URL: http://www.elsevier.nl/locate/entcs/volume52.html 11 pages

Privacy in Real-Time Systems

Ruggero Lanotte [1,2]

Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy

Andrea Maggiolo-Schettini [1,3]

Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy

Simone Tini [1,4]

Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy

Abstract

We study the problem of privacy in the framework of Timed Automata. By distinguishing between secret and observable actions we formulate a property of no-privacy in terms of a property of the language accepted by a Timed Automaton, and we give an algorithm checking such property.

1 Introduction

One of the main requirements of mobile code is that it must guarantee some kind of security to clients executing it. One of the security requirements is the client's *privacy*, namely that executing mobile code does not imply leaking of private information.

Several papers (see, among the others, [3],[4],[5], [6],[7]) dealing with privacy, consider *two-level* systems, where the *high level* (or *secret*) behavior is distinguished from the *low level* (or *observable*) one. In the mentioned papers, systems respect the property of privacy if there is no information flow from

[1] Research partially supported by MURST Progetto Cofinanziato TOSCA.
[2] Email: lanotte@di.unipi.it
[3] Email: maggiolo@di.unipi.it
[4] Email: tini@di.unipi.it

the high level to the low level. This means that the secret behavior cannot influence the observable one, or, equivalently, no information on the observable behavior permits to infer information on the secret one.

Our aim is to study the problem of privacy in real-time systems in the framework of Timed Automata [1]. When using this formalism, the possible behaviors of a system are described by a set of infinite *timed words*, namely infinite sequences of pairs (action performed, time of firing). In describing two-level systems, we distinguish between high-level and low-level actions. We formulate a *no-privacy property* as follows: if, whenever one can observe a given timed sequence of observable actions, one is sure that the system performs a certain secret action, then the system is insecure. The reason is that one can infer information on the secret behavior from the observation of the observable one.

We give an algorithm that exploits the *region graph* obtained from a Timed Automaton and checks the no-privacy property for a given sequence of observable actions and a given secret action.

2 HL Timed Automata

In this section we introduce the formalism of HL Timed Automata, as an extension of Alur and Dill's Timed Automata.

2.1 Security alphabet and timed words

A *security alphabet* is a pair consisting of two disjoint finite sets of *actions* (L, H). The set L contains the *low actions*, which can be performed by the system and can be observed by the external environment, and the set H contains the *high actions*, which can be performed by the system and are visible only inside the system.

Given any *time domain* T (non-negative rational numbers, or non-negative real numbers, as examples), a *timed word* ω on (L, H) and T is a pair of functions (ω_1, ω_2) such that $\omega_1 : \mathbb{N} \to (L \cup H)$ and $\omega_2 : \mathbb{N} \to T$. Intuitively, ω describes the behavior of a system that performs action $\omega_1(i)$ at time $\sum_{h=0}^{i} \omega_2(h)$. A timed word must satisfy the *time progress property*, namely for each time value $t \in T$, there is some index i such that $\sum_{h=0}^{i} \omega_2(h) > t$.

Given a timed word $\omega = (\omega_1, \omega_2)$, let us denote with ω_L the projection of ω on L, namely the (possibly finite) sequence $(\omega_1(i_1), \omega_2(i_1)), (\omega_1(i_2), \omega_2(i_2)), \ldots$ such that, for each index i_j, $\omega_1(i_j) \in L$ and, for each $i_j < k < i_{j+1}$, $\omega_1(k) \in H$. The sequence ω_L describes the part of ω that can be observed by the external environment.

Let us denote with F_ω the function that gives the index in ω of the low action in position j in ω_L, namely $F_\omega(j) = i_j$.

2.2 Clock valuations and clock constraints

We assume a set X of variables measuring time, called *clocks*. Intuitively, clocks increase uniformly with time when an automaton is in whatsoever state.

A *clock valuation* over a set of clocks X is a mapping $v : X \to T$ assigning time values to clocks. For a clock valuation v and a time value t, let $v + t$ denote the clock valuation such that $(v+t)(x) = v(x)+t$. For a clock valuation v and a subset of clocks $Y \subseteq X$, let $v[Y]$ denote the clock valuation such that $v[Y](x) = 0$, if $x \in Y$, and $v[Y](x) = v(x)$, otherwise.

