UNIVERSITE D'ORLEANS
Faculté des Sciences
LIFO

Laboratoire d'Informatique Fondamentale d'Orléans
4, rue Léonard de Vinci, BP 6759
F-45067 Orléans Cedex 2
FRANCE

Rapport de Recherche

www : http://www.univ-orleans.fr/SCIENCES/LIFO/

A Timed Synchronous Algebra for Information Flow Analysis

Siva ANANTHARAMAN, LIFO, Orléans (Fr.)
Jing CHEN, LIFO, Orléans (Fr.)
Gaétan HAINS, LIFO, Orléans (Fr.)

Rapport N^o **2004-03**

A Timed Synchronous Algebra for Information Flow Analysis

Siva Anantharaman, Jing Chen, Gaétan Hains

LIFO - Université d'Orléans (France),
e-mail: {siva, ghains, chen}@lifo.univ-orleans.fr

Abstract. The formalism of value-passing processes is well-suited for modeling and analyzing protocols. Timed processes account for the time consumed by the actions of the agents. In this paper we define process algebras with a view to cover both these aspects. Synchronous composition will be the central notion in our definition, since the role played by the synchronous branches is crucial in the analysis of information flow on value-passing processes. Algebraically, parallel synchronous composition will be distributive over the choice operator, and by adding the action prefixes we get a '$PACUID$'-signature on which we define our process algebras. With such a vision, we propose an approach based on natural observation criteria and associated notions of non-interference, and show that they are well adapted for the confidentiality and authentication analysis of cryptographic protocols. Our algebra can also be extended into a timed process algebra with a simple algebraic and operational semantics.

1 Introduction

In the context of information flow analysis, it is important to ensure the security of the data transmitted over distributed systems. Different formalisms have been proposed for modeling the various aspects of security in the information flow on such systems. The notions of non-interference and of admissible interference have been extensively studied with such an objective. On approaches based on process algebras – e.g. [9],[10],[20] –, these notions are defined modulo equivalence with respect to weak trace or weak bisimulation; and the internal CCS action τ keeps its 'magic' property of being unobservable even for the clock. This is in a sense unsatisfactory since the 'divergence' of a process – where an internal action τ loops for too long – will go unobservable then by definition, thus defeating one of the objectives of process algebras namely representation of liveness, deadlock or livelock.

We have proposed a different approach in [2] by designing the process algebra over a $PACUID$-equational theory: let '+' denote non-deterministic choice, then '+' is ACUI (Associative-Commutative, admits the null action as Unit, and is Idempotent), add a binary '$*$' denoting parallel synchronous composition assumed 2-sided Distributive over '+', and then add Prefixes symbolizing actions. There are two essential differences in such a vision as compared to those based on CCS: i) parallel composition of actions is defined with an explicit focus on the synchronous side; asynchronous parallel composition – although not formally part of the setup – can be 'recovered' in our formalism in complete conformity with CCS; and ii) the internal τ of CCS is replaced by an action denoted θ which will be 'internal *except* for the clock'. One of our objectives in doing so has been to introduce a notion of discrete time, and associate it to the 'internal action' of processes. This in particular eliminates the observational problems caused by the divergence of processes; and as we shall be showing below, such a vision allows us also to define 'timed process algebras' in an easier and more natural manner than in the works [22, 12, 11], by a simple extension of our algebraic setup.

The resulting calculus allows us to handle in a simple manner notions such as 'timeout' and 'idling' for any chosen set of actions. As we shall see, our synchronization operator '$*$' seems to capture the essence of the notion of 'time determinacy'; and the livelock loops encountered in the timed algebra of [11] disappear under our timed approach.

Bisimulation between processes reduces in our setup to congruence modulo the $PACUID$-equational theory (i.e. $ACUID$ extended with the action prefixes) underlying the process algebra. We proposed in [2] a method for the secrecy analysis via a notion of *robust non-interference*, based on strong bisimulation in our synchronous setup. In the current paper we extend in formal terms such a vision to the value-passing setup, and define appropriately the notions of observation criteria and that of non-interference. The power of our formalism as an efficient tool for information flow analysis is illustrated by applying it to the analysis of authentication and confidentiality of several cryptographic protocols. We in particular show that the issue of determining whether a given protocol has an 'information leak' for some intruder, can be formulated exclusively in terms of synchronous communication in the $PACUID$-equational syntax.

The paper is structured as follows: We begin by recalling (in Sections 2 and 3) the definition of our synchronous process algebra (as presented initially in [2]), its principal algebraic properties, and the notion of robust non-interference ('RNI'). In Sections 4 and 5, we extend and refine this algebraic formalism and the notion of RNI to the framework of value-passing processes; in this refinement, the discrete time-unit action θ gets indexed by suitable annotations (sequences of value-expressions) which keep track of the flow of data. In Sections 5,6,7, we apply such a vision to modeling and analyzing cryptographic protocols, with special attention to the issues of authentication and of confidentiality. We show in particular that if there is an attack of confidentiality on a process P then there is a linear attacker on a linear sub-process of P; an NP-algorithm for detecting confidentiality attacks on protocols modeled as value-passing processes is also derived. Several concrete examples of analysis are also given in Section 7. In Section 8 we show how our synchronous algebra can be extended in a natural and simple manner as a 'timed process algebra'; the operational and algebraic semantics of our timed algebra are much simpler than for those proposed in [22, 12, 11]. We illustrate the expressive power of our timed extension by modeling and analyzing two typical timed examples borrowed from the works cited. In a final section we show how to define partial evaluation techniques in our timed synchronous setup; such techniques are based on modal calculi using the Hennessy-Milner logic or its refinements (see [19, 24]) and are very useful for formal verification purposes, cf. [4, 11, 17, 18]. This part of our work is still on-going research, however.

2 A Synchronous Process Algebra

Process algebras are constructed here over a given set $\mathtt{EAct}$ of (extended) action symbols. Let $\mathtt{EAct} = \{\theta\} \biguplus \{a, \bar{a}, b, \bar{b}, \ldots\}$ where θ is a special 'internalizing' action similar to the τ in CCS, but its role in information flow will be different: it will 'count' a unit of time, meant to be observable. $\mathtt{Act}$ will denote the subset of non-θ actions of $\mathtt{EAct}$ (and referred to as 'standard' actions when necessary): 'bars' serve in pairing out synchronizing (or communicating) actions, in particular it is assumed $\bar{\bar{x}} = x$; we refer to $\bar{x}$. as the conjugate of x. The special action θ is assumed conjugate to itself. The grammar of ground process terms is defined as follows:

$$\mathtt{Proc} ::= 0 \mid \mathtt{EAct}.\mathtt{Proc} \mid \mathtt{Proc} + \mathtt{Proc} \mid \mathtt{Proc} * \mathtt{Proc} \mid P \setminus \mathtt{Act}.$$

Its operations are respectively called null process, prefix, (non-deterministic) choice, (parallel) synchronous composition, and restriction. Process $x.0$ will be abbreviated to x and $x.(y.0)$

to xy where no confusion is likely between processes and their traces. We will use variables $x, y, \ldots \in \mathtt{EAct}$, $\alpha, \beta, \ldots \in \mathtt{EAct}^*$ and $P, Q, R, \ldots \in \mathtt{Proc}$. The synchronization of actions is a (partial) binary operation denoted '$*$', defined as follows:

$$x * y = \begin{cases} \theta & \text{if } x = \bar{y} \\ x & \text{if } y = \theta \\ y & \text{if } x = \theta \\ \mathtt{undef} & \text{otherwise} \end{cases}$$

We shall agree to denote an undefined action as the null process (i.e. to say by 0) since no confusion is likely to arise. The operational semantics of our processes is a labeled transition system, corresponding to a relation on $\mathtt{Proc} \times \mathtt{EAct} \times \mathtt{Proc}$, denoted as $P \xrightarrow{x} Q$ and defined by the following rules:

(Prefix) $$\frac{}{x.P \xrightarrow{x} P}$$

(Sum) $$\frac{P \xrightarrow{x} P'}{P + Q \xrightarrow{x} P'} \qquad \frac{Q \xrightarrow{x} Q'}{P + Q \xrightarrow{x} Q'}$$

(Sync) $$\frac{P \xrightarrow{x} P' \quad Q \xrightarrow{y} Q' \quad x * y = z \neq 0}{P * Q \xrightarrow{z} P' * Q'}$$

(Restr) $$\frac{P \xrightarrow{x} P' \quad x \neq y, \quad y \neq \theta}{P \setminus y \xrightarrow{x} P' \setminus y}$$

In the current work we shall be essentially concerned with finite processes.

Definition 1 *For any process term P the set of possible actions for P is defined as the set:* $\mathtt{EAct}(P) = \{x \in \mathtt{EAct} \mid \exists P'.\ P \xrightarrow{x} P'\}$ *and the set of its direct successor processes as:* $\mathtt{Next}(P) = \{P' \in \mathtt{Proc} \mid \exists x.\ P \xrightarrow{x} P'\}$.

Remarks 1. i) Rule (Sync) is symmetric in $x, \bar{x}$ because we are assuming $\bar{\bar{x}} = x$.

ii) The semantics behind the condition $y \neq \theta$ in the rule (Restr) is that it is impossible to prevent a process from 'counting' its internal actions.

2.1 Bisimulation

Simulation and bisimulation are defined as usual. Simulation is a preorder relation on the set of all processes; and bisimulation is an equivalence relation. It is in fact a congruence: if $X \approx X'$ and $Y \approx Y'$, and x is any action symbol, then we have:

$$0 \approx 0, \quad x.X \approx x.X', \quad X + Y \approx X' + Y', \quad X * Y \approx X' * Y', \ \text{and} \ X \setminus x \approx X' \setminus x.$$

Bisimulation satisfies the following properties (where $P, Q, R, ..$ denote arbitrary processes, and $ab, b, ., ., x, y, ..$ are arbitrary actions); they are all easy consequences of our definitions. (The proofs of these, as well as of some other properties concerning the restriction operation $P \setminus x$, are given in the appendix of [2].)

(1) The choice operator '+' is associative, commutative, idempotent, and admits a unit element, up to bisimulation: That is to say, for any process terms P, Q, R we have:

$$P + (Q + R) \approx (P + Q) + R, \quad P + Q \approx Q + P, \quad P + P \approx P, \quad P + 0 \approx P$$

(2) Prefix and (parallel) synchronous composition 'commute': $a.P * b.Q \approx (a*b).(P*Q)$

(3) Any finite process term P is bisimilar to the process term: $\sum_{P \xrightarrow{a} P'} a.P'$. This gives an *additive (normal) form* for any process term, uniquely determined up to bisimulation.

(4) Synchronous composition '$*$' is commutative, and distributes over '$+$', up to bisimulation: We have:

$$P * Q \approx Q * P, \quad P*(Q+R) \approx P*Q + P*R, \quad P*0 \approx 0.$$

(5) Synchronous composition is *non-associative* in general: i.e., we do *not always* have $P*(Q*R) \approx (P*Q)*R$. For example: Let $P = a$, $Q = \bar{a}$ and $R = b$, with $b \notin \{a, \theta\}$. Then $P*(Q*R) = a*(\bar{a}*b)$ has no possible transition because $Q*R$ has none. But on the other hand $(P*Q)*R = (a*\bar{a})*b \xrightarrow{\theta*b=b} 0*0$, because $(P*Q) = (a*\bar{a}) \xrightarrow{\theta} 0$ and $R = b \xrightarrow{b} 0$.

