Process Environment Opacity

Damas P. GRUSKA

Institute of Informatics, Comenius University,
Mlynska dolina, 842 48 Bratislava, Slovakia,
gruska@fmph.uniba.sk.

Abstract. A security property called *process environment opacity* is formalized and studied. Properties of process's inputs (environments) are expressed by predicate and it is required that an intruder cannot learn their validity by observing process's behavior. This basic idea can be formulated in various ways. The resulting security properties are compared.

Keywords: security, opacity, process algebras, information flow

1 Introduction

Information flow based security properties (see [GM82]) assume an absence of any information flow between private and public systems activities. This means, that systems are considered to be secure if from observations of their public activities no information about private activities can be deduced. This approach has found many reformulations and among them opacity (see [BKR04,BKMR06]) could be considered as the most general one and many other security properties could be viewed as its special cases (see, for example, [Gru07]). A predicate is opaque if for any trace of a system for which it holds, there exists another trace for which it does not hold and the both traces are indistinguishable for an observer (which is expressed by an observation function). This means that the observer (intruder) cannot be sure whether a trace for which the predicate holds has been performed or not. In [Gru15] opacity is modified (the resulting property is called process opacity) in such a way that instead of process's traces we focus on properties of reachable states. Hence we assume an intruder who is not primarily interested whether some sequence of actions performed by a given process has some given property but we consider an intruder who wants to discover whether this process reaches a state which satisfied some given (classified) predicate. It turns out that in this way we could capture some new security flaws. Both opacity and process opacity are based on fixed (static) security policy which is not changed during system computation. In [Gru16] process opacity for dynamic security policies is formalized. All of these properties are defined in the framework of process algebras, but neither opacity, which represents security of process's traces nor process opacity, which represents security of resulting states, can capture security of process's inputs, similarly as security of inputs is defined for programming languages.

The aim of this paper is to formalize security of process's inputs. We adopt the approach which appeared in [SS15] for programming languages but we reformulate it for timed process algebras. Inputs will be modeled by processes which we call process's environments. The resulting security property will be called *process environment opacity*. It assumes that for every process's environment for which a given predicate holds there exists equivalent one, for which it does not hold and the process with these environments leads to equivalent states. This means that an intruder cannot learn validity of the predicate over process's inputs. We will study this security property as well as its variants. Moreover, we compare them with other security properties known in the literature.

The paper is organized as follows. In Section 2 we describe the timed process algebra TPA which will be used as a basic formalism. In Section 3 we present some notion on information flow security based on opacity and in the next section process environment opacity is defined, studied and compared to other security properties. Section 5 contains discussion and plans for future work.

2 Timed Process Algebra

In this section we define Timed Process Algebra, TPA for short. TPA is based on Milner's CCS but the special time action t which expresses elapsing of (discrete) time is added. The presented language is a slight simplification of Timed Security Process Algebra introduced in [FGM00]. We omit an explicit idling operator ι used in tSPA and instead of this we allow implicit idling of processes. Hence processes can perform either "enforced idling" by performing t actions which are explicitly expressed in their descriptions or "voluntary idling" (i.e. for example, the process $a.Nil$ can perform t action since it is not contained the process specification). But in both cases internal communications have priority to action t in the parallel composition. Moreover we do not divide actions into private and public ones as it is in tSPA. TPA differs also from the tCryptoSPA (see [GM04]). TPA does not use value passing and strictly preserves *time determinancy* in case of choice operator $+$ what is not the case of tCryptoSPA.

To define the language TPA, we first assume a set of atomic action symbols A not containing symbols τ and t, and such that for every $a \in A$ there exists $\overline{a} \in A$ and $\overline{\overline{a}} = a$. We define $Act = A \cup \{\tau\}, At = A \cup \{t\}, Actt = Act \cup \{t\}$. We assume that $a, b, \ldots$ range over A, $u, v, \ldots$ range over Act, and $x, y \ldots$ range over $Actt$. Assume the signature $\Sigma = \bigcup_{n \in \{0,1,2\}} \Sigma_n$, where

$$\begin{aligned}
\Sigma_0 &= \{Nil\} \\
\Sigma_1 &= \{x. \mid x \in A \cup \{t\}\} \cup \{[S] \mid S \text{ is a relabeling function}\} \\
&\quad \cup \{\backslash M \mid M \subseteq A\} \\
\Sigma_2 &= \{|, +\}
\end{aligned}$$

with the agreement to write unary action operators in prefix form, the unary operators $[S], \backslash M$ in postfix form, and the rest of operators in infix form. Relabeling functions, $S : Actt \to Actt$ are such that $\overline{S(a)} = S(\bar{a})$ for $a \in A, S(\tau) = \tau$ and $S(t) = t$.

