Fundamenta Informaticae 101 (2010) 1–14
DOI 10.3233/FI-2010-256
IOS Press

Process Algebra Contexts and Security Properties *

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

Abstract. A general framework for defining security properties is presented. It allows us to model many traditional security properties as well as to define new ones. The framework is based on process algebras contexts and processes relations. By appropriate choice of both of them we can model also probabilistic and quantified security properties.

Keywords: information flow, opacity, surprisal, uncertainty, security, quantified information flow

1. Introduction

The aim of this paper is to present a general framework which allows us to define various information flow based security properties. The framework is based on (timed and timed probabilistic) process algebras and relations on them. First we express the information flow by means of observation functions and by opacity ([3]). The observation functions can hide some system activities, can express unprecise observations and so on. The information flow will be expressed by opacity. A predicate is opaque if from any observation of system activities an observer cannot deduce whether the predicate holds or it does not hold. In general, opacity is undecidable, roughly speaking due to unlimited power of observational functions and the corresponding predicate. To overcome this problem we model both the observation function and the predicate by processes. We define concept *process defined security property* and we show that also many other security properties as NDC (see [8]), SNNI (see [7]), *Secrecy* (see [2], and hence also Perfect Security Property, see [20]) are special cases of this concept.

Traditional security properties are frequently criticized for being either too restrictive or too benevolent. For example, usually they consider a standard access control process to be insecure since there

*Work supported by the grant VEGA 1/0688/10
†Address for correspondence: Institute of Informatics, Comenius University, Mlynska dolina, 842 48 Bratislava, Slovakia

is always some (even very small) information flow for an attacker which tries to learn a password. To overcome these disadvantages we propose some alternative security properties which are based on probabilistic process alegebra and information theory. We show that they can be again defined by suitable timed probabilistic contexts. In this sense the presented work can be seen as a continuation of work presented in [12] (probabilistic opacity) and [11] (quantification of security properties).

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 and investigate general notion of information flow for different observation functions and security requirements. In Section 4 we define and study process defined security properties which are in Section 5 extended to probabilistic setting.

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 the Timed Security Process Algebra introduced in [7]. We omit the 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". But in the both cases internal communications have priority to action t in the case of the parallel operator. Moreover we do not divide actions into private and public ones as it is in tSPA. TPA differs also from the tCryptoSPA (see [10]). TPA does not use value passing and strictly preserves *time determinacy* 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 $Act = A \cup \{\tau\}, Actt = Act \cup \{t\}$. 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 subexpression $t.A$ i.e. between any two t actions only finitely many non

timed actions can be performed) are called TPA processes. Note that Nil will be often omitted from processes descriptions and hence, for example, instead of $a.b.Nil$ we will write just $a.b$.

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 $\rightarrow$ is a subset of TPA $\times$ $Actt$ $\times$ TPA. We write $P \xrightarrow{x} P'$ instead of $(P, x, P') \in \rightarrow$ and $P \not\xrightarrow{x}$ if there is no P' such that $P \xrightarrow{x} P'$. The meaning of the expression $P \xrightarrow{x} P'$ is that the term P can evolve to P' by performing action x, by $P \xrightarrow{x}$ we will denote that there exists a term P' such that $P \xrightarrow{x} P'$. We define the transition relation as the least relation satisfying the inference rules for CCS plus the following inference rules:

$$\frac{}{Nil \xrightarrow{t} Nil} \quad A1 \qquad \frac{}{u.P \xrightarrow{t} u.P} \quad A2$$

$$\frac{P \xrightarrow{t} P', Q \xrightarrow{t} Q', P \mid Q \not\xrightarrow{\tau}}{P \mid Q \xrightarrow{t} P' \mid Q'} \quad Pa1 \qquad \frac{P \xrightarrow{t} P', Q \xrightarrow{t} Q'}{P + Q \xrightarrow{t} P' + Q'} \quad S$$

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 ($Pa1$). A run of time is deterministic (S). Regarding behavioral relations we will work with the timed version of weak trace equivalence. Note that here we will use also a concept of observations which contain complete information which includes also τ actions and not just actions from A and t action as it is in [7]. For $s = x_1.x_2.\ldots.x_n, x_i \in Actt$ we write $P \xrightarrow{s}$ instead of $P \xrightarrow{x_1}\xrightarrow{x_2} \cdots \xrightarrow{x_n}$ and we say that s is a trace of P. The set of all traces of P will be denoted by $Tr(P)$. We will write $P \stackrel{x}{\Rightarrow} P'$ iff $P(\xrightarrow{\tau})^* \xrightarrow{x} (\xrightarrow{\tau})^* P'$ and $P \stackrel{s}{\Rightarrow}$ instead of $P \stackrel{x_1}{\Rightarrow}\stackrel{x_2}{\Rightarrow} \cdots \stackrel{x_n}{\Rightarrow}$. By ϵ we will denote the empty sequence of actions, by $Succ(P)$ we will denote the set of all successors of P and $Sort(P) = \{x | P \xrightarrow{s.x}$ for some $s \in Actt^\star$ and $x \neq \tau\}$. If the set $Succ(P)$ is finite we say that P is finite state.