(6) However, the following assertion is easily checked: let $a_i, i = 1..n$, be a family of actions *such that, for all* $i \neq j$, $a_i * a_j$ *is defined*; then the '$*$'-composition of the family is a unique action symbol, well-defined *independently of the way in which they synchronize two by two.*

2.2 Bisimulation as an Equational Congruence

Let $\equiv$ be the smallest congruence on the set `Proc` of all process terms defined by: $P \equiv Q$ iff $P = Q$ or P can be obtained from Q by applying one or more of the following equational axioms, where a, b range over the set of all action symbols:

(A) $X + (Y+Z) \doteq (X+Y)+Z$	(C) $X + Y \doteq Y + X$
(I) $X + X \doteq X$	(D) $X*(Y+Z) \doteq X*Y + X*Z$
(U) $X + 0 \doteq X$	(C*) $X * Y \doteq Y * X$
(Z) $X * 0 \doteq 0.$	(P) $a.X * b.Y \doteq (a*b).(X*Y).$

We claim that '$\equiv$' is the same as the congruence of bisimulation between processes. We begin by proving first that $\equiv$ is correct w.r.t. bisimulation. We denote by $P[R]$ a process term P having R as subterm; in this context $P[R']$ represents term P where the occurrences of R have been replaced by the process term R'.

Proposition 1 *$P \equiv Q$ implies $P \approx Q$*

Proof: Suppose $R \equiv R'$ be an instance of any one of the axioms. Then, one of the above propositions shows that $R \approx R'$ and since bisimulation is a congruence for process terms, this equivalence can be propagated to $P[R] \approx P[R']$. Any equational proof of $P \equiv Q$ is a finite sequence of such transformations. As all of them preserve bisimulation, transitivity of bisimulation implies $P \approx Q$. □

To prove the reverse implication, we need a preliminary lemma.

Lemma 1. *$P \approx 0$ implies $P \equiv 0$.*

Proof: Since P is bisimilar to 0, it follows that $\mathtt{EAct}(P) = \emptyset$. We will reason by induction on the size of terms, by cases on the syntax of P and use several times the lemma showing that any term is bisimilar to its additive normal form.

If $P = 0$ there is nothing to prove: such a P may not be of the form $a.P'$ because $\mathtt{EAct}(P) = \emptyset$.

If $P = P' + P''$ then we must have $\mathtt{EAct}(P') = \mathtt{EAct}(P'') = \emptyset$, otherwise there would be a transition for P, and since P', P'' are smaller terms than P we may use the induction hypothesis to conclude that $P' \equiv 0$, $P'' \equiv 0$ and hence $P = P' + P'' \equiv 0$.

If $P = P' * P''$, let the additive normal form for P' and P'' be respectively $\sum_i a_i.P'_i$, $\sum_j b_j.P''_j$. By induction hypothesis, P' and P'' are also equationally equivalent to those sums and distributivity implies $P = P' * P'' \equiv \sum_{i,j}(a_i.P'_i) * (b_j.P''_j)$. Moreover we know that every product $a_i * b_j$ is undefined because otherwise the rule for transitions of a product would have implied a transition for $P = P' * P'$. Hence the axioms imply that $\forall i, j.(a_i.P'_i) * (b_j.P''_j) \equiv 0$ and thus $P = P' * P'' \equiv \sum_{i,j} 0 \equiv 0$.

If $P = P' \setminus a$ then either $P' \approx 0$ or not. In the first case, by induction we have $P' \equiv 0$ so $P \equiv 0 \setminus a \equiv 0$. Otherwise there exists an action $b \in \mathtt{EAct}(P')$ and, since $P' \setminus a$ has no possible action, the only such b must be a and hence $P' \approx \sum_i a.P'_i$. Since P' is smaller than P we may apply the induction hypothesis to yield $P' \equiv \sum_i a.P'_i$. From this it follows that $P = P' \setminus a \equiv (\sum_i a.P'_i) \setminus a \equiv \sum_i (a.P'_i \setminus a) \equiv \sum_i 0 \equiv 0$. □

Proposition 2 $P \approx Q$ *implies* $P \equiv Q$.

Proof: If $P \approx 0$ then $P \equiv 0$ by the previous lemma and similarly for $Q \equiv 0 \equiv P$. So let us assume that $\mathtt{EAct}(P)$ and $\mathtt{EAct}(Q)$ are non-empty.

It is easy to verify that $\mathtt{EAct}(P) = \mathtt{EAct}(Q)$; let A denote this common set of actions. Then we know that $P \approx \sum_{a \in A} P_a$ and $Q \approx \sum_{a \in A} Q_a$. Bisimulation implies that for every $a \in A$ we have $P_a \approx Q_a$ and by induction hypothesis (the terms P_a, Q_a are smaller than P, Q respectively) $P_a \equiv Q_a$. One deduces then, by commutativity and associativity of $+$, that we have $P \equiv Q$. □

Putting together the above two propositions, we get:

Theorem 1 $P \approx Q$ `if and only if` $P \equiv Q$.

The $PAUCID$-equational theory given above can be turned into a *convergent* system of rewrite rules, modulo the $ACUI$-axioms for the choice operation '+', commutativity axiom for the synchronization operation '$*$', and visualizing each prefix (i.e. action symbol) as a monadic function symbol. In particular, bisimulation between finite process terms is decidable by normalization under this convergent rewrite system.

2.3 Asynchronous Parallel Composition

Although not formally part of the setup, the notion of asynchronous parallel composition between processes – for which we shall use the classical notation '$|$' - can be defined uniquely up to bisimulation in our formalism. This is done via structural induction. (It can also done by induction on '$*$' for finite processes, see Section 8.)

Let $P = \sum_{a_i} a_i.P_i$ and $Q = \sum_{b_j} b_j.Q_j$ be the respective additive normal forms of P, Q. Then $P \mid Q$ is the process term defined via structural induction, and uniquely up to bisimulation, as:

$$P \mid Q \;\;=\;\; \sum_{a_i} a_i.(P_i \mid Q) \;+\; \sum_{b_j} b_j.(P \mid Q_j) \;+\; \sum_{a_i \neq \theta \neq b_j} (a_i * b_j).(P_i \mid Q_j)$$

where the binary '$*$' is defined as above. The last branch concerns only synchronization between standard (non-θ) actions, so it is straightforward that we get the asynchronous composition of P and Q à la CCS, once every occurrence of θ in the resulting term is replaced by CCS's τ.

Parallel asynchronous composition of any finite family of processes $P_1, \ldots, P_r$ is defined similarly, by treating the processes successively in pairs; it is uniquely determined up to bisimulation, and is associative-commutative on $P_1, \ldots, P_r$; we shall employ the standard CCS notation $P_1 \mid \ldots \mid P_r$ for this parallel asynchronous composition.

3 Observation Criteria and Non-Interference (Pure CCS)

On a transition system which is 'abstracted' as a process algebra, a natural and simple way of ensuring secrecy in the flow of information is to attribute levels of secrecy to the action symbols, and check that there is no information flow from any given action to any other action of a lower level of secrecy(e.g. [5]).

For readability we shall limit the discussion to two levels, denoted respectively Hi (or H) and Lo (or L), and referred to as *high* and *low*. We thus assume that the set `EAct` of action symbols is partitioned into two subsets denoted by Hi and Lo, representing 'secret' and 'public' events respectively; θ is assumed to be in Lo ('clock is observable by anyone'). Both Hi and Lo are assumed stable under conjugacy, i.e. $Hi = \bar{H}i, Lo = \bar{L}o$. An arbitrary action symbol in Hi (resp. Lo) will be denoted hi (resp. lo). Following [5], we define then the *observation criterion* $\mathcal{O}_L$ for users of level L as follows: For any pure (non value-passing) process Q, its observable image $Q/\mathcal{O}_L$ is the process constructed inductively from Q under the following inference rules:

(`No_Hiding`) $$\frac{P \xrightarrow{x} P' \qquad x \in Lo}{P/\mathcal{O}_L \xrightarrow{x} P'/\mathcal{O}_L}$$

(`Hiding`) $$\frac{P \xrightarrow{x} P' \qquad x \in Hi}{P/\mathcal{O}_L \xrightarrow{\theta} P'/\mathcal{O}_L}$$

The semantics behind these inference rules is that for any high level action x, the part observable by a low level observer reduces to the time units that x consumes. For any given L, the observation criterion $\mathcal{O}_L$ is compatible with bisimulation (cf. the appendix in [2]).

Definition 2 *Let P be any process, and L any given set of ('low') actions. Then P is said to satisfy* non-interference *w.r.t. the observation criterion $\mathcal{O}_L$ iff the following holds:*

$$(\textbf{RNI}) \qquad P/\mathcal{O}_L \ \leq P$$

Intuitively this means that any branch of P observed modulo $\mathcal{O}_L$ is already a branch of P. We shall refer to this notion as RNI ('*Robust Non-Interference*'); the adjective '*Robust*' reflects the fact that the notion is based on exact – and not weak – simulation. We first give a characterization of RNI in terms of bisimulation. Let H be the set of actions not in L, i.e. $H = \texttt{Act} \setminus L$. Following usual convention, for any process Q, $Q\backslash H$ is the process obtained from Q by eliminating the branches of Q where the action prefixes are in H: if $H = \{h_1, \ldots, h_n\}$, then $Q\backslash H = Q\backslash h_1\backslash \ldots \backslash h_n$. We then have the following theorem:

Theorem 2 *A process P satisfies RNI if and only if: $P/\mathcal{O}_L \approx\ P\backslash H$.*

Proof: We first prove the 'only if' assertion, inductively. Assume that P satisfies RNI, i.e. $P/\mathcal{O}_L \leq P$, and let $P\backslash H \xrightarrow{x} P'\backslash H$ be a transition on the process $P\backslash H$ to some process $P'\backslash H$. Then x must be a lo: so we also have $P/\mathcal{O}_L \xrightarrow{x} P'/\mathcal{O}_L$, and by inductive hypothesis we have that $P'\backslash H \approx\ P'/\mathcal{O}_L$. we deduce therefore that $P\backslash H \leq P/\mathcal{O}_L$. On the other hand, if z is any action such that we have a transition $P/\mathcal{O}_L \xrightarrow{z} Q$ to some Q, then by definition of $\mathcal{O}_L$, the action z must be lo, so cannot be a hi; so we actually get a transition of the form $P\backslash H \xrightarrow{z} Q$; we thus deduce that $P\backslash H \geq P/\mathcal{O}_L$. In other words: $P\backslash H \approx\ P/\mathcal{O}_L$.

The proof of the 'if' assertion is even more straightforward: consider any transition on $P/\mathcal{O}_L$ labeled by an action symbol a; then a must be a lo, so there is a transition on $P\backslash H$ labeled by a; which actually must be a transition on P with the same label. So $P/\mathcal{O}_L$ is simulated by P, that is to say P satisfies RNI. $\square$

The notion RNI of robust non-interference defined above is a strict *generalization* of the notion of 'observation based non-deterministic non-interference' developed in the literature, e.g. the $SNNI$ or $BNNI$ of [9]: Processes satisfying $SNNI$ or $BNNI$ do *not* satisfy RNI in general, since they are based only on weak trace and weak bisimulation, respectively.

4 Value-Passing Process Algebras (VPPA's)

In the applications to the analysis of cryptographic protocols that we shall be interested in, processes are conveniently expressed in a so-called value-passing setup. Following Milner's treatment of CCS such value-passing terms are reduced to the pure process algebra by a non-deterministic enumeration of possible values for the value-variables. We recall briefly this extended syntax.

