Secure information release in timed automata

Vasilikos, Panagiotis; Nielson, Flemming; Nielson, Hanne Riis

Published in:
POST 2018: Principles of Security and Trust

Link to article, DOI:
10.1007/978-3-319-89722-6_2

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Secure Information Release in Timed Automata

Panagiotis Vasilikos(✉), Flemming Nielson, and Hanne Riis Nielson

Department of Applied Mathematics and Computer Science,
Technical University of Denmark, Kongens Lyngby, Denmark
{panva,fnie,hrni}@dtu.dk

Abstract. One of the key demands of cyberphysical systems is that they meet their safety goals. *Timed automata* has established itself as a formalism for modeling and analyzing the real-time safety aspects of cyberphysical systems. Increasingly it is also demanded that cyberphysical systems meet a number of security goals for confidentiality and integrity. Notions of security based on *Information flow control*, such as non-interference, provide strong guarantees that no information is leaked; however, many cyberphysical systems leak intentionally some information in order to achieve their purposes.

In this paper, we develop a formal approach of information flow for timed automata that allows intentional information leaks. The security of a timed automaton is then defined using a bisimulation relation that takes account of the non-determinism and the clocks of timed automata. Finally, we define an algorithm that traverses a timed automaton and imposes information flow constraints on it and we prove that our algorithm is sound with respect to our security notion.

1 Introduction

Motivation. Embedded systems are key components of cyberphysical systems and are often subject to stringent safety goals. Among the current approaches to the modeling and analysis of timed systems, the approach of *timed automata* [5] stands out as being a very successful approach with well-developed tool support – in particular the *UPPAAL* suite [28] of tools. As cyberphysical systems become increasingly distributed and interconnected through wireless communication links it becomes even more important to ensure that they meet suitable security goals.

In this paper, we are motivated by an example of a smart power grid system. In its very basic form, a smart grid system consists of a meter that measures the electricity consumption in a customer's (C) house and then sends this data to the utility company (UC). The detailed measurements of the meter provide more accurate billings for UC, while C receives energy management plans that optimize his energy consumption. Although this setting seems to be beneficial for both UC and C, it has been shown that high-frequent monitoring of the power flow poses a major threat to the privacy of C [14,23,27]. To deal with

L. Bauer and R. Küsters (Eds.): POST 2018, LNCS 10804, pp. 28–52, 2018.
https://doi.org/10.1007/978-3-319-89722-6_2

this problem many smart grid systems introduce a trusted third-party (TTP), on which both UC and C agree [27]. The data of the meter now is collected by the TTP and by the end of each month the TTP charges C depending on the tariff prices defined by UC. In this protocol, UC trusts TTP for the accurate billing of C, while C trusts TTP with its sensitive data. However, in some cases, C may desire an energy management plan by UC, and consequently he makes a clear statement to TTP that allows the latter to release the private data of C to UC. Therefore, it is challenging to formally prove that our trusted smart grid system leaks information only under C's decision.

Information Flow Control. [10,26,29] is a key approach to ensuring that software systems maintain the confidentiality and/or integrity of their data. Policies for secure information flow are usually formalized as non-interference [29] properties and systems that adhere to the stated policy are guaranteed to admit no flow of information that violates it. However, in many applications information is leaked by intention as in our smart grid example. To deal with such systems, information flow control approaches are usually extended with mechanisms that permit controlled information leakage. The major difficulty imposed by this extension is to formalize notions of security that are able to differentiate between the intentional and the unintentional information leakages in a system.

Contribution. It is therefore natural to extend the enforcement of safety properties of *timed automata* with the enforcement of appropriate Information Flow policies. It is immediate that the treatment of *clocks*, the *non-determinism*, and the *unstructured control flow* inherent in automata will pose a challenge. More fundamentally there is the challenge that *timed automata* is an automata-based formalism whereas most approaches to Information Flow take a language-based approach by developing type systems for programming languages with structured control flow or process calculi.

We start by giving the semantics of timed automata (Sect. 2) based on the ones used in UPPAAL [28]. Next, we formalize the security of a timed automaton using a bisimulation relation (Sect. 3). This notion describes the observations of a *passive attacker* and formally describes where an observation is allowed to leak information and where it is not. To deal with implicit flows we define a general notion of the post-dominator relation [18] (Sect. 4). We then develop a sound algorithm (Sect. 5) that imposes information flow constraints on the clocks and the variables of a timed automaton. We finish with our conclusions (Sect. 6) and the proofs of our main results (Appendix).

Related Work. There are other papers dealing with Information Flow using language based techniques for programs with a notion of time [2,9,16,22] or programs that leak information intentionally [6,13,19–21,24]. Our contribution focuses on the challenges of continuous time and the guarded actions of timed automata.

The work of [7,8] define a notion of non-interference for timed automata with high-level (secret) and low-level (public) actions. Their notion of security is expressed as a non-interference property and it depends on a natural number m,

representing a minimum delay between high-level actions such that the low-level behaviors are not affected by the high-level ones. The authors of [17] define a notion of timed non-interference based on bisimulations for probabilistic timed automata which again have high-level (secret) and low-level (public) actions. A somewhat different approach is taken in [12] that studies the synthesis of controllers. None of those approaches considers timed automata that have data variables, nor is their notion of security able to accommodate systems that leak information intentionally.

The authors of [25] take a language-based approach and they define a type-system for programs written in the language Timed Commands. A program in their language gives rise to a timed automaton, and type-checked programs adhere to a non-interference like security property. However, their approach is limited only to automata that can be described by their language and they do not consider information release.

2 Timed Automata

A *timed automaton* [1,5] $\mathsf{TA} = (\mathsf{Q}, \mathsf{E}, \mathsf{I}, q_\circ)$ consists of a set of nodes Q, a set of annotated edges E, and a labelling function I on nodes. A node $q_\circ \in \mathsf{Q}$ will be the initial node and the mapping I maps each node in Q to a condition (to be introduced below) that will be imposed as an invariant at the node.

The edges are annotated with actions and take the form $(q_s, g \rightarrow \boldsymbol{x} := \boldsymbol{a} \colon \boldsymbol{r}, q_t)$ where $q_s \in \mathsf{Q}$ is the source node and $q_t \in \mathsf{Q}$ is the target node. The action $g \rightarrow \boldsymbol{x} := \boldsymbol{a} \colon \boldsymbol{r}$ consists of a guard g that has to be satisfied in order for the multiple assignments $\boldsymbol{x} := \boldsymbol{a}$ to be performed and the clock variables $\boldsymbol{r}$ to be reset. We shall assume that the sequences $\boldsymbol{x}$ and $\boldsymbol{a}$ of program variables and expressions, respectively, have the same length and that $\boldsymbol{x}$ does not contain any repetitions. To cater for special cases we shall allow to write `skip` for the assignments of $g \rightarrow \boldsymbol{x} := \boldsymbol{a} \colon \boldsymbol{r}$ when $\boldsymbol{x}$ (and hence $\boldsymbol{a}$) is empty; also we shall allow to omit the guard g when it equals tt and to omit the clock resets when $\boldsymbol{r}$ is empty.

It has already emerged that we distinguish between (program) variables x and clock variables (or simply clocks) r. The arithmetic expressions a, guards g and conditions c are defined as follows using boolean tests b:

$$
\begin{aligned}
a &::= a_1 \,\mathsf{op}_a\, a_2 \mid x \mid n \\
b &::= \mathsf{tt} \mid \mathsf{ff} \mid a_1 \,\mathsf{op}_r\, a_2 \mid \neg b \mid b_1 \wedge b_2 \\
g &::= b \mid r \,\mathsf{op}_c\, n \mid (r_1 - r_2)\,\mathsf{op}_c\, n \mid g_1 \wedge g_2 \\
c &::= b \mid r \,\mathsf{op}_d\, n \mid (r_1 - r_2)\,\mathsf{op}_d\, n \mid c_1 \wedge c_2
\end{aligned}
$$

The arithmetic operators op_a and the relational operators op_r are as usual. For comparisons of clocks we use the operators $\mathsf{op}_c \in \{<, \leq, =, \geq, >\}$ in guards and the less permissive set of operators $\mathsf{op}_d \in \{<, \leq, =\}$ in conditions.

To specify the semantics of timed automata let σ be a state mapping variables to values (which we take to be integers) and let δ be a clock assignment mapping clocks to non-negative reals. We then have total semantic functions $[\![\cdot]\!]$ for evaluating the arithmetic expressions, boolean tests, guards and conditions;

the values of the arithmetic expressions and boolean expressions only depend on the states whereas that of guards and conditions also depend on the clock assignments.

The configurations of the timed automata have the form $\langle q, \sigma, \delta\rangle \in \mathbf{Config}$ where $[\![\mathsf{I}(q)]\!](\sigma, \delta)$ is true, and the transitions are described by an initial delay (possibly none) that increases the values of all the clocks followed by an action. Therefore, whenever $(q_s, g \rightarrow \boldsymbol{x} := \boldsymbol{a}\colon \boldsymbol{r}, q_t)$ is in E we have the rule:

$$\langle q_s, \sigma, \delta\rangle \xrightarrow{d} \langle q_t, \sigma', \delta'\rangle \begin{cases} d \geq 0 \\ [\![\mathsf{I}(q_s)]\!](\sigma, \delta + d) = \mathsf{tt}, \\ [\![g]\!](\sigma, \delta + d) = \mathsf{tt}, \\ \sigma' = \sigma[\boldsymbol{x} \mapsto [\![\boldsymbol{a}]\!]\sigma], \delta' = (\delta + d)[\boldsymbol{r} \mapsto \boldsymbol{0}], \\ [\![\mathsf{I}(q_t)]\!](\sigma', \delta') = \mathsf{tt} \end{cases}$$

where d corresponds to the initial delay. The rule ensures that after the initial delay the invariant and the guard are satisfied in the starting configuration and updates the mappings σ and δ where $\delta + d$ abbreviates $\lambda r.\, \delta(r) + d$. Finally, it ensures that the invariant is satisfied in the resulting configuration. Initial configurations assume that all clocks are initialized to 0 and have the form $\langle q_\circ, \sigma, \lambda r.0\rangle$.

Traces. We define a *trace* from $\langle q_s, \sigma, \delta\rangle$ to q_t in a timed automaton TA to have one of three forms. It may be a finite "successful" sequence

$$\begin{array}{l} \langle q_s, \sigma, \delta\rangle = \langle q'_0, \sigma'_0, \delta'_0\rangle \xrightarrow{d_1} \cdots \xrightarrow{d_n} \langle q'_n, \sigma'_n, \delta'_n\rangle \qquad (n > 0) \\ \text{such that } \{n\} = \{i \mid q'_i = q_t \wedge 0 < i \leq n\}. \end{array}$$

in which case at least one step is performed. It may be a finite "unsuccessful" sequence

$$\begin{array}{l} \langle q_s, \sigma, \delta\rangle = \langle q'_0, \sigma'_0, \delta'_0\rangle \xrightarrow{d_1} \cdots \xrightarrow{d_n} \langle q'_n, \sigma'_n, \delta'_n\rangle \qquad (n \geq 0) \\ \text{such that } \langle q'_n, \sigma'_n, \delta'_n\rangle \text{ is stuck and } q_t \notin \{q'_1, \cdots, q'_n\} \end{array}$$

where $\langle q'_n, \sigma'_n, \delta'_n\rangle$ is stuck when there is no action starting from $\langle q'_n, \sigma'_n, \delta'_n\rangle$. Finally, it may be an infinite "unsuccessful" sequence

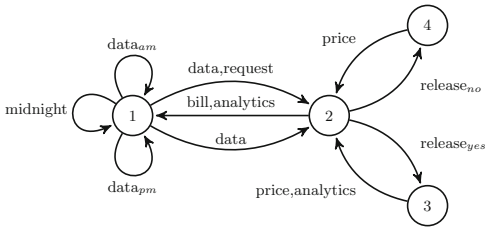
$$\begin{array}{l} \langle q_s, \sigma, \delta\rangle = \langle q'_0, \sigma'_0, \delta'_0\rangle \xrightarrow{d_1} \cdots \xrightarrow{d_n} \langle q'_n, \sigma'_n, \delta'_n\rangle \xrightarrow{d_{n+1}} \cdots \\ \text{such that } q_t \notin \{q'_1, \cdots, q'_n, \cdots\}. \end{array}$$