Given a set of clocks X, we consider the set of *clock constraints* over X, denoted $\Phi(X)$, which is defined by the following grammar, where ϕ ranges over $\Phi(X)$, $x \in X$, $c \in T$ and $\# \in \{<, \leq, =, \neq, >, \geq\}$:

$$\phi ::= x \,\#\, c \mid \phi \wedge \phi \mid \neg\phi \mid \phi \vee \phi \mid true\,.$$

We write $v \models \phi$ when *the clock valuation v satisfies the clock constraint ϕ*. More precisely, $v \models x \,\#\, c$ iff $v(x) \,\#\, c$, $v \models \phi_1 \wedge \phi_2$ iff both $v \models \phi_1$ and $v \models \phi_2$, $v \models \phi_1 \vee \phi_2$ iff either $v \models \phi_1$ or $v \models \phi_2$, $v \models \neg\phi$ iff $v \not\models \phi$, and $v \models true$.

2.3 The formalism

Definition 2.1 Given a security alphabet (H, L), a *HL Timed Automaton* (TA_{HL}) is a tuple $\mathcal{A} = ((L, H), A_1, \ldots, A_m)$, where, for each $1 \leq i \leq m$, $A_i = (Q_i, q_i^0, \delta_i, X_i)$ is a *sequential automaton*, with:

- a finite set of *states* Q_i
- an *initial state* $q_i^0 \in Q_i$
- a set of *clocks* X_i
- a set of *transitions* $\delta_i \subseteq Q_i \times \Phi(X_i) \times (L \cup H) \times 2^{X_i} \times Q_i$.

The sets of clocks $X_1, \ldots, X_m$ are pairwise disjoint.

Intuitively, a transition (q, ϕ, a, Y, q') of an automaton A_i fires in correspondence with the performance of action a when state q is active and the clock valuation of A_i satisfies the clock constraint ϕ. In such a case, state q' is entered and the clocks in Y are reset.

Let us describe now the behavior of a TA_{HL} $\mathcal{A} = ((L, H), A_1, \ldots, A_m)$.

A *configuration* of $\mathcal{A}$ is a tuple $s = ((q_1, v_1), \ldots, (q_m, v_m))$ such that, for each $1 \leq i \leq m$, q_i is a state in Q_i and v_i is a clock valuation over the set of clocks X_i.

The *initial configuration* s_0 is the tuple $((q_1^0, v_1^0), \ldots, (q_m^0, v_m^0))$, with q_i^0 the initial state of A_i and with v_i^0 the clock valuation such that $v_i(x)^0 = 0$ for each clock $x \in X_i$.

There is a *step* from configuration $s = ((q_1, v_1), \ldots, (q_m, v_m))$ to configuration $s' = ((q_1', v_1'), \ldots, (q_m', v_m'))$ through action a at time t, written $s \to_t^a s'$, if and only if, for each $1 \leq i \leq m$, there is a transition $(q_i, \phi_i, a, Y_i, q_i') \in \delta_i$ such

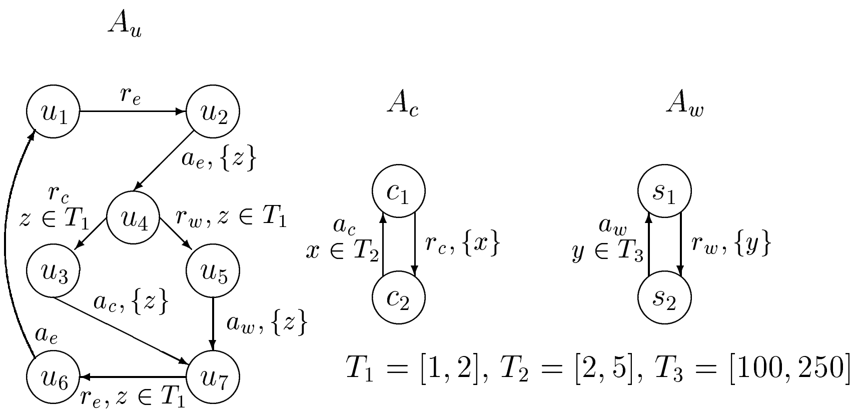

Fig. 1. The web system.

that $(v_i + t) \models \phi_i$ and $v'_i = (v_i + t)[Y_i]$. Intuitively, each clock constraint ϕ_i is satisfied by the clock valuation $v_i + t$ and all clocks in Y_i are reset.

