to $\ell^*_{(A)}$ or $\ell^*_{(B)}$ are **(b) first event** transitions; the transitions going from $\ell^*_{(A)}$ (resp: $\ell^*_{(B)}$) to $\ell_{(A,B)}$ (resp: $\ell_{(B,A)}$) are **(b) last event** transitions; the **(b) intermediary event** type is absent in the example as it only appears when $k \geq 3$ where intermediary locations would be necessary between $\ell^*_{(A)}$ (resp: $\ell^*_{(B)}$) and $\ell_{(A,B)}$ (resp: $\ell_{(B,A)}$) to account for more time-stamps memorized. Lastly, the **(a) event** transitions correspond to the transition between $\ell_{(A)}$ and $\ell_{(A,B)}$. Overall, the **(a)**-transitions serve the purpose of memorizing time-stamps in short runs whereas the **(b)**-transitions serve the same purpose for longer runs by allowing the attacker to ignore information about the part of the run they would discard anyway (by cycling on the `Prefix` location) so that only the last k events are associated with clocks $(y_i)_{i \in [\![1,k]\!]}$.

Because $\mathcal{A}$ and $\mathcal{B}_k$ do not use the same alphabet, we define the ϵ-automaton $\mathcal{A}^{\mathcal{O}} = (\{\epsilon\} \cup \Sigma_\Omega, L, l_0, \mathbb{X}, E^{\mathcal{O}})$ so that $\mathcal{O}_k(\mathcal{A}) = \mathcal{O}_k(\mathcal{A}^{\mathcal{O}})$ as follows:

$$\forall e = (l, g, a, R, l') \in E :$$
$$\mathcal{O}(l) = \mathcal{O}(l') \implies (l, g, \epsilon, R, l') \in E^{\mathcal{O}}$$
$$\mathcal{O}(l) \neq \mathcal{O}(l') \implies (l, g, \sigma_{\mathcal{O}(l')}, R, l') \in E^{\mathcal{O}}$$

We note $\mathcal{A}||_\Sigma\mathcal{B}$ the synchronized product à la Arnold Nivat ([Arn94]) of $\mathcal{A}$ with $\mathcal{B}$ on the set of action Σ.

Lemma 1. *$(o_1, t_1 = 0)(o_2, t_2)\dots(o_j, t_j)(o_j, t_{j+1}) \in \mathcal{O}_k(\mathcal{A})$ iff $\exists l \in L$, and a valuation ν on $Y \cup \mathbb{X}$ such that $\forall i \in [\![1, j]\!], \nu(y_i) = t_{j+1} - t_i$ and $((l, \ell_{(o_1,\dots,o_j)}), \nu) \in Q^{\mathcal{A}^\mathcal{O}||_{\Sigma_\Omega}\mathcal{B}_k}$.*

Proof (sketch). We first notice that $\mathcal{B}_k$ is universal on alphabet Σ_Ω. This is sufficient to show that any reachable state of $\mathcal{A}^\mathcal{O}$ is also a reachable state of $\mathcal{A}^\mathcal{O}||_{\Sigma_\Omega}\mathcal{B}_k$. We use this to associate to any run ρ of $\mathcal{A}$ a run ρ' of $\mathcal{A}^\mathcal{O}||_{\Sigma_\Omega}\mathcal{B}_k$ ending in a location of the form $(l \in L, \ell_{(o_1,\dots,o_n)})$ for some $n \leq k$. We ensure that such a run exists by splitting cases based on the number of events of ρ. If ρ contains $j \leq k$ events we simply consider the a run ρ' starting with the **(a) initialization** transition. If ρ contains$j > k$ events we consider a run ρ' starting with the **(b) initialization** transition and cycling on `Prefix` $k - j$ times. By the definition of $E^\mathcal{O}$ and $E_{\mathcal{B}_k}$, a new location of the form ℓ_ω or ℓ^*_ω can only be reached when an observed event would occur in the run of $\mathcal{A}$ that $\mathcal{A}^\mathcal{O}||_{\Sigma_\Omega}\mathcal{B}_k$ is reproducing. It follows that the values of clocks y_i are such that the difference between consecutive clocks are equal to the time elapsed between observable events of the run. The relation between time-stamps of the observed run with memory k of $\mathcal{A}$ and the clock valuations follows.

The other direction of the equivalence boils down to the same idea. To each state of $\mathcal{A}^\mathcal{O}||_{\Sigma_\Omega}\mathcal{B}_k$ can be associated plenty of runs of $\mathcal{A}$, possibly mapping to multiple observed runs, but to only a single observed run with memory k. □

Theorem 2. *$\forall k \in \mathbb{N}$, k-POS-NNI associated with an observation $\mathcal{O}$ is decidable for FDO automata w.r.t. $\mathcal{O}$.*

Proof (sketch). Given an FDO security automata $\mathcal{A}$ and its observation bound M, we can show that for any state $((l, \ell_\omega), \nu)$ reachable in $\mathcal{A}^\mathcal{O}||_{\Sigma_\Omega}\mathcal{B}_k$, $\forall i < |\omega|, \nu(y_i) < kM$. Therefore, the clocks y_i that are relevant to Lemma 1 do not suffer from the kind of "loss of information" caused by the region abstraction described on Figure 4 if the constant used to define regions is greater than kM.

