# A Simple Framework for Real-time Cryptographic Protocol Analysis with Compositional Proof Rules ⋆

Roberto Gorrieri[1] and Fabio Martinelli[2]

[1] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy.
e_mail: gorrieri@cs.unibo.it
[2] Istituto di Informatica e Telematica C.N.R., Pisa, Italy.
e_mail: Fabio.Martinelli@iit.cnr.it

**Abstract.** A real-time process algebra, enhanced with specific constructs for handling cryptographic primitives, is proposed to model cryptographic protocols in a simple way. We show that some security properties, such as authentication and secrecy, can be re-formulated in this timed setting. Moreover, we show that they can be seen as suitable instances of a general information flow-like scheme, called $tGNDC$, parametric w.r.t. the observational semantics of interest. We show that, when considering timed trace semantics, there exists a most powerful hostile environment (or enemy) that can try to compromise the protocol. Moreover, we present a couple of compositionality results for $tGNDC$, one of which is time dependent, and show their usefulness via a case study.

## 1 Introduction

In the last years there has been an increasing interest in the formal analysis of cryptographic protocols, as they have become the basic building blocks for many distributed services, such as home banking or electronic commerce. These analyzes have been very successful in many cases, uncovering subtle inaccuracies in many specifications of cryptographic protocols. However, such analyzes are usually restricted to very high abstractions of the real protocols, where concrete information about the timing of events are usually omitted (with some relevant exceptions as [2, 22]).

Our starting point is the work on $CryptoSPA$ [8, 10], which is an extension of $SPA$ [5] (a CCS-like process algebra with actions belonging to two different levels of confidentiality), with some new constructs for handling cryptographic primitives. On such a language a general schema for the definition of security properties, called $GNDC$, has been proposed [10]. It is based on the idea of checking the system against all the possible hostile environments. The general schema has the following form:

$$P \text{ satisfies } \mathcal{S}^{\alpha}_{\lhd} \text{ iff } \forall X \in Env : P || X \lhd \alpha(P)$$

⋆ This paper is a revised and extended version of [13]. This work has been partially supported by MURST Progetto "Metodi Formali per la Sicurezza" (MEFISTO); IST-FET project "Design Environments for Global ApplicationS (DEGAS)"; Microsoft Research Europe; by CNR project "Tecniche e Strumenti Software per l'analisi della sicurezza delle comunicazioni in applicazioni telematiche di interesse economico e sociale" and by a CSP grant for the project "SeTAPS II".

where the general property $\mathcal{S}_{\lhd}^{\alpha}$ requires that the system $P$ satisfies (via the behavioral pre-order $\lhd$) a specification $\alpha(P)$ when composed in parallel with any possible hostile environment (or enemy) $X$. The problem of the universal quantification is overcome when it is possible to show that there exists the "most powerful" enemy; hence, one check against the most powerful enemy is as discriminating as an infinity of different checks against all the possible enemies. This lucky case occurs when the behavioral pre-order $\lhd$ is a pre-congruence, e.g., for trace semantics.

The main goal of this paper is to show that the real-time information flow theory developed for $tSPA$ (a real-time extension of $SPA$ reported in [9]), can be extended to $CryptoSPA$, yielding $timedCryptoSPA$ ($tCryptoSPA$ for short). The main results from such an effort are the following:

- A language for describing cryptographic protocols, where information about the concrete timing of events is necessary, e.g., because of the presence of timeouts or time-stamps.
- A general scheme, called $tGNDC$, for describing uniformly the many security properties in a real-time setting; we will present three instances of such a general scheme, namely *timed authentication*, *timed integrity* and *timed secrecy*.
- Some specific results for the security properties based on semantics that are pre-congruences, such as the existence of a (real-time) most general enemy.

Moreover, we will propose some compositionality results, i.e., we will show some conditions under which secure real-time protocols can be safely composed. In particular, we show that under some conditions on the knowledge that a protocol accumulates during its lifetime, the $tGNDC$ property is compositional; in one case, such a condition is time dependent, hence more easily applicable to real-life case studies.

As a running example, we will consider the timed version of the Wide Mouther Frog protocol (WMF for short). The choice has been driven by its simplicity; nonetheless, the WMF protocol is expressive enough to discuss some interesting issues, e.g., the timed version of authentication, integrity and secrecy.

The paper is organized as follows: in Section 2 we define the $tCryptoSPA$ syntax, operational and behavioral semantics. In Section 3 we define the general schema $tGNDC$, hence the notion of hostile environment (or enemy) and we present some general results, such as the existence of a real-time most general enemy. In Section 4 we present some security properties, namely $tNDC$, *timed authentication*, *timed integrity* and *timed secrecy*. Section 5 reports some results about conditions for safe composition of real-time security protocols. In Section 6 we report some concluding remarks and comparison with related literature. Finally, in the Appendix, we give some detailed proofs of the papers.

## 2 The Model

In this section we present the model we will use for the specification of cryptographic protocols and security properties. It is a real-time extension of the *Cryptographic Security Process Algebra* ($CryptoSPA$ for short) proposed in [10, 8]. In $CryptoSPA$ it is possible to express qualitative ordering among events, while quantitative timing aspects cannot be expressed. Thus, we extend $CryptoSPA$ with operators that permit to express the elapsing of time.

### 2.1 The Language Syntax

We call the language *Timed Cryptographic Security Process Algebra* ($tCryptoSPA$ for short). Its syntax is based on the following elements:

- A set $Ch$ of channels, partitioned into a set $I$ of input channels (ranged over by $c$) and a set $O$ of output channels (ranged over by $\overline{c}$, the output corresponding to the input $c$);
- A set $\mathcal{M}$ of messages;
- A set $Var$ of variables, raged over by $x$;
- A set $Const$ of constants, ranged over by $A$.

The set $\mathcal{L}$ of tCryptoSPA terms (or processes) is defined as follows:

$$P ::= \mathbf{0} \mid c(x).P \mid \overline{c}e.P \mid \tau.P \mid tick.P \mid P + P \mid P||P \mid P \backslash L \mid$$

$$A(e_1, \ldots, e_n) \mid [\langle e_1, \ldots, e_r \rangle \vdash_{rule} x]P \mid \iota(P)$$

where $e, e_1, \ldots, e_r$ are messages or variables and $L$ is a set of channels. Both the operators $c(x).P$ and $[\langle e_1 \ldots e_r \rangle \vdash_{rule} x]P$ bind the variable $x$ in $P$.

Besides the standard (value-passing) CCS operators [19], we have an additional prefix action $tick$, used to model time elapsing, a delay operator $\iota(P)$, used to make lazy the initial actions of $P$, and the operator $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]P$ introduced in order to model message handling and cryptography. Informally, process $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]P$ tries to deduce a piece of information $z$ from the tuple of messages $\langle m_1 \ldots m_r \rangle$ through one application of rule $\vdash_{rule}$; if it succeeds, then it behaves like $P[z/x]$, otherwise it is stuck. See the next subsection for a more detailed explanation of inference rules.

The time model we use is known as the *fictitious clock* approach of, e.g., [23]. A global clock is supposed to be updated whenever all the processes agree on this, by globally synchronizing on action $tick$. All the other actions are assumed to take no time. This is reasonable if we choose a time unit such that the actual time of an action is negligible w.r.t. the time unit. Hence, the computation proceeds in lock-steps: between the two global synchronizations on action $tick$ (that represent the elapsing of one time unit), all the processes proceed asynchronously by performing durationless actions.

Let $Def : Const \longrightarrow \mathcal{L}$ be a set of defining equations of the form $A(x_1, \ldots, x_n) \stackrel{def}{=} P$, where $P$ may contain no free variables except $x_1, \ldots, x_n$, which must be distinct. Constants permit us to define recursive processes, but we have to be a bit careful in using them. A term $P$ is *closed* w.r.t. $Def$ if all the constants occurring in $P$ are defined in $Def$ (and, recursively, for their defining terms). A term $P$ is *guarded* w.r.t. $Def$ if all the constants occurring in $P$ (and, recursively, for their defining terms) occur in a prefix context [19].

The set $Act = \{c(m) \mid c \in I\} \cup \{\overline{c}m \mid \overline{c} \in O\} \cup \{\tau\} \cup \{tick\}$ of actions ($\tau$ is the internal, invisible action, $tick$ is the special action used to model time elapsing), ranged over by $a$ (with abuse of notation); we let $l$ range over $Act \backslash \{tick\}$. We call $\mathcal{P}$ the set of all the $tCryptoSPA$ *closed* terms (i.e., with no free variables), that are closed and guarded w.r.t. $Def$. We define $sort(P)$ to be the set of all the channels syntactically occurring in the term $P$. Moreover, for the sake of readability, we always omit the termination $\mathbf{0}$ at the end of process specifications, e.g., we write $a$ in place of $a.\mathbf{0}$.

We give an informal overview of $tCryptoSPA$ operators:

- $\mathbf{0}$ is a process that does nothing;
- $c(x).P$ represents the process that can get an input $m$ on channel $c$ behaving like $P$ where all the occurrences of $x$ are replaced by $m$ (written $P[m/x]$);
- $\overline{c}m.P$ is the process that can send $m$ on channel $c$, and then behaves like $P$;
- $\tau.P$ is the process that executes the invisible action $\tau$ and then behaves like $P$;
- $tick.P$ is a process willing to let one time unit pass and then behaves as $P$;
- $P_1 + P_2$ (*choice*) represents the nondeterministic choice between the two processes $P_1$ and $P_2$; time passes when both $P_1$ and $P_2$ are able to perform a *tick* action – and in such a case by performing $tick$ a configuration where both the derivatives of the summands can still be chosen is reached – or when only one of the two can perform $tick$ – and in such a case the other summand is discarded; moreover, $\tau$ prefixed summands have priority over $tick$ prefixed summands.
- $P_1||P_2$ (*parallel*) is the parallel composition of processes that can proceed in an asynchronous way but they must synchronize on complementary actions to make a communication, represented by a $\tau$. Both components must agree on performing a *tick* action, and this can be done even if a communication is possible;
- $P \backslash L$ is the process that cannot send and receive messages on channels in $L$; for all the other channels it behaves exactly like $P$;
- $A(m_1, \ldots, m_n)$ behaves like the respective defining term $P$ where all the variables $x_1, \ldots, x_n$ are replaced by the messages $m_1, \ldots, m_n$;
- $[\langle m_1, \ldots, m_r \rangle \vdash_{rule} x]P$ is the process used to model message handling and cryptography;
- $\iota(P)$ (*idling*) allows process $P$ to wait indefinitely. At every instant of time, if process $P$ performs an action $l$, then the whole system proceeds in this state, while dropping the idling operator.