We shall write $[\![\mathsf{TA} : q_s \mapsto q_t]\!](\sigma, \delta)$ for the set of traces from $\langle q_s, \sigma, \delta\rangle$ to q_t. We then have the following proposition

Proposition 1 [15]**.** *For a pair (σ, δ) whenever $[\![\mathsf{TA} : q_s \mapsto q_t]\!](\sigma, \delta)$ contains only successful traces, then there exists a trace $t \in [\![\mathsf{TA} : q_s \mapsto q_t]\!](\sigma, \delta)$ with maximal length.*

We also define the *delay* of a trace t from $\langle q_s, \sigma, \delta\rangle$ to q_t and we have that if t is a successful trace

$$\langle q_s, \sigma, \delta\rangle = \langle q'_0, \sigma'_0, \delta'_0\rangle \xrightarrow{d_1} \cdots \xrightarrow{d_n} \langle q'_n, \sigma'_n, \delta'_n\rangle = \langle q_t, \sigma', \delta'\rangle$$



Invariants:

1	$\mathsf{T} \leq 720$
2	$\mathsf{T} = 720$
3	$\mathsf{T} = 720$
4	$\mathsf{T} = 720$

data$_{am}$:	$\mathsf{t} \leq 12 \wedge \mathsf{r} = 1 \rightarrow \mathsf{e_1}{:=}\mathsf{e_1} + \mathsf{ed}\colon \mathsf{r}$	release$_{no}$:	$\mathsf{s} = 0 \rightarrow \mathtt{skip}\colon$
data$_{pm}$:	$\mathsf{t} > 12 \wedge \mathsf{t} < 24 \wedge \mathsf{r} = 1 \rightarrow \mathsf{e_2}{:=}\mathsf{e_2} + \mathsf{ed}\colon \mathsf{r}$	release$_{yes}$:	$\mathsf{s} = 1 \rightarrow \mathsf{y_1}, \mathsf{y_2}{:=}\mathsf{c_1}, \mathsf{c_2}\colon$
midnight:	$\mathsf{t} = 24 \wedge \mathsf{r} = 1 \rightarrow \mathsf{e_2}{:=}\mathsf{e_2} + \mathsf{ed}\colon \mathsf{r}, \mathsf{t}$	price,analytics:	$\mathsf{p_1}, \mathsf{p_2}, \mathsf{a}, \mathsf{f}{:=}\mathsf{v_1}, \mathsf{v_2}, \mathsf{z}, 1$
data,request:	$\mathsf{T} = 720 \rightarrow \mathsf{s}, \mathsf{c_1}, \mathsf{c_2}{:=}1, \mathsf{e_1}, \mathsf{e_2}\colon$	price:	$\mathsf{p_1}, \mathsf{p_2}, \mathsf{a}, \mathsf{f}{:=}\mathsf{v_1}, \mathsf{v_2}, 0, 1$
data:	$\mathsf{T} = 720 \rightarrow \mathsf{s}, \mathsf{c_1}, \mathsf{c_2}{:=}0, \mathsf{e_1}, \mathsf{e_2}\colon$	bill,analytics:	$\mathsf{f} = 1 \rightarrow \mathsf{b}, \mathsf{x}, \mathsf{e_1}, \mathsf{e_2}, \mathsf{f}{:=}$ $\mathsf{p_1} * \mathsf{c_1} + \mathsf{p_2} * \mathsf{c_2}$, $\mathsf{a}, 0, 0, 0\colon \mathsf{T}, \mathsf{t}, \mathsf{r}$

Fig. 1. The timed automaton SG (and the abbreviations used).

then

$$\Delta(t) = \sum_{i=1}^{n} d_i$$

In the case of t being an unsuccessful (finite or infinite) trace we have that

$$\Delta(t) = \infty$$

Finally for (σ_1, δ_1), (σ_2, δ_2) whenever for all $t_1 \in [\![\mathsf{TA} : q_s \mapsto q_t]\!](\sigma_1, \delta_1)$ and $t_2 \in [\![\mathsf{TA} : q_s \mapsto q_t]\!](\sigma_2, \delta_2)$ we have that $\Delta(t_1) = \Delta(t_2)$, we will say that (σ_1, δ_1) and (σ_2, δ_2) have the same *termination behaviour* with respect to q_s and q_t. Note that it is not necessarily the case that a pair (σ, δ) has the same termination behaviour as itself.

Example 1. To illustrate our development we shall consider an example automaton of a smart grid system as the one described in Sect. 1. The timed automaton SG is given in Fig. 1 and it uses the clocks t and T to model the time elapse of a day and a month respectively. Between midnight and noon, the electricity data ed is aggregated in the variable $\mathsf{e_1}$, while from noon to midnight the measurements are saved in the variable $\mathsf{e_2}$. The clock r is used to regulate the frequency of the measurements, by allowing one measurement every full hour. At the end of a day (midnight) the last measurement is calculated and the clock t is being reset to 0 indicating the start of a new day. At the end of each month ($\mathsf{T} = 720$) the trusted party TTP collects the data $\mathsf{e_1}$ and $\mathsf{e_2}$ of the meter and stores it in the collectors $\mathsf{c_1}$ and $\mathsf{c_2}$ respectively. At the same time, the customer C sends a service request s to TTP in case he desires to get some analytics regarding his energy consumption. The TTP then requests from the UC the prices $\mathsf{p_1}$, $\mathsf{p_2}$ of the electricity tariffs for the two time periods of interest and in case that C has made a request for his data to be analysed ($\mathsf{s} = 1$ otherwise $\mathsf{s} = 0$), TTP also reveals the collected data $\mathsf{c_1}$ and $\mathsf{c_2}$ to the UC where the latter stores them in the variables $\mathsf{y_1}$ and $\mathsf{y_2}$ respectively. The UC then responds back to the TTP by sending the values $\mathsf{v_1}$ and $\mathsf{v_2}$ of the electricity tariffs and also the result z of C's data analytics in case C made a request for that, otherwise it sends the value 0.

Once the TTP receives everything ($\mathsf{f} = 1$) he calculates the bill b for C, sends it to him together with the analysis result a (C stores it in x), the clocks and the variables of the meter are being reset to 0 and a new month starts. For simplicity here we assume that all the calculations done by the TTP and the UC by the end of the month are being completed in zero time.

3 Information Flow

We envisage that there is a security lattice expressing the permissible flows [10]. Formally this is a complete lattice and the permitted flows go in the direction of the partial order. In our development, it will contain just two elements, L (for low) and H (for high), and we set $L \sqsubseteq H$ so that only the flow from H to L is disallowed. For confidentiality, one would take L to mean public and H to mean private and for integrity one would take L to mean trusted and H to mean dubious.

A *security policy* is then expressed by a mapping $\mathcal{L}$ that assigns an element of the security lattice to each program variable and clock variable. An entity is called *high* if it is mapped to H by $\mathcal{L}$, and it is said to be *low* if it is mapped to L by $\mathcal{L}$. To express adherence to the security policy we use the binary operation $\rightsquigarrow$ defined on sets χ and χ' (of variables and clocks):

$$\chi \rightsquigarrow \chi' \Leftrightarrow \forall u \in \chi : \forall u' \in \chi' : \mathcal{L}(u) \sqsubseteq \mathcal{L}(u')$$

This expresses that all the entities of χ may flow into those of χ'; note that if one of the entities of χ has a high security level then it must be the case that all the entities of χ' have high security level.

Example 2. Returning to Example 1 of our smart grid system, we have that $\mathcal{L}$ maps the program variable ed of the electricity data, the variables $\mathsf{e}_1, \mathsf{e}_2$ that store this data, the collectors $\mathsf{c}_1, \mathsf{c}_2$ and the bill b to the security level H, while the rest of the program variables and clocks are mapped to L.

Information flow control enforces a security policy by imposing constraints of the form $\{y\} \rightsquigarrow \{x\}$ whenever the value of y may somehow influence (or flow into) that of x. Traditionally we distinguish between *explicit* and *implicit* flows as explained below. As an example of an *explicit* flow consider a simple assignment of the form $x{:=}a$. This gives rise to a condition $\mathsf{fv}(a) \rightsquigarrow \{x\}$ so as to indicate that the *explicit* flow from the variables of a to the variable x must adhere to the security policy: if a contains a variable with high security level then x also must have high security level. For an example of an *implicit* flow consider a conditional assignment $g \rightarrow x{:=}0$ where x is assigned the constant value 0 in case g evaluates to true. This gives rise to a condition $\mathsf{fv}(g) \rightsquigarrow \{x\}$ so as to indicate that the *implicit* flow from the variables of g to the variable x must adhere to the security policy: if g contains a variable with high security level then x also must have high security level.

As has already been explained, many applications as our smart grid example inevitably leak some information. In this paper we develop an approach to ensure

that the security policy is adhered to by the timed automaton of interest, however in certain conditions it can be bypassed. Thus, for a timed automaton $\mathsf{TA} = (\mathsf{Q}, \mathsf{E}, \mathsf{I}, q_\circ)$, we shall assume that there exists a set of *observable nodes* $Y \subseteq \mathsf{Q}$, that are the nodes where the values of program variables and clocks with low security are observable by an attacker. The observable nodes will be described by the union of two *disjoint* sets Y_s and Y_w, where a node q in Y_s (Y_w resp.) will be called *strongly observable* (*weakly observable* resp.). The key idea is to ensure that $\{x\} \rightsquigarrow \{y\}$ whenever there is an *explicit* flow of information from x to y (as illustrated above) or an *implicit* flow from x to y in computations that lead to strongly observable nodes, while computations that lead to weakly observable nodes are allowed to bypass the security policy $\mathcal{L}$.

To overcome the vagueness of this explanation we need to define a semantic condition that encompasses our notion of permissible information flow, where information leakage occurs only at specific places in our automaton.

Observable Steps. Since the values of low program variables and clocks are only observable at the nodes in Y, we collapse the transitions of the automaton that lead to non-observable nodes into one. Thus we have an *observable successful step*

$$\langle q_s, \sigma, \delta\rangle \overset{D}{\Longrightarrow}_Y \langle q_t, \sigma', \delta'\rangle$$

whenever there exists a successful trace t

$$\langle q_s, \sigma, \delta\rangle = \langle q_0, \sigma_0, \delta_0\rangle \xrightarrow{d_1} \cdots \xrightarrow{d_n} \langle q_n, \sigma_n, \delta_n\rangle = \langle q_t, \sigma', \delta'\rangle \quad (n > 0)$$

from $\langle q_s, \sigma, \delta\rangle$ to q_t in TA and $q_t \in Y$, $D = \Delta(t)$ and $\forall i \in \{1, ..., n-1\} : q_i \notin Y$.

And we have an *observable unsuccessful trace*

$$\langle q_s, \sigma, \delta\rangle \overset{\infty}{\Longrightarrow}_Y \bot$$

whenever there exists an unsuccessful finite trace

$$\langle q_s, \sigma, \delta\rangle = \langle q_0, \sigma_0, \delta_0\rangle \xrightarrow{d_1} \cdots \xrightarrow{d_n} \langle q_n, \sigma_n, \delta_n\rangle \quad (n \geq 0)$$

or an unsuccessful infinite trace

$$\langle q_s, \sigma, \delta\rangle = \langle q_0, \sigma_0, \delta_0\rangle \xrightarrow{d_1} \cdots \xrightarrow{d_n} \langle q_n, \sigma_n, \delta_n\rangle \xrightarrow{d_{n+1}} \cdots$$

from $\langle q_s, \sigma, \delta\rangle$ to any of the nodes in Y and $\forall i > 0 : q_i \notin Y$. From now on it should be clear that a configuration γ will range over $\mathbf{Config} \cup \{\bot\}$.

We write $(\sigma, \delta) \equiv (\sigma', \delta')$ to indicate that the two pairs are equal on low variables and low clocks:

$$(\sigma, \delta) \equiv (\sigma', \delta') \quad \text{iff} \quad \begin{array}{l} \forall x : \mathcal{L}(x) = L \Rightarrow \sigma(x) = \sigma'(x) \wedge \\ \forall r : \mathcal{L}(r) = L \Rightarrow \delta(r) = \delta'(r) \end{array}$$

It is immediate that this definition of $\equiv$ gives rise to an equivalence relation. Intuitively $\equiv$ represents the view of a *passive attacker* as defined in [24], a principal that is able to observe the computations of a timed automaton and deduce information.