Let $s \in Actt^\star$. By $|s|$ we will denote the length of s i.e. a number of action contained in s. By $s|_B$ we will denote the sequence obtained from s by removing all actions not belonging to B. For example, $|s|_{\{t\}}|$ denote a number of occurrences of t in s, i.e. time length of s.

Definition 2.1. The set of weak timed traces of process P is defined as $Tr_w(P) = \{s \in (A \cup \{t\})^\star | \exists P'.P \stackrel{s}{\Rightarrow} P'\}$. Two process P and Q are weakly timed trace equivalent ($P \approx_w Q$) iff $Tr_w(P) = Tr_w(Q)$ and process Q is a trace simulation of P ($P \preceq_w Q$) iff $Tr_w(P) \subseteq Tr_w(Q)$.

3. Information Flow

To formalize information flow we do not divide actions into public and private ones at the system description level, as it is done for example in [10], but we use a more general concept of observation and opacity. In [3] and [4] opacity was exploited for transition systems and Petri nets, respectively.

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 3.1. (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 $Op^{\phi}_{\mathcal{O}}$.

The notion of opacity is rather general. With its help many other security properties can be defined (anonymity, non-interference etc. see [3]). On the other side, opacity is undecidable even for the simplest possible observation function, namely for the constant one, and for finite state processes (see [15]). To obtain not only decidable but mainly more realistic security property we will modify opacity in several ways.

First, let us consider attackers which cannot see performing of internal actions but on the other side they can always see elapsing of time. So we assume that $t \in \Theta$ and if $\mathcal{O}(s) = o$ and $s = s_1.t^{n_1}.\ldots.s_{k-1}.t^{n_{k-1}}.s_k, s_i \in Act^\star$ then $o = o_1.t^{n_1}.\ldots.o_{k-1}.t^{n_{k-1}}.o_k, o_i \in (\Theta \setminus \{t\})^\star$. From now on we will consider only this type of observational functions $\mathcal{O}$.

Undecidability of opacity is caused by considering very powerful attackers (very powerful observation functions) and predicates which are possibly very difficult to be computed. So to overcome these obstacles, we will model both observational functions and predicates by processes.

Let us consider the set of atomic actions A. By the ghost set of actions to A (we will denote it by A^g) we will call the set $\{a^g | a \in A\}$ (we assume that $A \cap A^g = \emptyset$). For a sequence of action $s, s \in Actt^\star$ we will denote the corresponding sequence of ghost action by s^g where every elementary action is replaced by its ghost action. Note that actions t and τ have no a special ghost counterpart as well as a special unique action $\surd$ indicating a successful computation and which extends the set of actions ($\surd \in Actt$) i.e. $t^g = t, \tau^g = \tau, \surd^g = \surd$. By A^{gg} we will denote ghost actions of A^g i.e. $A^{gg} = (A^g)^g$.

Definition 3.2. The set of successful weak timed traces of process P is defined as $Tr_{w\surd}(P) = \{s.\surd | s \in (A \cup \{t\})^\star$ such that $P \stackrel{s.\surd}{\Rightarrow}\}$. Two process P and Q are successfully weakly timed trace equivalent ($P \approx_{w\surd} Q$) iff $Tr_{w\surd}(P) = Tr_{w\surd}(Q)$ and process Q is a successful trace simulation of P ($P \preceq_{w\surd} Q$) iff $Tr_{w\surd}(P) \subseteq Tr_{w\surd}(Q)$.

Definition 3.3. Process F_ϕ is called process definition of predicate ϕ over sequences of actions if for every $s \in A^\star$ such that $\phi(s)$ it holds

$$P \stackrel{s}{\Rightarrow} \text{ iff } (P|F_\phi) \setminus Sort(P) \stackrel{s^g.\surd}{\Rightarrow}$$

Example 3.1. Let $\phi(s)$ holds iff $s = s_1.h.s_2, h \in H$ i.e. if s contains a private action from the set private action H. Then the following process

$$F_\phi = \mu X.(\sum_{x \in Actt \setminus H} x.x^g.X + \sum_{x \in H} x.x^g.F')$$

where

$$F' = \mu X.(\sum_{x \in Actt} x.x^g.X + \surd.Nil)$$

is the process definition of predicate ϕ.

Example 3.2. Let $\phi^{n,m}(s)$ for $1 < n < m$ holds iff $s = s_1.h.s_2.h'.s_3, h, h' \in H$ such that $n \leq |s_2|_{\{t\}}| \leq m$ and $|s_2|_H| = \epsilon$, i.e. $\phi^{n,m}(s)$ holds if s contains two private actions from H and time elapsing between their occurrences is between n and m time units.