### 2.2 The Operational Semantics of tCryptoSPA

The data handling part of the language consists of a set of inference rules used to deduce messages from other messages. We consider a set of relations among closed messages as: $\vdash_r \subseteq \mathcal{P}^{fin}(\mathcal{M}) \times \mathcal{M}$, where $r$ is the name of the rule. Given a set $\mathcal{R}$ of inference rules, we consider the deduction relation $\mathcal{D}^{\mathcal{R}} \subseteq \mathcal{P}^{fin}(\mathcal{M}) \times \mathcal{M}$. Given a finite set of closed messages, say $\phi$, then $(\phi, M) \in \mathcal{D}^{\mathcal{R}}$ if $M$ can be derived by iteratively applying the rules in $\mathcal{R}$. For the sake of simplicity, we assume that $\vdash_r$ (for each $r \in \mathcal{R}$) and $\mathcal{D}^{\mathcal{R}} \subseteq \mathcal{P}^{fin}(\mathcal{M}) \times \mathcal{M}$ are decidable.

Note that $tCryptoSPA$ syntax, its semantics and the results obtained are completely parametric with respect to the inference system used. We present in Figure 1 the same inference system of [10]. This inference system can combine two messages obtaining a pair (rule $\vdash_{pair}$); it can extract one message from a pair (rules $\vdash_{fst}$ and $\vdash_{snd}$);

$$\frac{m \quad m'}{(m,m')}(\vdash_{pair}) \quad \frac{(m,m')}{m}(\vdash_{fst}) \quad \frac{(m,m')}{m'}(\vdash_{snd})$$

$$\frac{m \quad k}{\{m\}_k}(\vdash_{enc}) \quad \frac{\{m\}_k \quad k}{m}(\vdash_{dec})$$

**Fig. 1.** An example inference system for shared key cryptography.

it can encrypt a message $m$ with a key $k$ obtaining $\{m\}_k$ and finally decrypt a message of the form $\{m\}_k$ only if it has the same key $k$ (rules $\vdash_{enc}$ and $\vdash_{dec}$). In this framework, cryptography is completely reliable, i.e., that a crypted message can be deciphered only by knowing the suitable decryption key.

In a similar way, the inference system can contain rules for handling the basic arithmetic operations and boolean relations among numbers.

*Example 1.* Natural numbers may be encoded by assuming a single value 0 and a function symbol $S$, with the following rule:

$$\frac{n}{S(n)} inc$$

So we will often use $n + 1$ to indicate the application of rule $inc$ with $n$ as argument. Similarly, we can define summations and other operations on natural numbers.

*Example 2.* We do not explictly define equality check among messages in the syntax. However, this can be implemented through the usage of the inference construct. E.g., consider rule $\frac{n \quad n}{Equal(n,n)} equal$. Then $[m = m']A$ (with the expected semantics) may be equivalently expressed as $[m \quad m' \vdash_{equal} y]A$ where $y$ does not occur in $A$. Similarly, we can define inequalities, e.g., $\leq$, among natural numbers.

The operational semantics of $tCryptoSPA$ is described by means of the *labelled transition system* (*lts*, for short) $\langle \mathcal{P}, Act, \{\stackrel{a}{\longrightarrow}\}_{a \in Act} \rangle$, where $\{\stackrel{a}{\longrightarrow}\}_{a \in Act}$ is the least relation between $tCryptoSPA$ processes induced by the axioms and inference rules of Figure 2. Such a relation is well-defined even if negative premises occur in a rule for the idling operator and in one rule for $+$, because the relation is *strictly stratifiable* [15]. Note that $tCryptoSPA$ is $tick$*-deterministic* i.e., the time elapsing never moves a process to two different states. The proof of the following proposition can be easily given by inspecting the operational rules. In particular, the first two rules of the idling operator and the rules for nondeterministic choice are the key rules enforcing time determinacy.

**Proposition 1.** *For every tCryptoSPA process P we have:*
*If* $P \stackrel{tick}{\longrightarrow} P'$ *and* $P \stackrel{tick}{\longrightarrow} P''$ *then* $P' = P''$.

*Example 3.* In $tCryptoSPA$ there are processes, such as $\mathbf{0}$, that do not allow time to proceed; hence, as rule $||_3$ for parallel composition forces a global synchronisation on $tick$ actions, the effect of composing a process $P$ with $\mathbf{0}$ is to prevent $P$ from letting

$$(input)\frac{m \in \mathcal{M}}{c(x).P \xrightarrow{c(m)} P[m/x]} \qquad (output)\frac{}{\overline{c}m.P \xrightarrow{\overline{c}m} P}$$

$$(internal)\frac{}{\tau.P \xrightarrow{\tau} P} \qquad (tick)\frac{}{tick.P \xrightarrow{tick} P}$$

$$(||_1)\frac{P_1 \xrightarrow{l} P_1'}{P_1||P_2 \xrightarrow{l} P_1'||P_2} \qquad (||_2)\frac{P_1 \xrightarrow{c(x)} P_1' \quad P_2 \xrightarrow{\overline{c}m} P_2'}{P_1||P_2 \xrightarrow{\tau} P_1'||P_2'}$$

$$(||_3)\frac{P_1 \xrightarrow{tick} P_1' \quad P_2 \xrightarrow{tick} P_2'}{P_1||P_2 \xrightarrow{tick} P_1'||P_2'} \qquad (\backslash L)\frac{P \xrightarrow{c(m)} P' \quad c \notin L}{P\backslash L \xrightarrow{c(m)} P'\backslash L}$$

$$(+_1)\frac{P_1 \xrightarrow{l} P_1'}{P_1 + P_2 \xrightarrow{l} P_1'} \qquad (+_2)\frac{P_1 \xrightarrow{tick} P_1' \quad P_2 \xrightarrow{tick} P_2'}{P_1 + P_2 \xrightarrow{tick} P_1' + P_2'}$$

$$(+_3)\frac{P_1 \xrightarrow{tick} P_1' \quad P_2 \not\xrightarrow{tick} \quad P_2 \not\xrightarrow{\tau}}{P_1 + P_2 \xrightarrow{tick} P_1'}$$

$$(Def)\frac{P[m_1/x_1, \ldots, m_n/x_n] \xrightarrow{a} P' \quad A(x_1, \ldots, x_n) \stackrel{def}{=} P}{A(m_1, \ldots, m_n) \xrightarrow{a} P'}$$

$$(\mathcal{D})\frac{\langle m_1, \ldots, m_r\rangle \vdash_{rule} m \quad P[m/x] \xrightarrow{a} P'}{[\langle m_1, \ldots, m_r\rangle \vdash_{rule} x]P \xrightarrow{a} P'}$$

$$(\mathcal{I}_1)\frac{P \not\xrightarrow{tick}}{\iota(P) \xrightarrow{tick} \iota(P)} \qquad (\mathcal{I}_2)\frac{P \xrightarrow{tick} P'}{\iota(P) \xrightarrow{tick} \iota(P')} \qquad (\mathcal{I}_3)\frac{P \xrightarrow{l} P'}{\iota(P) \xrightarrow{l} P'}$$

**Fig. 2.** Structured Operational Semantics for tCryptoSPA (symmetric rules for $+_1, +_3, ||_1, ||_2$ and $\backslash L$ are omitted).

time pass. In other words, $\mathbf{0}$ acts as a time annihilator for its parallel context. On the contrary, $\iota(\mathbf{0})$ is a process that, even if functionally terminated, let time to proceed indefinitely. Hence, $\iota(\mathbf{0})$ acts as a neutral element for parallel composition.

*Example 4.* Consider a process $P = (\iota(a)||\iota(\overline{a}))\backslash a$ that can perform any sequence (possibly empty) of $tick$ actions followed by a $\tau$. It is worth-observing that, contrary to $tSPA$ [9], we do not assume maximal communication progress, i.e., $\tau$'s do not have priority over $tick$ actions or, equivalently, a process cannot idle if it can perform a $\tau$. Hence, in $tSPA$ process $P$ can perform only the sequence $\tau$.

*Example 5.* We can easily model timeout constructs in $tCryptoSPA$.
Assume $n_1 \leq n_2$ and define a process

$$Time\_out(n_1, n_2, A, B) = tick^{n_1}.\iota(A) + tick^{n_2}.\tau.B$$

By looking at the rules for choice and idling, we see that $Time\_out(n_1, n_2, A, B)$ first performs a sequence of $n_1$ $tick$ actions; then, the system may perform other $n_2 - n_1$ $tick$ actions, unless $A$ resolves the choice by performing an action; instead if $A$ does nothing,

$$
\begin{aligned}
Clock(n) &= tick.Clock(n+1) + \overline{time}\, n.Clock(n) \\
A(K) &= time(T_a).tick.\overline{c}_{AS}\{B, T_a, K\}_{K_{as}}.\iota(\mathbf{0}) \\
B &= \iota(c_{BS}(y)[\langle y, K_{bs}\rangle \vdash_{dec} z]time(T_b).tick.[\langle z\rangle \vdash_{snd} t] \\
&\quad [t+1 \leq T_b][\langle z\rangle \vdash_{trd} w]\tau) \\
S &= tick.c_{AS}(u).[\langle u, K_{as}\rangle \vdash_{dec} z]time(T_s).tick.[\langle z\rangle \vdash_{snd} t][t+1 \leq T_s] \\
&\quad [\langle z\rangle \vdash_{trd} x]\overline{c}_{BS}\{A, T_s, x\}_{K_{bs}}.tick \\
P(K_{ab}) &= (Clock(0)||A(K_{ab})||B||S)\backslash\{c_{AS}, c_{BS}, time\}
\end{aligned}
$$

**Fig. 3.** $tCryptoSPA$ specification for the WMF protocol.

after $n_2$ units of time, through the execution of action $\tau$, the process is forced to act as $B$. Note that rule $+_3$ is responsible for preventing the selection of process $A$ at timeout expiration. This semantics for the $+$ operator is different from the one in $tSPA$ (a $tick$ action can be performed *only if* both summands can do so) and is motivated by the need of a more flexible way of programming the choice between different components.