A timed word (ω_1, ω_2) is *accepted* by $\mathcal{A}$ if there exists a infinite sequence of steps $s_0 \rightarrow^{\omega_1(0)}_{\omega_2(0)} s_1 \rightarrow^{\omega_1(1)}_{\omega_2(1)} \ldots$ from the initial configuration s_0.

The *language* accepted by $\mathcal{A}$ (denoted by $\mathcal{L}(\mathcal{A})$) is the set of timed words accepted by $\mathcal{A}$.

By application of a cartesian product construction, any TA_{HL} can be transformed into an equivalent (namely, accepting the same language) TA_{HL} consisting of only one sequential component, namely into an Alur and Dill's Timed Automaton.

Proposition 2.2 *For any TA_{HL} $\mathcal{A}$ there exists a TA_{HL} $\mathcal{A}'$ composed by one only sequential automaton such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$.*

3 The No-privacy Property

Given sequences d and d', let $d \leq_P d'$ denote the fact that d is a prefix of d'.

Let $a \in H$, d be a finite sequence $(a_1, t_1), \ldots, (a_h, t_h)$ with $a_1, \ldots, a_h \in L$ and $t_1, \ldots, t_h \in T$, and i be an index $1 \leq i < h$. We define the $no-privacy$ property $NPr(d, i, a)$ for a TA_{HL} $\mathcal{A}$ as follows:

$$\text{for each } \omega \in \mathcal{L}(\mathcal{A}), d \leq_P \omega_L \text{ implies } a \in \{\omega_1(F_\omega(i)+1), \ldots, \omega_1(F_\omega(i+1)-1)\}.$$

Intuitively, $NPr(d, i, a)$ expresses that, whenever the sequence d of low symbols is read, the high symbol a is read between the low level actions a_i and a_{i+1}, and, therefore, there is an information flow from high level to low level, namely information on the secret behavior can be inferred from information on the observable behavior.

Example 3.1 We model the time attack on web privacy described in [2]. The attack compromises the privacy of user's web-browsing histories by allowing a malicious web site to determine whether or not the user has recently visited some other, unrelated, web page w. A Java applet is embedded in the malicious web site and is run by the user's browser. The applet first performs a request to a file of w, and then performs a new request to the malicious site. So, the malicious site can measure the time elapsed between the two requests which it receives from the user, and, if such a time is under a certain bound, it infers that w was in the cache of the browser of the user, thus implying that w has been recently visited by the user.

In Fig. 1 we model this problem. Automaton A_c represents the cache. The time elapsed between a request r_c and an answer a_c is in the interval $[2, 5]$. Automaton A_w represents the site w. The time elapsed between a request r_w and an answer a_w is in the interval $[100, 250]$. The automaton A_u represents the requests by the user that downloads the page of the malicious site. First of all, it performs a request r_e to the malicious site. Then, when it receives the answer a_e, it performs a communication either with the cache or with the site w in a time belonging to the interval $[1, 2]$. Finally, it performs another request r_e and it waits for an answer from the malicious site. We assume that there are transitions from state c_1 to state c_1 labeled with symbols r_e, a_e, r_w and a_w. Analogously there are transitions from state s_1 to state s_1 labeled with symbols r_e, a_e, r_c and a_c.

The only visible actions for the malicious site are r_e and a_e, so the alphabet (L, H) is $(\{r_e, a_e\}, \{r_c, a_c, r_w, a_w\})$.

Now, if we consider $d = (r_e, 10)(a_e, 20)(r_e, 200)$, then we have the no privacy property $NPr(d, 2, a_w)$.

3.1 Region graph

Our aim is to show that the property $NPr(d, i, a)$ is decidable.

To this purpose, let us recall first the notion of *region graph* of a timed automaton, as given in [1]. By proposition 2.2 it suffices to consider automata with only one sequential component.