Then the following process

$$F_\phi = \mu X.(\sum_{x \in Actt} x.x^g.X + \sum_{x \in H} x.x^g.F')$$

where

$$F' = \mu X.(\sum_{x \notin H} x.x^g.X + t.F_1),$$

$$F_i = \mu X.(\sum_{x \notin H} x.x^g.X + t.F_{i+1})$$

for $i < n$ and

$$F_i = \mu X.(\sum_{x \notin H} x.x^g.X + t.F'_{i+1})$$

for $i = n$,

$$F'_i = \mu X.(\sum_{x \notin H} x.x^g.X + \sum_{x \in H} x.x^g.F'' + t.F'_{i+1})$$

for $i < m$ and

$$F'' = \mu X.(\sum x.x^g.X + t.X + \surd.Nil)$$

is the process definition of predicate $\phi^{n,m}$.

Now we will define also observation functions by processes. Without loss of generality we can suppose that the set of observables $\mathcal{O}$ is a subset of $Actt$.

Definition 3.4. Process $F_\mathcal{O}$ is called process definition of observation function $\mathcal{O}$ on sequences of actions if for every $s \in Actt^\star$ such that $\mathcal{O}(s) = o$ it holds

$$P \stackrel{s}{\Rightarrow} \text{ iff } (P|F_\mathcal{O}) \setminus Sort(P) \stackrel{o^g}{\Rightarrow}$$

Example 3.3. Let $\mathcal{O}(s)$ is such that that all actions from the set of private actions H are not seen i.e. $s|_{Actt\setminus H} \approx_w \mathcal{O}(s)$. Then the following process

$$F_{\mathcal{O}} = \mu X. \sum_{x \in H} x.X$$

is the process definition of observation function $\mathcal{O}$.

Theorem 3.1. Let $F_{\mathcal{O}}$ is process definition of observation function $\mathcal{O}$, F_ϕ and $F_{\neg\phi}$ are process definitions of predicates ϕ and $\neg\phi$, respectively. Then $P \in Op^\phi_{\mathcal{O}}$ iff

$$(((P|F_\phi) \setminus Sort(P))|F_{\mathcal{O}}) \setminus Sort(P^g) \preceq_{w\surd} (((P|F_{\neg\phi}) \setminus Sort(P))|F_{\mathcal{O}}) \setminus Sort(P^g).$$

Proof:
Let $P \in Op^\phi_{\mathcal{O}}$ and $F_{\mathcal{O}}$ is process definition of observation function $\mathcal{O}$, F_ϕ and $F_{\neg\phi}$ are process definitions of predicates ϕ and $\neg\phi$, respectively. Let w is a trace of P such that $\phi(w)$ holds and $\mathcal{O}(w)$ is non empty sequence. Then there exists w' (Definition 4.2) such that $\neg\phi(w)$ holds and $\mathcal{O}(w) = \mathcal{O}(w')$. Then from Definitions 3.3 and 3.4 we have $(((P|F_\phi)\setminus Sort(P))|F_{\mathcal{O}})\setminus Sort(P^g) \preceq_{w\surd} (((P|F_{\neg\phi})\setminus Sort(P))|F_{\mathcal{O}})\setminus Sort(P^g)$. The oposite direction of the proof is similar. □

Note that since $Sort(P)) \cap Sort(F_{\mathcal{O}}) = \emptyset$ we have $(((P|F_\phi) \setminus Sort(P))|F_{\mathcal{O}}) \setminus Sort(P^g) = (P|F_\phi|F_{\mathcal{O}}) \setminus (Sort(P^g) \cup Sort(P))$.

4. Process defined information flow

Theorem 3.1 leads to the following concept of process defined security. By context we mean a process term with placeholder $\mathcal{H}$. Formally, the set of TPA contexts over the signature Σ is defined by the following BNF notation:

$$C \ ::= \ \mathcal{H} \ | \ X \ | \ op(C_1, C_2, \ldots C_n) \ | \ \mu X C$$

where $X \in Var$, Var is a set of process variables, $C, C_1, \ldots C_n$ are TPA contexts, $\mu X-$ is the binding construct, $op \in \Sigma$ and $\mathcal{H}$ is the place holder. By $C(P)$ we denote process obtained from process context C and process P by substituting P by place holder $\mathcal{H}$, i.e. $C(P) = C[P/\mathcal{H}]$.

Definition 4.1. Security property S is process defined if there are two process contexts C_1, C_2 and a relation between processes R such that for every process $P \in TPA$ it holds $P \in S$ iff $(C_1(P), C_1(P)) \in R$.

Clearly, from Theorem 3.1 we see that opacity is process defined property for precess defined observation function $\mathcal{O}$ and predicates ϕ and $\neg\phi$, respectively. The corresponding contexts are $C_1 = (((\mathcal{H}|F_\phi) \setminus A)|F_{\mathcal{O}}) \setminus A^g$, $C_2 = (((\mathcal{H}|F_{\neg\phi}) \setminus A)|F_{\mathcal{O}}) \setminus A^g$, respectively and $R = \preceq_{w\surd}$.