### 2.3 The Wide-Mouthed Frog Protocol

As a simple example, we recall the Wide-Mouthed Frog Protocol (WMF, for short). The aim of the protocol is to establish a session key from an initiator $A$ to a responder $B$ by means of a server $S$ and using shared key cryptography. The protocol, as reported in [3], is as follows:

1. $A \to S : A, \{B, T_a, K_{ab}\}_{K_{as}}$
2. $S \to B : \{A, T_s, K_{ab}\}_{K_{bs}}$

where $T_a$ and $T_s$ are timestamps, $K_{as}$ ($K_{bs}$) is the key shared between $A$ and $S$ ($B$ and $S$), and $K_{ab}$ is the new session key. Both the server $S$ and the responder $B$ check the timestamp. If it is too old, the message (hence, the session key) is ignored.

In order to model this protocol in $tCryptoSPA$, we use a process $Clock$, counting the time elapsing, which other processes can ask for the current time value. Moreover, we use $[\langle m, m', m''\rangle \vdash_{trd} x]$ as a shortcut to extract $m''$ and also $[t \leq t']P$, assuming to have an inference system that handles the boolean relations on arithmetic expressions. With $c_{XY}$ we denote the channel used by $X$ to communicate with $Y$ and *vice-versa*. The complete specification is reported in Figure 3. The $lts$ of $P(K_{ab})$ is reported below, where, for the sake of clarity, each $\tau$ labeled transition is decorated also with the channel on which the synchronization took place and, possibly, with the modified value of the timestamps:

$$
P(K_{ab}) \xrightarrow{\tau, time, T_a = 0} . \xrightarrow{tick} . \xrightarrow{\tau, c_{AS}} . \xrightarrow{\tau, time, T_s = 1} .
$$
$$
\xrightarrow{tick} . \xrightarrow{\tau, c_{BS}} . \xrightarrow{\tau, time, T_b = 2} . \xrightarrow{tick} . \xrightarrow{\tau}
$$

This protocol takes two time units to perform the distribution of the key. At the third time unit, $A$ and $B$ are ready to use the session key.

## 3 A General Schema for the Definition of Timed Security Properties

In this section we propose a general schema for the definition of timed security properties. We call it *Timed Generalized NDC* ($tGNDC$ for short), since it is a real-time generalization of *Generalized NDC* ($GNDC$ for short) [10], which is in turn a generalization of *Non Deducibility on Compositions* ($NDC$ for short) [5]. The main idea is the following: a system $P$ is $tGNDC^{\alpha}_{\lhd}$ iff for every hostile environment (or enemy) $X$ the composition of the system $P$ with $X$ satisfies the timed specification $\alpha(P)$. Essentially $tGNDC^{\alpha}_{\lhd}$ guarantees that the timed property $\alpha$ is satisfied, with respect to the $\lhd$ timed behavioral relation, even when the system is composed with any possible enemy.

This section is organized as follows. We first define timed trace semantics (the behavioral semantics of principal interest) and timed (bi-)simulation. Then, we discuss the issue of hostile environments, showing that it is necessary to restrict their initial knowledge. Finally, we present the $tGNDC$ schema and some general results on it, some of which are independent of the chosen behavioral notion.

### 3.1 Behavioural semantics

Here we define the semantic pre-order and equivalence we will use to formalize security properties, *timed trace* pre-order and equivalence, the timed version of the classic untimed semantics.

The expression $P \stackrel{a}{\Rightarrow} P'$ is a shorthand for $P(\stackrel{\tau}{\longrightarrow})^* P_1 \stackrel{a}{\longrightarrow} P_2(\stackrel{\tau}{\longrightarrow})^* P'$ where $(\stackrel{\tau}{\longrightarrow})^*$ denotes a (possibly empty) sequence of transitions labeled $\tau$. Let $\gamma = a_1, \ldots, a_n \in (Act\backslash\{\tau\})^*$ be a sequence of actions; then $P \stackrel{\gamma}{\Rightarrow} P'$ iff there exist $P_1, \ldots, P_{n-1} \in \mathcal{P}$ such that $P \stackrel{a_1}{\Rightarrow} P_1 \stackrel{a_2}{\Rightarrow}, \ldots, P_{n-1} \stackrel{a_n}{\Rightarrow} P'$.

**Definition 1.** *For any $P \in \mathcal{P}$ the set $T(P)$ of timed traces associated with $P$ is defined as follows $T(P) = \{\gamma \in (Act\backslash\{\tau\})^* \mid \exists P'.P \stackrel{\gamma}{\Rightarrow} P'\}$. The timed trace pre-order, denoted by $\leq_{ttrace}$, is defined as follows: $P \leq_{ttrace} Q$ iff $T(P) \subseteq T(Q)$. $P$ and $Q$ are* timed trace equivalent*, denoted by $P =_{ttrace} Q$, if $T(P) = T(Q)$.*

As an example, it is easy to see that $T(P(K_{ab})) = \{\epsilon, tick, tick\ tick, tick\ tick\ tick\}$, where $\epsilon$ denotes the empty sequence.

We define the concept of timed weak simulation as usual.

**Definition 2.** *We say that a relation R among processes is a timed weak simulation, if for every $(P, Q) \in \mathcal{R}$ we have:*

- *If $P \stackrel{a}{\longrightarrow} P', a \neq \tau$, then there exists $Q'$ s.t. $Q \stackrel{a}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$.*
- *If $P \stackrel{\tau}{\longrightarrow} P'$ then there exists $Q'$ s.t. $Q \Longrightarrow Q'$ and $(P', Q') \in \mathcal{R}$.*

If $\mathcal{R}$ and $\mathcal{R}^{-1}$ are weak timed simulations then $\mathcal{R}$ is called timed weak bisimulation . Let $\prec$ ($\approx$) the union of all timed weak simulations (bisimulations) among processes. Then, we have $\prec \subseteq \leq_{ttrace}$ and so $\approx \subseteq =_{ttrace}$.

### 3.2 Hostile Environments

Here we characterize the notion of admissible hostile environments similarly to what done in [10] for the untimed setting. Such a characterization is necessary to analyze protocols where some information is assumed to be secret, as in cryptographic protocols. A hostile environment, or *enemy*, is a process which tries to attack a protocol by stealing and faking information which is transmitted on *public* channels, say $C$. Such an agent is modeled as a generic process $X$ which can communicate only through channels in $C$, imposing some constraints on the initial data that are known by the enemy and requiring that such a protocol is *weakly time alive*, i.e., the agent may always perform $tick$ eventually. Otherwise $X$ could prevent time from elapsing when composed in parallel with some system, since in a compound system time can pass iff all components let it pass. So the enemy could block the time flow and we want to avoid this unrealistic case. Let $Der(P)$ be the set of all derivatives of $P$, i.e., all the $P'$s reachable from $P$ through a sequence of actions in $Act$.

**Definition 3.** *A process $P$ is directly weakly time alive* iff $P \stackrel{tick}{\Longrightarrow} P'$. *$P$ is weakly time alive* iff *for all $P' \in Der(P)$, we have $P'$ is directly weakly time alive.*

Now, let $ID(P)$ be the set of messages that appear in $P$ (see [6] for a formal definition) and $\phi \subseteq \mathcal{M}$ be the initial knowledge we would like to give to the enemies, i.e., the public information such as the names of the entities and the public keys, plus some possible private data of the enemies (e.g., their private key or nonces). For some enemy $X$, we want that all the messages in $ID(X)$ are deducible from $\phi$. If this is not the case, an enemy would be unrealistic, as it would know in advance the secrets. We thus define the set $t\mathcal{E}_C^{\phi_I}$ of timed hostile processes as:

$$t\mathcal{E}_C^{\phi} = \{X \in \mathcal{P} \mid sort(X) \subseteq C \text{ and } ID(X) \subseteq \mathcal{D}(\phi) \text{ and } X \text{ is weakly time alive}\}$$

### 3.3 The $tGNDC$ Schema

In this section we formally define the $tGNDC_\triangleleft^\alpha$ family of properties. We will use $A||_C B$ as a shortcut for $(A||B)\backslash C$. The proposed family of security properties is as follows.

**Definition 4.** *$P$ is $tGNDC_\triangleleft^\alpha$ iff $\forall X \in t\mathcal{E}_C^{\phi_I}\ :\ (P||_C X) \triangleleft \alpha(P)$ where $\triangleleft \in \mathcal{P} \times \mathcal{P}$ is a pre-order, $C$ is a set of channels and $\alpha\ :\ \mathcal{P} \mapsto \mathcal{P}$ is a function between processes defining the property specification for $P$ as the process $\alpha(P)$.*

We propose a sufficient criterion for a static characterization (i.e., not involving the universal quantification $\forall$) of $tGNDC_\triangleleft^\alpha$ properties. We will say that $\triangleleft$ is a *pre-congruence* w.r.t. $||_C$ if it is a pre-order and for every $P, Q, Q' \in \mathcal{P}$ if $Q \triangleleft Q'$ then $P||_C Q \triangleleft P||_C Q'$. Thus it is easy to prove the following:

**Proposition 2.** *If $\triangleleft$ is a pre-congruence w.r.t. $||_C$ and if there exist a process $Top \in t\mathcal{E}_C^{\phi_I}$ such that for every process $X \in t\mathcal{E}_C^{\phi_I}$ we have $X \triangleleft Top$, then $\forall\alpha$:*

$$P \in tGNDC_\triangleleft^\alpha \quad \text{iff} \quad (P||_C Top) \triangleleft \alpha(P)$$

*Proof.* ($\Leftarrow$)If $\lhd$ is a pre-congruence w.r.t. $||_C$, then $X \lhd X'$ implies $(P||_C X) \lhd (P||_C X')$. Thus if for every process $X \in t\mathcal{E}_C^{\phi_I}$ we have $X \lhd Top$ then we will also have $(P||_C X) \lhd (P||_C Top)$ for every process $X \in t\mathcal{E}_C^{\phi_I}$ and so, as by hypothesis $(P||_C Top) \lhd \alpha(P)$, we obtain $(P||_C X) \lhd (P||Top) \lhd \alpha(P),\ \forall X \in t\mathcal{E}_C^{\phi_I}$, i.e., $P$ is $tGNDC_{\lhd}^{\alpha}$.
($\Rightarrow$)By definition of $tGNDC_{\lhd}^{\alpha}$, since $Top \in t\mathcal{E}_C^{\phi_I}$.