As in [1], without loss of generality we assume clock constraints permitting only comparison with integer constants. In fact, given any automaton $\mathcal{A}$, there is a constant t such that, for each constant c appearing in a clock constraint in $\mathcal{A}$, $c \cdot t$ is an integer. Let $\mathcal{A} \cdot t$ be the automaton obtained by replacing each c appearing in a clock constraint in $\mathcal{A}$ by $c \cdot t$. In [1] it is proved that a word $(a_1, t_1) \dots (a_n, t_n) \dots$ is in the language $\mathcal{L}(\mathcal{A})$ if and only if the word $(a_1, t_1 \cdot t) \dots (a_n, t_n \cdot t) \dots$ is in the language $\mathcal{L}(\mathcal{A} \cdot t)$. As a consequence, $NPr(d, i, a)$ holds for $\mathcal{A}$ if and only if $NPr(d \cdot t, i, a)$ holds for $\mathcal{A} \cdot t$.

Let us consider the equivalence relation $\sim$ over clock valuations that contains each pair of clock valuations v and v' such that:

- for each clock x, either $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$, or both $v(x)$ and $v'(x)$ are greater than c_x, with c_x the largest integer appearing in clock constraints over x.
- for each pair of clocks x and y with $v(x) \leq c_x$ and $v(y) \leq c_y$, $fract(v(x)) \leq$

$fract(v(y))$ if and only if $fract(v'(x)) \leq fract(v'(y))$ ($fract(z)$ indicates the fractional part of z).

- for each clock x with $v(x) \leq c_x$, $fract(v(x)) = 0$ if and only if $fract(v'(x)) = 0$.

Note that for each pair of valuations v and v', and for each clock constraint ϕ in $\mathcal{A}$, it holds that:

$$\text{if } v \sim v' \text{ then } v \models \phi \text{ iff } v' \models \phi.$$

A *clock region* is an equivalence class of clock valuations induced by $\sim$. We denote by $[v]$ the equivalence class of $\sim$ containing v. Note that the set of clock regions is finite.

Clock regions can be expressed with conditions of the form $x = c$, $c < x < c+1$ and $x > c_x$.

A *region* is a pair $(q, [v])$, with q a state and $[v]$ a clock region. The *initial region* is the pair $(q^0, [v^0])$ with q^0 the initial state and v^0 the valuation such that $v^0(x) = 0$, for each clock x.

The *region graph* $R(\mathcal{A})$ is a graph having the regions of $\mathcal{A}$ as set of nodes and having an edge $\langle (q,\alpha), a, I, (q', \alpha')\rangle$, with I an interval of the form (c, c'),$[c, c]$ or (c_m, ∞) (where $c < c_m$, $c \in \mathbb{N}$ and c_m is the largest constant that appears in the constraints of $\mathcal{A}$) if and only if, for each pair of valuations $v \in \alpha$ and $v' \in \alpha'$, $(q, v) \rightarrow_t^a (q', v')$ for some time $t \in I$.

Note that, differently from [1], we label edges of the region graph also with an interval.

Without loss of generality, we can consider only the nodes that can be reached from the initial region without cycles of edges labeled with interval $[0, 0]$ which would violate the assumption of progress of time.

3.2 Checking no-privacy

Our algorithm checking no-privacy constructs a set of intervals by applying operations, which are defined below, to intervals of the region graph.

Given two intervals I and I', let us denote with $I \pm I'$ the interval

$$I \pm I' = \{t \pm t' \mid t \in I \text{ and } t' \in I'\}.$$

Given a step leading to a region (q, α) in a time in the interval I, and a step from (q, α) to another region (q', α') in the interval I', a time in $I + I'$ is needed to reach (q', α') through (q, α), provided that the waiting time in q does not depend on the time consumed to reach q.

Given two intervals I and I', let us denote with $I \oplus I'$ the interval such that $inf(I \oplus I') = inf(I) + inf(I') + 1$, $sup(I \oplus I') = sup(I) + sup(I') - 1$. Moreover, if $inf(I \oplus I') = sup(I \oplus I')$ then we assume that $(I \oplus I')$ is left-closed and right-closed. Otherwise, $I \oplus I'$ is left-closed if and only if both I and I' are, and $I \oplus I'$ is right-closed if and only if both I and I' are.