We will now define our security notion with the use of a bisimulation relation. Our notion shares some ideas from [19,21], where a bisimulation-based security is defined for a programming language with threads. In their approach, the bypassing of the security policy is localized on the actions, and that is because their attacker model is able to observe the low variables of a program at any of its computation steps (e.g. in a timed-automaton all of the nodes would have been observable). In contrast to [19,21], we localize bypassing of policies at the level of the nodes, while we also define a more flexible notion of security with respect to the attacker's observability.

Definition 1 *(*$Y-$Bisimulation*).* *For a timed automaton* $\mathsf{TA} = (\mathsf{Q}, \mathsf{E}, \mathsf{I}, q_\circ)$ *and a set of nodes* $Y = Y_s \cup Y_w$*, a relation* $\simeq_Y \subseteq (\boldsymbol{Config} \cup \{\bot\}) \times (\boldsymbol{Config} \cup \{\bot\})$ *will be called a* $Y-$*bisimulation relation if* $\simeq_Y$ *is symmetric and we have that if* $\gamma_1 = \langle q_1, \sigma_1, \delta_1\rangle \simeq_Y \langle q_2, \sigma_2, \delta_2\rangle = \gamma_2$ *then*

$$
\begin{aligned}
(\sigma_1, \delta_1) \equiv (\sigma_2, \delta_2) \Rightarrow \textit{if } \gamma_1 \overset{D_1}{\Longrightarrow}_Y \gamma_1' \textit{ then } \exists \gamma_2', D_2 : \\
\gamma_2 \overset{D_2}{\Longrightarrow}_Y \gamma_2' \wedge \gamma_1' \simeq_Y \gamma_2' \wedge \\
(\gamma_1' \neq \bot \wedge \gamma_2' \neq \bot) \Rightarrow ((\mathit{node}(\gamma_1') \in Y_w \wedge \mathit{node}(\gamma_2') \in Y_w) \vee \\
\mathit{pair}(\gamma_1') \equiv \mathit{pair}(\gamma_2'))
\end{aligned}
$$

where $\mathit{node}(\langle q, \sigma, \delta\rangle) = q$*,* $\mathit{pair}(\langle q, \sigma, \delta\rangle) = (\sigma, \delta)$*, and if* $\gamma_1 \simeq_Y \gamma_2$ *then*

$$(\gamma_1 = \bot \Leftrightarrow \gamma_2 = \bot)$$

We write $\sim_Y$ for the union of all the $Y-$bisimulations and it is immediate that this definition of $\sim_Y$ is both a $Y-$bisimulation and an equivalence relation. Intuitively, when two configurations are related in $\sim_Y$, and they are low equivalent then they produce distinguishable pairs of states only at the weakly observable nodes. Otherwise, observations made at strongly observable nodes should be still indistinguishable. In both cases, the resulting configurations of two $Y-$bisimilar configurations should also be $Y-$bisimilar. We are now ready to define our security notion.

Definition 2 *(*Security of Timed Automata*).* *For a timed automaton* $\mathsf{TA} = (\mathsf{Q}, \mathsf{E}, \mathsf{I}, q_\circ)$ *and a set* $Y = Y_s \cup Y_w$ *of observable nodes, we will say that* TA *satisfies the* information security policy $\mathcal{L}$ *whenever:*

$$
\begin{aligned}
&\forall q \in \{q_\circ\} \cup Y : \forall (\sigma, \delta), (\sigma', \delta') : \\
&\quad (\llbracket \mathsf{I}(q) \rrbracket(\sigma, \delta) \ \wedge \ \llbracket \mathsf{I}(q) \rrbracket(\sigma', \delta')) \Rightarrow \langle q, \sigma, \delta\rangle \sim_Y \langle q, \sigma', \delta'\rangle
\end{aligned}
$$

Whenever $Y_w = \emptyset$ our notion of security coincides with standard definitions of non-interference [29], where an automaton that satisfies the information security policy $\mathcal{L}$ does not leak any information about its high variables.

Example 3. For the smart grid automaton SG of the Example 1, we have the set of observable nodes $Y = \{2, 3, 4\}$, where the strongly observable ones are the nodes 2 and 4 ($Y_s = \{2, 4\}$), and the weakly one is the node 3 ($Y_w = \{3\}$), where the TTP is allowed to release the secret information of C.

4 Post-dominators

For the implicit flows arising from conditions, we are interested in finding their end points (nodes) that are the points where the control flow is not dependent on the conditions anymore. For that, we define a generalized version of the post-dominator relation and the immediate post-dominator relation [18].

Paths. A *path* π in a timed automaton $\mathsf{TA} = (\mathsf{Q}, \mathsf{E}, \mathsf{I}, q_\circ)$ is a finite $\pi = q_0 act_1 q_1 ... q_{n-1} act_n q_n$ $(n \geq 0)$ or infinite $\pi = q_0 act_1 q_1 ... q_{n-1} act_n q_n ...$ sequence of nodes and actions such that $\forall i > 0 : (q_{i-1}, act_i, q_i) \in \mathsf{E}$. We say that a path is *trivial* if $\pi = q_0$ and we say that a node q belongs to the path π, or π contains q, and we will write $q \in \pi$, if there exists some i such that $q_i = q$. For a finite path $\pi = q_0 act_1 q_1 ... q_{n-1} act_n q_n$ we write $\pi(i) = q_i act_{i+1} q_{i+1} ... q_{n-1} act_n q_n$ $(i \leq n)$ for the suffix of π that starts at the i-th position and we usually refer to it as the i-th suffix of π. Finally, for a node q and a set of nodes $Y \subseteq \mathsf{Q}$ we write

$$\mathit{\Pi}_{(q,Y)} = \{\pi \mid \pi = q_0 act_1 q_1 ... q_{n-1} act_n q_n : n > 0 \wedge q_0 = q \wedge q_n \in Y \wedge \forall i \in \{1, ..., n-1\} : q_i \notin Y\}$$

for the set of all the non-trivial finite paths that start at q, end at a node y in Y and all the intermediate nodes of the path do not belong in Y.

Definition 3 *(*Post-dominators*).* *For a node q and a set of nodes $Y \subseteq \mathsf{Q}$ we define the set*

$$pdom_Y(q) = \{q' \mid \forall \pi \in \mathit{\Pi}_{(q,Y)} : q' \in \pi(1)\}$$

and whenever $q' \in pdom_Y(q)$, we will say that q' is a Y post-dominator *of q.*

Intuitively whenever a node q' is a Y post-dominator of a node q it means that every non-trivial path that starts at q has to visit q' before it visits one of the nodes in Y. We write $\mathrm{pdom}_y(q)$ whenever $Y = \{y\}$ is a singleton and we have the following facts

Fact 1. *For a set of nodes $Y \subseteq \mathsf{Q}$ and for a node q we have that*

$$pdom_Y(q) = \bigcap_{y \in Y} pdom_y(q)$$

Fact 2. *The post-dominator set for a singleton set $\{y\}$ can be computed by finding the greatest solution of the following data-flow equations:*

$$\begin{array}{ll} pdom_y(q) = \mathsf{Q} & \text{if } \mathit{\Pi}_{(q,\{y\})} = \emptyset \\ pdom_y(q) = \{y\} & \text{if } y \in succ(q) \\ pdom_y(q) = \bigcap_{q' \in succ(q)} \left(\{q'\} \cup pdom_y(q')\right) & \text{otherwise} \end{array}$$

For a node q, we are interested in finding the Y post-dominator "closest" to it.

Definition 4. *For a node q and a set of nodes Y we definite the set*

$$ipdom_Y(q) = \{q' \in pdom_Y(q) \mid pdom_Y(q) = \{q'\} \vee q' \notin Y \wedge (\forall q'' \in pdom_Y(q) : q'' \neq q' \Rightarrow q'' \in pdom_Y(q'))\}$$

and a node $q' \in ipdom_Y(q)$ will be called an immediate Y post-dominator *of q.*

The following fact gives us a unique immediate Y post-dominator for the nodes that can reach Y ($\Pi_{(q,Y)} \neq \emptyset$). Intuitively this unique immediate Y post-dominator of a node q is the node that is the "closest" Y post-dominator of q, meaning that in any non-trivial path starting from q and ending in Y, the Y immediate post-dominator of q will always be visited first before any other Y post-dominator of q.

Fact 3. *For a set of nodes Y and a node q, whenever $\Pi_{(q,Y)} \neq \emptyset$ and $pdom_Y(q) \neq \emptyset$ then there exists node q' such that $ipdom_Y(q) = \{q'\}$.*

For simplicity, whenever a node q' is the unique immediate Y post-dominator of a node q and $\Pi_{(q,Y)} \neq \emptyset$ we shall write $\mathrm{ipd}_Y(q)$ for q' and we will say that *the unique immediate Y post-dominator of q is defined*. For any other case where q can either not reach Y ($\Pi_{(q,Y)} = \emptyset$) or $\mathrm{pdom}_Y(q) = \emptyset$ we will say that *the unique immediate post-dominator of q is not defined*.

Example 4. For the timed automaton SG and for the set of observable nodes $Y = \{2, 3, 4\}$, we have that $\mathrm{pdom}_Y(q) = \mathrm{ipd}_Y(q) = \{2\}$ for q being 1, 3 and 4 while $\mathrm{pdom}_Y(2) = \mathrm{ipd}_Y(2) = \emptyset$. Therefore for the nodes 1,3 and 4 their unique immediate Y post-dominator is defined and it is the node 2, while the unique immediate Y post-dominator of the node 2 is not defined.

5 Algorithm for Secure Information Flow

We develop an algorithm (Fig. 2) that traverses the graph of a timed automaton $\mathsf{TA} = (\mathsf{Q}, \mathsf{E}, \mathsf{I}, q_\circ)$ and imposes information flow constraints on the program variables and clocks of the automaton with respect to a security policy $\mathcal{L}$ and a Y post-dominator relation, where $Y = Y_s \cup Y_w$ is the set of observable nodes. Before we explain the algorithm we start by defining some auxiliary operators.

Auxiliary Operators. For an edge $(q_s, g \rightarrow \boldsymbol{x} := \boldsymbol{a}\colon \boldsymbol{r}, q_t) \in \mathsf{E}$ we define the auxiliary operator $\mathsf{ass}(.)$, $\mathsf{expr}(.)$ and $\mathsf{con}(.)$ as

$$\begin{aligned}
\mathsf{ass}((q_s, g \rightarrow \boldsymbol{x} := \boldsymbol{a}\colon \boldsymbol{r}, q_t)) &= \{\boldsymbol{x}, \boldsymbol{r}\} \\
\mathsf{expr}((q_s, g \rightarrow \boldsymbol{x} := \boldsymbol{a}\colon \boldsymbol{r}, q_t)) &= \{\boldsymbol{a}\} \\
\mathsf{con}((q_s, g \rightarrow \boldsymbol{x} := \boldsymbol{a}\colon \boldsymbol{r}, q_t)) &= \mathsf{I}(q_s) \wedge g \wedge \mathsf{I}(q_t)[\boldsymbol{a}/\boldsymbol{x}][\boldsymbol{0}/\boldsymbol{r}]
\end{aligned}$$

where $\mathsf{ass}(.)$ gives the modified variables and clocks of the assignment performed by TA using that edge, $\mathsf{expr}(.)$ gives the expressions used for the assignment, and the operator $\mathsf{con}(.)$ returns the condition that has to hold in order for the assignment to be performed. We finally lift the $\mathsf{ass}(.)$ operator to finite paths and thus for a finite path $\pi = q_0 act_1 q_1 ... q_{n-1} act_n q_n$ we define the auxiliary operators $\mathsf{Ass}(.)$ as

$$\mathsf{Ass}(q_0 act_1 q_1 ... q_{n-1} act_n q_n) = \bigcup_{i=1}^{n} \mathsf{ass}((q_{i-1}, act_i, q_i))$$

We write

$$\mathsf{Q}_{\rightsquigarrow \mathsf{w}} = \{q \mid \forall \pi = q..q' \in \Pi_{(q,Y)} : q' \in Y_w\}$$