In particular if the hypotheses of the proposition above hold it is sufficient to check that $\alpha(P)$ is satisfied when $P$ is composed with the most general hostile environment $Top$. This permits to make only one single check, in order to prove that a property holds for all possible attackers. We also have the following corollary for the congruence induced by $\lhd$:

**Corollary 1.** *Let $\lhd$ be a pre-congruence w.r.t. $||_C$ and let $\equiv\ =\ \lhd \cap \lhd^{-1}$. If there exist two processes $Bot, Top \in t\mathcal{E}_C^{\phi_I}$ such that for every process $X \in t\mathcal{E}_C^{\phi_I}$ we have $Bot \lhd X \lhd Top$ then*

$$P \in tGNDC_{\equiv}^{\alpha} \quad \text{iff} \quad (P||_C Bot) \equiv (P||_C Top) \equiv \alpha(P)$$

Given these very general results, we show that they are instanciable in the model we presented so far. Indeed, this is the case, at least for the trace pre-order $\leq_{ttrace}$, which is a pre-congruence.

**Proposition 3.** *Timed trace pre-order is a pre-congruence w.r.t. $||_C$.*

*Proof.* Given $P, Q, Q' \in \mathcal{P}$ such that $Q \leq_{ttrace} Q'$, we have to prove $P||_C Q \leq_{ttrace} P||_C Q'$. This can be done by induction on the length of the timed traces generated by $P||_C Q$.

Note that in the $tSPA$ model, timed trace pre-order is not a pre-congruence, since the semantic rules enforce the so called *maximal communication progress*, i.e., when a communication is possible it must start immediately, and it is not possible to perform a $tick$ [9].

Now we single out the minimal element $Bot$ and the maximum element $Top$ in $t\mathcal{E}_C^{\phi_I}$ w.r.t. $\leq_{ttrace}$. As for $Bot$ it is clear that the minimum set of traces is generated by the weakly time alive process that does nothing, that is generated by process $\iota(\mathbf{0})$. As a matter of fact, $(P||\iota(\mathbf{0})) =_{ttrace} P$ for timed trace equivalence and most other equivalences. We thus define the $Top$ element using a family of processes $Top_{ttrace}^{C,\phi}$ each representing an instance of the enemy with knowledge $\phi$:

$$Top_{ttrace}^{C,\phi} = \sum_{c \in C} \iota(c(x).Top_{ttrace}^{C,\phi \cup \{x\}}) + \sum_{c \in C, m \in \mathcal{D}(\phi) \cap \mathcal{M}} \iota(\overline{c}m.Top_{ttrace}^{C,\phi})$$

The initial element of the family is $Top_{ttrace}^{C,\phi_I}$ as $\phi_I$ is the initial knowledge of the enemy. This may accept any input message to be bound to the variable $x$ which is then added to the knowledge set that becomes $\phi_I \cup \{x\}$, and may output only messages that can pass on the channel $c$ and that are deducible from the current knowledge set $\phi$ via the deduction function $\mathcal{D}$. Furthermore it can let time pass. Note that $\tau$ summands are not considered, as inessential for trace pre-order. We may prove the following:

**Proposition 4.** *If* $X \in t\mathcal{E}_C^\phi$ *then* $X \leq_{ttrace} Top_{ttrace}^{C,\phi}$.

*Proof.* By induction on the length of the timed traces generated by $X$.

## 4 Some Timed Security Properties

In this section we show how to redefine four timed security properties as suitable instances of the $tGNDC_\lhd^\alpha$ schema, by suitably defining function $\alpha$. As for the behavioral semantics $\lhd$, we will always consider the timed trace semantics. The four properties we consider are:

- The timed version of Non Deducibility on Compositions [5], which has been proposed to study information flow security; we will show that $tNDC$ is the strongest property in the $tGNDC_\lhd^\alpha$ family, under some mild assumptions.
- A timed notion of authentication, called *timed agreement* (see also [17]), according to which agreement must be reached within a certain deadline, otherwise authentication does not hold.
- A timed notion of secrecy, we call *timed secrecy*, according to which a message is secret only within a time interval and after the deadline it can become a public piece of information.
- A timed notion of integrity, called *timed integrity*, which simply requires a correct delivery of messages within a certain amount of time.

### 4.1 Timed Non Deducibility on Compositions

We start with $tNDC$ since $tGNDC_\lhd^\alpha$ is a generalization of such a property. Its underlying idea is that the system behavior must be invariant w.r.t. the composition with every hostile environment. Indeed, there is no possibility of establishing a communication (i.e. sending information). In the $CryptoSPA$ untimed setting the $NDC$ [1] idea can thus be defined as follows (see [10]):

**Definition 5.** $P \in NDC$ *if and only if* $\forall X \in \mathcal{E}_C^{\phi_I}$, *we have* $(P||_C X) =_{trace} P \backslash C$.

where $=_{trace}$ is trace equivalence and the only difference with the definition given in SPA is that the knowledge of process $X$ is bounded by $\phi_I$. Now we present *timed* $NDC$ ($tNDC$, for short) ([9]) which is the natural extension of $NDC$ to a timed setting.

**Definition 6.** $P \in tNDC$ *if and only if* $\forall X \in t\mathcal{E}_C^{\phi_I}$ *we have* $(P||_C X) =_{ttrace} P \backslash C$.

where the difference is that we use the timed hostile environment and timed trace equivalence. Note that $tNDC$ corresponds to $tGNDC_{=_{ttrace}}^{P\backslash C}$. It is also possible to apply Corollary 1 obtaining the following static characterization.

**Proposition 5.** $P \in tNDC$ *if and only if* $(P||_C Top_{ttrace}^{C,\phi_I}) =_{ttrace} P \backslash C$.

[1] As for $tGNDC_\lhd^\alpha$, also $NDC$ and $tNDC$ are implicitly parametric w.r.t. the set $C$ of public channels and the set $\phi_I$ of initial knowledge. We usually omit these parameters when clear from the context.

Now we suggest that $tNDC$ is the most restrictive $\alpha(P)$ hence inducing the strongest property for timed trace semantics. The most restrictive $\alpha(P)$ should return an *encapsulation* of protocol $P$, i.e., a version of $P$ which is completely isolated from the environment, corresponding to the execution of $P$ in a perfectly secure network where only the honest parties are present. In our process algebra setting, this corresponds to the *restriction* of all public channels in $C$ along which protocol messages are sent.

Note that for every process $P$ we have $(P||\iota(\mathbf{0})) \setminus C =_{ttrace} P \setminus C$. This means that $P$ restricted on $C$ is equivalent to the protocol in composition with the enemy that does nothing. Note also that, by definition, $\iota(\mathbf{0}) \in t\mathcal{E}_C^\phi$ for every $\phi$. So it is very natural to consider $\alpha$ functions and processes $P$ such that $P \setminus C \leq_{ttrace} \alpha(P)$. This simply means that the protocol $P$ is correct (as it satisfies its specification $\alpha(P)$) at least when it is not under attack. This condition can be somehow seen as a reasonable criterion for any *good* protocol: it must be correct at least when it is not under attack! Under this mild assumption, it is clear that $P \in tNDC$ implies $P \in tGNDC^{\alpha}_{\leq_{ttrace}}$.

This form of reasoning shows one of the main advantages of the GNDC idea that tries to unify concepts from different areas as information flow and cryptographic protocol analysis. The uniform schema, in this paper extended in a timed setting, allows us to easily compare different security properties.

Another way to avoid universal quantification over all the admissible enemies is to show the equivalence between $tNDC$ and *Timed Strong Nondeterministic Non-Interference* ($tSNNI$, for short); such equivalence result holds in the untimed case [5], but that does not hold for $tSPA$ [9] because of the maximal communication assumption of that language.

A $CryptoSPA$ process is $SNNI_C^\phi$ if $P \backslash C$, where all actions in $C$ are forbidden, behaves like the system $P$ where all the actions in $C$ are *hidden* (i.e., transformed into internal $\tau$ actions). To express this second system we need to introduce first the *hiding* operator $P/^{\phi}C$:

$$\frac{P \xrightarrow{a} P'}{P/^{\phi}C \xrightarrow{a} P'/^{\phi}C}(a \notin C \cup \overline{C}) \quad \frac{P \xrightarrow{\overline{c}m} P' \quad c \in C \cup \overline{C}}{P/^{\phi}C \xrightarrow{\tau} P'/^{\phi \cup \{m\}}C}$$

$$\frac{P \xrightarrow{c(m)} P' \quad c \in C \cup \overline{C} \quad m \in \mathcal{D}(\phi)}{P/^{\phi}C \xrightarrow{\tau} P'/^{\phi}C}$$

Now we are ready to define the property *timed* $SNNI_C^\phi$ as follows.

**Definition 7.** *A process is* $tSNNI_C^\phi$ *if* $P \backslash C =_{ttrace} P/^{\phi}C$.

It is rather intuitive that $P/^{\phi}C$ can be seen as $P||_C Top$, where $Top$ is the top element of the trace pre-order for $CryptoSPA$. Hence, such a static characterization can be seen as a corollary of the existence of a top element in the trace pre-order (together with the fact that trace pre-order is a pre-congruence).