Given a step leading to a region (q,α) in a time in the interval I, and a step from (q,α) to another region (q',α') in the interval I', a time in $I \oplus I'$ is needed to reach (q',α') through (q,α), provided that the waiting time in q depends on the time consumed to reach q, in the sense that the longer is the time consumed before entering q, the shorter is the waiting time in q. This happens if there is a constraint $c < x < c+1$ in α, for some clock x.

Finally, let us denote with $(I)_t$ the interval:

$$(I)_t = \begin{cases} I & \text{if } sup(I) \leq t \\ I \cup [sup(I),\infty) & \text{if } inf(I) \leq t < sup(I) \\ (t,\infty) & \text{otherwise.} \end{cases}$$

Note that $t \in I$ if and only if $t \in (I)_t$. We have introduced notation $(I)_t$ since the set of the intervals $(I)_t$, such that I is a sum (obtained by means of either $+$ or $\oplus$) of intervals of a region graph, is finite.

Let us define now the algorithm *Ch-path-symb*. Given regions p and q, a high symbol a, a constant t and a low symbol a', *Ch-path-symb* checks whether there exists a sequence of steps labeled with symbols in $H \setminus \{a\}$ followed by a step labeled with a' and taking from p to q at time t.

Algorithm 1

```
Ch-path-symb(p,q:region, a:high-symbol, t:T, a': low symbol): boolean
(i)      tovisit:={$(p,[0,0],false)$};
(ii)     visited:=$\emptyset$;
(iii)    while true do
(iv)        if empty(tovisit) then return false
(v)         else
(vi)           $(r,I,tt)$:=extract(tovisit);
(vii)          add($(r,I,tt)$,visited);
(viii)         if (tt=true and $(r=q)$ and $t \in I$) then return true;
(ix)           if $tt = false$ then
(x)            for each edge $\langle r,a'',I',r'\rangle \in R(\mathcal{A})$ with $a'' \in H \setminus \{a\} \cup \{a'\}$
(xi)              if $x = t'$ is a constraint in $r$ then c:=$(r',(I+I')_t,(a''=a'))$
(xii)             else if $I' = [0,0]$ then c:=$(r',I,(a''=a'))$
(xiii)                 else c:=$(r',(I \oplus I')_t,(a''=a'))$
(xiv)             if c$\notin$ visited $\wedge$ $((I+I')_t \neq (t,\infty))$ then Add(c,tovisit).
```

A tuple $(r,I,false)$ means that the region r can be reached from p in a time in the interval I by reading symbols in $H \setminus \{a\}$. A tuple $(r,I,true)$ means that the region r can be reached from p in a time in the interval I by reading symbols in $H \setminus \{a\}$ and, subsequently, symbol a'.

So the algorithm considers firstly the pair $(p, [0,0], false)$. Given a pair $(r, I, false)$ and an edge $\langle r, a'', I', r' \rangle$ in $R(\mathcal{A})$ for some symbol $a'' \in H \setminus \{a\} \cup \{a'\}$, if the clock region in r satisfies $x = t'$ for some clock x and constant t', then the algorithm considers either the pair $(r', (I+I')_t, false)$, if $a'' \neq a'$, or the pair $(r', (I+I')_t, true)$, if $a'' = a'$. The condition $x = t'$ in r ensures that a time in I' must be elapsed after r is entered. We use interval $(I+I')_t$ instead of $(I+I')$ to guarantee that *Ch-path-symb* generates finite intervals.

If, on the contrary, the clock region in r does not satisfy $x = t'$ for any clock x and constant t', then there are two cases. If $\langle r, a'', I', r' \rangle$ is such that $I' = [0,0]$, then the clock regions of r and r' coincide, and, therefore, we consider the tuple $(r', I, (a'' = a'))$. Otherwise, the tuple $(r, (I \oplus I')_t, (a'' = a'))$ is considered. The interval $(I \oplus I')_t$ takes into account that the minimal (resp. maximal) waiting time in r follows the maximal (resp. minimal) waiting time needed to reach r. Also in this case we use interval $(I \oplus I')_t$ instead of $(I \oplus I')$ to guarantee that *Ch-path-symb* generates finite intervals.