C1. For all $q \in \mathsf{Q}_{\rightsquigarrow w}$:
(a) $\bigwedge_{e \in \mathsf{E}_q} \forall y \in \mathsf{fv}(\mathsf{con}(e)) : \mathcal{L}(y) = L \wedge (\Psi_e \vee A_e)$
C2. For all $q \in \mathsf{Q}^c_{\rightsquigarrow w}$ such that their unique immediate Y post-dominator is defined :
(a) $\bigwedge_{e \in \mathsf{E}_q} \bigcup_{\pi \in \Pi_{(e,\{\mathrm{ipd}_Y(q)\})}} \mathsf{fv}(\mathsf{con}(e)) \rightsquigarrow \mathsf{Ass}(\pi) \wedge A_e$
(b) $\bigwedge_{e \neq e' : e \in \mathsf{E}_q,\ e' \in \mathsf{E}_q,\ \underline{\mathrm{sat}}(\mathsf{con}(e) \wedge \mathsf{con}(e'))} \mathsf{fv}(\mathsf{con}(e)) \rightsquigarrow \bigcup_{\pi \in \Pi_{(e',\{\mathrm{ipd}_Y(q)\})}} \mathsf{Ass}(\pi)$
(c) $\Phi_q \Rightarrow \bigwedge_{e \in \mathsf{E}_q} \forall y \in \mathsf{fv}(\mathsf{con}(e)) : \mathcal{L}(y) = L$
C3. For all $q \in \mathsf{Q}^c_{\rightsquigarrow w}$ such that their unique immediate Y post-dominator is not defined :
(a) $\bigwedge_{e \in \mathsf{E}_q} \forall y \in \mathsf{fv}(\mathsf{con}(e)) : \mathcal{L}(y) = L \wedge$
(b) $((e \rightsquigarrow w \wedge \Psi_e) \vee A_e)$

Fig. 2. Security of $\mathsf{TA} = (\mathsf{Q}, \mathsf{E}, \mathsf{I}, q_\circ)$ with respect to $\mathcal{L}$ and the Y post-dominator relation

for the set of nodes, where whenever the automaton performs a successful observable step starting from a node $q \in \mathsf{Q}_{\rightsquigarrow \mathsf{w}}$ and ending in an observable node $q' \in Y$, then it is always the case that q' is weakly observable.

Condition **C1**. We start by looking at the nodes in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$. According to our security notion (Definition 2), for two low equivalent configurations at a node q, whenever the first one performs a successful (or unsuccessful) observable step that ends at a weakly observable node, then also the second should be able to perform an observable step that ends at a weakly observable node (or an unsuccessful one resp.). For that, the condition **C1** (a) first requires that the conditions of the outgoing edges in E_q where $\mathsf{E}_q = \{(q, act, q') \mid (q, act, q') \in \mathsf{E}\}$ contain only low variables. However, this is not enough.

To explain the rest of the constraints imposed by the condition **C1** (a) consider the automaton (**a**) of Fig. 3, where the node 3 is weakly observable, h and l is a high and a low variable respectively, and all the invariants of the nodes are set to tt. This automaton is not secure with respect to Definition 2. To see this, we have $([\mathsf{l} \mapsto 0, \mathsf{h} \mapsto 1], \delta) \equiv ([\mathsf{l} \mapsto 0, h \mapsto 0], \delta)$ (for some clock state δ) but the pair $([\mathsf{l} \mapsto 0, \mathsf{h} \mapsto 1], \delta)$ always produces $\perp$ since we will have an infinite loop at the node 2, while $([\mathsf{l} \mapsto 0, \mathsf{h} \mapsto 0], \delta)$ always terminates at the node 3. That is because even if both edges of the node 2 contain only the low variable l in their condition, the assignment $\mathsf{l}{:=}\mathsf{h}$ bypasses the policy $\mathcal{L}$ and thus, right after it, the two pairs stop being low equivalent.

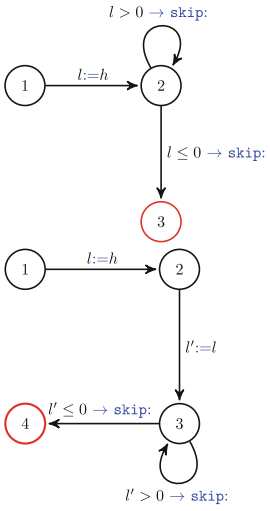


Fig. 3. Example automata **(a)** (top) and **(b)** (bottom)

As another example, consider the automaton (**b**) of Fig. 3. Here the node 4 is weakly observable, h is a high variable, l, l' are two low variables and all the invariants of nodes are set to tt again. We have $([\mathsf{l} \mapsto 0, \mathsf{l}' \mapsto 0, \mathsf{h} \mapsto 1], \delta) \equiv ([\mathsf{l} \mapsto 0, \mathsf{l}' \mapsto 0, \mathsf{h} \mapsto 0], \delta)$ (for some clock state δ)

and again the first pair produces $\perp$ by looping at the node 3, whereas the second pair always terminates. Here even if the variable l is not used in any condition after the assignment $\mathsf{l}{:=}\mathsf{h}$, it influences the value of l' and consequently, since l' appears on the condition of the edges of the node 3 we get this behavior.

To cater for such cases, for an edge $e = (q_s, g \rightarrow \boldsymbol{x} := \boldsymbol{a}\colon \boldsymbol{r}, q_t)$ we first define the predicate

$$A_e = \bigwedge_i \mathsf{fv}(a_i) \rightsquigarrow \{x_i\}$$

that takes care of the explicit flows arising from the assignments. We then define

$$\Pi_{(e,Y)} = \{\pi \mid e = (q_0, act_1, q_1) : \pi = q_0 act_1 q_1 ... q_{n-1} act_n q_n \in \Pi_{(q_0,Y)}\}$$

to be set of paths (the ones defined in Sect. 4) that start with e and end in Y, and all the intermediate nodes do not belong to Y. Finally, whenever an assignment bypasses the security policy $\mathcal{L}$ due to an explicit flow and thus A_e is false, we then impose the predicate

$$\begin{aligned}\Psi_e = \forall \pi \in \Pi_{(e,Y)} : \forall q' \in \pi(1) :\\ q' \notin Y \Rightarrow (\forall e' \in E_{q'} : (\mathsf{ass}(e) \setminus R) \cap (\mathsf{fv}(\mathsf{con}(e')) \cup \mathsf{fv}(\mathsf{expr}(e'))) = \emptyset)\end{aligned}$$

The predicate Ψ_e demands that the assigned program variables of $e = (q_s, act, q_t)$ cannot be used in any expression or condition that appears in a path that starts with q_t and goes to an observable node. Note here that even if Ψ_e quantifies over a possibly infinite set of paths ($\Pi_{(e,Y)}$), it can be computed in *finite time* by only looking at the paths where each cycle occurs at most once.

We will now look at the nodes where the automaton may perform a successful observable step that ends in a strongly observable node. Those nodes are described by the set $\mathsf{Q}^c_{\rightsquigarrow w} = \mathsf{Q} \setminus \mathsf{Q}_{\rightsquigarrow \mathsf{w}}$, that is the complement of $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$.

Condition **C2**. For a node q in $\mathsf{Q}^c_{\rightsquigarrow w}$, whose immediate Y post-dominator is defined, condition **C2** (a) takes care of the explicit and the implicit flows generated by the assignment and the control dependencies respectively, arising from the edges of q. Note here that we do not propagate the implicit flows any further after $\mathrm{ipd}_Y(q)$. This is because $\mathrm{ipd}_Y(q)$ is the point where all the branches of q are joining and any further computation is not control-dependent on them anymore. Those constraints are along the line of Denning's approach [10] of the so-called *block-labels*.

To understand condition **C2** (b) consider the automaton (**c**) of Fig. 4, where h and l is a high and a low variable respectively, the node 2 is strongly observable, and both nodes 1 and 2 have their invariant set to $\mathtt{tt}$. Next take $([\mathsf{l} \mapsto 0, \mathsf{h} \mapsto 1], \delta) \equiv ([\mathsf{l} \mapsto 0, \mathsf{h} \mapsto 0], \delta)$ (for some clock state δ) and note that the first pair can result in a configuration in 2 with $([l \mapsto 0, h \mapsto 1], \delta)$ (taking the top branch) while the second pair always ends in 2 with $[l \mapsto 1, h \mapsto 0]$. Therefore this automaton is not secure with respect to our Definition 2.

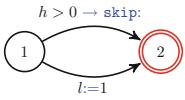


Fig. 4. Example automaton (**c**)

To take care of such behaviours we write $\underline{\mathsf{sat}}(\cdots)$ to express the satisfiability of the $\cdots$ formula. Whenever there are two branches (induced by the edges e and e' both leaving q) that are not mutually exclusive (that is, where $\underline{\mathsf{sat}}(\mathsf{con}(e) \wedge \mathsf{con}(e'))$ we make sure to record the information flow arising from *bypassing* the branch that would otherwise perform an assignment. This is essential for dealing with *non-determinism*.

Fact 4. *For a timed automaton* $\mathsf{TA} = (\mathsf{Q}, \mathsf{E}, \mathsf{I}, q_\circ)$*, we have that if*

$$\langle q, \sigma, \delta \rangle \overset{D}{\Longrightarrow}_{\{q'\}} \langle q', \sigma', \delta' \rangle$$

then

$$\{x \mid \sigma(x) \neq \sigma'(x)\} \cup \{r \mid \delta'(r) \neq \delta(r) + D\} \subseteq \bigcup_{\pi \in \Pi_{(e,\{q'\})}} \mathsf{Ass}(\pi)$$

where e *corresponds to the initial edge of this observable step.*

Condition **C2** (c) takes care of cases where a *timing/termination* side channel [2] could have occurred.

As an example of such a case consider the automaton (**d**) of Fig. 5, where h and t is a high program variable and a low clock respectively, node 2 is strongly observable and both 1 and 2 have their invariant set to tt. Next, for $([\mathsf{h} \mapsto 1], [\mathsf{t} \mapsto 0]) \equiv ([\mathsf{h} \mapsto 0], [\mathsf{t} \mapsto 0])$ we have that the first pair always delays at least 30 units and ends in 2 with a clock state that has $t > 30$, whereas the second pair can go to 2, taking the lower branch immediately without any delay, and thus the resulting pairs will not be low equivalent. To take care of such behaviours, we stipulate a predicate Φ_q such that

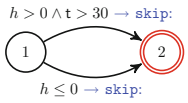


Fig. 5. Example automaton (**d**)

$$\exists t_1, t_2 \in \bigcup_{(\sigma,\delta):[\![\mathsf{I}(q)]\!](\sigma,\delta)} [\![\mathsf{TA} : q \mapsto \mathrm{ipd}_Y(q)]\!](\sigma, \delta) : \Delta(t_1) \neq \Delta(t_2)$$
$$\Downarrow$$
$$\Phi_q$$

Using this predicate we demand that whenever the TA does not have a "constant" *termination behavior* from the node q to the node $\mathrm{ipd}_Y(q)$, then variables that influence the termination behavior should not be of high security level.

Condition **C3**. We are now left with the nodes in $\mathsf{Q}^c_{\rightsquigarrow w}$, whose immediate Y post-dominator is not defined. Since for such a node q, we cannot find a point (the unique immediate Y post-dominator) where the control dependencies from the branches of q end, condition **C3** (a) requires that the conditions of the edges of q should not be dependent on high security variables.

Condition **C3** (b) caters for the explicit flows, of an edge e using the predicate A_e. However we are allowed to dispense A_e, whenever further computations after taking the edge e may lead only to weakly observable nodes and Ψ_e holds. To express this for an edge $e = (q_s, g \rightarrow \boldsymbol{x} := \boldsymbol{a}\colon \boldsymbol{r}, q_t)$ we write

$$e \rightsquigarrow w$$

whenever $q_t \in Y_w$ or $q_t \in \mathsf{Q}_{\rightsquigarrow \mathsf{w}}$.

Example 5. Consider now the automaton SG of Example 1, and the Y post-dominator relation of Example 4.