An action in the value-passing syntax consists of either θ or combined names like $c(v), \bar{c}(v)$ where c is any *channel*, and v a value from an externally defined domain (e.g. integers). The '$*$'-product of actions is defined so that any $\bar{c}(v)$ is the conjugate of $c(v)$: $c(v) * \bar{c}(v) = \theta$; all the *other* products are deemed null. In the value-passing setup, the set of transition rules is augmented with the following additional rule:

$$(\texttt{Input}) \qquad \frac{}{c(x).P \xrightarrow{c(v)} P[v/x]}$$

where $P[v/x]$ is considered as synonymous with a term of the form " $\texttt{let}\ x = v\ \texttt{in}\ P$"; the issue of defining expression v's syntax and evaluating it to a value is assumed settled. The case of an 'Output' action is covered by the ($\texttt{Prefix}$) rule, as defined earlier. It must be observed that the operational semantics corresponding to our inference rules is the so-called '*early semantics*' for VPPA's, cf. [19, 13].

We shall be assuming henceforth that all values come from a given domain D, and for our current purpose of finite processes, also that the domain D is finite.

Remarks 2. i) For describing cryptographic protocols we also need some special value terms of the form (m_1, m_2), $\pi_1(m)$, $\pi_2(m)$, $\texttt{enc}(m, k)$, $\texttt{dec}(m, k')$, representing respectively: a message-pair, the first component of a message-pair, the second component of a message-pair, message m encrypted with key (integer) k and the decryption of m by k'. The value-terms in these protocols are then assumed to satisfy the following identities:

$$(1) \qquad \pi_1\ (x, y) = x$$
$$(2) \qquad \pi_2\ (x, y) = y$$
$$(3) \qquad \texttt{dec}(\ \texttt{enc}(x, k), k^{-1}\ ) = x$$

which can be used – once oriented from left to right – as a convergent rewrite system on value terms. The operational semantics of this rewrite system is meant to cover what is known as the *Dolev-Yao model* of intruder agents in cryptographic protocols (cf. [7]). We shall agree here that when the arguments of π_1, π_2 are lists of messages, these are forced into pairs by implicit parenthesizing to the left; this will allow us to drop parentheses, for better readability.

ii) Suppose given two value-passing process P, Q, and C a given set of action channels shared between P or Q. The process $(P \mid Q) \setminus C$ has no branch corresponding to any of the channels in C (because of restriction); however the interleaving semantics of the parallel composition '$|$' defined above leaves intact any branch where P and Q have a synchronized action along any channel in C. Therefore $(P \mid Q) \setminus C$ is said to be the process deduced from P, Q under forced synchronization along C. We shall be using this notion below.

5 Non-Interference in Value-Passing Process Algebras

We shall be principally interested in the issues of confidentiality and of authentication, in analyzing information flows. We propose to show that such an analysis can be based on RNI, for a suitable choice of the observation criterion in any given problem. It is generally agreed that the secrecy analysis of information flow in any process P modeled as VPPA can be done under the following assumption (e.g. [10]): the intruder I has access to all the channels of P, and has at his private disposal an output channel on which he can transmit ('to the outside world') information exchanged between the agents of P that was intended secret but which has been intercepted by I (as clear text). Under such a vision, the observation criterion is to be applied to the process denoted by P_I and defined as $(P \mid I) \setminus \mathcal{X}$ where $\mathcal{X}$ is defined as the channel set of P: the only information observable on P_I is what I can get by synchronizing with the agents of P, that is to say I is in forced synchronization with P.

Let us define (tentatively) process P to satisfy RNI w.r.t. I if and only if: $P_I/\mathcal{O}_{\mathcal{X}} \approx P_I \setminus \mathcal{X}^c$, where $\mathcal{X}^c$ is the set of all channels of P_I not in $\mathcal{X}$ (i.e. to say the private channels of I), and the observation criterion $\mathcal{O}_{\mathcal{X}}$ is to 'hide' the data transmitted along such channels. In the value-passing setup we need to be more precise in defining observation: for the analysis to be meaningful, it should also take into account the data flowing along the channels. We propose to do that by annotating suitably the actions. Here is an illustrative example.

Example 1. Consider $P = a(x).0 \mid \bar{a}(1).0$ where two principals just exchange the value 1; and two possible intruders $I_1 = a(z).\bar{c}(z).0$ and $I_2 = a(z).(\bar{c}(z).0 + \bar{a}(2).0)$; both share with P the 'public' channel $a, \bar{a}$, so $\mathcal{X} = \{a, \bar{a}\}$, but may use $c, \bar{c}$ as 'private' channels.

Case of I_1: Consider $P_{I_1} = (P \mid I_1) \setminus a = (\ \bar{a}(1).0 \mid a(x).0 \mid a(z).\bar{c}(z).0\ ) \setminus a$, where I_1 is in 'forced synchronization with' P. If no data flow is taken into account, we get: $P_{I_1} = \theta.0 + \theta.\bar{c}(1).0$. But here $\mathcal{X}^c = \{c, \bar{c}\}$; so we get $P_{I_1}/\mathcal{O}_{\mathcal{X}} = \theta.0 + \theta.\theta.0$, while $P_{I_1} \setminus \mathcal{X}^c = \theta.0 + \theta.0 = \theta.0$. So the condition RNI is *not* satisfied in this case; this seems in order, since there has indeed been an information leak on P for I_1: the message-value 1 exchanged between the principals in P has been 'intercepted' by I_1, who uses his 'private channel c to transmit it outside'.

Case of I_2: Observe first that process I_2 has I_1 as a 'branch'. Consider now P_{I_2}; the set of channels shared with P is $\mathcal{X} = \{a, \bar{a}\}$. And the processes to compare for the RNI condition are:

$P_1 = (P \mid I_2) \setminus \mathcal{X} \setminus \mathcal{X}^c = \theta + \theta.\theta$, and

$P_2 = ((P \mid I_2) \setminus \mathcal{X})/\hat{\mathcal{O}}_{\mathcal{X}} = \theta + \theta.(\theta + \theta)$

But these *are bisimilar* since $\theta + \theta \approx \theta$! This shows that if the data flow is unaccounted for, information leaks along a branch might well go undetected. To avoid such pathologies, we propose to introduce annotations as suffixes for the actions, in such a way that we shall actually get in the case of I_2 above:

$P_1 = (P \mid I_2) \setminus \mathcal{X} \setminus \mathcal{X}^c = \theta_{\{x:=1\}} + \theta_{\{z:=1\}}.\theta_{\{x:=2\}}$, and

$P_2 = ((P \mid I_2) \setminus \mathcal{X})/\hat{\mathcal{O}}_{\mathcal{X}} = \theta_{\{x:=1\}} + \theta_{\{z:=1\}}.(\theta_{\{z:=1\}} + \theta_{\{x:=2\}})$ □

5.1 Annotating Actions

For any value-passing process E, we want to associate a set $fv(E)$ and $bv(E)$ of variables of E, called respectively: the set of *free variables*, and the set of *bound variables* in E. Intuitively a bound variable x in E can only be introduced by a construction of the form $a(x).E'$, where E' is a sub-process of E. Without loss of generality we assume $bv(E) \cap fv(E) = \emptyset$. The set $fv(E)$ can be computed with the help of the following rules:

$$fv(0) = \emptyset, \quad fv(\bar{a}(y).E) = fv(E) \cup \{y\}, \quad fv(a(y).E) = fv(E) \setminus \{y\}$$

$$fv(E_1 + E_2) = fv(E_1) \cup fv(E_2), \ fv(E_1 * E_2) = fv(E_1) \cup fv(E_2), \ fv(E \setminus C) = fv(E)$$

In the applications that we will be interested in (e.g., analysis of cryptographic protocols), the input or output values transmitted along the channels will not be atomic values in general; they will be more complex expressions formed from such atomic values. Such expressions can be defined formally (cf. e.g. [13]) as follows. Assume given a finite set $\mathcal{F}$ of function symbols on the value-domain. Then a *data-expression* (in short: an *expression*) is any word in the 'language' derivable by the following grammar where v stands for any value-constant or value-variable, and f is any function symbol in $\mathcal{F}$ of suitable arity:

$$e ::= v \mid e, v \mid f(e)$$

An *annotation* is by definition a set of assignments, denoted under the form $\{assignments\}$; an *assignment* is by definition of the form $x := e$; in any assignment $x := e$, the right part e can be any expression, while the left part x can only be a variable.

To keep track of the data-dependence relation in the value-passing setup, we shall index each action with an *annotation* as subscript. Such an annotation is meant to record the assignment events accompanying an action, or the content of an output action. We may 'reformulate' the synchronous product '$*$' in the annotated setup, so as to keep track of the annotations, as follows:

$$\alpha_A * \beta_B = \begin{cases} \theta_{A\cup B} & if\ \alpha = \bar{\beta}\ (or\ \bar{\alpha} = \beta), \\ \alpha_{A\cup B} & if\ \beta = \theta \\ \beta_{A\cup B} & if\ \alpha = \theta \\ 0 & otherwise \end{cases}$$

The operational semantics for the annotated actions on processes remain as previously. The output actions on VPPA's are annotated naturally by the "`let ..=.. in`" instructions preceding them; for instance, an action of the form "`let` $v = e$ `in` $\bar{a}(v)$" is the same as the annotated action: $\bar{a}_{\{v:=e\}}(v)$. An `Input` action of the form $c(x).P \xrightarrow{c(v)} P[v/x]$ gets annotated with the assignment $\{x := v\}$. It is assumed tacitly, that any assigned value-expression gets propagated over the right-hand-sides of all assignments. Bisimulation between VPPA's is henceforth assumed defined w.r.t. the annotated actions. This implies in particular, that two θ-actions with two different annotations are to be distinguished.

Definition 3 *Let P be any process, I a second ('intruder') process, and let $\mathcal{X}$ be the set of 'public' actions of P (thus in particular shared with I). Denote by P_I be the process $(P \mid I) \setminus \mathcal{X}$, and let $\mathcal{O}_{\mathcal{X}}$ denote the observation criterion on P_I as defined above. Then P is said to be* `secrecy preserving` *for I, iff P_I satisfies the property:*

$$(\mathbf{RNI}) \qquad P_I/\mathcal{O}_{\mathcal{X}} \ \approx\ P_I \setminus \mathcal{X}^c$$

Otherwise P is said to present an `information leak` *for I.*

Remark 3. The two processes considered for bisimulation in the above definition of secrecy preservation, namely $(P \mid I) \setminus \mathcal{X} \setminus \mathcal{X}^c$ and $(\,(P \mid I) \setminus \mathcal{X}\,)/\mathcal{O}_{\mathcal{X}}$, both have only branches labeled with (annotated) θ's. It is immediate that the former is simulated by the latter since every sub-term of the former is a sub-term of the latter. So P presents an information leak for I `if and only if` $(\,(P \mid I) \setminus \mathcal{X}\,)/\mathcal{O}_{\mathcal{X}}$ is *not* simulated by $(P \mid I) \setminus \mathcal{X} \setminus \mathcal{X}^c$.