Finally, if a tuple (r, I, tt) with $r = q$, $t \in I$ and $tt = true$ is generated, then *Ch-path-symb* terminates successfully.

The following lemmata state the correctness of the algorithm.

Lemma 3.2 *For any pair of regions p and q, high symbol a, time t and low symbol a', Ch-path-symb(p,q,a,t,a') terminates.*

Proof. Both the regions of the graph and the intervals that can been generated by the algorithm are finite and, as a consequence, also the tuples (r, I, tt) that are generated are finite. $\square$

Lemma 3.3 *If $(\omega_1, \omega_2) \in \mathcal{L}(\mathcal{A})$ with $\{\omega_1(i), \omega_1(i+1), \ldots, \omega_1(j-1), \omega_1(j)\} \subseteq H \setminus \{a\}$ and $\omega_1(j+1)$ the low symbol a', then there exists an infinite sequence of steps $(q_0, v_0) \rightarrow^{\omega_1(0)}_{\omega_2(0)} (q_1, v_1) \rightarrow^{\omega_1(1)}_{\omega_2(1)} \ldots$ if and only if Ch-path-symb($(q_i, [v_i])$,$(q_{j+2}, [v_{j+2}])$,a, $\sum_{h=i}^{j+1} \omega_2(h)$,$a'$).*

Lemma 3.3 is a direct consequence of the properties of the region graph proved in [1].

We define also the algorithm *Ch-path* that checks whether there exists a sequence of steps labeled with high symbols and followed by a step labeled with the low symbol a' taking from a given region p to a given region q in a given time t. We obtain it from *Ch-path-symb* by replacing the condition $a'' \in H \setminus \{a\} \cup \{a'\}$ in line (x) with the condition $a'' \in H \cup \{a'\}$.

The following results are the analogous of Lemma 3.2 and Lemma 3.3.

Lemma 3.4 *For any pair of regions p and q, time t and low symbol a', Ch-path(p,q,t,a') terminates.*

Lemma 3.5 *Let $(\omega_1, \omega_2) \in \mathcal{L}(\mathcal{A})$ with $\{\omega_1(i), \omega_1(i+1), \ldots, \omega_1(-1j), \omega_1(j)\} \subseteq$*

H and $\omega_1(j+1)$ the low symbol a', then there exists an infinite sequence of steps $(q_0, v_0) \rightarrow^{\omega_1(0)}_{\omega_2(0)} (q_1, v_1) \rightarrow^{\omega_1(1)}_{\omega_2(1)} \ldots$ if and only if Ch-path $((q_i, [v_i]),(q_{j+2}, [v_{j+2}]), \sum_{h=i}^{j+1} \omega_2(h),\, a')$.

The following algorithm *Ch-NPriv* checks whether there exists a sequence of steps from the initial region $(q^0, [v^0])$ due to a low sequence d that does not perform the high symbol a between the low symbols in $d(i)$ and $d(i+1)$. At iteration k, the set A contains the regions that can be reached from $(q_0, [v_0])$ by reading $d(0), \ldots, d(k)$, by reading symbols in $H \setminus \{a\}$ between $d(i)$ and $d(i+1)$, and by reading symbols in H between $d(j)$ and $d(j+1)$, for each $j \neq i$. So, if A is empty then the sequence of steps we were looking for does not exist, and $NPr(d, i, a)$ holds.

Algorithm 2

```
Ch-NPriv(d:L − sequence, i:ℕ, a:high − symbol): boolean
(i)      k:=0;
(ii)     A:={$(q^0, [v^0])$};
(iii)    while $k \leq length(d)$ do
(iv)        $(a', t)$:=d(k);
(v)         B:=∅;
(vi)        while A≠ ∅ do
(vii)          $(q, [v])$:=extract(A);
(viii)         for each region $(q', [v']) \in R(\mathcal{A})$
(ix)              if ($k = i$ and Ch-path-symb($(q, [v])$,$(q', [v'])$,a,t,a')) OR
(x)                 ($k \neq i$ and Ch-path($(q, [v])$, $(q', [v'])$,t,a'))
(xi)               then Add($(q', [v'])$,B);
(xii)       A:=B;
(xiii)      k:=k+1;
(xiv)    return $A = \emptyset$.
```

The following results state the correctness of the algorithm *Ch-NPriv*.