We have that the nodes 1, 3 and 4 are in $\mathsf{Q}^{c}_{\rightsquigarrow w}$ and also that their immediate unique Y post-dominator is defined. Condition **C2** (a) and **C2** (b) impose the following constraints

$$\{\mathsf{T},\mathsf{t},\mathsf{r}\} \rightsquigarrow \{\mathsf{ed},\mathsf{e_i},\mathsf{c_i},\mathsf{s},\mathsf{t},\mathsf{r}\},\{\mathsf{ed},\mathsf{e_i}\} \rightsquigarrow \{\mathsf{e_i}\},\{\mathsf{e_i}\} \rightsquigarrow \{\mathsf{c_i}\},\{\mathsf{v_i}\} \rightsquigarrow \{\mathsf{p_i}\} \quad (i=1,2)$$
$$\{\mathsf{T}\} \rightsquigarrow \{\mathsf{p_1},\mathsf{p_2},\mathsf{a},\mathsf{f}\},\{\mathsf{z}\} \rightsquigarrow \{\mathsf{a}\},\{\} \rightsquigarrow \{\mathsf{s},\mathsf{f},\mathsf{a}\}$$

Finally, for the node 1, because Φ_1 (**C2** (c)) all the clocks need to be of low security level.

Next, the node 2 is in $\mathsf{Q}^{c}_{\rightsquigarrow w}$ and since its unique immediate Y post-dominator is not defined, condition **C3** (b) impose the constraints

$$\{\mathsf{p_1},\mathsf{p_2},\mathsf{c_1},\mathsf{c_2}\} \rightsquigarrow \{\mathsf{b}\},\{\mathsf{a}\} \rightsquigarrow \{\mathsf{x}\},\{\} \rightsquigarrow \{\mathsf{e_1},\mathsf{e_2},\mathsf{f}\}$$

and condition **C3** (a) imposes that T, s and f should be of low security level. Notice here that since for the edge $e = (2, \mathsf{s}=1 \rightarrow \mathsf{y_1},\mathsf{y_2}{:=}\mathsf{c_1},\mathsf{c_2}{:}\ ,3)$ that releases the sensitive information of C we have that $e \rightsquigarrow w$, we are not imposing the constraint $\{\mathsf{c_i}\} \rightsquigarrow \{\mathsf{y_i}\}$ $(i=(1,2))$. Those constraints are easy to verify for the security assignment of Example 2.

Now if we were to change the node 3 from being a weakly observable to a strongly observable node, the automaton SG will not be secure with respect to Definition 2. In that case our algorithm will reject it, since for the edge e we would have that $e \not\rightsquigarrow w$ and the predicate A_e would have resulted in false.

Finally, we shall write $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$ whenever the constraints arising from our algorithm (Fig. 2) are satisfied and thus we have the following lemmas

Lemma 1. *For a timed automaton* $\mathsf{TA} = (\mathsf{Q},\mathsf{E},\mathsf{I},q_\circ)$*, if* $sec_{Y,\mathcal{L}}(\mathsf{TA})$ *then for* (σ_1,δ_1), (σ_2,δ_2) *such that* $[\![\mathsf{I}(q)]\!](\sigma_1,\delta_1)$ *and* $[\![\mathsf{I}(q)]\!](\sigma_2,\delta_2)$ *and* $(\sigma_1,\delta_1) \equiv (\sigma_2,\delta_2)$ *we have that*

$$\textit{if } \langle q,\sigma_1,\delta_1\rangle \overset{D_1}{\Longrightarrow}_Y \langle q',\sigma_1',\delta_1'\rangle \textit{ then } \exists(\sigma_2',\delta_2'), D_2 : \langle q,\sigma_2,\delta_2\rangle \overset{D_2}{\Longrightarrow}_Y \langle q',\sigma_2',\delta_2'\rangle \wedge (q' \in Y_w \vee (\sigma_1',\delta_1') \equiv (\sigma_2',\delta_2'))$$

Lemma 2. *For a timed automaton* $\mathsf{TA} = (\mathsf{Q},\mathsf{E},\mathsf{I},q_\circ)$*, if* $sec_{Y,\mathcal{L}}(\mathsf{TA})$ *then for* (σ_1,δ_1), (σ_2,δ_2) *such that* $[\![\mathsf{I}(q)]\!](\sigma_1,\delta_1)$ *and* $[\![\mathsf{I}(q)]\!](\sigma_2,\delta_2)$ *and* $(\sigma_1,\delta_1) \equiv (\sigma_2,\delta_2)$ *we have that*

$$\textit{if } \langle q,\sigma_1,\delta_1\rangle \overset{\infty}{\Longrightarrow}_Y \bot \textit{ then also } \langle q,\sigma_2,\delta_2\rangle \overset{\infty}{\Longrightarrow}_Y \bot$$

The following theorem concludes the two lemmas from above to establish the soundness of our algorithm with respect to the notion of security of Definition 2.

Theorem 1. *For a timed automaton* $\mathsf{TA} = (\mathsf{Q},\mathsf{E},\mathsf{I},q_\circ)$*, if* $sec_{Y,\mathcal{L}}(\mathsf{TA})$ *then* TA *satisfies the information security policy* $\mathcal{L}$*.*

6 Conclusion

We have shown how to successfully enforce *Information Flow Control* policies on timed automata. This has facilitated developing an algorithm that prevents unnecessary *label creep* and that deals with *non-determinism*, *non-termination*, and *continuous real-time*. The algorithm has been proved sound by means of a bisimulation result, that allows controlled information leakage.

We are exploring how to automate the analysis and in particular how to implement (a sound approximation of) the Φ_q predicate. There has been a lot of research [3,4] done for determining the maximum (max_t) or minimum (min_t) execution time that an automaton needs to move from a location q_s to a location q_t. One possibility is to make use of this work [3,4] and thus the predicate Φ_q would amount to checking if the execution time between the two nodes of interest (q and $\text{ipd}_Y(q)$) is constant (e.g. $max_t = min_t$).

A longer-term goal is to allow policies to simultaneously deal with safety and security properties of cyberphysical systems.

Appendix

Proposition 1

Assume that all the traces in $[\![\mathsf{TA} : q_s \mapsto q_t]\!](\sigma, \delta)$ are successful and we want to show that there exists $t \in [\![\mathsf{TA} : q_s \mapsto q_t]\!](\sigma, \delta)$ with a maximal length m.

We use results from model-checking for timed automata [15]. As in [15] we first transform our automaton to an equivalent diagonal-free automaton, that is an automaton where clocks appearing in its guards and invariants can be compared only to integers (e.g. $r_1 - r_2 \leq 5$ is not allowed). We then define the region graph $RG(\mathsf{TA})$ of TA, that is a finite graph where nodes of the region graph are of the form (q, reg) where reg is a clock region, that is an equivalence class defined on the clock states (for details we refer to [15]). Configurations of $RG(\mathsf{TA})$ are of the form $\langle (q, reg), \sigma \rangle$ and we have that $\langle (q, reg), \sigma \rangle \Longrightarrow \langle (q', reg'), \sigma' \rangle$ if there are $\delta \in reg$, $\delta' \in reg'$, $d \geq 0$, σ' such that the automaton TA performs the transition $\langle q, \sigma, \delta \rangle \xrightarrow{d} \langle q', \sigma', \delta' \rangle$. Lemma 1 of [15] then states that each abstract run (finite or infinite) in the region graph $RG(\mathsf{TA})$ can be instantiated by a run (finite or infinite resp.) in TA and vice verca. This is based on the property of the region graph of being *pre-stable* that is that $\langle (q, reg), \sigma \rangle \Longrightarrow \langle (q', reg'), \sigma' \rangle$ if $\forall \delta \in reg$ there are $\delta' \in reg'$, $d \geq 0$, σ' such that $\langle q, \sigma, \delta \rangle \xrightarrow{d} \langle q', \sigma', \delta' \rangle$. Therefore the computation tree T of $\langle q, \sigma, \delta \rangle$ in TA has the same depth as the computation tree T' of $\langle (q, [\delta]), \sigma \rangle$ in $RG(\mathsf{TA})$ where $[\delta]$ is the region that contains all the clock states that are equivalent to δ. We then recall König's infinity lemma as it applies to trees – that every tree who has infinitely-many vertices but is locally finite (each vertex has finitely-many successor vertices), has at least one infinite path [11]. It is immediate that T' is a locally finite tree. Now if T' is infinite then by König's infinity lemma we have that T' has an infinite path and thus using Lemma 1 of [15] we have also that T has an infinite path that corresponds

to a trace $\langle q, \sigma, \delta \rangle$ in TA which contradicts our assumptions that all the traces of $\langle q, \sigma, \delta \rangle$ are finite. Therefore we can conclude that T' has a finite depth and therefore also T and that they are equal to the number m.

Proof of Fact 2

Proof. The first equation is straightforward by the definition of the post-dominator relation. For the second one, that is when y is a successor (an immediate one) of q then the only post-dominators of q is the node y, since there exists a non-trivial path $\pi = q\,act\,y \in \Pi_{(q,y)}$ (for some action act) such that the trivial path $\pi(1) = y$ contains only y, and therefore for any other path $\pi' \in \Pi_{(q,y)}$ in which a node q' different from y is contained in $\pi'(1)$, q' can not be a post-dominator of q since it is not contained in the trivial path $\pi(1)$. To understand the last equation notice that if a node q'' post-dominates all of the successors of q or it is a successor of q that post-dominates all the other successors of q then all the non-trivial paths from q to y will always visit q'' and thus $q'' \in \mathrm{pdom}_y(q)$; similarly if $q'' \notin \bigcap_{q' \in \mathrm{succ}(q)} \left(\{q'\} \cup \mathrm{pdom}_y(q')\right)$ then there exists a successor of q, $q' \neq q''$ such that q'' does not post-dominate q' and thus we can find a non-trivial path $\pi \in \Pi_{(q,Y)}$ that starts with $q\,act\,q'$ (for some action act) and does not contain q'' and thus q'' is not a post-dominator of q.

Proof of Fact 3

Proof. To prove that $\mathrm{ipdom}_Y(q)$ is singleton we consider two cases. In the case that $\mathrm{pdom}_Y(q) = \{q'\}$ then the proof is trivial.

Assume now that $\mathrm{pdom}_Y(q) = \{q_1, ..., q_n\}$ $(n \geq 2)$ and take an arbitrary non-trivial path $\pi \in \Pi_{(q,Y)}$ and find the closest to q (the one that appears first in the path) Y post-dominator $q_j \in \mathrm{pdom}_Y(q)$ in that path. Next note that $q_j \notin Y$ since if $q_j \in Y$, we could shorten that path to the point that we meet q_j for the first time and thus we have found a non trivial path $\pi' \in \Pi_{(q,Y)}$ (since $q_j \in Y$) in which $\forall i \neq j : q_i \notin \pi'(1)$ and thus $\forall i \neq j : q_i \notin \mathrm{pdom}_Y(q)$ which contradicts our assumption. Next to prove that $\forall i \neq j : q_i \in \mathrm{pdom}_Y(q_j)$ assume that this is not the case and thus we can find $q_l \neq q_j : q_l \notin \mathrm{pdom}_Y(q_j)$. Therefore we can find a path $\pi'' \in \Pi_{(q_j,Y)}$ such that $q_l \notin \pi''(1)$, but this means that if we concatenate the paths π' and π'' we have a path in $\Pi_{(q,Y)}$ in which q_l does not belong to it and thus q_l does not belong in its 1-suffix either and therefore $q_l \notin \mathrm{pdom}_Y(q)$, which again contradicts our assumption.

Finally to prove that $\mathrm{ipdom}_Y(q)$ is singleton assume that there exists another Y post-dominator of q, q_l such that $q_l \neq q_j$ and $q_l \notin Y$ and $q_j \in \mathrm{pdom}(q_l)$. Then this means that q_j belongs in all the 1-suffixes of the paths in the set $\Pi_{(q_l,Y)}$. Therefore take $\pi = q_l ... q_j ... y \in \Pi_{(q_l,Y)}$ (for some $y \in Y$) such that π contains no cycles (e.g. each node occurs exactly once in the path) but then there exists a path $\pi' = q_j ... y$ (the suffix of the path π) such that $q_l \notin \pi'$ and thus $q_l \notin \mathrm{pdom}_Y(q_j)$ which contradicts our assumption. Therefore we have proved that q_j is the unique immediate Y post-dominator of q.