**Proposition 6.** *For every* $P \in \mathcal{P}$ *we have that* $(P||_C Top^{C,\phi}_{ttrace}) =_{ttrace} P/^{\phi}C$.

*Proof.* We have to prove $(P||_C Top^{C,\phi}_{ttrace}) \leq_{ttrace} P/^{\phi}C$ and the symmetric relation $(P||_C Top^{C,\phi}_{ttrace}) \leq_{ttrace} P||_C Top^{C,\phi}_{ttrace})$. These can be done by induction on the length of the traces generated by the left-hand-side processes.

**Proposition 7.** $P \in tNDC_C^\phi$ iff $P \in tSNNI_C^\phi$.

*Proof.* By Proposition 5 and Proposition 6.

We can also extend the *Bisimulation Strong Nondeterministic Non-interference* (BSNNI, for short) property defined in [5] in our setting.

**Definition 8.** *A process is* $tBSNNI_C^\phi$ *if* $P \backslash C \approx P/^\phi C$.

Indeed, this property requires that the equivalences used to compare $P \backslash C$ and $P/^\phi C$ is timed weak bisimulation. We can prove a proposition similar to Prop. 6.

**Proposition 8.** *For every* $P \in \mathcal{P}$ *we have that* $(P||_C Top_{ttrace}^{C,\phi}) \approx P/^\phi C$.

*Proof.* By checking that the following relation is a timed weak bisimulation:

$$\mathcal{R} = \{(P||_C Top_{ttrace}^{C,\phi}, P/^\phi C) \mid P \in \mathcal{P}, \phi \subseteq \mathcal{M}\}$$

Unfortunately, we cannot prove a proposition as Prop. 5. As a matter of fact, also in the case without cryptography and time, it is not true that checking a system only against the $Top$ process is enough to avoid the universal quantification (e.g., see [5]).

However, checking $tBSNNI_C^\phi$ may be an efficient method for checking $tSNNI_C^\phi$. Indeed, we have $tBSNNI_C^\phi \subseteq tSNNI_C^\phi$ and timed weak bisimulation is decidable in polynomial time for finite processes while checking trace inclusion is a PSPACE-complete problem.

### 4.2 Timed Agreement

We now present the *Timed Agreement* Property [17]:

> "A protocol guarantees *Timed Agreement* between a responder $B$ and an initiator $A$ on a set of data items $ds$ if, whenever $B$ (acting as responder) completes a run of the protocol, apparently with initiator $A$, then $A$ has previously been running the protocol, apparently with $B$, in the last $n\ ticks$ (where $n$ is a prefixed timeout value) and the two agents agreed on the data values corresponding to all the variables in $ds$, and each such a run of $B$ corresponds to a unique run of $A$."

As done in [10] for the non real-time version of *Agreement*, what we do is to have for each party an action representing the running of a protocol and another one representing the completion of it. We consider an action $commit\_res(B, A, d)$ representing a correct termination of $B$ as a responder, convinced to communicate with $A$ that agrees on data $d$. On the other hand we have an action $running\_ini(A, B, d)$ that represents the fact that $A$ is running the protocol as initiator, apparently with $B$, with data $d$. If we specify these two actions in the protocol, the *Timed Agreement* property requires that when $B$ executes $commit\_res(B, A, d)$ then $A$ has previously executed $running\_ini(A, B, d)$ and at most $n\ tick$ actions, where $n$ is the prefixed timeout value, occurred between these actions. We assume that the actions representing the running and the commit are correctly specified in the protocol. We can see them as output actions over two particular channels $running\_ini$ and $commit\_res$. We assume that $d$ can assume values in a set $D$. Let $NotObs(P) = sort(P) \backslash (C \cup \{running\_ini, commit\_res\})$ be the set of channels in $P$ that are not public and are different from $running\_ini$ and $commit\_res$, i.e., that will not be observed. Function $\alpha^{t(n)}_{tAgree}$ can be thus defined as follows:

$$P'(x,y) = \sum_{d \in D, i \in 0..n} \iota(\overline{running\_ini}(x,y,d).tick_1..tick_i.\overline{commit\_res}(y,x,d).\iota(\mathbf{0}))$$
$$P'' = \sum_{c \in NotObs(P)} \iota(c(x).P'') + \sum_{c \in NotObs(P), m \in \mathcal{M}} \iota(\overline{c}m.P'')$$
$$\alpha^{t(n)}_{tAgree}(P) = P'' || P'(A,B)$$

Note that $P''$ is essentially the process that executes every possible action over channels in $sort(P)$ which are not in $C$ and are different from $running\_ini$ and $commit\_res$, or let time pass. Given $P$, $\alpha^{t(n)}_{tAgree}(P)$ represents the most general system which satisfies the *Timed Agreement* property and has the same sort of $P$. In fact in $\alpha^{t(n)}_{tAgree}(P)$ action $\overline{running\_ini}(x,y,d)$ always precedes $\overline{commit\_res}(y,x,d)$ for every datum $d$, and every combination of the other actions of $P$ can be executed. Finally the number of $tick$ actions is at most $n$. In order to analyze more than one session, it suffices to consider an extended $\alpha$ which has several processes $P'$ in parallel.
We want that even in the presence of a hostile process, $P$ does not execute traces that are not in $\alpha^{t(n)}_{tAgree}(P)$. So we can give the following definition:

**Definition 9.** *$P$ satisfies Timed Agreement* iff *$P$ is $tGNDC^{\alpha^{t(n)}_{tAgree}(P)}_{\leq_{ttrace}}$, i.e.,*

$$\forall\, X \in t\mathcal{E}^{\phi_I}_C \ : \ (P||_C X) \leq_{ttrace} \alpha^{t(n)}_{tAgree}(P)$$

*Example 6.* We consider here the Wide Mouthed Frog protocol presented in Section 2.3. In this specification we consider a server that can receive requests from both $A$ and $B$ as initiators. Note that time parameter $n$ is 3 because in a correct execution 3 ticks are performed in between $running\_ini$ and $commit\_res$.

$$\begin{aligned}
Clock(n) &= tick.Clock(n+1) + \overline{time}\, n.Clock(n)\\
A(K) &= time(T_a).\overline{running_ini}(A,B,K).tick.\overline{c}_{AS}\{B,T_a,K\}_{K_{as}}.\iota(\mathbf{0})\\
B &= \iota(c_{BS}(y)[\langle y, K_{bs}\rangle \vdash_{dec} z]time(T_b).tick.[\langle z\rangle \vdash_{snd} t][t+1 \leq T_b]\\
&\quad [\langle z\rangle \vdash_{trd} w]\overline{commit_res}(B,A,w))\\
S &= tick.c_{AS}(u).[\langle u, K_{as}\rangle \vdash_{dec} z]time(T_s).tick.[\langle z\rangle \vdash_{snd} t][t+1 \leq T_s]\\
&\quad [\langle z\rangle \vdash_{trd} x]\overline{c}_{BS}\{A,T_s,x\}_{K_{bs}}.S+\\
&\quad tick.c_{BS}(u).[\langle u, K_{bs}\rangle \vdash_{dec} z]time(T_s).tick.[\langle z\rangle \vdash_{snd} t][t+1 \leq T_s]\\
&\quad [\langle z\rangle \vdash_{trd} x]\overline{c}_{AS}\{B,T_s,x\}_{K_{as}}.S\\
P(K_{ab}) &= (Clock(0)||A(K_{ab})||B||S) \backslash \{c_{AS}, c_{BS}, time\}
\end{aligned}$$

Unfortunately, this protocol could be attacked by an intruder $P$ in the following way:

1. $A \to S : A, \{B, T_a, K_{ab}\}_{K_{as}}$
2. $S \to E(B) : \{A, T_a+1, K_{ab}\}_{K_{bs}}$
3. $E(B) \to S : B, \{A, T_a+1, K_{ab}\}_{K_{bs}}$
4. $S \to E(A) : \{B, T_a+3, K_{ab}\}_{K_{as}}$

5. $E(A) \rightarrow S : A, \{B, T_a + 3, K_{ab}\}_{K_{as}}$
6. $S \rightarrow B : \{B, T_a + 5, K_{ab}\}_{K_{bs}}$

Indeed, the protocol $P$, in parallel with the enemy $E$, produces the following trace:

$$\overline{running\_ini}(A, B, K_{ab}).tick.tick.tick.tick.tick.\overline{commit\_res}(B, A, K_{ab})$$

that is not in $\alpha^{t(n)}_{tAgree}$ since the number of $ticks$ is greater than $n = 3$. Note that, in fact, the actions $\overline{running\_ini}(A, B, K_{ab})$ and $\overline{commit\_res}(B, A, K_{ab})$ are correctly executed, but the enemy forces $B$ to accept an old key. In an untimed setting this would be a correct execution of the protocol. The problem here is, on the one hand that the message produced by $S$ can easily be used by the enemy to start a new run of the protocol having as initiator the intended receiver of the previous run, and on the other hand that when $B$ receives the key, he has no way to know when $A$ forged it.

### 4.3 Timed Secrecy

We now present the *Timed Secrecy* Property:

> "A protocol guarantees to an initiator $A$ the property of *Timed Secrecy* on a set of data items $ds$ within a time $n$ if, whenever a data item in $ds$ becomes public, at least $n$ ticks passed since $A$ started the protocol"

As done for *Timed Agreement*, what we do is to have an action representing the running of a protocol and another one representing that a secret is revealed. We consider an action $running\_ini(A, d)$ that represents the fact that $A$ is running the protocol as initiator, with data $d$. On the other hand we have an action $\overline{public}(d)$ representing that data item $d$ is made public. If we specify these two actions in the protocol, the *Timed Secrecy* property requires that when someone executes $\overline{public}(d)$ then $A$ has executed $running\_ini(A, d)$ and at least $n$ $tick$ actions, where $n$ is the prefixed timeout value, occurred between them. We assume that the actions representing the running and the publication are correctly specified in the protocol. We can see the first as an output action over a particular channel $running\_ini$. The second action, following the approach of [7] is performed by a particular process called *Key Expired Notifier* ($KEN$, for short) that reads from a public channel $c$ not used in the protocol and performs the output of what it has read on the channel $\overline{public}$, i.e. $KEN = c(x).\overline{public}(x)$.