For any $\omega \in \Omega^j$ for some $j < k$, we can compute a set Γ_ω of pairs (ω, ν) for every ν that is the projection on clocks $\{y_i, \dots, y_{|\omega|}\}$ of a valuation contained in any region R such that a vertex $((l, \ell_\omega), R)$ is reachable in the region graph of $\mathcal{A}^\mathcal{O}||_{\Sigma_\Omega}\mathcal{B}_k$. by Lemma 1, Γ_ω is in bijection with the set of observed runs with memory k of the form $(\omega[1], 0)\dots(\omega[|\omega|], t_{|\omega|})(\omega[|\omega|], t_{|\omega|+1})$. Because there is a finite amount of possible ω, the set $\Gamma = \bigcup_\omega \Gamma_\omega$ is in bijection with $\mathcal{O}_k(\mathcal{A})$

Repeating this process for $\mathcal{A}_{\setminus\Sigma_{\mathsf{high}}}$ provides a set $\Gamma^{\setminus\Sigma_{\mathsf{high}}}$ in bijection with $\mathcal{O}_k(\mathcal{A}_{\setminus\Sigma_{\mathsf{high}}})$. Therefore, $\mathcal{A}$ verifies k-POS-NNI if and only if $\Gamma^{\setminus\Sigma_{\mathsf{high}}} = \Gamma$, which can be computed using standard techniques on polyhedra, because those infinite sets can be represented by finite unions of regions. □

Although the method given above would not be efficient if implemented as it is, a PSPACE algorithm can be found.

Theorem 3. *With k represented in unary, k-POS-NNI for FDO automata is PSPACE-complete.*

Proof (sketch). To prove PSPACE-membership we use the PSPACE-membership of region reachability in a region-graph. Because $\mathcal{B}_k$ is only defined with respect to $\mathcal{O}$ and k it is not necessary to compute it explicitly, instead the transition function of the Turing machine can account for the sequence of observations as $\mathcal{O}$ is given in the input. Because the sequence of observation has to be memorized, an overhead of size linear with k is induced, hence the necessity for k to be represented in unary to preserve PSPACE complexity.

To prove PSPACE-hardness we reduce from location reachability in TAs. Given a TA with a given target location, we create an augmented TA with an extra location associated with a unique observation. We ensure this additional location is reachable either by using a high-level action or be using a low-level action from the target location so that this unique observation appears in the set of observed runs restricted to low level actions if and only if the target location is reachable. Some other details are added to ensure the augmented automaton is FDO and to ensure the strict equality between observed runs.

5 Discussion and future work

We have introduced a new property of state-based non-interference. We believe that this property is more realistic than previously studied state-based approach to non-interference as it models a smarter attacker with a more reasonable observation power. We have shown that this property is not decidable in the general case. However, we have provided a decidable class of automata for which the property is decidable given a finite memory of any length. We can also decide membership in that class.

For future work consists first in investigating other policies for observation replacement in the memory of the attacker. We also want to find more efficient, e. g. DBM-based, algorithms to decide k-POS-NNI for FDO automata.

Lastly, we want to settle the general case of k-POS-NNI decidability for non-FDO automata, and of the related problem of the decidability of St-NNI when clocks are not bounded which, as we have shown, is actually still open.

References

AD94. Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

AETYM21. Ikhlass Ammar, Yamen El Touati, Moez Yeddes, and John Mullins. Bounded opacity for timed systems. *Journal of Information Security and Applications*, 61:102926, 2021.

AK20. Étienne André and Aleksander Kryukov. Parametric non-interference in timed automata. In *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 37–42. IEEE, 2020.

ALMS22. Étienne André, Didier Lime, Dylan Marinho, and Jun Sun. Guaranteeing timed opacity using parametric timed model checking. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(4):1–36, 2022.

Arn94. André Arnold. Finite transition systems. *Prentice Hall*, 1994.

BB07. Andrew Bortz and Dan Boneh. Exposing private information by timing web applications. In *Proceedings of the 16th international conference on World Wide Web*, pages 621–628, 2007.

BCLR15. Gilles Benattar, Franck Cassez, Didier Lime, and Olivier H Roux. Control and synthesis of non-interferent timed systems. *International Journal of Control*, 88(2):217–236, 2015.

BKMR08. Jeremy Bryans, Maciej Koutny, Laurent Mazare, and Peter Y. A. Ryan. Opacity generalised to transition systems. *International Journal of Information Security*, 7(6):421–435, 2008.

BT03. Roberto Barbuti and Luca Tesei. A decidable notion of timed non-interference. *Fundamenta Informaticae*, 54(2-3):137–150, 2003.

FS00. Edward W. Felten and Michael A. Schneider. Timing attacks on web privacy. In *Proceedings of the 7th ACM Conference on Computer and Communications Security*, CCS '00, page 25–32, New York, NY, USA, 2000. Association for Computing Machinery.

GM84. Joseph A Goguen and José Meseguer. Unwinding and inference control. In *1984 IEEE Symposium on Security and Privacy*, pages 75–75. IEEE, 1984.

GMR07. Guillaume Gardey, John Mullins, and Olivier H Roux. Non-interference control synthesis for security timed automata. *Electronic Notes in Theoretical Computer Science*, 180(1):35–53, 2007.

GSB18. Christopher Gerking, David Schubert, and Eric Bodden. Model checking the information flow security of real-time systems. In *Engineering Secure Software and Systems: 10th International Symposium, ESSoS 2018, Paris, France, June 26-27, 2018, Proceedings 10*, pages 27–43. Springer, 2018.

Koc96. Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Annual International Cryptology Conference*, pages 104–113. Springer, 1996.

KPJJ13. Robert Kotcher, Yutong Pei, Pranjal Jumde, and Collin Jackson. Cross-origin pixel stealing: timing attacks using css filters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1055–1062, 2013.

WZ18. Lingtai Wang and Naijun Zhan. Decidability of the initial-state opacity of real-time automata. In *Symposium on Real-Time and Hybrid Systems: Essays Dedicated to Professor Chaochen Zhou on the Occasion of His 80th Birthday*, pages 44–60. Springer, 2018.