The definition of opacity (see Definition 4.2) of predicate ϕ is asymmetric in the sense that if $\phi(w)$ does not hold than it is not required that there exists another trace for which it holds (in general $Op^\phi_{\mathcal{O}} \neq Op^{\neg\phi}_{\mathcal{O}}$). This means that opacity says something to an intruder which tries to detect only validity of ϕ (if it is opaque, than validity cannot be detected) but not its non-validity i.e. it says nothing about predicate $\neg\phi$. Hence we define strong variant of opacity.

Definition 4.2. (Strong Opacity)
Given process P, a predicate ϕ over $Actt^\star$ is strongly 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 $\mathcal{O}(w) = \mathcal{O}(w')$. Moreover, for for every sequence w, $w \in Tr(P)$ such that $\neg\phi(w)$ holds and $\mathcal{O}(w) \neq \epsilon$, there exists a sequence $w', w' \in Tr(P)$ such that $\phi(w')$ 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 $sOp^{\phi}_{\mathcal{O}}$.

Theorem 4.1. Strong opacity $sOp^{\phi}_{\mathcal{O}}$ is process defined if $F_{\mathcal{O}}$ is process definition of observation function $\mathcal{O}$, F_ϕ and $F_{\neg\phi}$ are process definitions of predicates ϕ and $\neg\phi$, respectively.

Proof:
The proof follows from the fact that $P \in sOp^{\phi}_{\mathcal{O}}$ iff $(((P|F_\phi) \setminus Sort(P))|F_{\mathcal{O}}) \setminus Sort(P^g) \approx_{w\surd} (((P|F_{\neg\phi}) \setminus Sort(P))|F_{\mathcal{O}}) \setminus Sort(P^g)$. □

Now we show process definability of Strong Nondeterministic Non-Interference (SNNI, for short). We recall its definition (see [7]). Suppose that all actions are divided in two groups, namely public (low level) actions L and private (high level) actions H i.e. $A = L \cup H, L \cap H = \emptyset$. Then process P has SNNI property if $P \setminus H$ behaves like P for which all high level actions are hidden for an observer. To express this hiding we introduce hiding= operator $P/M, M \subseteq A$, for which 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 $a \in M \cup \bar{M}$. Formal definition of SNNI follows.

Definition 4.3. Let $P \in TPA$. Then $P \in SNNI$ iff $P \setminus H \approx_w P/H$.

Theorem 4.2. Property $SNNI$ is process defined.

Proof:
Let $F_{\mathcal{O}}$ is the process defined in Example 3.3 and let $C_1 = (\mathcal{H}) \setminus H$, $C_2 = (\mathcal{H}|F_{\mathcal{O}}) \setminus H$ and $R = \approx_w$. It is easy to see that $P \in SNNI$ iff $(C_1(P), C_2(P)) \in R$. □

Note. Let us consider processes F_ϕ defined in Example 3.1 and $F_{\mathcal{O}}$ defined in Example 3.3. It is easy to check that SNNI property can be seen as a special case of opacity for corresponding process defined predicate ϕ and process defined observation function $\mathcal{O}$ (see also [13]).

Non-Deducibility on Composition (NDC for short, see in [8]) is another 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 4.4. (NDC) $P \in NDC$ iff for every $A, Sort(A) \subseteq H \cup \{\tau, t\}$

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

Theorem 4.3. Property NDC is process defined.

Proof:
Let $C_1 = (\mathcal{H}|Top) \setminus H, C_2 = (\mathcal{H}) \setminus H$, and $R = \approx_w$ where $Top = \mu X. \sum_{h \in H} h.X$. In [8] it is proved that to show NDC property it is enough to check it with respect to so called the most powerful attacker (Top). Then it holds $P \in NDC$ iff $(C_1(P), C_2(P)) \in R$, and hence property NDC is process defined. □

NDC property is based on an idea that process P communicates with (one) high level user or alternatively has high level inputs from the top level. By process defined security we can refine this concept by allowing more high level users nested in various*secrecy*, was introduced in [2] for labeled transition systems. Now we repeat its definition slightly modified for processes. Function IP*-inferable properties*, is a function that, given a trace w, a property ϕ over traces and an equivalence relation R over traces, represents the knowledge of the observer about the property ϕ from w. $IP(w, \phi, R) = \top$ if $\forall w' : (w, w') \in R \implies \phi(w')$, $IP(w, \phi, R) = \bot$ if $\forall w' : (w, w') \in R \implies \neg\phi(w')$ and $IP(w, \phi, R) = m$ otherwise, where $\top, \bot, m$ are three different symbols. Let φ is predicate over traces specifying a subset of traces of interest. Now we are ready to formalize *secrecy*.

Definition 4.5. (Secrecy) Let P be a process, ϕ and φ be two properties over traces. The property ϕ is a secret in φ for P w.r.t. R if for all w such that $\varphi(w)$ holds, we have $IP(w, \phi, R) = m$.

It can be shown that secrecy can capture several standard information properties such as noninterference or Perfect Security Property (see [20]). subprocesses (see [14]) of P or that P can send/receive high level outputs/inputs on several nested levels not necessary on the top level. These situations could be modeled by contexts of the form $C = (\mathcal{H}S_1 \dots S_n|Top_1| \dots Top_n)$ where S_i are relabeling functions and Top_i processes similar to Top but exploiting a dedicated alphabet given by S_i.