The set of TPA terms over the signature Σ is defined by the following BNF notation:

$$P \ ::= \ X \ \mid \ op(P_1, P_2, \ldots P_n) \ \mid \ \mu X P$$

where $X \in Var$, Var is a set of process variables, $P, P_1, \ldots P_n$ are TPA terms, $\mu X-$ is the binding construct, $op \in \Sigma$.

The set of CCS terms consists of TPA terms without t action. We will use an usual definition of opened and closed terms where μX is the only binding operator. Closed terms which are t-guarded (each occurrence of X is within some subterm $t.A$ i.e. between any two t actions only finitely many non timed actions can be performed) are called TPA processes.

We give a structural operational semantics of terms by means of labeled transition systems. The set of terms represents a set of states, labels are actions from $Actt$. The transition relation $\to$ is a subset of TPA $\times$ $Actt$ $\times$ TPA. We write $P \stackrel{x}{\to} P'$ instead of $(P, x, P') \in \to$ and $P \stackrel{x}{\not\to}$ if there is no P' such that $P \stackrel{x}{\to} P'$. The meaning of the expression $P \stackrel{x}{\to} P'$ is that the term P can evolve to P' by performing action x, by $P \stackrel{x}{\to}$ we will denote that there exists a term P' such that $P \stackrel{x}{\to} P'$. We define the transition relation as the least relation satisfying the inference rules for CCS plus the following inference rules:

$$\frac{}{Nil \stackrel{t}{\to} Nil} \ A1 \qquad \frac{}{u.P \stackrel{t}{\to} u.P} \ A2$$

$$\frac{P \stackrel{t}{\to} P', Q \stackrel{t}{\to} Q', P \mid Q \stackrel{\tau}{\not\to}}{P \mid Q \stackrel{t}{\to} P' \mid Q'} \ Pa \quad \frac{P \stackrel{t}{\to} P', Q \stackrel{t}{\to} Q'}{P + Q \stackrel{t}{\to} P' + Q'} \ S$$

Here we mention the rules that are new with respect to CCS. Axioms $A1, A2$ allow arbitrary idling. Concurrent processes can idle only if there is no possibility of an internal communication (Pa). A run of time is deterministic (S) i.e. performing of t action does not lead to the choice between summands of $+$. In the definition of the labeled transition system we have used negative premises (see Pa). In general this may lead to problems, for example with consistency of the defined system. We avoid these dangers by making derivations of τ independent of derivations of t. For an explanation and details see [Gro90].