Let $NotObs(P) = sort(P) \setminus (C \cup \{c, running\_ini, public\})$ be the set of channels in $P$ that are not public and are different from $running\_ini$ and $public$, i.e., that will not be observed. We assume that $d$ can take values in a set of secret values $D$. Function $\alpha^{t(n)}_{tSec}$ can be thus defined as follows:

$$P'(x) = \sum_{d \in D} \iota(\overline{running\_ini}(x, d).tick_1 \dots tick_n.(\iota(\overline{public}(d).\iota(\mathbf{0})) \; + \iota(\tau.\iota(\mathbf{0}))))$$
$$P'' = \sum_{c \in NotObs(P)} \iota(c(x).P'') + \sum_{c \in NotObs(P), m \in \mathcal{M}} \iota(\overline{c}m.P'')$$
$$\alpha^{t(n)}_{tSec}(P) = P'' || P'(A)$$

Given $P$, $\alpha^{t(n)}_{tSec}(P)$ represents the most general system which satisfies the *Timed Secrecy* property and has the same sort of $P$. In fact in $\alpha^{t(n)}_{tSec}(P)$, the action $\overline{public}(d)$

may be executed only after $n$ $ticks$ from the action $\overline{running\_ini}(x,d)$ for every datum $d$, and every combination of the other actions of $P$ can be executed. In order to analyze more than one session, it suffices to consider an extended $\alpha$ which has several processes $P'$ in parallel.
We want that, even in the presence of a hostile process, $P$ does not execute traces that are not in $\alpha^{t(n)}_{tSec}(P)$. So we can give the following definition:

**Definition 10.** *$P$ satisfies Timed Secrecy iff $P$ is $tGNDC^{\alpha^{t(n)}_{tSec}(P)}_{\leq_{ttrace}}$, i.e.,*

$$\forall\, X \in t\mathcal{E}^{\phi_I}_C \;:\; (P||_C X) \leq_{ttrace} \alpha^{t(n)}_{tSec}(P)$$

*Example 7.* We consider the following protocol:

1. $A \rightarrow S : A, \{B, T_a, K_{ab}, Exp\}_{K_{as}}$
2. $S \rightarrow B : \{A, T_s, K_{ab}, Exp'\}_{K_{bs}}$

which is a variant of the Wide Mouthed Frog Protocol in which we add a field $Exp$ used to state how long the key will last before expiring. After that time the key is no longer valid and will be made public by $B$. Both the server $S$ and $B$ check the value $Exp$ and upgrade it according to the amount of time passed. $c$ is the public channel used to throw away expired keys while $KEN$ is the process that we use to write on the private channel $public$.

We assume that we have two servers that update the field $Exp$ in a different way. In particular server $S_2$ decreases at double speed (w.r.t. $S_1$) the key expiration time, so that it is possible to have secure keys that will last less.

$$\begin{aligned}
Clock(n) &= tick.Clock(n+1) + \overline{time}\, n.Clock(n)\\
A(K) &= time(T_a).\overline{running_ini}(A,K).tick.\overline{c}_{AS_1}\{B,T_a,K,n\}_{K_{as}}.\iota(\mathbf{0})\\
B &= \iota(c_{BS}(y)[\langle y, K_{bs}\rangle \vdash_{dec} z]time(T_b).tick.[\langle z\rangle \vdash_{snd} t_s][\langle z\rangle \vdash_{trd} w][\langle z\rangle \vdash_{frt} t]\\
&\quad [t-(T_b-t_s)\leq 0]\overline{c}w)\\
KEN &= \iota(c(x).\overline{public}(x))\\
S &= S_1||S_2\\
S_1 &= \iota(c_{AS_1}(u).[\langle u, K_{as}\rangle \vdash_{dec} z]time(T_s).tick.[\langle z\rangle \vdash_{snd} t_a][\langle z\rangle \vdash_{trd} x]\\
&\quad [\langle z\rangle \vdash_{frt} t][1 < t-(T_s-t_a)]\overline{c}_{BS}\{A,T_s,x,t-(T_s-t_a)\}_{K_{bs}}.tick)\\
S_2 &= \iota(c_{AS_2}(u).[\langle u, K_{as}\rangle \vdash_{dec} z]time(T_s).tick.[\langle z\rangle \vdash_{snd} t_a][\langle z\rangle \vdash_{trd} x]\\
&\quad [\langle z\rangle \vdash_{frt} t][0 < t-(T_s-t_a)]\overline{c}_{BS}\{A,T_s,x,t-2*(T_s-t_a)\}_{K_{bs}}.tick)\\
P(K_{ab}) &= (Clock(0)||A(K_{ab})||B||S||KEN)\backslash\{c_{as},c_{bs},c,time\}
\end{aligned}$$

An enemy $E$, by using server $S_2$, could make the key appear as expired to the intended recipient, by simply redirecting to server $S_2$ the message intended for server $S_1$. We suppose that $n = 4$. The enemy could attack the protocol in the following way:

1. $A \rightarrow E(S_1) : A, \{B, 0, K_{ab}, 4\}_{K_{as}}$
2. $E(A) \rightarrow S_2 : A, \{B, 0, K_{ab}, 4\}_{K_{as}}$
3. $S_2 \rightarrow B : \{A, 1, K_{ab}, 2\}_{K_{bs}}$

As $B$ receives the third message and upgrades the value of $Exp$, it becomes 1 and so the key will be made public even if only 3 $ticks$ passed.

The protocol $P$ composed in parallel with enemy $E$ could produce the following trace:

$$\overline{running\_ini}(A, K_{ab}).tick.tick.tick.\overline{public}(K_{ab})$$

that is not in $\alpha^{t(n)}_{tSec}$ since the number of $ticks$ is less than $n = 4$.

### 4.4 Timed Integrity

We now present the *Timed Integrity* Property:

> "A protocol guarantees to the user $B$ the property of *Timed Integrity* on a set of data items $ds$ within a time $n$ if $B$ only accepts data items in $ds$ and this may only happen in at most $n$ ticks since the beginning of the protocol"

For instance, imagine that you would like to receive your favorite newspaper each day before noon. This may be expressed as an integrity property rather than an authenticity one, since you are not actually interested in the sender but simply on the data (the newspaper). Consider a channel $out$ used for expressing the reception of a message and let $NotObs(P) = sort(P) \backslash (C \cup \{out\})$ be the set of channels in $P$ that are not public and $d$ ranging over a set of data $D$. Then, *Timed Integrity* may be formally specified as follows:

$$\begin{array}{l} P'(y, n) = ||_{d \in D} tick_1. \ldots .tick_n.\tau.\iota(\mathbf{0}) + \iota(\overline{out}(y, d).\iota(\mathbf{0})) \\ P'' = \sum_{c \in NotObs(P)} \iota(c(x).P'') + \sum_{c \in NotObs(P), m \in \mathcal{M}} \iota(\overline{c}m.P'') \\ \alpha^{t(n)}_{tInt}(P) = P''||P'(B, n) \end{array}$$

## 5 Compositional analysis

In this section we illustrate some compositional proof rules for establishing that a system enjoys a $tGNDC$ property, in particular $tSNNI^{\phi}_{C}$. However, remember that, as it is equivalent to $tNDC$, this property implies all the other ones based on trace semantics, that are the most frequently used in security analysis.

A compositional proof rule for checking a property *f* works as follows: in order to check if a system $P||Q$ satisfies *f* it is enough to check whether both $P$ and $Q$ satisfy *f*. Compositional reasoning is useful in many cases. For instance, an interesting field of application is the analysis of systems with an arbitrary number of equal components. Let us consider the parallel composition of multiple instances of process $P$:

$$\overbrace{P|| \ldots ||P}^{n}$$

To prove that this system enjoys *f* (for whatever $n$) it is sufficient to prove that $P$ enjoys *f*.

Within the $SPA$ theory, $SNNI$ is compositional, i.e. if $P, Q \in SNNI$ then $(P||Q) \in SNNI$. Unfortunately, the same does not hold when considering enemies with limited knowledge, as for $tSNNI^{\phi}_{C}$. For instance, consider the processes:

$$P = \overline{c_1}m_1.c_2(x)[x = m_2].\overline{c_3}m_2 \qquad Q = \overline{c_1}m_2.c_2(x)[x = m_1].\overline{c_3}m_1$$

Now, assuming $C = \{c_1, c_2\}$ and $\phi = \emptyset$, we have that $P, Q \in tSNNI_C^{\phi}$. However, $P||Q \notin tSNNI^{\phi}$. As a matter of fact, $(P||Q)\backslash C$ is equivalent to $\mathbf{0}$, while $(P||Q)/^{\phi}C$ may perform both $\overline{c_3}m_1$ and $\overline{c_3}m_2$. The problem is the following: The information acquired by an intruder interacting with one process is useful to interfere with the other one.

However, if we strengthen the assumptions we can get a compositional rule for establishing that a process belongs to $tSNNI_C^{\phi}$. The *stability* assumption we make is that the process cannot significantly increase its knowledge.

**Definition 11.** *We say that a process $P$ is stable w.r.t. $\phi$, whenever if $P/^{\phi}C \stackrel{\gamma}{\Longrightarrow} P'/^{\phi'}C$ then $\mathcal{D}(\phi) = \mathcal{D}(\phi')$.*

Stability (w.r.t. a knowledge $\phi$) is a compositional property.

**Proposition 9.** *Assume that $P, Q \in tSNNI_C^{\phi}$ and that $P, Q$ are stable w.r.t. $\phi$. Then $P||Q$ is stable w.r.t. $\phi$.*

This is useful to show the compositional principle on the whole observational behavior of the processes.