Another very general security property, called *secrecy*, was introduced in [2] for labeled transition systems. Now we repeat its definition slightly modified for processes. Function IP*-inferable properties*, is a function that, given a trace w, a property ϕ over traces and an equivalence relation R over traces, represents the knowledge of the observer about the property ϕ from w. $IP(w, \phi, R) = \top$ if $\forall w' : (w, w') \in R \implies \phi(w')$, $IP(w, \phi, R) = \bot$ if $\forall w' : (w, w') \in R \implies \neg\phi(w')$ and $IP(w, \phi, R) = m$ otherwise, where $\top, \bot$ and m are three different symbols. Let φ is predicate over traces specifying a subset of traces of interest. Now we are ready to formalize *secrecy*.

Definition 4.6. (Secrecy) Let P be a process, ϕ and φ be two properties over traces. The property ϕ is a secret in φ for P w.r.t. R if for all w such that $\varphi(w)$ holds, we have $IP(w, \phi, R) = m$.

It can be shown that secrecy can capture several standard information properties such as noninterference or Perfect Security Property (see [20]).

Similarly to opacity if we can restrict secrecy to predicates $\phi, \neg\phi$ and φ which are process defined then secrecy becomes precess defined property.

Theorem 4.4. Secrecy is process defined if corresponding predicates $\phi, \neg\phi$ and φ which are process defined.

Proof:
Let $F_\phi, F_{\neg\phi}, F_\varphi$ are corresponding process defined predicates $\phi, \neg\phi$ and φ, respectively. Corresponding contexts are as follows: $C_1 = (\mathcal{H}|F'_\phi|F'_\varphi) \setminus \{Sort(F_\phi) \cup Sort(F_\varphi))\}$ and $C_2 = (\mathcal{H}|F'_{\neg\phi}|F'_\varphi) \setminus \{Sort(F_{\neg\phi}) \cup Sort(F_\varphi)\}$ where processes with comas use relabeled ghost alphabets (see Definition 3.3). □

Process definability of a security property allows us to use traditional tools developed for process algebras. Moreover in many cases, for example for finite state contexts and processes, corresponding security property is decidable. We repeat that opacity is not decidable even for finite state processes but it is not true for process defined opacity.

Theorem 4.5. Let $F_\mathcal{O}$ is process definition of observation function $\mathcal{O}$, F_ϕ and $F_{\neg\phi}$ are process definitions of predicates ϕ and $\neg\phi$, respectively and $F_\mathcal{O}$, F_ϕ and $F_{\neg\phi}$ are finite state processes. Then $P \in Op^\phi_\mathcal{O}$ is decidable for finite state processes.

Proof:
From Theorem 3.1 we know that the corresponding contexts are finite state contexts.

□

Note that similar theorem can be formulated also for strong opacity. By process defined security concept we can also model many other non-standard security properties.

Example 4.1. Let us assume that an attacker cannot observe processes for longer than n time units from its start. That means that in that case processes are considered to be secure if the attack cannot be performed on sequences s shorter than n times units ($|s|_{\{t\}}| < n$. We can model this situation by adding the following process to the context given in Theorem 3.1.

$$F_{\mathcal{O}_n} = t^n.\surd.Nil.$$

Example 4.2. Let $\mathcal{O}(s)$ is such that that all actions from the set of private actions H are not seen as in Example 3.3 and moreover elapsing of one time unit can be seen with unprecision up to three time units, for example, due to some delays of an interconnection network.

Then the following process

$$F_\mathcal{O} = \mu X.\sum_{x\in H} x.X + \mu X.\sum_{x\in A\setminus H} x.x^g.X + \mu X.t.(X + t.t.X + t.t.t.X)$$

is the process definition of observation function $\mathcal{O}$.

It is easy to see that above mentioned properties - say of time limited attacks and time unprecise observations, are process defined.

A special class of process defined security properties are those ones which are defined by means of relation which is a congruence. For such properties we have the following compositional theorem.

Theorem 4.6. Let $\mathcal{S}$ is process defined security property such that the corresponding relation R is a congruence. Let $P \in \mathcal{S}$. Then we have $Q \in \mathcal{S}$ for every $Q, (P, Q) \in R$.

Proof:
Straightforward from definition of process definability and congruency properties. □

Moreover, to check security of processes which are defined by means of one of usual process relations we can exploit existing software tools.

Traditional security properties as opacity, SNNI or NDC are frequently criticized to be either too benevolent or too restrictive. In the former case they might assign a process to be secure despite the fact that practically its security is very limited and in the later case, they refuse as insecure processes generally considered to be secure. For example, usually they consider a standard access control process to be insecure since there is always some (even very small) information flow for an attacker which tries to learn a password - at least after each attempt a space of possible passwords is reduced. On the other side they might accept a process as secure despite the fact that an attacker can learn, for example, a whole private key except one bit and hence the size of the space of possible private keys is reduced to two.