Remark 4. Suppose P is any (value-passing) process, $\mathcal{X}$ its set of actions all assumed public. Then the process $P \setminus \mathcal{X}$ can be seen as reflecting the 'internal behavior' of P (since only the synchronized actions along the channels in $\mathcal{X}$ remain visible on this restriction). In [10], where the approach is based on weak trace equivalence, a notion called *NDC* ('*Non-Deducibility under Composition*') was defined as follows: (With the same notation as above) a process P is said to have the property **NDC** for I if and only if the following holds: $(P \mid I) \setminus \mathcal{X} \approx P \setminus \mathcal{X}$; the semantics is that the internal behavior of P is not modified by the presence of I. We show now that **NDC** is stronger than $RNI_{\mathcal{X}}$.

Proposition 3 *Let P be any VPPA, $\mathcal{X}$ the set of actions of P all assumed public, and I any VPPA such that $(P \mid I) \setminus \mathcal{X} \approx P \setminus \mathcal{X}$. Then P satisfies $RNI_{\mathcal{X}}$, i.e. to say $(P \mid I) \setminus \mathcal{X} \setminus \mathcal{X}^c \approx (\,(P \mid I) \setminus \mathcal{X}\,)/\mathcal{O}_{\mathcal{X}}$.*

Proof: Restriction is compatible with bisimulation; so if P satisfies **NDC** we have: $(P \mid I) \setminus \mathcal{X} \setminus \mathcal{X}^c \approx P \setminus \mathcal{X} \setminus \mathcal{X}^c$; but P has no channel in $\mathcal{X}^c$, so we have $P \setminus \mathcal{X} \setminus \mathcal{X}^c = P \setminus \mathcal{X}$. Similarly, the observation criterion $\mathcal{O}_\mathcal{X}$ being compatible with bisimulation (cf. [2]), we deduce that, if P satisfies **NDC**, we also have: $(\,(P \mid I) \setminus \mathcal{X}\,)/\mathcal{O}_\mathcal{X} \approx P \setminus \mathcal{X}$. □

Example 2. The notion of **NDC** is *strictly* stronger than secrecy preservation: Let P be a simple 'ping-pong protocol' where two principals exchange a value back and forth: $P = \bar{a}(1).a(y) \mid a(x).\bar{a}(x)$; assume that all the actions along the channels $\{a, \bar{a}\}$, are public. Then the internal behavior of P is $P \setminus a \approx \theta_{\{x:=1\}}.\theta_{\{y:=x\}}.0$. The process P does *not* satisfy **NDC** for $I = a(z)$; indeed: $P_I = (P \mid I) \setminus a \approx \theta_{\{z:=1\}}.0 + \theta_{\{x:=1\}}.(\theta_{\{z:=1\}}.0 + \theta_{\{y:=1\}}.0)$. However, $(P \mid I)$ has no channels outside $\{a, \bar{a}\}$, so the process P_I trivially satisfies $RNI_\mathcal{X}$: $P_I/\mathcal{O}_\mathcal{X} = P_I = P_I \setminus \mathcal{X}^c$. Note that there has been *no leak* here: no information has been intercepted and 'transmitted to the outside world' by I, although the presence of I has been the cause for some 'parasite' branches. □

As we shall be seeing in the coming sections, observation criteria based on annotated actions have sufficient expressive power as tools of analysis of authentication and confidentiality in cryptographic protocols.

6 Authentication Analysis based on *RNI*

Our considerations here will be restricted to cryptographic protocol analysis. In contrast with confidentiality, there does not seem to be a 'uniformly accepted' definition of authentication. We shall see in the following section that 'authentication with *public* keys' can actually be modeled as confidentiality; and the analysis can then be based on RNI by an appropriate choice of the observation criterion. Our purpose in this section is to consider 'authentication based on *private* keys' with the help of a 'trusted' server. In such an approach, there is an *initiator* P, a *respondor* Q, and a server A; each agent X shares a symmetric key k_{XA} with the server A, which is assumed secret to any other agent $Y \neq X$.

A session is always started by the initiator P, towards the respondor Q. The role of the server is to help Q to be sure that the session he is running is indeed with P, for this purpose he sends a certain message to Q encrypted with k_{QS}; if 'mutual authentication' is needed, then the server in addition includes a symmetric session key (denoted k_{PQ} and supposed fresh for the given session between P and Q) inside the message that it sends to Q.

Following essentially the work [27] – and less restrictively than in [16] – we *define such an authentication to be correct if and only if*:

- *When Q is brought to 'believe' that he is talking to P* – i.e. to say when Q receives the authenticating message from the server A – *then indeed P is the initiator of the session.*

Our aim is to define, for any given protocol, such a correction as RNI w.r.t. a suitable observation criterion. We begin by analyzing a couple of well-known examples of authentication protocols based on private keys, where authentication bugs or 'attacks' have been detected, arising from impersonations (or masquerading) by agents that are regular participants in other protocol sessions.

In the following two subsections, we shall denote by c_{XY} (resp. by $\overline{c}_{XY}$) a symmetric input (resp. output) communication channel between any two distinct agents X, Y participating in a session of any protocol modeled as a VPPA. All actions using such channels will be assumed public; the actions that an agent X performs for his own sake will be denoted as $\texttt{Self}_X(..)$, and most of them will be assumed private. We shall also be using the names of the agents as value-constants (instead of assigning to each agent a value-constant as identity).

6.1 The Woo and Lam One-Way Authentication protocol

The specification of the Woo and Lam protocol is as follows (cf. [27]):

$$\begin{array}{llcl}
(1) & P \rightarrow Q & : & \text{"I am P"} \\
(2) & Q \rightarrow P & : & \text{(generate and) send nonce } n \\
(3) & P \rightarrow Q & : & \{n\}_{K_{PA}} \\
(4) & Q \rightarrow A & : & \{P, \{n\}_{K_{PA}}\}_{K_{QA}} \\
(5) & A \rightarrow Q & : & \{n\}_{K_{QA}}
\end{array}$$

which can be modeled as a VPPA, as follows:

$$\begin{array}{lcl}
INIT_{XY} & = & Init_{X2Y}.\overline{c}_{XY}(X).c_{YX}(x).\overline{c}_{XY}(\{x\}_{K_{XA}}). \\
RESP_{XY} & = & c_{XY}(X).\overline{c}_{YX}(n).c_{XY}(\{n\}_{K_{XA}}).\overline{c}_{YA}(\{X, \{n\}_{K_{XA}}\}_{K_{YA}}).c_{AY}(\{n\}_{K_{YA}}). \\
SERV_{XY} & = & c_{YA}(x).\mathtt{Self}_A(Y, x, x').\mathtt{Self}_A(\pi_1(x'\!\downarrow), \pi_2(x'\!\downarrow), x'').\overline{c}_{AY}(\{x''\}_{K_{YA}}).
\end{array}$$

where: by $Init_{X2Y}$ we denote an atomic action indicating that X is starting a protocol session with Y, and by $\mathtt{Self}_A(X, x, x')$ we denote an action of the server A meant for himself, with the semantics that m is the message received from agent X and $x' := \{x\}_{K_{XA}^{-1}}$. And as earlier, for any value-expression e, its Dolev-Yao normal form is denoted by $e\!\downarrow$. The entire protocol is then modeled as the VPPA where the above three are in forced synchronization along the public channels. That is to say:

$$PROC_{XY} \;=\; (INIT_{XY} \mid RESP_{XY} \mid SERV_{XY})$$

For checking the correctness of authentication of a given agent P w.r.t. a given Q, we define an observation criterion denoted as $\mathcal{O}_{P\rightarrow Q}$ or as $\mathcal{O}_L$, by specifying a set L of value-passing actions referred to as 'low-level' (for authenticating P w.r.t. Q during a given session), as below:

- Actions of the form $Init_{X2Y}$ are in L if and only if $X = P$ and $Y = Q$.
- Actions of the form $\mathtt{Self}_A(X, m, m')$ are in L if and only if $X = P$ or $X = Q$.
- Every other value-passing action is in L.

The set of value-passing actions not in L, will be denoted by H. An action of the form $\mathtt{Self}_A(X, m, m')$ where $X \neq P$, $X \neq Q$ is in H, so will be 'hidden' under $\mathcal{O}_L$; observed via $\mathcal{O}_L$ it will be an annotated theta, denoted as θ_A^X (a private action of A intended for himself, but related to agent X).

We can now express the correctness condition for the above protocol as follows: The *authentication of* P *w.r.t.* Q *is* `correct` - in presence of a given finite set $\mathcal{P}$ of (other) principals - *if and only if the following holds, for* `any` Z:

$$\begin{array}{lc}
(\textbf{RNI}): & ((S \mid Z)\setminus C)/\mathcal{O}_L \;\leq\; ((S \mid Z)\setminus C) \\
\text{or equivalently:} & ((S \mid Z)\setminus C)/\mathcal{O}_L \;\approx\; ((S \mid Z)\setminus C)\setminus H
\end{array}$$

where S is of the form $(PROC_{PQ} \mid PROC_{QR} \mid PROC_{RS}..)$, with $R, S, \ldots \in \mathcal{P}$; and Z is a process using some of the actions of S; and C= set of all public actions of S and Z.

If Z is a principal w.r.t. whom the above criterion for authentication fails, we say that Z is an 'enemy' or that Z 'defines an (authentication) attack' on the protocol. The following three attacks on the Woo and Lam One-way Authentication protocol have been reported in the literature; all the three in fact fail to satisfy our RNI-criterion for some Z. We shall be using below the uniform notation $X(Y)$ to indicate that agent X is masquerading as agent Y, both for reception and emission of messages.

<u>Attack n^o 1</u>: This attack corresponds to the following sequence where an enemy Z masquerading as P w.r.t. Q, manages to get authenticated as P by intercepting a session initiated later by Q with some regular agent R; this masquerading of P is modeled as Z using the channel $\overline{c}_{PQ}$ to send his initiator message to Q (as will be done also in Example 3 farther down).

$$
\begin{array}{llcl}
(PQ.1) & Z(P) \to Q & : & \text{"I am P."} \\
(PQ.2) & Q \to Z(P) & : & \text{(generate and) send nonce} \quad n \\
(PQ.3) & Z(P) \to Q & : & G \\
(PQ.4) & Q \to Z(A) & : & \{P, G\}_{K_{QA}} \\
(QR.1) & Q \to Z(R) & : & \text{"I am Q."} \\
(QR.2) & Z(R) \to Q & : & (\textit{send as nonce})\ Z, \{n\}_{K_{ZA}} \\
(QR.3) & Q \to Z(R) & : & \{Z, \{n\}_{K_{ZA}}\}_{K_{QA}} \\
(QR.4) & Z(Q) \to A & : & \{Z, \{n\}_{K_{ZA}}\}_{K_{QA}} \\
(QR.5) & A \to Z(Q) & : & \{n\}_{K_{QA}} \\
(PQ.5) & Z(A) \to Q & : & \{n\}_{K_{QA}}
\end{array}
$$

We can model the enemy Z as the following VPPA:

$$
\begin{aligned}
Z \ == \ & \overline{c}_{PQ}(P).c_{QP}(n).\overline{c}_{PQ}(G).c_{QA}(\{P,G\}_{K_{QA}}).c_{QR}(Q).\overline{c}_{RQ}(Z,\{n\}_{K_{ZA}}). \\
& c_{QR}(\{Z,\{n\}_{K_{ZA}}\}_{K_{QA}}).\overline{c}_{QA}(\{Z,\{n\}_{K_{ZA}}\}_{K_{QA}}).c_{AQ}(y).\overline{c}_{AQ}(y).
\end{aligned}
$$