For $s = x_1.x_2.\dots.x_n, x_i \in Actt$ we write $P \xrightarrow{s}$ instead of $P \xrightarrow{x_1}\xrightarrow{x_2}\dots\xrightarrow{x_n}$ and we say that s is a trace of P. The set of all traces of P will be denoted by $Tr(P)$. By ϵ we will denote the empty sequence of actions, by $Succ(P)$ we will denote the set of all successors of P i.e. $Succ(P) = \{P' | P \xrightarrow{s} P', s \in Actt^*\}$. If set $Succ(P)$ is finite we say that P is finite state process. We define modified transitions $\overset{x}{\Rightarrow}_M$ which "hide" actions from M. Formally, we will write $P \overset{x}{\Rightarrow}_M P'$ for $M \subseteq Actt$ iff $P \xrightarrow{s_1}\xrightarrow{x}\xrightarrow{s_2} P'$ for $s_1, s_2 \in M^\star$ and $P \overset{s}{\Rightarrow}_M$ instead of $P \overset{x_1}{\Rightarrow}_M \overset{x_2}{\Rightarrow}_M \dots \overset{x_n}{\Rightarrow}_M$. We will write $P \overset{x}{\Rightarrow}_M$ if there exists P' such that $P \overset{x}{\Rightarrow}_M P'$. We will write $P \overset{\widehat{x}}{\Rightarrow}_M P'$ instead of $P \overset{\epsilon}{\Rightarrow}_M P'$ if $x \in M$. Note that $\overset{x}{\Rightarrow}_M$ is defined for arbitrary action but in definitions of security properties we will use it for actions (or sequence of actions) not belonging to M. We can the extend the definition of $\Rightarrow_M$ for sequences of actions similarly to $\xrightarrow{s}$. By $Sort(P)$ we will denote the set of actions from A which can be performed by P. The set of weak timed traces of process P is defined as $Tr_w(P) = \{s \in (A \cup \{t\})^\star | \exists P'.P \overset{s}{\Rightarrow}_{\{\tau\}} P'\}$. Two process P and Q are weakly timed trace equivalent ($P \simeq_w Q$) iff $Tr_w(P) = Tr_w(Q)$. We conclude this section with definitions M-bisimulation and weak timed trace equivalence.

Definition 1. *Let $(TPA, Actt, \rightarrow)$ be a labelled transition system (LTS). A relation $\Re \subseteq TPA \times TPA$ is called a* M-bisimulation *if it is symmetric and it satisfies the following condition: if $(P, Q) \in \Re$ and $P \xrightarrow{x} P', x \in Actt$ then there exists a process Q' such that $Q \overset{\widehat{x}}{\Rightarrow}_M Q'$ and $(P', Q') \in \Re$. Two processes P, Q are M-bisimilar, abbreviated $P \approx_M Q$, if there exists a M-bisimulation relating P and Q.*

3 Information flow and opacity

In this section we will present motivations for new security concepts which will be introduced in the next section. First we define an absence-of-information-flow property called Strong Nondeterministic Non-Interference

(SNNI, for short, see [FGM00]). Suppose that all actions are divided into two groups, namely public (low level) actions L and private (high level) actions H. It is assumed that $L \cup H = A$. Process P has SNNI property (we will write $P \in SNNI$) if $P \setminus H$ behaves like P for which all high level actions are hidden (by action τ) for an observer. To express this hiding we introduce hiding operator $P/M, M \subseteq A$, for which it holds if $P \xrightarrow{a} P'$ then $P/M \xrightarrow{a} P'/M$ whenever $a \notin M \cup \bar{M}$ and $P/M \xrightarrow{\tau} P'/M$ whenever $a \in M \cup \bar{M}$. Formally, we say that P has SNNI property, and we write $P \in SNNI$ iff $P \setminus H \simeq_w P/H$. SNNI property assumes an intruder who tries to learn whether a private action was performed by a given process while (s)he can observe only public ones. If this cannot be done then the process has SNNI property. A generalization of this concept is given by opacity (this concept was exploited in [BKR04], [BKMR06] and [Gru07] in a framework of Petri Nets, transition systems and process algebras, respectively. Actions are not divided into public and private ones at the system description level, as it is done for example in [GM04], but a more general concept of observations and predicates is exploited. A predicate is opaque if for any trace of a system for which it holds, there exists another trace for which it does not hold and the both traces are indistinguishable for an observer (which is expressed by an observation function). This means that the observer (intruder) cannot say whether a trace for which the predicate holds has been performed or not.

First we assume an observation function i.e. a function $\mathcal{O} : Actt^\star \rightarrow \Theta^\star$, where Θ is a set of elements called observables (note that we have no other requirements on $\mathcal{O}$ except that it has to be total, i.e. defined for every sequence of actions). Now suppose that we have some security property. This might be an execution of one or more classified actions, an execution of actions in a particular classified order which should be kept hidden, etc. Suppose that this property is expressed by a predicate ϕ over sequences. We would like to know whether an observer can deduce the validity of the property ϕ just by observing sequences of actions from $Actt^\star$ performed by system of interest. The observer cannot deduce the validity of ϕ if there are two sequences $w, w' \in Actt^\star$ such that $\phi(w), \neg\phi(w')$ and the sequences cannot be distinguished by the observer i.e. $obs(w) = obs(w')$. We formalize this concept by the notion of opacity.