There are several ways how to deal with the above mentioned situations in the framework of process algebras. For example, we can modify concept of opacity in such a way that it will be defined not for predicates over traces but over mappings $\phi(s) \rightarrow \{0, \ldots, k\}$ (or predicates of many-valued logics) and it will be required that for every s such that $\phi(s) = n$ there exists s' such that $\phi(s') < n$ such that $\mathcal{O}(s) = \mathcal{O}(s')$. Or we work with predicates $\phi(s) = \phi_1(s) \wedge \cdots \wedge \phi_k(s)$ or $\phi(s) = \phi_1(s_1) \wedge \cdots \wedge \phi_k(s_k)$ where $s = s_1 \ldots s_k$ and evaluate opacity of each ϕ_i separately. Each of predicates ϕ_i may say, for example, something about i-th bit of a private key.

Alternatively, we use probabilistic process algebras to deal with the above mentioned criticism.

5. Probabilistic process defined information flow

Security properties as opacity, SNNI or NDC do not take into account neither attacks which take into account changes of distribution of public outputs due to changes of private inputs nor security of systems based on very low probabilities for attacker to learn some private information. There are several ways how to solve the above mentioned problems. To explain them, to discuss them and to suggest an alternative approach, we need some preparatory work.

First we add probabilities to TPA calculus. We will follow alternating model (the approach presented in [16]) which is neither reactive nor generative nor stratified (see [17]) but instead of that it based on separation of probabilistic and nondeterministic transitions and states (see [12]). Probabilistic transitions are not associated with performing of actions but labeled only with probabilities. In so called probabilistic states a next transition is chosen according to probabilistic distribution. For example, process $a.(0.3.b.Nil \oplus 0.7.(a.Nil + b.Nil))$ can perform action a and after that it reaches the probabilistic state and from this state it can reach with probability 0.3 the state where only action b can be performed or with probability 0.7 it can reach the state where it can perform either a or b.

Note that resented approach slightly differs from the Calculus for Communicating with Time and Probabilities ([16]), where first probabilities are added to CCS and later time is added but without an explicit special time action.

Formally, to add probabilities to TPA calculus we introduce a new operator $\bigoplus_{i\in I} q_i.P_i, q_i$ being real numbers in $(0,1]$ such that $\sum_{i\in I} q_i = 1$. Processes which can perform as the first action probabilistic transition will be called probabilistic processes or states (to stress that P is non-probabilistic process we will sometimes write P_N if necessary). Hence we require that all P_i processes in $\bigoplus_{i\in I} q_i.P_i$ and in $P_1 + P_2$ are non-probabilistic ones. By pTPA we will denote the set of all probabilistic and non-probabilist processes and all definitions and notations for TPA processes are extended for pTPA ones. We need new transition rules for pTPA processes. We mention only three rules which are significantly different from those ones for TPA.

$$\frac{}{P_N \xrightarrow{1} P_N} \quad A3 \qquad\qquad \frac{}{\bigoplus_{i\in I} q_i.P_i \xrightarrow{q_i} P_i} \quad A4$$

$$\frac{P \xrightarrow{q} P', Q \xrightarrow{r} Q'}{P \mid Q \xrightarrow{q.r} P' \mid Q'} \quad Pa2$$

For probabilistic choice we have the rule $A4$ and for a probabilistic transition of two processes running in parallel we have the rule $Pa2$. The technical rule $A3$ enables parallel run of probabilistic and non-probabilistic processes by allowing to non-probabilistic processes to perform $\xrightarrow{1}$ transition and hence the rule $Pa2$ could be applied.

Introducing probabilities to process algebras usually causes several technical complications. For example, an application of the restriction operator to probabilistic process may lead to unwanted deadlock states or to a situation when a sum of probabilities of all outgoing transitions is less than 1. A normalization is usually applied to overcome similar situations. We do not need to resolve such situations on the level of pTPA calculus since we will use only relative probabilities of sets of computations. To compute these probabilities normalization will be also exploited but only as the very last step.

To add probabilities to traces one has to assign to each possible path of computation resulting to a given sequence of action corresponding probability and to sum up all of them. There are several ways how to do that we refer reader to [18, 9, 12].

Some security properties which are defined in the previous section could be directly and meaningfully translated to probabilistic setting. For example, if we take probabilistic processes and probabilistic version of the weak trace equivalence ($\approx_{pw}$) which require that process have the same traces with the same probabilities we obtain probabilistic version of SNNI and NDC property.

Definition 5.1. Let $P \in pTPA$. Then $P \in pSNNI$ iff $P \setminus H \approx_{pw} P/H$.

Definition 5.2. (pNDC) $P \in pNDC$ iff for every $A \in pTPA, Sort(A) \subseteq H \cup \{\tau, t\}$

$$(P|A) \setminus H \approx_{pw} P \setminus H.$$

Security property pSNNI says that an observer which can observe process P for a long time cannot distinguish whether private actions are not sees or are restricted.

Security property pNDC has also a natural motivation - process P has this property if private process A cannot influence not only set of possible traces but also their distribution.