Lemma 3.6 *For any finite sequence d, index i and high symbol a, the algorithm Ch-NPriv(d,i,a) terminates.*

Theorem 3.7 *For any finite sequence d, index i and high symbol a, it holds that $NPr(d, i, a)$ if and only if Ch-NPriv(d,i,a).*

Note that *Ch-NPriv(d,i,a)* performs at most k times the body of the external cycle. The internal cycle calls either the algorithm *Ch-path-symb* or the algorithm *Ch-path* at most $|\mathcal{R}(\mathcal{A})|$ times, with $|\mathcal{R}(\mathcal{A})|$ the number of regions of $\mathcal{R}(\mathcal{A})$. Finally, both *Ch-path-symb* and *Ch-path* construct at most $O(|\mathcal{R}(\mathcal{A})| \times t^2)$ tuples.

Corollary 3.8 *It is decidable in polynomial time whether $NPr(d,i,a)$.*

4 Further Work

In this paper we have introduced the "no-privacy" property, which corresponds to the ability, by an attacker, to infer information on the private behavior of a system from the observable behavior.

To model a real time system that respects privacy, we can consider properties derived from the property $NPr(d,i,a)$ considered in the paper.

The first step is to consider properties such as $\exists a \in H.NPr(d,i,a)$, meaning that the performance of some secret action follows a sequence of observable actions, and $\exists i \in [1, length(d)-1].NPr(d,i,a)$, meaning that the performance of some secret action is implied by a sequence of observable actions. Both properties are decidable, since $NPr(d,i,a)$ is decidable and it is sufficient to enumerate the cases.

An interesting property is $\exists d.NPr(d,i,a)$. This property holds if there is a sequence of observable actions implying a secret action. In this case we cannot enumerate the cases. Moreover, even if such a property would be decidable, we cannot enumerate to prove the property $\exists d.\exists i \in [1, length(d)-1].NPr(d,i,a)$.

So, one may consider weaker properties. As an example, the property $\exists d.\ length(d) \leq n$ and $NPr(d,i,a)$ considers only finite-length sequences of observable actions. (It is usually sufficient to observe a finite number of observable actions to describe time attacks on protocols). Note that the sequences d such that $length(d) \leq n$ are not finite because times are infinitely many.

We might also consider sequences d in $(\Sigma \times Interval)^*$ instead of $(\Sigma \times Time)^*$. This kind of sequences permit to consider more general behaviors. As an example, a possible sequence is $(a,[0,\infty))(b,[3,3])(c,[2,5))$, meaning that a is performed in the interval $[0,\infty)$, b is performed 3 units of time after a, and c is performed when a time in $[2,5)$ after b is elapsed.

Our aim is to study the decidability of such properties.

References

[1] Alur, R., and D.L. Dill: *A theory of timed automata.* Theoretical Computer Science **126** (1994), 183–235.

[2] Felten, E.W., and M.A. Schneider: Timing attacks on Web privacy. Proc. 7th ACM Conference on Computer and Communications Security, 25–32, 2000.

[3] Focardi, R., and R. Gorrieri: Automatic compositional verification of some security properties. Proc. Second International Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science 1055, Springer, Berlin, 1996, 167-186.

[4] Focardi, R., and R. Gorrieri: *A classification of security properties for process algebras.* Journal of Computer Security **3** (1995), 5–33.

[5] Focardi, R., R. Gorrieri, and F. Martinelli: Information flow analysis in a discrete-time process algebra. Proc. 13^{th} Computer Security Foundation Workshop, IEEE Computer Society Press, 2000.

[6] Volpano, D., and G. Smith: *Confinement properties for programming languages.* SIGACT News **29** (1998), 33–42.

[7] Smith, G., and D. Volpano: Secure information flow in a multi-threaded imperative language. Proc. ACM Symposium on Principles of Programming Languages, 1998, 355–364.