Definition 2 (Opacity). *Given process P, a predicate ϕ over $Actt^\star$ is opaque w.r.t. the observation function $\mathcal{O}$ if for every sequence w, $w \in Tr(P)$ such that $\phi(w)$ holds and $\mathcal{O}(w) \neq \epsilon$, there exists a sequence $w', w' \in Tr(P)$ such that $\neg\phi(w')$ holds and $\mathcal{O}(w) = \mathcal{O}(w')$. The set of processes* *for which the predicate ϕ is opaque with respect to $\mathcal{O}$ will be denoted by $Op(\phi, \mathcal{O})$.*

$$\begin{array}{ll} P & \stackrel{w}{\Longrightarrow} \\ & \quad \mathcal{O}(w) = \mathcal{O}(w') \ \texttt{and} \ \phi(w), \neg\phi(w') \\ P & \stackrel{w'}{\Longrightarrow} \end{array}$$

Fig. 1. Opacity

Now let us assume that an intruder tries to discover, instead of a property over process's traces, whether a given process can reach a state with some given property expressed by a (total) predicate. This might be process deadlock, capability to execute only traces s with time length less then n, capability to perform at the same time actions form a given set, incapacity to idle (to perform t action) etc. Again we do not put any restriction on such predicates but we only assume that they are consistent with some suitable behavioral equivalence. The formal definition follows.

Definition 3. *We say that the predicate ϕ over processes is consistent with respect to relation $\cong$ if whenever $P \cong P'$ then $\phi(P) \Leftrightarrow \phi(P')$.*

As consistency relation $\cong$ we could take bisimulation ($\approx_{\emptyset}$), weak bisimulation ($\approx_{\{\tau\}}$) or any other suitable equivalence. A special class of such predicates are such ones (denoted as $\phi^Q_{\cong}$) which are defined by a given process Q and equivalence relation $\cong$ i.e. $\phi^Q_{\cong}(P)$ holds iff $P \cong Q$.

We suppose that the intruder can observe only some activities performed by the process. Hence we suppose that there is a set of public actions which can be observed and a set of hidden (not necessarily private) actions. To model observations we exploit the relation $\stackrel{s}{\Rightarrow}_M$. The formal definition of process opacity (see [Gru15]) is the following.

Definition 4 (Process Opacity). *Given process P, a predicate ϕ over processes is process opaque w.r.t. the set M if whenever $P \stackrel{s}{\Rightarrow}_M P'$ for $s \in (Actt \setminus M)^*$ and $\phi(P')$ holds then there exists P'' such that $P \stackrel{s}{\Rightarrow}_M P''$ and $\neg\phi(P'')$ holds. The set of processes for which the predicate ϕ is process opaque w.r.t. to the M will be denoted by $POp(\phi, M)$.*

Note that if $P \cong P'$ then $P \in PPOp(\phi, M) \Leftrightarrow P' \in POp(\phi, M)$ whenever ϕ is consistent with respect to $\cong$ and $\cong$ is such that it a subset of the trace equivalence (defined as $\simeq_w$ but instead of $\stackrel{s}{\Rightarrow}_{\{\tau\}}$ we use $\stackrel{s}{\Rightarrow}_{\emptyset}$).

$$P \quad \overset{s}{\Longrightarrow}_M \quad \phi(P')$$

$$P \quad \overset{s}{\Longrightarrow}_M \quad \neg\phi(P'')$$

Fig. 2. Process opacity

Properties opacity and process opacity are depicted in Fig. 1 and 2, respectively. In the former case, a set of actions performed by a process is of interest, say classified, in the later case, it is a property of reachable states which is protected by process opacity.