Proof of Lemma 1

Proof. Assume that $\langle q, \sigma_1, \delta_1\rangle \overset{D_1}{\Longrightarrow}_Y \langle q', \sigma_1', \delta_1'\rangle$ because of the trace

$$\langle q, \sigma_1, \delta_1\rangle = \langle q, \sigma_{01}, \delta_{01}\rangle \xrightarrow{d_1} ... \xrightarrow{d_k} \langle q_{k1}, \sigma_{k1}, \delta_{k1}\rangle = \langle q', \sigma_1', \delta_1'\rangle \quad (*)$$

where $k > 0$ and $\forall i \in \{1, .., k-1\} : q_{i1} \notin Y$ and $D_1 = \sum_{j=1}^{k} d_j$ and the first transition of the trace has happened because of the edge $e \in \mathsf{E}_q$.

We shall consider two main cases. The one where q is in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$ and one where it is not.

Main Case 1: q is in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$. In that case $q' \in Y_w$ and thus we only have to prove that (σ_2, δ_2) can reach q'. We start by proving a small fact.

First for a set of variables and clocks $\mathcal{Z}$, and two pairs (σ, δ), (σ', δ') we write

$$(\sigma, \delta) \equiv^{\mathcal{Z}} (\sigma', \delta') \quad \text{iff} \quad \begin{array}{l} \forall x : (x \in \mathcal{Z} \wedge \mathcal{L}(x) = L) \Rightarrow \sigma(x) = \sigma'(x) \wedge \\ \forall r : (r \in \mathcal{Z} \wedge \mathcal{L}(r) = L) \Rightarrow \delta(r) = \delta'(r) \end{array}$$

Next, for a finite path $\pi = q_0 act_1 q_1 ... q_{n-1} act_n q_n$ we define the auxiliary operator $\mathcal{Z}(.)$ as $\mathcal{Z}(\pi) = \bigcup_{i=0}^{n-1} (\bigcup_{e' \in \mathsf{E}_{q_i}} \mathsf{fv}(\mathsf{con}(e')) \cup \mathsf{fv}(\mathsf{expr}(e')))$.

Now we will prove that for a path $\pi = q'_{01} act'_1 q'_{11} ... q'_{(n-1)1} act'_n q'_n \in \Pi_{(e,Y)}$, if

$$\langle q, \sigma_1, \delta_1\rangle = \langle q'_{01}, \sigma'_{01}, \delta'_{01}\rangle \xrightarrow{d'_1} ... \xrightarrow{d'_l} \langle q'_{l1}, \sigma'_{l1}, \delta'_{l1}\rangle \quad (l \leq n) \tag{1}$$

using the edges $(q'_{01}, act'_1, q'_{11}), ..., (q'_{(l-1)1}, act'_l, q'_l)$ and $(\sigma_1, \delta_1) \equiv^{\mathcal{Z}(\pi)} (\sigma_2, \delta_2)$ then $\exists (\sigma'_{l2}, \delta'_{l2})$:

$$\langle q, \sigma_2, \delta_2\rangle = \langle q'_{01}, \sigma'_{02}, \delta'_{02}\rangle \xrightarrow{d'_1} ... \xrightarrow{d'_l} \langle q'_{l1}, \sigma'_{l2}, \delta'_{l2}\rangle \quad (a)$$

and

$$l < n \Rightarrow (\sigma'_{l1}, \delta'_{l1}) \equiv^{\mathcal{Z}(\pi(l))} (\sigma'_{l2}, \delta'_{l2}) \qquad (b)$$

where recall that $\pi(l)$ is the $l-$suffix of π. The proof proceeds by induction on l.

Base Case $l = 1$. To prove (a), let $e = (q'_{01}, g \rightarrow \boldsymbol{x} := \boldsymbol{a}\colon \boldsymbol{r}, q'_{11})$ and note that because $(\sigma_1, \delta_1) \equiv^{\mathcal{Z}(\pi)} (\sigma_2, \delta_2)$ and $\mathsf{con}(e)$ contains only low variables (since $q'_{01} = q \in \mathsf{Q}_{\rightsquigarrow \mathsf{w}}$ and **C1** (a)) it is immediate that there exists $\sigma'_{12} = \sigma_2[\boldsymbol{x} \mapsto [\![\boldsymbol{a}]\!]\sigma_2]$, $\delta'_{12} = (\delta_2 + d'_1)[\boldsymbol{r} \mapsto \boldsymbol{0}]$ such that $[\![\mathsf{I}(q'_{01})]\!](\sigma_2, \delta_2 + d'_1) = \mathsf{tt}$ and $[\![\mathsf{I}(q'_{11})]\!](\sigma'_{12}, \delta'_{12}) = \mathsf{tt}$, and $\langle q'_{01}, \sigma_2, \delta_2\rangle \xrightarrow{d'_1} \langle q'_{11}, \sigma'_{12}, \delta'_{12}\rangle$.

Now if $l < n$, to prove (b) we consider two cases. One where A_e is true and one where it is false. If A_e is true we note that $(\sigma'_{11}, \delta'_{11}) \equiv^{\mathcal{Z}(\pi)} (\sigma'_{12}, \delta'_{12})$, and then it is immediate that also $(\sigma'_{11}, \delta'_{11}) \equiv^{\mathcal{Z}(\pi(1))} (\sigma'_{12}, \delta'_{12})$ as required. Otherwise, if A_e is false then Ψ_e is true and thus $(\sigma'_{11}, \delta'_{11}) \equiv^{\mathcal{Z}(\pi(1))} (\sigma'_{12}, \delta'_{12})$, because the two pairs are still low equivalent for the variables that are not used in the assignment of e, while the ones used in the assignment of e they do not appear in any condition (or expression) of an edge of a node q that belongs in $\pi(1)$.

Inductive Case $l = l_0 + 1$ $(l_0 > 0)$. Because of the trace in (1) we have that $t_1 = \langle q'_{01}, \sigma'_{01}, \delta'_{01}\rangle \xrightarrow{d'_1} \langle q'_{11}, \sigma'_{11}, \delta'_{11}\rangle$ and $t_2 = \langle q'_{11}, \sigma'_{11}, \delta'_{11}\rangle \xrightarrow{d'_2} ... \xrightarrow{d'_l} \langle q'_{l1}, \sigma'_{l1}, \delta'_{l1}\rangle$.

Using our induction hypothesis on t_1 we have that there exists $(\sigma'_{12}, \delta'_{12})$ such that $\langle q'_{01}, \sigma_2, \delta_2\rangle \xrightarrow{d'_1} \langle q'_{11}, \sigma'_{12}, \delta'_{12}\rangle$ and $(\sigma'_{11}, \delta'_{11}) \equiv^{\mathcal{Z}(\pi(1))} (\sigma'_{12}, \delta'_{12})$ and the proof is completed using our induction hypothesis on t_2. The proof of *Main Case 1* follows by the result (a) of the fact from above, taking the path π that corresponds to the trace $(*)$ and using that $(\sigma_1, \delta_1) \equiv^{\mathcal{Z}(\pi)} (\sigma_2, \delta_2)$ (since $(\sigma_1, \delta_1) \equiv (\sigma_2, \delta_2)$ and all the nodes in π except q_{k1} have edges whose conditions contain only low variables). Therefore, since (σ_1, δ_1) creates the trace (*) we also have that $\exists(\sigma'_2, \delta'_2)$:

$$\langle q, \sigma_2, \delta_2\rangle = \langle q_{01}, \sigma_{02}, \delta_{02}\rangle \xrightarrow{d_1} ... \xrightarrow{d_k} \langle q_{k1}, \sigma_{k2}, \delta_{k2}\rangle = \langle q', \sigma'_2, \delta'_2\rangle$$

and thus for $D_2 = d_1 + ... + d_k$ we have that

$$\langle q, \sigma_2, \delta_2\rangle \overset{D_2}{\Longrightarrow}_Y \langle q', \sigma'_2, \delta'_2\rangle$$

where $q' \in Y_w$ and this completes the proof for this case.

Main Case 2: When q is not in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$. The proof proceeds by induction on the length k of the trace $(*)$.

Base Case ($k = 1$). We have that

$$\langle q, \sigma_1, \delta_1\rangle \xrightarrow{d_1} \langle q', \sigma'_1, \delta'_1\rangle$$

and let $e = (q, g \rightarrow \boldsymbol{x} := \boldsymbol{a}\colon \boldsymbol{r}, q')$, then it is immediate that $D_1 = d_1$, $\sigma'_1 = \sigma_1[\boldsymbol{x} \mapsto [\![\boldsymbol{a}]\!]\sigma_1]$, $\delta'_1 = (\delta_1 + d_1)[\boldsymbol{r} \mapsto \mathbf{0}]$ and $[\![\mathsf{I}(q)]\!](\sigma_1, \delta_1 + d_1) = \mathsf{tt}$ and $[\![\mathsf{I}(q')]\!](\sigma'_1, \delta'_1) = \mathsf{tt}$.

We shall consider two subcases one where the unique immediate Y post-dominator of q is defined and one where it is not.

Subcase 1: When the unique immediate Y post-dominator $\mathrm{ipd}_Y(q)$ *is defined.* It has to be the case then that $q' = \mathrm{ipd}_Y(q)$ since $q' \in Y$ and in particular, we have that $q' \in Y_s$. We will proceed by considering two other subcases of the *Subcase 1*, one where the condition Φ_q is *true* and one which it is *false*.

Subcase 1(a): When Φ_q is true. Then it is the case that all the variables of the condition $\mathsf{con}(e)$ are low and thus it is immediate that there exists $d_2 = d_1$ and $\sigma'_2 = \sigma_2[\boldsymbol{x} \mapsto [\![\boldsymbol{a}]\!]\sigma_2]$, $\delta'_2 = (\delta_2 + d_2)[\boldsymbol{r} \mapsto \mathbf{0}]$ and $[\![\mathsf{I}(q)]\!](\sigma_2, \delta_2 + d_2) = \mathsf{tt}$ and $[\![\mathsf{I}(q')]\!](\sigma'_2, \delta'_2) = \mathsf{tt}$ such that $\langle q, \sigma_2, \delta_2\rangle \xrightarrow{d_2} \langle q', \sigma'_2, \delta'_2\rangle$ which implies that for $D_2 = d_2$

$$\langle q, \sigma_2, \delta_2\rangle \overset{D_2}{\Longrightarrow}_Y \langle q', \sigma'_2, \delta'_2\rangle$$

Finally, because $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, condition **C2** (a) gives us that A_e is true, and thus all the *explicit flows* arising from the assignments $\boldsymbol{x} := \boldsymbol{a}$ are permissible and thus $(\sigma'_1, \delta'_1) \equiv (\sigma'_2, \delta'_2)$ as required.

Subcase 1(b): When Φ_q is false. If it is the case that all the variables in the condition $\mathsf{con}(e)$ are low then the proof proceeds as in *Subcase 1(a)*.

For the case now that at least one variable in the condition $\mathsf{con}(e)$ is high then because $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, condition **C2** (a) and Fact 4 ensure that $\forall x : \mathcal{L}(x) = L \Rightarrow \sigma_1'(x) = \sigma_1(x)$ and $\forall r : \mathcal{L}(r) = L \Rightarrow \delta_1'(r) = \delta_1(r) + d_1$. Since Φ_q is false (σ_1, δ_1) and (σ_2, δ_2) have the same *termination behaviour* and thus there exists $d_2 = d_1$ and (σ_2', δ_2') such that $\langle q, \sigma_2, \delta_2\rangle \xrightarrow{d_2} \langle q', \sigma_2', \delta_2'\rangle$ and therefore for $D_2 = d_2$ we have that

$$\langle q, \sigma_2, \delta_2\rangle \overset{D_2}{\Longrightarrow}_Y \langle q', \sigma_2', \delta_2'\rangle$$

We just showed that $(\sigma_1', \delta_1') \equiv (\sigma_1, \delta_1 + d_1) \equiv (\sigma_2, \delta_2 + d_2)$ and we will now show that $(\sigma_2', \delta_2') \equiv (\sigma_2, \delta_2 + d_2)$.