Then the following process does not satisfy the RNI condition:

$$Proc' = (PROC_{PQ} \mid PROC_{QR} \mid PROC_{ZQ} \mid Z) \setminus C'$$

where $C' = \{c_{XY}, \overline{c}_{XY} \mid X, Y \in \{P, Q, R, A\}, X \neq Y\}$. Indeed, the following is a subprocess of $Proc'$:

$$\mathtt{Self}_A(Q, x, x').\mathtt{Self}_A(\pi_1(x'\!\downarrow), \pi_2(x'\!\downarrow), x'').\theta_{\{y:=\{x''\}_{K_{QA}}\}}.\theta$$

where $x' = \{Z, \{n\}_{K_{ZA}}\}$, so $\pi_1(x'\!\downarrow) = Z \notin \{P, Q\}$; consequently, in the process $Proc' \setminus H$ the branch $\mathtt{Self}_A(\pi_1(x'\!\downarrow), \pi_2(x'\!\downarrow), x'').\theta_{\{y:=\{x''\}_{K_{QA}}\}}.\theta$ will be cut out, but it will subsist as $\theta^Z_A.\theta_{\{y:=\{x''\}_{K_{QA}}\}}.\theta$ in the process $Proc'/\mathcal{O}_L$, with the $\mathtt{Self}_A(Z, ...)$ action replaced by a θ^Z_A. This leads to the violation of the RNI-condition. □

<u>*Attack n^o* 2</u>: Here the enemy Z attacks by initiating two parallel sessions with Q, one regular and the other where he masquerading as P; he in addition manoeuvres in such a way that the regular session gets aborted (at the server's level). This corresponds to the following sequence:

$$
\begin{array}{llcl}
(PQ.1) & Z(P) \to Q & : & \text{"I am P."} \\
(ZQ.1) & Z \to Q & : & \text{"I am Z."} \\
(PQ.2) & Q \to Z(P) & : & n_P \\
(ZQ.2) & Q \to Z & : & n_Z \\
(PQ.3) & Z(P) \to Q & : & G \\
(ZQ.3) & Z \to Q & : & \{n_P\}_{K_{ZA}} \\
(PQ.4) & Q \to A & : & \{P, G\}_{K_{QA}} \\
(ZQ.4) & Q \to A & : & \{Z, \{n_P\}_{K_{ZA}}\}_{K_{QA}} \\
(ZQ.5) & A & : & \text{abort} \\
(PQ.5) & A \to Q & : & \{n_P\}_{K_{QA}}
\end{array}
$$

We model Z as the following VPPA:

$$Z \ == \ \overline{c}_{PQ}(P).c_{QP}(n_P).\overline{c}_{PQ}(G) \ \mid \ \overline{c}_{ZQ}(Z).c_{QZ}(n_Z).\overline{c}_{ZQ}(\{n_P\}_{K_{ZA}})$$

Then the following process does not satisfy the RNI condition:

$$Proc'' = (PROC_{PQ} \mid PROC_{ZQ} \mid Z) \setminus C''$$

where $C'' = \{c_{XY}, \overline{c}_{XY} \mid X, Y \in \{P, Q, A, Z\}, X \neq Y\}$. The reason is essentially the same as for the process $Proc'$.

<u>*Attack n^o* 3</u>: This is the so-called *Man-in-the-middle* attack, i.e., P starts the protocol with Z, and Z exploits this session to impersonate P in a new session he starts with Q:

$$
\begin{array}{llcl}
(PZ.1) & P \rightarrow Z & : & \text{"I am P."} \\
(PQ.1) & Z(P) \rightarrow Q & : & \text{"I am P."} \\
(PQ.2) & Q \rightarrow Z(P) & : & n \\
(PZ.2) & Z \rightarrow P & : & n \\
(PZ.3) & P \rightarrow Z & : & \{n\}_{K_{PA}} \\
(PQ.3) & Z(P) \rightarrow Q & : & \{n\}_{K_{PA}} \\
(PQ.4) & Q \rightarrow A & : & \{P, \{n\}_{K_{PA}}\}_{K_{QA}} \\
(PQ.5) & A \rightarrow Q & : & \{n\}_{K_{QA}}
\end{array}
$$

Here the enemy Z is modeled as:

$$Z \ == \ c_{PZ}(P).Init_{P2Q}.\overline{c}_{PQ}(P).c_{QP}(n).\overline{c}_{QP}(n).c_{PZ}(y).\overline{c}_{PZ}(y).$$

And the entire run of the system is modeled by:

$$Proc''' \ = \ (PROC_{PZ} \mid PROC_{PQ} \mid Z) \setminus C'''$$

where $C''' = \{c_{XY}, \overline{c}_{XY} \mid X, Y \in \{P, Q, A, Z\}, X \neq Y\}$. As is easy to see, this attack can be detected by the first rule of the criterion $\mathcal{O}_{P \rightarrow Q}$, w.r.t. action $Init_{X2Y}$.

6.2 Woo and Lam Mutual Authentication protocol

The specification of the Woo and Lam Mutual Authentication protocol is as follows (cf. the SPORE repository at the url: `http://www.lsv.ens-cachan.fr/spore/table.html`):

$$
\begin{array}{llcl}
(1) & P \rightarrow Q & : & P, N_1 \\
(2) & Q \rightarrow P & : & Q, N_2 \\
(3) & P \rightarrow Q & : & \{P, Q, N_1, N_2\}_{K_{PA}} \\
(4) & Q \rightarrow A & : & \{P, Q, N_1, N_2\}_{K_{PA}},\ \{P, Q, N_1, N_2\}_{K_{QA}} \\
(5) & A \rightarrow Q & : & \{Q, N_1, N_2, K_{PQ}\}_{K_{PA}},\ \{P, N_1, N_2, K_{PQ}\}_{K_{QA}} \\
(6) & Q \rightarrow P & : & \{Q, N_1, N_2, K_{PQ}\}_{K_{PA}},\ \{N_1, N_2\}_{K_{PQ}} \\
(7) & P \rightarrow Q & : & \{N_2\}_{K_{PQ}}
\end{array}
$$

Here is an attack reported in the SPORE repository; it arises from the fact that, during a given session from an initiator P to a certain respondor Z, the latter starts – before getting 'committed' to P, i.e. before the authentication is complete – a new session with P by using the same nonce as the one he received from P !! :

$$
\begin{array}{llcl}
(PZ.1) & P \rightarrow Z & : & P, N_1 \\
(ZP.1) & Z \rightarrow P & : & Z, N_1 \\
(ZP.2) & P \rightarrow Z & : & P, N_2 \\
(PZ.2) & Z \rightarrow P & : & Z, N_2 \\
(PZ.3) & P \rightarrow Z & : & \{P, Z, N_1, N_2\}_{K_{PA}} \\
(PZ.4) & Z \rightarrow A & : & \{P, Z, N_1, N_2\}_{K_{PA}},\ \{P, Z, N_1, N_2\}_{K_{ZA}} \\
(PZ.5) & A \rightarrow Z & : & \{Z, N_1, N_2, K_{PZ}\}_{K_{PA}},\ \{P, N_1, N_2, K_{PZ}\}_{K_{ZA}} \\
(PZ.6) & Z \rightarrow P & : & \{Z, N_1, N_2, K_{PZ}\}_{K_{PA}},\ \{N_1, N_2\}_{K_{PZ}} \\
(PZ.7) & P \rightarrow Z & : & \{N_2\}_{K_{PZ}} \\
(ZP.3) & Z \rightarrow P & : & \{Z, P, N_1, N_2\}_{K_{ZA}} \\
(ZP.4) & P \rightarrow Z(A) & : & \{Z, P, N_1, N_2\}_{K_{ZA}},\ \{Z, P, N_1, N_2\}_{K_{PA}} \\
(ZP.5) & Z(A) \rightarrow P & : & \{Z, N_1, N_2, K_{PZ}\}_{K_{ZA}},\ \{Z, N_1, N_2, K_{PZ}\}_{K_{PA}} \\
(ZP.6) & P \rightarrow Z & : & \{P, N_1, N_2, K_{PZ}\}_{K_{ZA}},\ \{N_1, N_2\}_{K_{PZ}} \\
(ZP.7) & Z \rightarrow P & : & \{N_2\}_{K_{PZ}}
\end{array}
$$

We model here the protocol via the following VPPAs:

$$
\begin{aligned}
INIT_{PQ} &== Init_{P2Q}(i).\overline{c}_{PQ}(P,N_1).c_{QP}(Q,N_2).\overline{c}_{PQ}(\{P,Q,N_1,N_2\}_{K_{PA}})\\
&\quad .c_{QP}(\{Q,N_1,N_2,x_K\}_{K_{PA}},\{N_1,N_2\}_{x_K}).\overline{c}_{PQ}(\{N_2\}_{x_K}).Commit_{P2Q}(i)\\
RESP_{PQ} &== c_{PQ}(P,N_1).\overline{c}_{QP}(Q,N_2).c_{PQ}(w).\overline{c}_{QA}(w,\ \{P,Q,N_1,N_2\}_{K_{QA}})\\
&\quad .c_{AQ}(y_1,\{P,N_1,N_2,x_K\}_{K_{QA}})\overline{c}_{QP}(y_1,\ \{N_1,N_2\}_{x_K}).c_{PQ}(\{N_2\}_{x_K})\\
SERV_{PQ} &== c_{QA}(x_1,x_2).\mathtt{Self}_A(P,x_1,x').let\ (X_1,X_2)=(\pi_2(\pi_1(x')),\pi_2(x'))\ in\\
&\quad \overline{c}_{AQ}(\{Q,X_1,X_2,K_{pq}\}_{K_{PA}},\ \{P,X_1,X_2,K_{pq}\}_{K_{QA}})
\end{aligned}
$$

Note here that in the modeling, we have put a parameter i in both $Init_{P2Q}$ and $Commit_{P2Q}$, which is meant to reflect the 'liveness' of the current session from P to Q. We assume that each agent P keeps a private counter for authentication with respect to any other given agent, e.g. Q; the counter initially has the value 0, and gets increased by 1 each time P does $Init_{P2Q}$, and decreased by 1 at the end of the session: when P does $Commit_{P2Q}$. Thus, the first time P can only do $Init_{P2Q}(1)$ (i.e. the current value is 1), and if P does again $Init_{P2Q}$ before $Commit_{P2Q}(1)$, i.e. before finishing the first session, then the counter will be increased and the action will be $Init_{P2Q}(2)$. But if the second session begins after the first session finished, the counter will never have value more than 1; this simple technique is meant to detect if P has initiated more than 1 session with the same agent Q 'at the same time'.