4 Process environment opacity

Now we will formulate security properties which protect inputs of a given process. We start with property called *process environment opacity* which is a modification of the the property formulated for programming languages which appeared in [SS15]. Process's inputs will be modeled also by processes, called environments, running in parallel with the process. Moreover, we force (by restriction operator) the process to communicate exclusively with its environment by means of channels contained in given set M. Formally, we assume a set of environments $\mathcal{E}$, i.e. processes which represent possible input environments for a given process. We extend definition of predicates over environments to sets of environments $\mathcal{K}, \mathcal{K} \subseteq \mathcal{E}$ in the following way. $\phi(\mathcal{K})$ holds iff $\phi(E)$ holds for every $E \in \mathcal{K}$. Moreover, we assume two equivalence relations over processes, $\cong^1, \cong^2$, which express capability of an intruder to distinguish among processes.

Definition 5 (Process Environment Opacity). *Given process P and set of environments $\mathcal{E}$, a predicate ϕ over processes is environment opaque w.r.t. the set M and $\cong^1, \cong^2$ if whenever $(P|E) \setminus M \overset{s}{\Rightarrow} R$ for $s \in (Actt \setminus M)^+$ and $\phi(E)$ holds then there exists E' such that $(P|E') \setminus M \overset{s}{\Rightarrow} R'$, $E \cong^1 E'$, $\neg\phi(E')$ holds and $R \cong^2 R'$. The set of processes for which the predicate ϕ is environment opaque w.r.t. to the M and $\cong^1, \cong^2$ will be denoted by $PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$.*

Note that process environment opacity says nothing about security of inputs if $\phi(\mathcal{E})$ or $\neg\phi(\mathcal{E})$ holds. As regards choice of relations $\cong^1, \cong^2$ it depends on capability of possible intruders. For example, it could be bisimulation, trace equivalence or just the relation equal to $TPA \times TPA$. Schematically, process environment opacity is depicted in Fig. 3.

$$\begin{array}{llll}
(P|E) \setminus M \overset{s}{\Longrightarrow} & R, \texttt{ and } \phi(E) & & E \\
& \cong^2 & & \cong^1 \\
(P|E') \setminus M \overset{s}{\Longrightarrow} & R' \texttt{ and } \neg\phi(E') & & E'
\end{array}$$

Fig. 3. Process environment opacity

4.1 Properties of process environment opacity

In this subsection we will formulate some properties of process environment opacity.

Proposition 1. *Let $\phi_1 \Rightarrow \phi_2$. Then*

$$PEOp(\phi_2, \mathcal{E}, M, \cong^1, \cong^2) \subseteq PEOp(\phi_1, \mathcal{E}, M, \cong^1, \cong^2).$$

Proof. Let $P \in PEOp(\phi_2, \mathcal{E}, M, \cong^1, \cong^2)$ and $(P|E) \setminus M \overset{s}{\Rightarrow} R$ for $s \in (Actt \setminus M)^+$ and $\phi_1(P')$ holds. Then also $\phi_2(P')$ holds since $\phi_1 \Rightarrow \phi_2$. We know that there exists E' such that $(P|E') \setminus M \overset{s}{\Rightarrow} R'$, $E \cong^1 E'$ and $\neg\phi_2(E')$ holds and $R \cong^2 R'$. Since $\neg\phi_2 \Rightarrow \neg\phi_1$ we have that also $\neg\phi_1(E')$ holds and hence $P \in PEOp(\phi_1, \mathcal{E}, M, \cong^1, \cong^2)$. □

Proposition 2. *Let $\mathcal{E}_1 \subseteq \mathcal{E}_2$ and $\neg\phi(\mathcal{E}_2 \setminus \mathcal{E}_1)$. Then*

$$PEOp(\phi, \mathcal{E}_1, M, \cong^1, \cong^2) \subseteq PEOp(\phi, \mathcal{E}_2, M, \cong^1, \cong^2).$$

Proof. Sketch. Let $P \in PEOp(\phi, \mathcal{E}_1, M, \cong^1, \cong^2)$ and $(P|E) \setminus M \overset{s}{\Rightarrow} R$ for $s \in (Actt \setminus M)^+$ and $\phi(E)$ holds for $E \in \mathcal{E}_1$. Since $\neg\phi(\mathcal{E}_2 \setminus \mathcal{E}_1)$ all environments for which ϕ holds are from $\mathcal{E}_1$. We know that there exists $E' \in \mathcal{E}_1$, $E \cong^1 E'$, such that $(P|E') \setminus M \overset{s}{\Rightarrow} R'$ and $\neg\phi(E')$ holds and $R \cong^2 R'$. Since $\mathcal{E}_1 \subseteq \mathcal{E}_2$ we have that also $P \in PEOp(\phi, \mathcal{E}_2, M, \cong^1, \cong^2)$. □