Note that property pNDC is defined also for non probabilistic properties. Moreover, it is easy to see that for every $P, P \in TPA$ we have that if $P \in pNDC$ then $P \in NDC$.

In general, we define probabilistically process defined security properties as follows.

Definition 5.3. Security property S is probabilistically process defined if there are two probabilistic process contexts C_1^p, C_2^p and a relation between probabilistic processes R^p such that for every process $P \in pTPA$ it holds $P \in S$ iff $(C_1^p(P), C_2^p(P)) \in R^p$.

Note that probabilistic process definability can be applied also on non probabilistic processes. By choosing either suitable probabilistic contexts or by suitable probabilistic process relation we can model many naturally motivated security notions. For example, we can choose some probabilistic process relation to admit some level of tolerance behind which an attacker cannot observe processes (see for example ([1, 12]). We can model unprecise observations or observations which might have some known distribution of errors etc.

Example 5.1. Let $\mathcal{O}(s)$ be an observation function from Example 4.2 with probabilities associated with unprecise observations. Then the following timed probabilistic process

$$F_{\mathcal{O}} = \mu X. \sum_{x \in H} x.X + \mu X. \sum_{x \in A \setminus H} x.x^g.X + \mu X.t.(0.7.X \oplus 0.2.t.t.X \oplus 0.1.t.t.t.X)$$

is the probabilistic process definition of observation function $\mathcal{O}$.

Another approach how to add some quantities to security properties is represented by various measures of an amount of information flow. To express quantity of information flow we will exploit Schannon information theory (see [19]). Let X be a discrete random variable and let x ranges over the set of values which X may take. By $p(x)$ we will denote probability that X takes the value x. Self-information (or surprisal) is a measure of the information content associated with the outcome of the random variable X. It is defined as $\mathcal{H}(x) = \log_b \frac{1}{p(x)}$. We put $\mathcal{H}(x) = \infty$ if $p(x) = 0$. The information entropy (also called self-information or a measure of uncertainty) of the variable X is denoted $\mathcal{H}(X)$ and is defined as $\mathcal{H}(X) = \sum_x p(x). \log_b \frac{1}{p(x)}$. We define $p(x). \log_b \frac{1}{p(x)} = 0$ if $p(x) = 0$. We will work with the base b of $\log_b$ equal to 2 and hence the unit of the information entropy will be one bit. Sometimes we will write $\mathcal{H}(p_1, \ldots, p_n)$ instead of $\mathcal{H}(X)$ if probabilities of values of X are $p_1, \ldots, p_n$. Given two random variables X and Y, the mutual information between them, written $\mathcal{I}(X;Y)$, is defined as $\mathcal{I}(X;Y) = \sum_x \sum_y p(x,y). \log \frac{p(x,y)}{p(x).p(y)}$. It can be easily shown that $\mathcal{I}(X;Y) = \mathcal{H}(X) + \mathcal{H}(Y) - \mathcal{H}(X,Y) = \mathcal{H}(X) - \mathcal{H}(X|Y) = \mathcal{H}(Y) - \mathcal{H}(Y|X)$. Conditional entropy of X given knowledge of Y is defined as $\mathcal{H}(X|Y) = \sum_y p(y).\mathcal{H}(X|Y = y)$, and conditional mutual information between X and Y given knowledge of Z is defined as $\mathcal{I}(X;Y|Z) = \mathcal{H}(Y|Z) - \mathcal{H}(Y|X,Z)$.

Information theory can be used to measure an amount of information which can attacker gain (see [5] and [11] for quantification of information flow in imperative languages and for process algebras, respectively). Information flow is quantified as a surprisal of value of private variable or predicates validity or its entropy. Alternatively, it can be quantified as a conditional mutual information flow between private and public inputs (H_{in}, L_{in}) and the resulting distribution of a corresponding public output (L_{out}) as $\mathcal{I}(H_{in}, L_{out}|L_{in})$ or as difference between entropy of H_{in} before and after "attack" i.e. observing L_{out}

knowing L_{in}. That means as $H(H_{in}|L_{in}) - H(H_{in}|L_{in}, L_{out})$. In all this cases it is not assumed that an attacker might have some belief about private data or actions what is not always the case.

Suppose that an intruder tries to learn a secrete password. (S)he starts an attack with a prebelief that the password is very likely P. This prebelief is given by information received from past experiences, obtained form some cooperator etc. Say that there are 100 possible passwords and the attackers prebelief B is expressed by the following probability distribution $p(P) = 0.9$ and $p(x) = 1/990$ for $x \neq P$. Hence entropy of B, $\mathcal{H}(B)$ is about 1. Suppose that the attacker tries as the first the password P and it fails. After that the attacker has to change his/her prebelief. Now a new belief B' is such that all possible password have the equal probability $1/99$ and $\mathcal{H}(B')$ is roughly seven times higher. Hence according to traditional view there is no information flow since uncertainty is higher after an attack what contradicts our intuition (in [6] this problem is formalized for a simple imperative language).