Now if

$$\langle q, \sigma_2, \delta_2\rangle \xrightarrow{d_2} \langle q', \sigma_2', \delta_2'\rangle$$

using the edge e or an edge $e' \neq e$ such that $\mathsf{con}(e')$ contains a high variable, since $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, condition **C2** (a) and Fact 4 gives that $\forall x : \mathcal{L}(x) = L \Rightarrow \sigma_2'(x) = \sigma_2(x)$ and $\forall r : \mathcal{L}(r) = L \Rightarrow \delta_2'(r) = \delta_2(r) + d_2$ and therefore $(\sigma_2', \delta_2') \equiv (\sigma_2, \delta_2 + d_2)$ as required. If now $\mathsf{con}(e')$ contains only low variables, (σ_1, δ_1) is a witness of $\underline{\mathrm{sat}}(\mathsf{con}(e) \wedge \mathsf{con}(e'))$ and therefore because $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, using the condition **C2** (b) and Fact 4 we work as before and we obtain that $(\sigma_2', \delta_2') \equiv (\sigma_2, \delta_2 + d_2)$.

Subcase 2: When the unique immediate Y post-dominator of q is not defined. In that case, all the variables in $\mathsf{con}(e)$ are low. If q' is in Y_w we have that $e \rightsquigarrow w$ and we proceed as in *Main Case 1*. Otherwise, we proceed as in *Subcase 1(a)*.

This completes the case for $k = 1$.

Inductive Case $(k = k_0 + 1)$. We have that

$$\langle q, \sigma_1, \delta_1\rangle = \langle q, \sigma_{01}, \delta_{01}\rangle \xrightarrow{d_1} \ldots \xrightarrow{d_k} \langle q_{k1}, \sigma_{k1}, \delta_{k1}\rangle = \langle q', \sigma_1', \delta_1'\rangle$$

and recall that the first transition happened because of the edge e and that q is not in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$.

We shall consider two cases again, one where the unique immediate Y post-dominator of q is defined and one where it is not.

Subcase 1: When the unique immediate-post dominator $\mathrm{ipd}_Y(q)$ *is defined.* We will proceed by considering two subcases of *Subcase 1*, one where Φ_q is *true* and one where Φ_q is *false*.

Subcase 1(a): When Φ_q is true. Since Φ_q is true we have that all the variables in $\mathsf{con}(e)$ are low and thus $\exists d_1' = d_1$ and $(\sigma_{12}, \delta_{12}) \equiv (\sigma_{11}, \delta_{11})$ (this is ensured by our assumptions that $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$ and the predicate A_e of the condition **C2** (a) that takes care of the *explicit flows* arising from the assignment in the edge e) such that

$$\langle q, \sigma_2, \delta_2\rangle = \langle q_{01}, \sigma_{02}, \delta_{02}\rangle \xrightarrow{d_1'} \langle q_{11}, \sigma_{12}, \delta_{12}\rangle \tag{1}$$

Since q is not in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$, note that it is also the case that q_{11} is not in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$ and thus using that $(\sigma_{12}, \delta_{12}) \equiv (\sigma_{11}, \delta_{11})$ and our induction hypothesis on the trace

$$\langle q_{11}, \sigma_{11}, \delta_{11}\rangle \xrightarrow{d_2} \ldots \xrightarrow{d_k} \langle q_{k1}, \sigma_{k1}, \delta_{k1}\rangle$$

we have that $\exists(\sigma_2', \delta_2')$ and D_2' such that

$$\langle q_{11}, \sigma_{12}, \delta_{12}\rangle \overset{D_2'}{\Longrightarrow}_Y \langle q', \sigma_2', \delta_2'\rangle \tag{2}$$

and therefore by (1) and (2) and for $D_2 = d_1' + D_2'$ we have that

$$\langle q, \sigma_2, \delta_2\rangle \overset{D_2}{\Longrightarrow}_Y \langle q', \sigma_2', \delta_2'\rangle$$

and $(\sigma_1', \delta_1') \equiv (\sigma_2', \delta_2') \vee q' \in Y_w$ as required.

Subcase 1(b): When Φ_q is false. In the case that all the variables in con(e) are low then the proof proceeds as in Subcase 1(a).

Assume now that at least one variable in $\mathsf{con}(e)$ is high. Since $\mathrm{ipd}_Y(q)$ is defined then there exists $j \in \{1, ..., k\}$ such that $q_{j1} = \mathrm{ipd}_Y(q)$ and $\forall i \in \{1, .., j-1\} : q_{i1} \neq \mathrm{ipd}_Y(q)$. Therefore we have that

$$\langle q_{01}, \sigma_{01}, \delta_{01}\rangle \xrightarrow{d_1} ... \xrightarrow{d_j} \langle q_{j1}, \sigma_{j1}, \delta_{j1}\rangle \xrightarrow{d_{j+1}} ... \xrightarrow{d_k} \langle q_{k1}, \sigma_{k1}, \delta_{k1}\rangle$$

Next, using that $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, condition **C2** (a) and Fact 4 gives us that $\forall x : \mathcal{L}(x) = L \Rightarrow \sigma_{j1}(x) = \sigma_{01}(x)$ and $\forall r : \mathcal{L}(r) = L \Rightarrow \delta_{j1}(r) = \delta_{01}(r) + d_1 + ... + d_j$. Since Φ_q is false, (σ_1, δ_1) and (σ_2, δ_2) have the same *termination behaviour* and thus there exists trace $t' \in [\![\mathsf{TA} : q \mapsto \mathrm{ipd}_Y(q)]\!](\sigma_2, \delta_2)$ and $d_1', ..., d_l'$ such that $d_1 + ... + d_j = d_1' + ... + d_l'$ and $(\sigma_{l2}, \delta_{l2})$ such that t' is

$$\langle q, \sigma_2, \delta_2\rangle = \langle q, \sigma_{02}, \delta_{02}\rangle \xrightarrow{d_1'} ... \xrightarrow{d_l'} \langle q_{l2}, \sigma_{l2}, \delta_{l2}\rangle \tag{3}$$

and $q_{l2} = \mathrm{ipd}_Y(q)$.

It is immediate that $\forall x : \mathcal{L}(x) = L \Rightarrow \sigma_{l2}(x) = \sigma_{02}(x)$ and $\forall r : \mathcal{L}(r) = L \Rightarrow \delta_{l2}(r) = \delta_{02}(r) + d_1' + ... + d_l'$. To see how we obtain this result, we have that if t' has started using the edge e or an edge $e' \neq e$, where $\mathsf{con}(e')$ contains at least one high variable, then this result follows by our assumptions that $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, condition **C2** (a) and Fact 4. Now if the t' has started using an edge $e' \neq e$ and $\mathsf{con}(e')$ contains only low variables then (σ_1, δ_1) is a witness of $\underline{\mathrm{sat}}(\mathsf{con}(e) \wedge \mathsf{con}(e'))$ and the result follows by our assumptions that $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, condition **C2** (b) and Fact 4. Therefore in any case $(\sigma_{j1}, \delta_{j1}) \equiv (\sigma_{l2}, \delta_{l2})$.

Now if $\mathrm{ipd}_Y(q) = q_{k1}$ the proof has been completed. Otherwise we have that $\mathrm{ipd}_Y(q)$ is not in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$ and the proof follows by an induction on the trace

$$\langle q_{j1}, \sigma_{j1}, \delta_{j1}\rangle \xrightarrow{d_j} ... \xrightarrow{d_k} \langle q_{k1}, \sigma_{k1}, \delta_{k1}\rangle$$

using that $(\sigma_{j1}, \delta_{j1}) \equiv (\sigma_{l2}, \delta_{l2})$

Subcase 2: When the unique immediate Y post-dominator of q is not defined. In that case, all the variables in $\mathsf{con}(e)$ are low. Therefore, if $e \rightsquigarrow w$ we proceed similar to *Main Case 1*, otherwise we proceed as in Subcase 1(a).

This completes our proof.

Proof of Lemma 2

Proof. Assume that $\langle q, \sigma_1, \delta_1 \rangle \overset{\infty}{\Longrightarrow}_Y \bot$ and thus either there exists a finite unsuccessful trace t

$$\langle q, \sigma_1, \delta_1 \rangle = \langle q_{01}, \sigma_{01}, \delta_{01} \rangle \xrightarrow{d_1} ... \xrightarrow{d_n} \langle q_{n1}, \sigma_{n1}, \delta_{n1} \rangle \quad (n \geq 0)$$

such that $\forall i \in \{1, ..., n\} : q_{i1} \notin Y$ and $\langle q_{n1}, \sigma_{n1}, \delta_{n1} \rangle$ is stuck, or there exists an infinite unsuccessful trace t

$$\langle q, \sigma_1, \delta_1 \rangle = \langle q_{01}, \sigma_{01}, \delta_{01} \rangle \xrightarrow{d_1} ... \xrightarrow{d_n} \langle q_{n1}, \sigma_{n1}, \delta_{n1} \rangle \xrightarrow{d_{n+1}} ...$$

such that $\forall i > 0 : q_{i1} \notin Y$.

Assume now that all the traces from $\langle q, \sigma_2, \delta_2 \rangle$ to a node $q' \in Y$ are successful, which means that $\langle q, \sigma_2, \delta_2 \rangle \overset{\infty}{\not\Longrightarrow}_Y \bot$ and thus by Proposition 1 the set

$$\{k \mid \langle q'_0, \sigma'_0, \delta'_0 \rangle \xrightarrow{d'_1} ... \xrightarrow{d'_k} \langle q'_k, \sigma'_k, \delta'_k \rangle : \langle q'_0, \sigma'_0, \delta'_0 \rangle = \langle q, \sigma_2, \delta_2 \rangle \wedge q'_k \in Y \wedge \forall i \in \{1, ..., k-1\} : q'_i \notin Y\}$$

has a maximum m.

The proof proceeds by contradiction where we show that we can either construct an unsuccessful trace of $\langle q, \sigma_2, \delta_2 \rangle$ or a "long" trace t'

$$\langle q, \sigma_2, \delta_2 \rangle = \langle q_{02}, \sigma_{02}, \delta_{02} \rangle \xrightarrow{d'_1} ... \xrightarrow{d'_l} \langle q_{l2}, \sigma_{l2}, \delta_{l2} \rangle \quad (l > 0)$$

where $\forall i \in \{1, ..., l\} : q_{i2} \notin Y$ and $m \leq l$ and that would mean that this trace will either terminate later (at a node in Y) and thus it will have a length greater than m, or it will result into an unsuccessful trace.

We consider two main cases one where q is in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$ and one where it isn't.

Main Case 1: When q is in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$. If the trace t of $\langle q, \sigma_1, \delta_1 \rangle$ visits only nodes that can reach Y ($\forall i : \Pi_{q_{i1}} \neq \emptyset$) then we proceed similar to the proof of Main Case 1 of Lemma 1, using the result (a) and (b) of the fact proven there. Therefore if t is infinite we can show that (σ_2, δ_2) can simulate the first m steps of (σ_1, δ_1) and this give us the desired trace t'. Similarly, in case of t being a finite unsuccessful trace that stops at the node q_{n1}, and $\langle q_{n1}, \sigma_{n1}, \delta_{n1} \rangle$ is a stuck, we can also show that (σ_2, δ_2) can reach the node q_{n1} (using the result (a)) and the resulting configuration will be stuck (using the result (b)).

Now if the first $j > 0$ nodes $q_{01}...q_{j1}$ (visited by t) can reach Y and then for the node $q_{(j+1)1}$ we have that $\Pi_{(q_{(j+1)1}, Y)} = \emptyset$, we can show similarly as before that (σ_2, δ_2) can reach the node $q_{(j+1)1}$ (using the results (a) and (b)), and thus any further computation will lead to an unsuccessful trace since $\Pi_{(q_{(j+1)1}, Y)} = \emptyset$.

Finally if t visits only nodes that cannot reach Y ($\forall i : \Pi_{q_{i1}} = \emptyset$) and thus also q cannot reach Y, the proof is trivial since all the traces of $\langle q, \sigma_2, \delta_2 \rangle$ will be unsuccessful with respect to Y. This completes the proof of Main Case 1.

Main Case 2: When q is not in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$. We will now present a finite construction strategy for the desired trace t'.