The enemy action can be modeled as the following VPPA:

$$
\begin{aligned}
Z \quad == \quad & c_{PZ}(P,N_1).\overline{c}_{ZP}(Z,N_1).c_{PZ}(P,N_2).\overline{c}_{ZP}(Z,N_2).c_{PZ}(u).\\
& \overline{c}_{ZA}(u,\{P,Z,N_1,N_2\}_{K_{ZA}}).c_{AZ}(y,\{P,N_1,N_2,x_K\}_{K_{ZA}}).\overline{c}_{ZP}(y,\{N_1,N_2\}_{x_K}).\\
& c_{PZ}(z).\overline{c}_{ZP}(\{Z,P,N_1,N_2\}_{K_{ZA}}).c_{PA}(z_1,z_2).\overline{c}_{AP}(\{Z,N_1,N_2,x_K\}_{K_{ZA}},y).\\
& c_{PZ}(z_1',z_2').\overline{c}_{ZP}(z)
\end{aligned}
$$

which is then a branch of the process:

$$PROC_m = (PROC_{PZ} \mid PROC_{ZP} \mid Z)\setminus C_m$$

where $C_m = \{c_{XY},\overline{c}_{XY} \mid X,Y \in \{P,Z,A\},\ X \neq Y\}$. Now, for analyzing $Proc_m$, it suffices to introduce an integer parameter j in our observation criterion as follows: Given any j, all actions of the form $Init_{P2Q}(i)$ with $i \neq j$, and all actions $Init_{Q2P}(r)$ for any r, are declared 'high' with respect to $\mathcal{O}^j_{P\to Q}$; and use the criterion $\mathcal{O}^1_{P\to Z}$ to detect the above enemy attack. Associating a session number, in such a manner, with the observation criterion $\mathcal{O}_{P\to Q}$ has a well-defined semantical meaning; it can of course be done in all cases, although in our earlier examples the attack detection was not dependent on the session number.

7 Observation Criteria for Confidentiality Analysis

The criterion $\mathcal{O}_{\mathcal{X}}$ can be refined further for analyzing confidentiality, as we shall show now. In any (data-)expression e, we define $Vars(e)$ (resp $Const\_args(e)$) as the set of value-variables appearing in e (resp. value-constants appearing as arguments in the sequence e). Given an annotation A, for each variable x appearing in A, we define $Parent_A(x)$ to be the minimal set such that:

$Parent_A(x) = \emptyset$ if A contains no assignment of the form $x := e$;
$Parent_A(x) = Vars(e) \cup Const\_args(e) \cup \bigcup_{y\in Vars(e)} Parent_A(y)$, if $x := e \in A$.

Since the only operator involved is $\cup$ which is monotonic, the above equations have a *least fixed point* as the solution for the set $Parent_A(x)$.

(Without loss of generality, we can assume - up to α-conversion - that different input actions use distinct bound variables; so it will never arise that an annotation A of actions on a VPPA contains two different assignments of the form: $x := e$, $x := e'$ on a same variable x; therefore the above definition of $Parent_A$ is operationally unambiguous.)

Let E be any *given* process. If $\Gamma = \alpha^1_{A_1}\alpha^2_{A_2}\ldots\alpha^n_{A_n}$ is any sequence of annotated actions in E, we shall write: $E \xrightarrow{\Gamma} E'$ for $E \xrightarrow{\alpha^1_{A_1}}\xrightarrow{\alpha^2_{A_2}} \cdots \xrightarrow{\alpha^n_{A_n}} E'$; and refer to E' as the Γ-successor or the Γ-sub-process of E. For any given value-variable x of E, we set: $Ancestor_E(\Gamma, x) = Parent_{A_1\cup\cdots\cup A_n}(x)$. The set $Ancestor_E(\Gamma, x)$ is the set of 'ancestor' value-variables and Const_args along the path Γ on E, on which the value-variable x depends; if no confusion is likely, we will omit the E in the subscript, and write simply $Ancestor(\Gamma, x)$. Let $\mathcal{X}$ be a *given set of low-level actions in* E, and S a *given set of value-constants* (meant to be kept 'secret'). We propose now to refine the observation criterion $\mathcal{O}_\mathcal{X}$ on E into a criterion denoted $\hat{\mathcal{O}}_\mathcal{X} = \hat{\mathcal{O}}_{\mathcal{X},S}$, with the following intuition: the observation criterion $\hat{\mathcal{O}}_\mathcal{X} = \hat{\mathcal{O}}_{\mathcal{X},S}$ must *'shortcut'* (or 'jump over') all high-level **output** actions, *if the value-variables of these output actions have no ancestors in the set S*. We adopt the following notational convention:

- if Γ is a path of annotated actions and P the Γ-sub-process of E, then we write (Γ, P); the empty path is denoted by Γ_0;
- if $P \xrightarrow{\alpha} P'$ for an annotated action α, then we write: $(\Gamma, P) \xrightarrow{\alpha} (\Gamma.\alpha, P')$.

The following transition rules are then applied to a system *rooted at* E, that we denote as the pair (Γ_0, E):

$$\texttt{(Keep)} \qquad \frac{(\Gamma, P) \xrightarrow{\alpha} (\Gamma.\alpha, P')}{P/\hat{\mathcal{O}}_\mathcal{X} \xrightarrow{\alpha} P'/\hat{\mathcal{O}}_\mathcal{X}} \qquad if\ \alpha \in \mathcal{X}\ or\ \alpha = \theta_A.$$

$$\texttt{(Hide-Input)} \qquad \frac{(\Gamma, P) \xrightarrow{a_A(v)} (\Gamma.a_A(v), P')}{P/\hat{\mathcal{O}}_\mathcal{X} \xrightarrow{\theta_A} P'/\hat{\mathcal{O}}_\mathcal{X}} \qquad if\ a_A(v) \notin \mathcal{X}.$$

$$\texttt{(Hide-Output)} \qquad \frac{(\Gamma, P) \xrightarrow{\overline{a}_A(e)} (\Gamma.\overline{a}_A(e), P')}{P/\hat{\mathcal{O}}_\mathcal{X} \xrightarrow{\theta_A} P'/\hat{\mathcal{O}}_\mathcal{X}}$$

$$\text{if} \quad \overline{a}_A(e) \notin \mathcal{X}, \ \text{ and } \ \exists x \in Vars(e), \text{ with } \ Ancestor(\Gamma, x) \cap S \neq \emptyset.$$

$$\texttt{(Jump)} \qquad \frac{(\Gamma, P) \xrightarrow{\overline{a}_A(e)} (\Gamma.\overline{a}_A(e), P')}{P/\hat{\mathcal{O}}_\mathcal{X} \xrightarrow{\epsilon} P'/\hat{\mathcal{O}}_\mathcal{X}}$$

$$\text{if} \quad \overline{a}_A(e) \notin \mathcal{X}, \ \text{ and } \ \forall x \in Vars(e), \ Ancestor(\Gamma, x) \cap S = \emptyset.$$

For example, let $G = a(x)|\bar{a}(1)|a(z).\bar{d}(2).\bar{d}(z)$; we have $G \xrightarrow{\theta_{\{z:=1\}}} P \xrightarrow{\bar{d}_{\{z:=1\}}(2)} P' \xrightarrow{\bar{d}_{\{z:=1\}}(z)} P''$, but since we only care about whether the message 1 is intercepted and sent out, we may *shortcut* the useless output transition $\xrightarrow{\bar{d}_{\{z:=1\}}(2)}$; we will thus have $P/\hat{\mathcal{O}}_{\{a,\bar{a}\},\{1\}} \xrightarrow{\theta_{\{z:=1\}}} P''/\hat{\mathcal{O}}_{\{a,\bar{a}\},\{1\}}$, where the θ comes from `Hide-Output`; there is no transition on $P/\hat{\mathcal{O}}_{\{a,\bar{a}\},\{1\}}$ under the action $\bar{d}_{\{z:=1\}}(2)$.

Definition 4 *With the notation of Definition* 3*, let $\mathcal{X}$ be a set of low-level actions of P, all assumed shared with I; and let S be a given set of value-constants in P. Denote by $\hat{\mathcal{O}}_\mathcal{X}$ the refined observation criterion $\hat{\mathcal{O}}_{\mathcal{X},S}$ on $P_I = (P \mid I) \setminus \mathcal{X}$. Then P is said to be* `confidentiality preserving` *for I (w.r.t. S), iff P_I satisfies the following (refined non-interference) property:*

$$(\mathbf{R\hat{N}I}_\mathcal{X}) \qquad P_I/\hat{\mathcal{O}}_\mathcal{X} \ \approx \ P_I \setminus \mathcal{X}^c.$$

Otherwise P is said present an `information leak` *for I.*

7.1 Confidentiality Analysis based on $\hat{\mathrm{RNI}}$

Given a VPPA P, our objective in this section is to propose an approach using *only* the syntax of our synchronous algebra, to find out if P presents an information leak for some intruder I.

A process J will be said to be *linear* if it has no choice operator '+', that is to say, as a term it is of the form $\alpha_1 \dots \alpha_n.0$, where each α_i is an input or output action or θ (with annotations as suffixes).

Definition 5 *Let $\mathcal{X}$ be the set of public actions of P, $\bar{d}$ an output channel not used by the actions in $\mathcal{X}$ and with variables disjoint from P, and let S denote a given set of value-constants appearing in P. An* `intrusion witness` *for P (w.r.t. $\mathcal{X}$ and S) is a linear process Q of the form $\alpha_1 \dots \alpha_n.0$, such that:*

i) For every $i = 1..(n-1)$, α_i is a θ suffixed with an annotation A_i;

ii) α_n is an output action of the form $\bar{d}_{A_n}(z)$ where z is a variable (and `not` *a constant);*

iii) For every $i = 2..n$, and every variable *u in A_i, the following condition holds:*

$$Ancestor_{P*Q}(\alpha_1 \dots \alpha_{i-1},\ u) \bigcap S \neq \emptyset.$$

A linear process $\beta_1 \dots \beta_n.0$ is said (by abuse of terminology) to be a *linear sub-process* of P iff: on the labeled tree representing P, there is a path along which all the actions $\beta_i, i = 1..n$, appear *in the increasing order of indices, but with some θ's possibly interspersed in between.*

We propose to show now that any information leak from a process P can be traced down to a leak from some linear sub-process. More precisely:

Theorem 3 *Let P be any process which presents an information leak for some process I (in the sense of the above definition). Then there is a linear process J and a linear sub-process P' of P, such that $P' * J$ is an intrusion witness for P.*

Conversely: if there is a linear sub-process P' of P and a linear process J such that $P' * J$ is an intrusion witness for P, then P presents an information leak for process J.

Proof: For proving the first assertion, we start with the assumption that $((P \mid I) \setminus \mathcal{X})/\hat{\mathcal{O}}_{\mathcal{X}}$ is *not* simulated by $(P \mid I) \setminus \mathcal{X} \setminus \mathcal{X}^c$, where $\mathcal{X}$ is the set of actions in P, all assumed public. Let us denote the former process by P_2 and the latter by P_1. Now (all our processes being assumed finite) both P_1 and P_2 can be represented as a finite labeled tree. The labels are all θ; but some of these labels on P_2 are attributed by $\hat{\mathcal{O}}_{\mathcal{X}}$ by hiding an output action on the 'private' channel $\bar{d}$, while the other θ-labels represent synchronization under '$*$'. Let us call an edge on the tree P_2 *special* iff its θ-label comes from $\hat{\mathcal{O}}_{\mathcal{X}}$ hiding a $\bar{d}$-action; a sub-tree of P_2 will be called special if it contains a special edge. Edges, or sub-trees, which are not special will be referred to as *non-special*.

Note first that for any son of the root node on P_2, the edge leading from the root node to this son cannot be special: otherwise it would mean that the criterion $\hat{\mathcal{O}}_{\mathcal{X}}$ has hidden a $\bar{d}$-action which outputs ancestorless data, instead of shortcutting across such an action as it ought to have, by definition. But we are assuming P_2 is not simulated by P_1, so P_2 must have special edges somewhere; so there is at least one special son of the root node of P_2.