Proposition 3. *Let $\cong^1 \subseteq \cong^{1'}$. Then*

$$PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2) \subseteq PEOp(\phi, \mathcal{E}, M, \cong^{1'}, \cong^2).$$

Proof. Let $P \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$ and $(P|E) \setminus M \overset{s}{\Rightarrow} R$ for $s \in (Actt \setminus M)^+$ and $\phi(E)$ holds for $E \in \mathcal{E}$. We now that there exists E' $E' \in \mathcal{E}$ such that $(P|E') \setminus M \overset{s}{\Rightarrow} R'$, $E \cong^1 E'$ and $\neg\phi(E')$ holds and $R \cong^2 R'$. Since $\cong^1 \subseteq \cong^{1'}$ we have that also $E \cong^{1'} E'$ and hence we have also $P \in PEOp(\phi, \mathcal{E}, M, \cong^{1'}, \cong^2)$. □

Proposition 4. *Let $\cong^2 \subseteq \cong^{2'}$. Then*

$$PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2) \subseteq PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^{2'}).$$

Proof. Let $P \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$ and $(P|E) \setminus M \overset{s}{\Rightarrow} R$ for $s \in (Actt \setminus M)^+$ and $\phi(E)$ holds for $E \in \mathcal{E}$. We now that there exists E' $E' \in \mathcal{E}$ such that $(P|E') \setminus M \overset{s}{\Rightarrow} R'$, $E \cong^1 E'$ and $\neg\phi(E')$ holds and $R \cong^2 R'$. Since $\cong^2 \subseteq \cong^{2'}$ we have that also $R \cong^{2'} R'$ and hence we have also $P \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^{2'})$. □

As regards compositionality properties of process environment opacity, only a few of them hold. In general, process environment opacity is not compositional even for the prefix operator. Let $P \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$ for ϕ such that $\phi(A)$ holds iff $A \xrightarrow{\bar{x}}$ and $x \in M$. Clearly, $(x.P|A') \setminus M$ cannot perform any action for A' such that $\neg\phi(A')$, i.e. $x.P \notin PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$. A bit more sophisticated argument leads to the same result also for $x \notin M$. On the other side, process environment opacity is compositional with respect to non-deterministic choice as it is stated by the following Proposition.

Proposition 5. *Let* $P, Q \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$. *Then*

$$P + Q \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2).$$

Proof. Let $(P+Q|E) \setminus M \overset{s}{\Rightarrow}_M R$ for $s \in (Actt \setminus M)^+$ and $\phi(E)$ holds. Then without loss of generality we can assume that $(P|E) \setminus M \overset{s}{\Rightarrow}_M R$. Since $P \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$ there exists E' such that $(P|E') \setminus M \overset{s}{\Rightarrow}_M R'$, $E \cong^1 E'$ and $\neg\phi(E')$ holds and $R \cong^2 R'$. Hence also $(P+Q|E') \setminus M \overset{s}{\Rightarrow}_M R'$ what implies $P + Q \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$.

□

As regards the rest of TPA operators, the situation is similar as for the prefix operator.

4.2 Process environment opacity and Non-Deducibility

In this subsection we will compare process environment opacity with Non-Deducibility on Composition (NDC for short, see in [FGM03]) which is a widely studied security property. It is based on the idea of checking the system against all high level potential interactions, representing every possible high level process. System is NDC if for every high level user A, the low level view of the behaviour of P is not modified (in terms of trace equivalence) by the presence of A. The idea of NDC can be formulated as follows.

Definition 6. (NDC) $P \in NDC$ *iff for every* $A, Sort(A) \subseteq H \cup \{\tau, t\}$

$$(P|A) \setminus H \simeq_w P \setminus H.$$

Now we will formulate relationship between NDC and $PEOp(\phi, \mathcal{E}, H, \cong^1, \cong^2)$ properties. Note that this relationship is similar to the one which appeared in [SS15] for programming languages.