**Proposition 10.** *Assume that $P, Q \in tSNNI_C^{\phi}$ and that $P, Q$ are stable w.r.t. $\phi$. Then $P||Q \in tSNNI_C^{\phi}$.*

**Proposition 11.** *Assume that $P, Q \in tBSNNI_C^{\phi}$ and that $P, Q$ are stable w.r.t. $\phi$. Then $P||Q \in tBSNNI_C^{\phi}$.*

*Example 8.* We consider a simple example of the application of the principle above. Consider the processes $P = \overline{c}\{m\}_k.0$ and $Q = c?x.[x \quad k \vdash_{dec} z]\tau$. Consider also the process $KEN' = c?x.[x = m].\overline{public}m$. Assuming $\phi = \{\{m\}_k\}$ and $C = \{c\}$, we can establish that $P, Q, KEN' \in tSNNI_C^{\phi}$ and are all stable; hence $P||Q||KEN' \in tSNNI_C^{\phi}$, which means that $P||Q$ keeps $m$ secret. As a matter of fact, we have:
$(P||Q||KEN')\backslash C =_{ttrace} \mathbf{0}$.

We may derive a proof principle for the $tGNDC^{\alpha}_{\leq_{ttrace}}$ schema, again under the assumption that the involved processes are stable.

**Proposition 12.** *Given the set of initial knowledge $\phi$ and the set of public channels $C$, assume $P_i \in tGNDC^{\alpha_i(P_i)}_{\leq_{ttrace}}$, with $i = 1, 2$, and $P_1, P_2$ are stable w.r.t. $\phi$. It follows that $P_1||P_2 \in tGNDC^{\alpha_1(P_1)||\alpha_2(P_2)}_{\leq_{ttrace}}$ and $P_1||P_2$ is stable w.r.t. $\phi$.*

*Proof.* The proof follows by noticing that $(P_1||P_2)/^{\phi}C \leq_{ttrace} P_1/^{\phi}C||P_2/^{\phi}C$. Then, by hypothesis and Proposition 6, we have $P_i/^{\phi}C \leq_{ttrace} \alpha_i(P_i)$, with $i = 1, 2$, and the thesis then follows because $\leq_{ttrace}$ is a pre-congruence w.r.t. $||$.

Note that whenever $\alpha(P_1||P_2) \leq_{ttrace} \alpha(P_1||P_2)$ then we get a compositional proof rule for $tGNDC^{\alpha}$.

*Example 9.* Consider the following family of processes $S(i)$, each sending a message $(m_i, i)$ after every time unit:

$$S(i) = \iota(tick.[(m_i, i) \quad pk_A \vdash_{enc} z].\overline{c}z.S(i+1))$$

Consider also a family of receivers of this kind:

$$R(i) = tick.c(y)[y \quad pk_A^{-1} \vdash_{dec} t][t \vdash_{snd} t_2][t_2 = i][t \vdash_{fst} t_1]\overline{out}\, t_1.R(i+1)$$

Basically, we have that $(S(0)||R(0))\backslash C$, where $C = \{c\}$, is trace included into $Spec(0)$ where

$$Spec(i) = tick.\overline{out}\, m_i.Spec(i+1)$$

Consider $\phi = \{\{(m_i, i)\}_{pk_A} \mid 0 \leq i\} \cup \{pk_A\}$ as the intruder's knowledge set. Then, we have that $S(0)$ and $R(0)$ are stable w.r.t. $\phi$ and $S(0) \in tNDC_C^{\phi}$, with $S(0)\backslash C =_{ttrace} \iota(\mathbf{0})$, and $R(0) \in tGNDC_{\leq_{ttrace}}^{Spec(0)}$. By Proposition 12, we have that

$$(S(0)||R(0)||Top_C^{\phi})\backslash C \leq_{ttrace} \iota(\mathbf{0})||Spec(0) \leq_{ttrace} Spec(0)$$

### 5.1 Time-dependent stability and compositional results

We give a new result about conditions for safe composition of cryptoprotocols where time plays an essential role. In order to achieve this result, we should refine the concept of stability given above, by introducing the one of $time - dependent$ stability.

In the previous section we noticed that if we assume that the intruder knowledge does not increase when composing the intruder process with $P$ (i.e. $P||X$) and with $Q$ (i.e. $Q||X$) (using the same communication channels) then the intruder knowledge does not increase when composing the intruder itself with the process $P||Q$. Unfortunately, such a form of stability is not time dependent and so it is unable to check properties based on a timed notion of secrecy. This does not make it feasible to check protocols such as $\mu$TESLA [21], whose security features exactly depend on a form of *timed* secrecy. To clarify this point consider the following example.

*Example 10.* The process $P = tick.\overline{c}k.\iota(\mathbf{0})$ enjoys the secrecy of $k$ for one time unit. In the more complex process $Q = (c(x).[x = k].\overline{c}m) + tick.\iota(\mathbf{0})$ the secrecy of $k$ in the first time unit is crucial to get the secrecy of $m$. Indeed, either $Q$ is willing to receive the key $k$ only in the first time unit (if so, it releases $m$) or it starts to idle. Unfortunately, each stable set $\phi$ must contain the message $k$, and so also the message $m$. Thus, our compositional proof principles cannot be applied. However, we may note that if we could reason about stability in each time slot then we can refine the proof principle in order to cope with such cases.

We let $\gamma$ be a sequence of actions (possibly empty) ranging over $Act\backslash\{\tau\}$. Let $\#^{tick}(\gamma)$ be the number of occurrences of $tick$ actions in the sequence $\gamma$. The following new definition will allow us to cope with *timed* secrecy and security properties of protocols that rely on it.

**Definition 12.** *We say that a process $P$ is time-dependent stable w.r.t. the sequence $\{\phi_i\}_{i\geq 0}$ if, whenever $(P||X_{\phi_0})\backslash C \stackrel{\gamma}{\Longrightarrow} (P'||X'_{\phi'})\backslash C$ and $\#^{tick}(\gamma) = i$, then $\mathcal{D}(\phi') = \mathcal{D}(\phi_i)$.*

Basically, a process $P$ is time-dependent stable when an enemy cannot increase significantly its knowledge when $P$ runs in the space of a time slot.

The following proposition holds.

**Proposition 13.** *Given a sequence $\{\phi_i\}$ and a set of public channels $C$, assume $P_r \in tGNDC^{\alpha_r(P_r)}_{\leq_{ttrace}}$ with $1 \leq r \leq n$. Assume also $P_r$ t. d. stable w.r.t. $\{\phi_i\}$. It follows that $(P_1||P_2||\ldots||P_n) \in tGNDC^{\alpha_1(P_1)||\alpha_2(P_2)||\ldots||\alpha_n(P_n)}_{\leq_{ttrace}}$ and $(P_1||P_2||\ldots||P_n)$ is t. d. stable w.r.t. $\{\phi_i\}$.*

*Example 11.* Consider again the processes in Example 10. We have that $P$ and $Q$ are t.d. stable w.r.t. $\phi_0 = \{\emptyset\}$, $\phi_i = \mathcal{D}(\{k\})$ for $i \geq 1$. Then, $P||Q$ is t.d. stable w.r.t. $\{\phi_i\}_{i\geq 0}$ (by Proposition 13) and so $m$ will never belong to the knowledge of the intruder (whose initial knowledge is $\emptyset$).

## 6 Conclusions

We have shown how to extend the $GNDC$ schema to a real time setting while preserving the properties of the untimed schema. In particular, we have shown the existence of a "most powerful" timed enemy, at least for the timed trace semantics. We have also shown how to express uniformly in this general schema some timed security properties, such as timed Non Deducibility on Compositions, (one definition of) timed authentication, timed secrecy and also timed integrity. We have also introduced a compositional proof principle that allows us to compose safely two security protocols, preserving the security properties they enjoy. Such a compositional proof principle has been extended to cope with real-time. One may wonder if the stability condition is too restrictive. As a matter of fact, our compositional proof principles can be successfully applied for checking security properties, e.g. message integrity, in stream signature protocols as the ones in [4, 11, 20, 21]. See [12, 14, 18] for some preliminary results.

Related literature on real-time security include the prominent papers [22, 2]. The former paper presents tock-CSP – a real-time language similar to tSPA – that is used to specify real-time cryptographic protocols. The main differences consists of a different treatment of timed operators, in the absence of a mechanism for handling crypto-primitives, in the lack of a uniform schema, and in the absence of compositionality results. The latter paper [2] is mainly concerned with the model checking of the interesting case study of TESLA, a protocol for stream broadcasting over the internet. The main focus is on showing that it is possible to give a finite model for the unbounded supply of fresh cryptographic keys used during the protocol. The so-called *security condition* of the protocol is similar to timed agreement.

Compositional proof techniques for reasoning about cryptographic protocols in an untimed setting may be found in [1, 16]. In [1], a compositional proof system for an environment-sensitive bisimulation has been developed. One main difference from ours, is that we consider a weak notion of observation where the internal actions are not visible. This permits us to have more abstract specifications. (As a matter of fact, the authors of [1] leave as future work the treatment of such a form of weak equivalence.) In [16], the authors develop the concept of disjoint encryption and, under this hypothesis, are able to perform compositional reasoning both for secrecy and authentication properties. While on the one hand, their approach seems to deal better with authentication properties than ours, on the other one it seems that there are situations where stability holds while disjoint encryption does not. We leave as future work the comparison of the effectiveness of the different approaches for compositional analysis of security properties.

Future work will be also devoted to study other security properties in a timed setting, such as non repudiation, for which apparently there is the need for using semantics more discriminating than timed trace semantics, for which most of the results in this paper are developed.