Since P_2 is not simulated by P_1, there is at least one son at depth 1 on P_2 which is not simulated by any son of P_1 at depth 1. Choose one such son at depth 1 on P_2; then the sub-tree of P_2 corresponding to this son has to be special, let us denote it by $P_2^{(1)}$. The edge leading from the root node of P_2 to the root of $P_2^{(1)}$ is not special, so corresponds to synchronization (under '$*$') of two actions along two conjugate actions of P. Without loss of generality, we just consider the case when the conjugate actions are of the form $a_A(v)$ and $\bar{a}_B(v)$, the former being an input action on P (the other cases are treated similarly), and the θ-label of the chosen edge is indexed by an annotation $A \cup B$. We then set $A' = A \setminus \{x := v\}$, let $\alpha_1 = a_{A'}(x)$ and $\bar{\alpha}_1 = \bar{a}_B(v)$, and then set: $P' = \alpha_1.P'^{(1)}$ and $J = \bar{\alpha}_1.J^{(1)}$. This means that we have guessed non-deterministically the first actions for the processes P' and J that we are looking for.

The rest of the construction is done inductively: assume that after m such steps, we have gone down to a special subtree $P_2^{(m)}$ at depth m on P_2 such that: $P_2^{(m)}$ is not simulated by any sub-tree at depth m on P_1, and all the edges leading from the root of P_2 to the root of $P_2^{(m)}$ are non-special. We would also have at this stage: $P' = \alpha_1 \ldots \alpha_m.P'^{(m)}$ and $J = \bar{\alpha}_1 \ldots \bar{\alpha}_m.J^{(m)}$. The $(m+1)$-st step goes then as follows:

<u>*case i)*</u>: There is a son of $P_2^{(m)}$ such that the edge going from the root of $P_2^{(m)}$ to the root of this son is special. In this case, the θ-label of this edge comes from $\hat{\mathcal{O}}_{\mathcal{X}}$ hiding a $\bar{d}$-action, and this must necessarily be an output of data with ancestors (otherwise $\hat{\mathcal{O}}_{\mathcal{X}}$ would have shortcut). This output action is thus of the form $\bar{d}_A(v)$, where v must be a variable: otherwise v will be ancestorless by definition, and the action would be shortcut by $\hat{\mathcal{O}}_{\mathcal{X}}$. Write then $J^{(m)} = \bar{d}_A(v).0$ and $P'^{(m)} = \theta.0$; return P' and J completed respectively with these definitions of $P'^{(m)}$ and $J^{(m)}$, and stop.

<u>*case ii)*</u>: No edge on $P_2^{(m)}$ leading to any of its sons is special. In this case, proceed exactly as in step 1 to guess the first action to do for $P'^{(m)}$ and $J^{(m)}$, and proceed to to step $(m+2)$.

This proves the first assertion of the theorem. The converse assertion is proved via straightforward reasoning, based directly on the definition of information leak. $\square$

<u>**Remark 5**</u>. Following the refined definition of robust non-interference ($\mathbf{R\hat{N}I}_{\mathcal{X}}$), the process $a(x).0 \mid \bar{a}(1).0$ will *not* present an information leak for the intruder $I_5 = a(z).\bar{d}(1)$. This is so because the value transmitted on the channel $\bar{d}$ is a constant which will have no ancestor, and $\hat{\mathcal{O}}_{\mathcal{X}}$ would have cut out the corresponding branch.

7.2 An NP-Algorithm for Deciding Confidentiality Attacks

The approach described above is in particular applicable to the analysis of confidentiality of cryptographic protocols modeled as VPPA's. We assume given here encryption and decryption functions over the value-domain, and that all the input or output values are in normal form w.r.t. a 'Dolev-Yao rewrite system'; for such purposes, we shall keep to the notation and conventions made in Remark 2.i). Suppose that a process P presents an information leak for an intruder I; then the leak may be called *admissible* if for some witness $J = \alpha_1 \ldots \alpha_{n-1}.\bar{d}(z)$, e.g. as constructed in the above proof for this leak, the final output action detecting the violation of $\mathbf{R\hat{N}I}_{\mathcal{X}}$, transmits a value which appears *as clear text* on the right hand side of an assignment for an input variable of P in the annotations of the α_i's. The term "*attack*" is generally used for an information leak which is *not admissible* in this sense. Such a notion of attack is defined below in formal terms. For any value-term u, we shall denote by $u\!\downarrow$ the normal form of u under the Dolev-Yao rewrite system of Section 4, Remark 2.i); if z is any value-variable bound to a value-term v, we also agree to denote by $z\!\downarrow$ the Dolev-Yao normal form $v\!\downarrow$ of v.

Definition 6 *Let $\mathcal{X}$ be the set of all public actions of P, and S a given set of 'secret value-constants', and suppose P presents an information leak for some I w.r.t. the observation criterion $\hat{\mathcal{O}}_{\mathcal{X},S}$. Then the leak is called an* `attack on` P `by` I *iff for some intrusion witness Q for this leak, of the form $Q = \alpha_1 \ldots . \alpha_{n-1}.\bar{d}_{A_n}(z)$, where each $\alpha_i, i = 1..(n-1)$, is a θ annotated with a set A_i, the following holds:*

- $z\!\downarrow \in S$, `and`
- $A_1 \cup \ldots \cup A_{n-1}$ *contains* `no` *assignment of the form* $x := v, z\!\downarrow, v'$ *for any* $x \in Parent_{A_1,\ldots A_n}(z) \cap Vars(P)$.

Algorithm for Attack Detection:

We assume given a VPPA P of finite height h, and a finite set S of value-terms (those intended to remain 'secret'), with respect to which possible 'attacks' are to be detected. The algorithm given below is based on the proof of Theorem 4 ("Any confidentiality attack on a VPPA P, is an attack on a suitable linear sub-process of P").

$VARS$ = Set of value-variables appearing in P; `Chan` = Channel set of P;

N = Bound for the size of data transmitted along the channels of P;

T = Path traversed so far (for attack guessing) along P;

$Assn$ = Data (assignments) accumulated along T;

J = Linear process guessed so far for the attacker ;

$Data_N(J)$ = set of expressions of length $\leq N$,
involving only variables and constants appearing in J.

The attack detection algorithm is the instance of that given below, with the instantiations $T = 0 = J$, and $Assn = \emptyset$. (For any path Γ along P, we shall be denoting the sub-process of P 'at the end of' Γ also as Γ, since no confusion is likely).

Guess_Attack$(T, J, Assn) = \{$
  Guess $e \in Data_N(J)$;
  `If` $e\downarrow \in S$ and $\{x := v, e\downarrow, v'\} \notin Assn$, where $x \in VARS \cap Parent_{Assn}(e)$
    `then` { choose channels $d, \bar{d} \notin$ `Chan`, and a fresh variable z;
        `RETURN` $J.\bar{d}_{\{z:=e\}}(z)$; `EXIT`; } /* Found an Attack !! */
    `else` {
      If `EAct`$(T) = \emptyset$ then RETURN(); fi; /* No Attack found along T */
      Guess a $T \xrightarrow{\alpha} T'$:
        Case i) $\alpha = a_A(e')$, for some data-expression $e' \in Data_N(J)$:
            set $B = \emptyset$; let $\bar{\alpha} = \bar{a}_B(e')$;
        Case ii) $\alpha = \bar{a}_A(e'')$: choose a new variable $y \notin Vars(P) \cup Vars(J)$;
            set $B = \{y := e''\}$; let $\bar{\alpha} = a_B(y)$;
        Case iii) $\alpha = \theta_A$: set $B = \emptyset$; let $\bar{\alpha} = \theta_B$;
      Set $J' = J.\bar{\alpha}$, $Assn' = Assn \cup A \cup B$;
      `RETURN` (*Guess_Attack*$(T', J', Assn')$); }
}

This algorithm is *a priori* of complexity DEXPTIME w.r.t. $(h + N)$, indeed the total number of ways of guessing the paths in P is $O(exp(h))$ and the *number of all possible guesses* for $\alpha = a_A(e')$ in Case i) is $O(exp(N))$. These (absolute) exponential bounds w.r.t. h, N can however be 'lowered' to non-deterministic polynomial ones, by proceeding as follows:
- in Case i) limit the guess of e' to those which are "rewrite reachable" from $Data_N(J)$, in a suitable sense;
- manipulate processes as *dags* instead of as terms, as in [23].

Under such an adapted data structure, the algorithm is of complexity NP w.r.t. $h + N$. □

7.3 An Example of Confidentiality Analysis

The following known attack on the Needham-Schroeder public key authentication protocol, illustrates well the notion of intrusion witness (Definition 5) and also the above algorithm which is based on the reasoning behind the proof of Theorem 4.

Example 3. The Needham-Schroeder public key authentication protocol is specified as follows. (Principals are denoted by uppercase letters, the nonces they send have corresponding lowercase subscripts, the messages sent are encrypted with the public key of the addressee.) A, B exchange fresh nonces 'for the current session' in an encrypted form, that only the addressee is supposed to be able to decrypt with his private key.

$$\begin{array}{l} 1.\ A \to B : \{N_a, A\}_{K_b} \\ 2.\ B \to A : \{N_a, N_b\}_{K_a} \\ 3.\ A \to B : \{N_b\}_{K_b} \end{array}$$

Here is an attack by a third principal I who manages to intercept in clear text (modulo Dolev-Yao) the nonce N_b addressed by B to A:

$$\begin{array}{lll} 1.1. & A \to I & : \{N_a, A\}_{K_i} \\ 2.1. & I(A) \to B & : \{N_a, A\}_{K_b} \\ 2.2. & B \to I(A) & : \{N_a, N_b\}_{K_a} \\ 1.2. & I \to A & : \{N_a, N_b\}_{K_a} \\ 1.3. & A \to I & : \{N_b\}_{K_i} \\ 2.3. & I(A) \to B & : \{N_b\}_{K_b} \end{array}$$

The attack involves two concurrent runs of the protocol: in run 1 (whose i-th step is marked $1.i$), A establishes a valid session with I; in run 2 (steps marked $2.i$), I impersonates A to talk to B. The attack is actually 'effective' at step 1.3, where I captures in clear text (modulo Dolev-Yao) the nonce N_b that B has intended for A, so sent encrypted with the key K_a. All these can be formulated as VPPA's as follows, where c is the communication channel between A and B, c_1 is the channel between A and I, and c_2 is that between B and I; these channels are public by assumption:

$$\begin{array}{ll} A :: & \mathtt{let}\ v_1 = \mathtt{enc}(\{N_a, A\}, K_i)\ \mathtt{in}\ \bar{c}_1(v_1).c_1(x). \\ & \quad \mathtt{let}\ v_3 = \mathtt{enc}(\pi_2(\mathtt{dec}(x, K_a)), K_i)\ \mathtt{in}\ \bar{c}_1(v_3).0 \\ B :: & c(y).\ \mathtt{let}\ v_2 = \mathtt{enc}(\{\pi_1(\mathtt{dec}(y, K_b)), N_b\}, K_a)\ \mathtt{in}\ \bar{c}(v_2).c(z).0 \\ I\ :: & c_1(y').\ \mathtt{let}\ v_1' = \mathtt{enc}(\mathtt{dec}(y', K_i), K_b)\ \mathtt{in}\ \bar{c}(v_1').c(x').\bar{c}_1(x').c_1(z'). \\ & \quad \mathtt{let}\ v_3' = \mathtt{enc}(\mathtt{dec}(z', K_i), K_b)\ \mathtt{in}\ \bar{c}(v_3').\ \mathtt{let}\ u = \mathtt{dec}(z', K_i)\ \mathtt{in}\ \bar{d}(u).0 \end{array}$$

The output action $\bar{d}(u)$ at the end of process I is to represent the behavior that I sends out – on *his* private channel $\bar{d}$ – some (private) information between A and B that he has intercepted. Set $\mathcal{X} = \{c, \bar{c}, c_1, \bar{c}_1, c_2, \bar{c}_2, c_3, \bar{c}_3\}$ $S = \{N_a, N_b\}$, and let $P_I = (A \mid B \mid I) \setminus \mathcal{X}$; then it is easily checked that: $P_I \setminus \mathcal{X}^c \not\approx P_I / \hat{\mathcal{O}}_{\mathcal{X}}$, where $\hat{\mathcal{O}}_{\mathcal{X}}$ is the observation criterion defined as $\hat{\mathcal{O}}_{\mathcal{X},S}(P_I)$.