Proposition 6. *$P \in NDC$ iff $P \in PEOp(\phi, \mathcal{E}, H, \cong^1, \cong^2)$ for $\mathcal{E} = \{E \in TPA | Sort(A) \subseteq H\}$, $\cong^2 = TPA \times TPA$ and for every ϕ such that every kernel of $\cong^1$ contains with every process E from $\mathcal{E}$ for which $\phi(E)$ holds also process E' for which $\neg\phi(E')$ holds.*

Proof. Sketch. Let $P \in NDC$ that means that $(P|E) \setminus H \simeq_w (P|E') \setminus H$ for every two E, E' such that $Sort(E) \subseteq H, Sort(E') \subseteq H$. But by the assumption we know that for every ϕ such that for every E for which $\phi(E)$ holds there exists E' for which $\neg\phi(E')$ holds and $E \cong^1 E'$, and this implies that $P \in PEOp(\phi, \mathcal{E}, H, \cong^1, \cong^2)$.

Now suppose that $P \notin NDC$ i.e. there exist E, E' such that $(P|E) \setminus H \not\simeq_w (P|E') \setminus H$. Without loss of generality we can assume that $(P|E) \setminus H \overset{s}{\Rightarrow}$ but $(P|E') \setminus H \overset{s}{\not\Rightarrow}$. Hence we can choose predicate ϕ such that $\phi(E)$ holds and only environment for which ϕ does not hold is equal to E'. Hence $P \notin PEOp(\phi, \mathcal{E}, H, \cong^1, \cong^2)$. □

4.3 Variants of process environment opacity

In this subsection we will formulate several variants of process environment opacity. We will start with its persistent variant.

Definition 7 (Persistent Process Environment Opacity). *Given process P and set of environments $\mathcal{E}$, a predicate ϕ over processes is environment persistently opaque w.r.t. the set M and $\cong^1, \cong^2$ if whenever for every $P', P' \in Succ(P)$ we have $P' \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$.*

The set of processes for which the predicate ϕ is persistent environment opaque w.r.t. to the M and $\cong^1, \cong^2$ will be denoted by $PPEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$.

The relationship between process environment opacity and its persistent variant is formulated in by the following Proposition.

Proposition 7. *$PPEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2) \subseteq PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$*

Proof. The proof follows directly from Definitions 5 and 7. In general, we cannot guarantee that this inclusion is proper. □

Process environment opacity expresses security of inputs for which given predicate holds but does not say anything about those ones for which it does not hold, i.e. says nothing about security of $\neg\phi$. So we define a stronger variant of process environment opacity.

Definition 8 (Strong Process Environment Opacity). *Given process P and set of environments $\mathcal{E}$, a predicate ϕ over processes is strongly environment persistently opaque w.r.t. the set M and $\cong^1, \cong^2$ if whenever $P \in PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$ and $P \in PEOp(\neg\phi, \mathcal{E}, M, \cong^1, \cong^2)$.*

The set of processes for which the predicate ϕ is strongly environment opaque w.r.t. to the M and $\cong^1, \cong^2$ will be denoted by $SPEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$.

Proposition 8. $SPEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2) \subseteq PEOp(\phi, \mathcal{E}, M, \cong^1, \cong^2)$

Proof. Again the proof follows directly from Definitions 5 and 8. and we cannot guarantee that the inclusion is proper as well. □

The most of properties stated for process environment opacity can be formulated also for its strong variant, some with a slight modification.

Till now we have not expected that environments are fully "consumed" by processes. For cases when an intruder can see only results of complete computations we have to modify definition of process environment opacity. Suppose that all process in $\mathcal{E}$ are finite and moreover, the last action performed before reaching (sub)state Nil is a new auxiliary action $\surd$. That means that for every environment E every occurrence of Nil is preceded by $\surd$ i.e. every Nil is always a subterm of a term $\surd.Nil$. For example, $E = a.b.c.\surd.Nil + b.a.\surd.Nil$ is such a process. We indicate corresponding environments by $\mathcal{E}_{\surd}$.