Here we formalize this concept in the presented framework. Suppose that attacker beliefs is expressed by probabilistic process A which can perform only sequences of public input action $l_1 \dots l_n$ of length n each with a probability corresponding to attackers belief. On the other side let G be similar process but with probabilities for each sequence to be equal. By similar technique as it was used in Examples 3.1 and 3.3 we can force process to perform always the whole sequence of inputs. Then there is an information flow for attacker with belief A if uncertainty of ϕ under observation o for $(P|A) \setminus L_{in}$ is lower than for $(P|G) \setminus L_{in}$. This can be expressed by contexts and $C_1' = C_1((\mathcal{H}|A) \setminus L_{in})$, $C_2' = C_2((\mathcal{H}|G) \setminus L_{in})$, C_1, C_2 being context from opacity definition (see Theorem 3.1) and relation $\preceq_{pw\surd}$ on processes such that $P \preceq_{pw\surd} Q$ if for every weak trace of P there is a weak trace of Q such that corresponding overall probabilities are not higher for Q. Hence the property is probabilistic process defined.

6. Conclusions

We have presented the concept *process defined security property* and we have showed that many of already studied information flow based security properties (NDC, SNNI, Secrecy, Perfect Security Property) can be seen as its special cases. This concept offers not only an uniform framework for security theory but can be used to model more elaborated security properties than traditional ones and moreover, by careful choice of process contexts and process relation, we can obtain properties which can be effectively checked (note that in general, opacity is undecidable). By this concept we can also naturally model security with respect to limited time length of an attack, with a limited number of attempts to perform an attack and so on.

Moreover, by exploiting probabilistic process algebra contexts we can model various probabilistic security properties and quantify an amount of information flow as well.

The presented approach allows us to use also other types of process algebras enriched by operators expressing also other properties (space, distribution, networking architecture, processor or power consumption and so on) and in this way also other types of attacks which exploit these information to detect information flow through various covert channels can be described.

Process defined security limits us to predicates and observation functions (i.e. observers) which can be expressed by process algebra processes. In fact, this restriction does not represent any real limitation. Practically all predicates and observation function of interest (used in known attacks) can be described by finite state processes and there is even no need to exploit full universal power of process algebras. In

other words, it has no practical meaning to consider predicates and observation functions which cannot be computed.

References

[1] A. Aldini: Probabilistic Information Flow in a Process Algebra. CONCUR'01, Springer LNCS 2154, 2001.

[2] R. Alur, P. Černý and S. Zdancewic: Preserving Secrecy under Refinement. In Proc. of 33rd International Colloquium on Automata, Languages and Programming (ICALP), 2006.

[3] J. Bryans, M. Koutny, L. Mazare and P. Ryan: Opacity generalised to transition systems. International Journal of Information Security, Vol. 7, No. 6, 2008.

[4] J. Bryans, 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.

[5] D. Clark, S. Hunt and P. Malacaria: A Static Analysis for Quantifying the Information Flow in a Simple Imperative Programming Language. The Journal of Computer Security, 15(3). 2007.

[6] M.R. Clarkson, A.C. Myers, F.B. Schneider: Quantifying Information Flow with Beliefs. Journal of Computer Security, to appear, 2009.

[7] R. Focardi, 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.

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

[9] R. J. van Glabbeek, S. A. Smolka and B. Steffen: Reactive, Generative and Stratified Models of Probabilistic Processes Inf. Comput. 121(1): 59-80, 1995.

[10] R. Gorrieri and F. Martinelli: A simple framework for real-time cryptographic protocol analysis with compositional proof rules. Science of Computer Programming archive Volume 50, Issue 1-3, 2004.

[11] D.P. Gruska: Quantifying Security for Timed Process Algebras, Fundamenta Informaticae, vol. 93, Numbers 1-3, 2009.

[12] D.P. Gruska: Probabilistic Information Flow Security. Fundamenta Informaticae, vol. 85, Numbers 1-4, 2008.

[13] D.P. Gruska: Observation Based System Security, Fundamenta Informaticae, 79 (2007), Numbers 3-4, pp. 335-346, 2007.

[14] D.P. Gruska: Network Information Flow, Fundamenta Informaticae, Volume 72, Numbers 1-3, pp 167-180, 2006.

[15] D.P. Gruska: Information Flow in Timing Attacks. Proceedings CS&P'04, 2004.

[16] H. Hansson and B. Jonsson: A Calculus for Communicating Systems with Time and Probabilities. In Proceedings of 11th IEEE Real - Time Systems Symposium, Orlando, 1990.

[17] N. López and Núñez: An Overview of Probabilistic Process Algebras and their Equivalences. In Validation of Stochastic Systems, LNCS 2925, Springer-Verlag, Berlin, 2004.

[18] R. Segala and N. Lynch: Probabilistic Simulations for Probabilistic Processes. Nord. J. Comput. 2(2): 250-273, 1995.

[19] C. E. Shannon: A mathematical theory of communication. Bell System Technical Journal, vol. 27, 1948.

[20] A. Zakinthinos and E. S. Lee: A general theory of security properties. In Proc. of SP'97, 1997.