## References


1. M. Boreale and D. Gorla. on Compositional Reasoning in the Spi-calculus In proc. of FOSSACS'02, LNCS 2303, 2002.
2. P. Broadfoot and G. Lowe. Analysing a Stream Authentication Protocol using Model Checking In Procs. ESORICS'02, LNCS, Springer, 2002.
3. J. Clark and J. Jacob. A Survey of Autentication Protocol Literature: Version 1.0. http://www.cs.york.ac.uk/jac/papers/drareview.ps.gz
4. N. De Francesco and M. Petrocchi. Authenticity in a Reliable Protocol for Mobile Computing. in Proc. of the 18th ACM Symposium on Applied Computing (SAC03). U.S., March 2003.
5. R. Focardi and R. Gorrieri. A classification of security properties. *Journal of Computer Security*, 3(1):5–33, 1995.
6. R. Focardi, R. Gorrieri and F. Martinelli. A comparison of three authentication properties. To appear on Theoretical Computer Science.
7. R. Focardi, R. Gorrieri and F. Martinelli. Secrecy in Security Protocols as Non Interference. *Electronic Notes in Theoretical Computer Science*, 32. (2000)
8. R. Focardi, R. Gorrieri and F. Martinelli. Non-interference for the analysis of cryptographic protocols. In Procs. *Int.l Colloquium on Automata, Languages and Programming* (ICALP'00), LNCS 1853, 354-372, Springer, 2000.
9. R. Focardi, R. Gorrieri and F. Martinelli. Real-time Information Flow Analysis *Journal of Selected Areas of Communications*, IEEE Press, 2002, to appear.
10. R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proceedings of World Congress on Formal Methods (FM'99)*, pages 794–813. Springer, LNCS 1708, 1999.
11. R. Gennaro and P. Rohatgi. How to Sign Digital Streams. Information and Computation 165(1), pp.100-116 (2001).
12. R. Gorrieri, F. Martinelli, M. Petrocchi and A. Vaccarelli. Using compositional reasoning and Non-interference for verifying integrity in digital stream protocols. In Proc. of ACSD'03. IEEE Press, Lisbon 2003.

13. R. Gorrieri, E. Locatelli, F. Martinelli. A simple language for real-time cryptographic protocol analysis In Proc. of ESOP'03. LNCS 2618, Warsaw 2003.
14. R. Gorrieri, F. Martinelli, M. Petrocchi and A. Vaccarelli. Formal analysis of some security properties in wireless/sensor protocols. Submitted for publication.
15. J.F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, 118:263–299, 1993.
16. J. Guttman and F.J. Thayer. Protocol Independence through Disjoint Encryption. In Proc. CSFW'00. IEEE Press, 2000.
17. G. Lowe. A Hiearchy of Autentication Specifications. In Proc. CSFW'97, pp. 31-43, IEEE Press.
18. F. Martinelli, M. Petrocchi and A. Vaccarelli. Compositional Verification of Secure Streaming Data: a Case Study with EMSS To appear in proc. of ICTCS'03, LNCS, 2003.
19. R. Milner. Communication and Concurrency. Prentice Hall, 1989.
20. A. Perrig, R. Canetti, J. D. Tygar, and D. X. Song. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *Proc. of IEEE S&P'00*, pages 56–73, 2000.
21. A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. Culler. SPINS: Security Protocols for Sensor Networks. *Wireless Networks Journal (WINE)*, 8:521–534, 2002.
22. S. Schneider. Analysing Time-Dependent Security Properties in CSP using PVS. In Procs. ESORICS, LNCS 1895, 2000.
23. I. Ulidowski and S. Yuen. Extending process languages with time. In *Proceedings of AMAST*, LNCS 1349, pages 524–538, 1997.

# A Proofs

Here we list some proofs omitted from Section 3 and Section 5.

**Lemma 1.** *Assume that $P_1$ and $P_2$ are t.d. stable w.r.t. $\{\phi_i\}_{i\geq 0}$; then $P_1||P_2$ is t.d. stable w.r.t. $\{\phi_i\}_{i\geq 0}$.*

*Proof.* For the sake of simplicity, we assume that $\phi_i = \mathcal{D}(\phi_i), \forall i$. By contradiction, consider a computation $\gamma$ s.t.

$$(P_1||P_2||X_{\phi_0})\backslash C \stackrel{\gamma}{\Longrightarrow} (P_1''||P_2''||X_{\phi''})\backslash C$$

and $\mathcal{D}(\phi'') \neq \mathcal{D}(\phi_i) = \phi_i$. Then, it means that $P_1||P_2$ outputs at least one message $m$ s.t. $m \notin \phi_i$. Let $m_1$ be the first (and last) of such messages sent during the computation under consideration. Assume, without loss of generality, that $m_1$ has been sent by $P_1$. Thus, we must have:

$$(P_1||P_2||X_{\phi_0})\backslash C \stackrel{\gamma'}{\Longrightarrow} (P_1'||P_2'||X_{\phi_i})\backslash C \stackrel{\overline{c}m_1}{\longrightarrow} (P_1''||P_2''||X_{\phi''})\backslash C$$

During the prefix of the computation, $P_2$ emitted over channels in $C$ only messages in $\phi_i$, for each time slot $i$. Otherwise $P_2$ would not be t.d. stable w.r.t. $\{\phi_i\}_{i\geq 0}$. So, the trace may be simulated by $P_1$ in composition with $X_{\phi_i}$. But this implies that $(P_1||X_{\phi_0})\backslash C \stackrel{\gamma}{\Longrightarrow} (P_1''||X_{\phi''})\backslash C$ and $\mathcal{D}(\phi'') \neq \mathcal{D}(\phi_i)$. This leads to a contradiction since we assumed $P_1$ t.d. stable w.r.t. $\{\phi_i\}_{i\geq 0}$. The generic case with an arbitrary number of processes can be proved by induction (on the number $n$ of processes).

**Lemma** *Assume that $P_1$ and $P_2$ are t.d. stable w.r.t. $\{\phi_i\}$; then,*

$$(P_1||P_2||Top_{\phi_0}) \setminus C \leq_{trace} (P_1||Top_{\phi_0}) \setminus C||(P_2||Top_{\phi_0}) \setminus C$$

*Proof.* The proof consists in checking that the following relation is a weak simulation:

$$\begin{aligned}\mathcal{R} = \{((P'||P''||Top_{\phi_i}) \setminus C, (P'||Top_{\phi_i}) \setminus C||(P''||Top_{\phi_i}) \setminus C) \mid \\ (P_1||P_2||Top_{\phi_i}) \setminus C \stackrel{\gamma}{\Longrightarrow} (P'||P''||Top_{\phi_i}) \setminus C, \#^{tick}(\gamma) = i\}\end{aligned}$$

- Assume $(P'||P''||Top_{\phi_i}) \setminus C \xrightarrow{\overline{a}m} (P_1'||P''||Top_{\phi_i}) \setminus C$ due to a transition $P' \xrightarrow{\overline{a}m} P_1'$, with $a \notin C$. Then, also $(P'||Top_{\phi_i}) \setminus C||(P''||Top_{\phi_i}) \setminus C \xrightarrow{\overline{a}m} (P_1'||Top_{\phi_i}) \setminus C||(P''||Top_{\phi_i}) \setminus C$. If the transition is due to a receiving action by $P'$ or one of the processes performs an internal computation then the reasoning is similar.
- Assume $(P'||P''||Top_{\phi_i}) \setminus C \xrightarrow{\tau} (P_1'||P_1''||Top_{\phi_i}) \setminus C$ due to a transition $P' \xrightarrow{\overline{a}m} P_1'$ and $P'' \xrightarrow{a(m)} P_1''$. If $a \notin C$ then the case is easy. Otherwise, since $m \in \mathcal{D}(\phi_i)$ (recall that $P_1'||P_1''$ is t.d. stable), we have that $(P'||Top_{\phi_i}) \setminus C \xrightarrow{\tau} (P_1'||Top_{\phi_i}) \setminus C$ and $(P''||Top_{\phi_i}) \setminus C \xrightarrow{\tau} (P_2''||Top_{\phi_i}) \setminus C$. Thus, we get

$$\begin{aligned}&(P'||Top_{\phi_i}) \setminus C||(P''||Top_{\phi_i}) \setminus C \xrightarrow{\tau} \\ &(P_1'||Top_{\phi_i}) \setminus C||(P''||Top_{\phi_i}) \setminus C \xrightarrow{\tau} \\ &(P_1'||Top_{\phi_i}) \setminus C||(P_1''||Top_{\phi_i}) \setminus C\end{aligned}$$

  The case of synchronization between $P'$ ($P''$) and $Top_{\phi_i}$ is a simplified instance of the previous case.
- Assume $(P'||P''||Top_{\phi_i}) \setminus C \xrightarrow{tick} (P_1'||P_1''||Top_{\phi_{i+1}}) \setminus C$ is due to a time synchronization, i.e., $P' \xrightarrow{tick} P_1'$, $P'' \xrightarrow{tick} P_1''$ and $Top_{\phi_i} \xrightarrow{tick} Top_{\phi_{i+1}}$. Then, $(P'||Top_{\phi_i}) \setminus C \xrightarrow{tick} (P_1'||Top_{\phi_{i+1}}) \setminus C$ and $(P_1'||Top_{\phi_i}) \setminus C \xrightarrow{tick} (P_1''||Top_{\phi_{i+1}}) \setminus C$ (due to time stability of $P_1$ and $P_2$ and henceforth of $P'$ and $P''$). Thus, we get

$$(P'||Top_{\phi_i}) \setminus C||(P''||Top_{\phi_i}) \setminus C \xrightarrow{tick} (P_1'||Top_{\phi_{i+1}}) \setminus C||(P_1''||Top_{\phi_{i+1}}) \setminus C$$

We can generalize the previous result as follows.

**Lemma** *Assume that $\{P_j\}_{j=1,\ldots,n}$ are t.d. stable w.r.t. $\{\phi_i\}_{i\geq 0}$; then,*

$$(P_1||\ldots||P_n||Top_{\phi_0}) \setminus C \leq_{trace} (P_1||Top_{\phi_0}) \setminus C||\ldots||(P_n||Top_{\phi_0}) \setminus C$$

*Proof.* By induction on $n$. The base case is trivial. The case $n+1$ may be treated as follows:

$$\begin{array}{ll}(P_1||\ldots||P_n||P_{n+1}||Top_{\phi_0}) \setminus C & \leq_{trace} \text{ Lemma } \textbf{??} \\ (P_1||\ldots||P_n||Top_{\phi_0}) \setminus C||(P_{n+1}||Top_{\phi_0}) \setminus C & \leq_{trace} \text{ Induction hypothesis} \\ (P_1||Top_{\phi_0}) \setminus C||\ldots||(P_{n+1}||Top_{\phi_0}) \setminus C & \end{array}$$