Definition 9 (Termination Process Environment Opacity). *Given process P and set of environments $\mathcal{E}_{\surd}$, a predicate ϕ over processes is environment opaque w.r.t. the set M and $\cong^1, \cong^2$ if whenever $(P|E)\setminus M \overset{s.\surd}{\Rightarrow} R$ for $s \in (Actt \setminus M)^+$ and $\phi(E)$ holds then there exists E' such that $(P|E') \setminus M \overset{s.\surd}{\Rightarrow} R'$, $E \cong^1 E'$ and $\neg\phi(E')$ holds and $R \cong^2 R'$. The set of processes for which the predicate ϕ is termination environment opaque w.r.t. to the M and $\cong^1, \cong^2$ will be denoted by $TPEOp(\phi, \mathcal{E}_{\surd}, M, \cong^1, \cong^2)$.*

As regards decidability of proposed security properties, in general they are undecidable. To obtain properties which are decidable, we need restrictions on predicates which should be decidable, limitation to finite set of environments and/or restrictions on equivalences $\cong^1, \cong^2$.

5 Discussion and further work

We have presented the new security concept called process environment opacity and we have formalized it in the timed process algebra framework.

It expresses different aspects of processes behaviour as opacity or process opacity. Instead of sequences of actions or resulting processes, it focuses on process's inputs and their properties. We have presented some properties of the resulting security property as well as its variants. Since we worked with timed process algebras we can model security with respect to limited time length of an attack, with a limited number of attempts to perform an attack and so on. Besides investigation of decidable variants of the proposed properties we also plan to study variants of process environment opacity which assume intruders which are not only observers but can actively interact with the systems to be attacked.

References

[BKR04] Bryans J., M. Koutny and P. Ryan: Modelling non-deducibility using Petri Nets. Proc. of the 2nd International Workshop on Security Issues with Petri Nets and other Computational Models, 2004.

[BKMR06] Bryans J., M. Koutny, L. Mazare and P. Ryan: Opacity Generalised to Transition Systems. In Proceedings of the Formal Aspects in Security and Trust, LNCS 3866, Springer, Berlin, 2006.

[DHS15] van Delft B., S. Hunt, D. Sands: Very Static Enforcement of Dynamic Policies. In Principles of Security and Trust, LMCS 9036, 2015.

[FGM03] Focardi R., R. Gorrieri and F. Martinelli: Real-Time information flow analysis. IEEE Journal on Selected Areas in Communications 21 (2003).

[FGM00] 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.

[GM04] Gorrieri R. and F. Martinelli: A simple framework for real-time cryptographic protocol analysis with compositional proof rules. Science of Computer Programming, Volume 50, Issues 13, 2004.

[GM82] Goguen J.A. and J. Meseguer: Security Policies and Security Models. Proc. of IEEE Symposium on Security and Privacy, 1982.

[GM04] Gorrieri R. and F. Martinelli: A simple framework for real-time cryptographic protocol analysis with compositional proof rules. Science of Computer Programming, Volume 50, Issues 13, 2004.

[Gro90] Groote, J. F.: Transition Systems Specification with Negative Premises. Proc. of CONCUR'90, Springer Verlag, Berlin, LNCS 458, 1990.

[Gru15] Gruska D.P.: Process Opacity for Timed Process Algebra. In Perspectives of System Informatics, LNCS 8974, 2015.

[Gru16] Gruska D.P.: Dynamic Security Policies and Process Opacity, In Proc of Perspectives of System Informatics, to appeare in LNCS, 2016.

[Gru10] Gruska D.P.: Process Algebra Contexts and Security Properties. Fundamenta Informaticae, vol. 102, Number 1, 2010.

[Gru07] Gruska D.P.: Observation Based System Security. Fundamenta Informaticae, vol. 79, Numbers 3-4, 2007.

[KS83] Kanellakis, P. C. and S.A. Smolka: CCS expressions, finite state processes, and three problems of equivalence. Proc. of The second annual ACM symposium on Principles of distributed computing, ACM, 1983.

[SS15] Schoepe D. and A. Sabelfeld: Understanding and Enforcing Opacity. Proc. of IEEE 28th Computer Security Foundations Symposium, 2015.