Construction. We start by looking at the configurations $\langle q, \sigma_1, \delta_1\rangle$, $\langle q, \sigma_2, \delta_2\rangle$ the unsuccessful trace t of (σ_1, δ_1), and we remember that so far we have created a trace $t' = \langle q, \sigma_2, \delta_2\rangle$ of length $l = 0$. We proceed according to the following cases:

Case 1: When the unique immediate Y post-dominator $\mathrm{ipd}_Y(q)$ *of q is defined.* We then consider two subcases, one where Φ_q is *false* and one where Φ_q is *true.*

Subcase (a): Φ_q is false. Now if the trace t does *not* visit $\mathrm{ipd}_Y(q)$, we have that (σ_1, δ_1) and (σ_2, δ_2) have the same *termination behaviour* (using that Φ_q is false) and thus there exists a trace t' of (σ_2, δ_2) that never visits $\mathrm{ipd}_Y(q)$. However, then we would have the case that t' is an unsuccessful trace with respect to q and the set Y which contradicts our assumptions.

If the trace t does visit $\mathrm{ipd}_Y(q)$, then it has to be the case that $\mathrm{ipd}_Y(q)$ is not in Y. Assume now that t starts with an edge $e \in \mathsf{E}_q$. If $\mathsf{con}(e)$ contains only low variables then $\exists d'_1 = d_1$ and $(\sigma_{12}, \delta_{12}) \equiv (\sigma_{11}, \delta_{11})$ (this is ensured by our assumptions that $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$ and the predicate A_e of condition **C2** (a) that takes care of the *explicit flows* arising from the assignment in the edge e) such that

$$\langle q, \sigma_2, \delta_2\rangle = \langle q_{02}, \sigma_{02}, \delta_{02}\rangle \xrightarrow{d'_1} \langle q_{12}, \sigma_{12}, \delta_{12}\rangle$$

where $q_{12} = q_{11}$. If now $m \leq l+1$ then we have our desired trace t' and we stop.

Otherwise, notice that also q_{11} is not in $\mathsf{Q}_{\rightsquigarrow \mathsf{w}}$ and we repeat the *Construction* by looking at the configurations $\langle q_{11}, \sigma_{11}, \delta_{11}\rangle$, $\langle q_{11}, \sigma_{12}, \delta_{12}\rangle$, the suffix of t that starts with $\langle q_{11}, \sigma_{11}, \delta_{11}\rangle$ and we remember that so far we have created the trace

$$t' = \langle q_{02}, \sigma_{02}, \delta_{02}\rangle \xrightarrow{d'_1} \langle q_{12}, \sigma_{12}, \delta_{12}\rangle \quad (\langle q, \sigma_2, \delta_2\rangle = \langle q_{02}, \sigma_{02}, \delta_{02}\rangle)$$

that has length equal to $l+1$.

Now if $\mathsf{con}(e)$ contains at least one high variable then we look at the first occurrence of $\mathrm{ipd}_Y(q)$ in t and let that to be the configuration $\langle q_{h1}, \sigma_{h1}, \delta_{h1}\rangle$ for some $h > 0$. Therefore, since $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, using the condition **C2** (a) and Fact 4 we have that $\forall x : \mathcal{L}(x) = L \Rightarrow \sigma_{h1}(x) = \sigma_{01}(x)$ and $\forall r : \mathcal{L}(r) = L \Rightarrow \delta_{h1}(r) = \delta_{01}(r) + d_1 + ... + d_h$. Since Φ_q is false (σ_1, δ_1) and (σ_2, δ_2) have the same *termination behaviour* and thus there exists trace $t' \in [\![\mathsf{TA} : q \mapsto \mathrm{ipd}_Y(q)]\!](\sigma_2, \delta_2)$ and $d'_1, ..., d'_j$ such that $d_1 + ... + d_h = d'_1 + ... + d'_j$ and $(\sigma_{j2}, \delta_{j2})$ such that t' is

$$\langle q, \sigma_2, \delta_2\rangle = \langle q_{02}, \sigma_{02}, \delta_{02}\rangle \xrightarrow{d'_1} ... \xrightarrow{d'_j} \langle q_{j2}, \sigma_{j2}, \delta_{j2}\rangle$$

where $q_{j2} = \mathrm{ipd}_Y(q)$.

Now if $j + l \geq m$ we have constructed the required trace t'.

Otherwise, we have that $\forall x : \mathcal{L}(x) = L \Rightarrow \sigma_{j2}(x) = \sigma_{02}(x)$ and $\forall r : \mathcal{L}(r) = L \Rightarrow \delta_{j2}(r) = \delta_{02}(r) + d'_1 + ... + d'_j$. To see how we obtain this result, we have that if t' has started using the edge e or an edge $e' \neq e$, where $\mathsf{con}(e')$ contains at least one high variable, then this result follows by our assumptions that $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, condition **C2** (a) and Fact 4. Now if the t' has started using an edge $e' \neq e$ and $\mathsf{con}(e')$ has only low variables then (σ_1, δ_1) is a witness of $\underline{\mathrm{sat}}(\mathsf{con}(e) \wedge \mathsf{con}(e'))$ and the result follows again by our assumptions that $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, condition **C2**

(b) and Fact 4. Therefore in any case $(\sigma_{h1}, \delta_{h1}) \equiv (\sigma_{j2}, \delta_{j2})$ and thus we repeat the *Construction* by looking at the configurations $\langle q_{h1}, \sigma_{h1}, \delta_{h1}\rangle$, $\langle q_{j2}, \sigma_{j2}, \delta_{j2}\rangle$ the suffix of t that starts with $\langle q_{h1}, \sigma_{h1}, \delta_{h1}\rangle$ and we remember that so far we have created the trace t'

$$\langle q, \sigma_2, \delta_2\rangle = \langle q_{02}, \sigma_{02}, \delta_{02}\rangle \xrightarrow{d'_1} ... \xrightarrow{d'_j} \langle q_{j2}, \sigma_{j2}, \delta_{j2}\rangle$$

of length equal to $l + j$.

Subcase (b): Φ_q is true. Then if t starts with the edge e, because $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$, $\mathsf{con}(e)$ contains only low variables and we proceed as in Subcase (a).

Case 2: When the unique immediate Y post-dominator $\mathrm{ipd}_Y(q)$ of q is not defined. In this case, if t starts with the edge e, because $\mathrm{sec}_{Y,\mathcal{L}}(\mathsf{TA})$ we have that $\mathsf{con}(e)$ contains only low variables. Now if $e \rightsquigarrow w$ working as in Main Case 1 we can get an unsuccessful trace t', otherwise we proceed as in Subcase (a).

Proof of Theorem 1

Proof. Let

$$Z = \{(\langle q, \sigma, \delta\rangle, \langle q, \sigma', \delta'\rangle) \mid [\![\mathsf{I}(q)]\!](\sigma, \delta) \wedge [\![\mathsf{I}(q)]\!](\sigma', \delta')\} \cup \{(\bot, \bot)\}$$

It is immediate by Lemmas 1 and 2 that Z is a $Y-$bisimulation and that

$$\forall q \in \{q_\circ\} \cup Y : \forall (\sigma, \delta), (\sigma', \delta') : [\![\mathsf{I}(q)]\!](\sigma, \delta) \ \wedge \ [\![\mathsf{I}(q)]\!](\sigma', \delta') \Downarrow (\langle q, \sigma, \delta\rangle, \langle q, \sigma', \delta'\rangle) \in Z$$

Therefore since $\sim_Y$ is the largest $Y-$bisimulation we have that $Z \subseteq \sim_Y$ and thus TA satisfies the information security policy $\mathcal{L}$.

References

1. Aceto, L., Ingolfsdottir, A., Larsen, K.G., Srba, J.: Reactive Systems: Modelling, Specification and Verification. Cambridge University Press, Cambridge (2007)
2. Agat, J.: Transforming out timing leaks. In: Proceedings of the POPL, pp. 40–53 (2000)
3. Al-Bataineh, O.I., Reynolds, M., French, T.: Finding minimum and maximum termination time of timed automata models with cyclic behaviour. CoRR, abs/1610.09795 (2016)
4. Al-Bataineh, O.I., Reynolds, M., French, T.: Finding minimum and maximum termination time of timed automata models with cyclic behaviour. Theor. Comput. Sci. **665**, 87–104 (2017)
5. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994)

6. Askarov, A., Sabelfeld, A.: Localized delimited release: combining the what and where dimensions of information release. In: Proceedings of the 2007 Workshop on Programming Languages and Analysis for Security, PLAS 2007, San Diego, California, USA, 14 June 2007, pp. 53–60 (2007)
7. Barbuti, R., De Francesco, N., Santone, A., Tesei, L.: A notion of non-interference for timed automata. Fundam. Inform. **51**(1–2), 1–11 (2002)
8. Barbuti, R., Tesei, L.: A decidable notion of timed non-interference. Fundam. Inform. **54**(2–3), 137–150 (2003)
9. Barthe, G., Rezk, T., Warnier, M.: Preventing timing leaks through transactional branching instructions. Electron Notes Theor. Comput. Sci. **153**(2), 33–55 (2006)
10. Denning, D.E., Denning, P.J.: Certification of programs for secure information flow. Commun. ACM **20**(7), 504–513 (1977)
11. Franchella, M.: On the origins of Dénes König's infinity lemma. Arch. Hist. Exact Sci. **51**, 3–27 (1997)
12. Gardey, G., Mullins, J., Roux, O.H.: Non-interference control synthesis for security timed automata. Electron Notes Theor. Comput. Sci. **180**(1), 35–53 (2007)
13. Grewe, S., Lux, A., Mantel, H., Sauer, J.: A formalization of declassification with what-and-where-security. Archive of Formal Proofs (2014)
14. Gupta, B.B., Akhtar, T.: A survey on smart power grid: frameworks, tools, security issues, and solutions. Annales des Télécommunications **72**(9–10), 517–549 (2017)
15. Herbreteau, F., Srivathsan, B., Walukiewicz, I.: Efficient emptiness check for timed büchi automata. Form. Methods Syst. Des. **40**(2), 122–146 (2012)
16. Kashyap, V., Wiedermann, B., Hardekopf, B.: Timing- and termination-sensitive secure information flow: exploring a new approach. In: 32nd IEEE Symposium on Security and Privacy, S&P 2011, 22–25 May 2011, Berkeley, California, USA, pp. 413–428 (2011)
17. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Time and probability-based information flow analysis. IEEE Trans. Softw. Eng. **36**(5), 719–734 (2010)
18. Lengauer, T., Tarjan, R.E.: A fast algorithm for finding dominators in a flowgraph. ACM Trans. Program. Lang. Syst. **1**(1), 121–141 (1979)
19. Lux, A., Mantel, H., Perner, M.: Scheduler-independent declassification. In: Gibbons, J., Nogueira, P. (eds.) MPC 2012. LNCS, vol. 7342, pp. 25–47. Springer, Heidelberg (2012)
20. Magazinius, J., Askarov, A., Sabelfeld, A.: Decentralized delimited release. In: Yang, H. (ed.) APLAS 2011. LNCS, vol. 7078, pp. 220–237. Springer, Heidelberg (2011)
21. Mantel, H., Sands, D.: Controlled declassification based on intransitive noninterference. In: Chin, W.-N. (ed.) APLAS 2004. LNCS, vol. 3302, pp. 129–145. Springer, Heidelberg (2004)
22. Mantel, H., Starostin, A.: Transforming out timing leaks, more or less. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 447–467. Springer, Cham (2015)
23. McMillin, B.M., Roth, T.P.: Cyber-Physical Security and Privacy in the Electric Smart Grid. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan & Claypool Publishers, San Rafael (2017)
24. Myers, A.C., Sabelfeld, A., Zdancewic, S.: Enforcing robust declassification and qualified robustness. J. Comput. Secur. **14**(2), 157–196 (2006)
25. Nielson, F., Nielson, H.R., Vasilikos, P.: Information flow for timed automata. In: Aceto, L., Bacci, G., Bacci, G., Ingólfsdóttir, A., Legay, A., Mardare, R. (eds.) Models, Algorithms, Logics and Tools. LNCS, vol. 10460, pp. 3–21. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63121-9_1

26. Sabelfeld, A., Myers, A.C.: Language-based information-flow security. IEEE J. Sel. Areas Commun. **21**(1), 5–19 (2003)
27. Baumgart, I., Finster, S.: Privacy-aware smart metering: a survey. IEEE Commun. Surv. Tutor. **17**(2), 1088–1101 (2015)
28. UPPAAL. http://www.uppaal.com/index.php?sida=200&rubrik=95
29. Volpano, D.M., Smith, G., Irvine, C.E.: A sound type system for secure flow analysis. J. Comput. Secur. **4**(2/3), 167–188 (1996)