The information leak here is from the processes A, B in parallel composition with the 'regular process' I, towards the impersonating part of I. It can be visualized as following intrusion witness $Q = \theta_{A_1}.\theta_{A_2}.\theta_{A_3}.\theta_{A_4}.\theta_{A_5}.(\mathtt{let}\ u = \mathtt{dec}(z', K_i)\ \mathtt{in}\ \bar{d}(u)).0$

$$
\begin{array}{lll}
A \to I: & \bar{c}_1(\mathtt{enc}(\{N_a, A\}, K_i)) * c_1(y') & \\
 & \Rightarrow \qquad \theta_{A_1}, & \textit{where } A_1 = \{y' := \mathtt{enc}(\{N_a, A\}, K_i)\} \\
I(A) \to B: & \bar{c}(\mathtt{enc}(\mathtt{dec}(y', K_i), K_b)) * c(y) & \\
 & \Rightarrow \qquad \theta_{A_2}, & \textit{where } A_2 = \{y := \mathtt{enc}(\mathtt{dec}(y', K_i), K_b)\} \\
B \to I(A): & \bar{c}(\mathtt{enc}(\{\pi_1(\mathtt{dec}(y, K_b)), N_b\}, K_a)) * c(x') & \\
 & \Rightarrow \qquad \theta_{A_3}, & \textit{where } A_3 = \{x' := \mathtt{enc}(\{\pi_1(\mathtt{dec}(y, K_b)), N_b\}, K_a)\} \\
I \to A: & \bar{c}_1(x') * c_1(x) & \\
 & \Rightarrow \qquad \theta_{A_4}, & \textit{where } A_4 = \{x := x'\} \\
A \to I: & \bar{c}_1(\mathtt{enc}(\pi_2(\mathtt{dec}(x, K_a)), K_i)) * c_1(z') & \\
 & \Rightarrow \qquad \theta_{A_5}, & \textit{where } A_5 = \{z' := \mathtt{enc}(\pi_2(\mathtt{dec}(x, K_a)), K_i)\}
\end{array}
$$

The annotated action θ_{A_2} is obtained under synchronization on channel c: the impersonation of A by I is modeled here by the fact that I communicates with B on the channel c (the channel between A and B); θ_{A_3} is also obtained under synchronization on c, but when I intercepts a message sent by B to A. Following the reasoning of the proof of Theorem 4, the intrusion witness $Q = \theta_{A_1}.\theta_{A_2}.\theta_{A_3}.\theta_{A_4}.\theta_{A_5}.(\mathtt{let}\ u = \mathtt{dec}(z', K_i)\ \mathtt{in}\ \bar{d}(u)).0$ can be obtained as $P' * J$, where:

$$
\begin{array}{rl}
 & P' = \bar{c}_1(v_1).c(y).\bar{c}(v_2).c_1(x).\bar{c}_1(v_3).\theta.0, \\
\text{and} & J\ = c_1(y').\bar{c}(v_1').c(x').\bar{c}_1(x').c_1(z').\bar{d}_{\{u:=\mathtt{dec}(z',K_i)\}}(u).0.
\end{array}
$$

where the values assigned to the variables are respectively those given by the expressions "`let ..=.. in`" in the above definitions of A, B, I as VPPA's. □

8 A Timed Synchronous Algebra

In this section, we propose to extend our synchronous process algebra by introducing notions such as '*idling*' and '*timeout*', comparable to those developed in [22, 12, 11]; it will appear that such an extension is easier, and much more natural in our formalism than in the works just cited. The principal reasons for this seem be the following: (i) in each of these works 'the unit time counting' action is introduced as an addition, whereas we have a natural candidate for that purpose inside our algebra: namely θ; (ii) additional 'time determinacy axioms' are needed in all the cited approaches; but in our setup the synchronization operator '$*$' seems to capture the essence of this notion; (iii) the definitions we propose for the notion of the 'idling' copy of any process and for 'timeout', are algebraic.

For the action intended to reflect unit time passing, which was labeled 'tick' in [22, 11] (and σ in [12]), we shall use our θ without any (or the empty) annotation, meaning: 'no data flow'; in other words *we define 'tick' to be a synonym for θ*. For introducing properly the notion of 'idling', we need recursion, and for that we resort to the possibility of defining any process – as in CCS – via an operator denoted '$\stackrel{def}{=}$', with the following operational semantics. The operational rule for '$\stackrel{def}{=}$' is:

$$
\frac{A\{\overline{v}/\overline{x}\} \xrightarrow{\alpha} P'}{P(\overline{v}) \xrightarrow{\alpha} P'} \quad if\ P(\overline{x}) \stackrel{def}{=} A
$$

The idling copy of any process P, denoted as $i(P)$, is then defined as follows: $i(P) \stackrel{def}{=} P + tick.i(P)$. To understand its semantics more intuitively: $i(P)$ can be viewed or interpreted as the infinite sum: $i(P) \;=\; P + tick.(P + tick.(P + tick.(P + tick.(\ldots))))$. It is easy to see that for any E, we have: $E \;*\; i(0) \;\approx\; E$.

For modeling '*timeout*', we define – for any two processes P, Q and any non-negative integer n – a process $i(P, Q, n)$, as follows:

$$i(P, Q, 0) \stackrel{def}{=\!=} P + Q$$
$$i(P, Q, n) \stackrel{def}{=\!=} P + tick.i(P, Q, n-1)$$

Semantics: $i(P, Q, n)$ is the process for which *all* the actions of P remain available up to n units of time, i.e., n is the bound of timeout for process P; and if P doesn't act within the time bound n, the system will behave as Q. Note that the idling copy $i(P)$ of any P that we defined above, can also be seen as $i(P, P, \omega)$, with the natural convention that $\omega - n = \omega$.

We propose now to illustrate the expressive power of the timed extension that we have thus obtained, by modeling in our calculus two typical examples borrowed from [12] and [11]. (Both used notions of 'timed process', 'tick', and 'idling', all defined very differently but operationally similar to ours.)

8.1 The Security Costs Protocol (SCP)

The "Security Costs Protocol (SCP)", developed and analyzed in [12], models the transmission of a message through two different media, one assumed reliable, while the other is not. The sender initially sends the message across the unreliable medium; he sends it again via the reliable medium iff an acknowledgement from the receiver has not arrived before a given timeout. The acknowledgement (by the receiver) is always sent via the reliable medium.

Recall that we have defined $\theta_{\{\}} = tick$; in what follows, we will in general use $tick$ in the specifications, but only θ in the calculations. In our model of SCP given below, we have chosen as timeout the bound of 3 units of time, based on the assumption that each of the actions: sending a message, or receiving the acknowledgement, takes 1 unit of time. This timeout can obviously be parametrized by other considerations.

The Sender A:

$$A(v) \stackrel{def}{=\!=} \overline{mess^{ur}_{ad}}(v).i(ack_{ca}.A(v), \overline{mess^{r}_{ac}}(v).tick^2.ack_{ca}.A(v), 3)$$

The Reliable Medium C:

$$C \stackrel{def}{=\!=} i(mess^{r}_{ac}(x).\overline{mess^{r}_{cb}}(x).C + ack_{bc}.\overline{ack_{ca}}.C)$$

The Unreliable Medium D:

$$D \stackrel{def}{=\!=} i(mess^{ur}_{ad}(y).(D + \overline{mess^{ur}_{db}}(y).D))$$

The Receiver B:

$$B \stackrel{def}{=\!=} i(mess^{ur}_{db}(z).\overline{ack_{bc}}.B + mess^{r}_{cb}(z).\overline{ack_{bc}}.B)$$

If S is the set of all the non-θ actions, the process we are interested in is the following VPPA:

$$Sys \stackrel{def}{=\!=} (((A(v) * D) * C) * B) \setminus S$$

The restriction under S here reflects the fact that we are only looking at the branches where the message/acknowledgement gets transmitted. (The parentheses reflect the fact that A first sends the message via D etc.; but, as we shall be seeing shortly, they have no real influence on the calculations: in other words, because of the restriction under S, '$*$' will have an associative-commutative behavior in this example. See remark (6) of Section 2.1).

For any $k \in 0..3$, let us define:

$$A_k(v) = i(ack_{ca}.A(v), \overline{mess^{r}_{ac}}(v).\theta^2.ack_{ca}.A(v), k)$$

Then we have the following, where equality '=' is up to bisimulation '$\approx$':

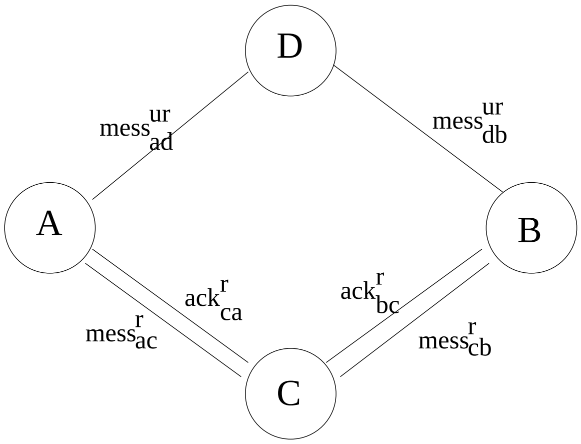

Fig. 1. The Simple Cost Protocol

$$Sys \;=\; \theta_{\{y:=v\}}.(((A_3(v) * (D + \overline{mess^{ur}_{db}}(v).D)) * C) * B) \setminus S \;=\; \theta_{\{y:=v\}}.(X+Y),$$

where:

$$\begin{aligned}
X \;&=\; (((A_3(v) * D) * C) * B) \setminus S \;=\; \theta.(((A_2(v) * D) * C) * B) \setminus S \\
&=\; \theta^2.(((A_1(v) * D) * C) * B) \setminus S \;=\; \theta^3.(((A_0(v) * D) * C) * B) \setminus S \\
&=\; \theta^3.\theta_{\{x:=v\}}.((((\theta^2.ack_{ca}.A(v)) * D) * (\overline{mess^r_{cb}}(x).C)) * B) \setminus S \\
&=\; \theta^3.\theta_{\{x:=v\}}.\theta_{\{z:=x\}}.((((\theta.ack_{ca}.A(v)) * D) * C) * (\overline{ack_{bc}}.B)) \setminus S \\
&=\; \theta^3.\theta_{\{x:=v\}}.\theta_{\{z:=x\}}.\theta.((((ack_{ca}.A(v)) * D) * (\overline{ack_{ca}}.C)) * B) \setminus S \\
&=\; \theta^3.\theta_{\{x:=v\}}.\theta_{\{z:=x\}}.\theta^2.(((A(v) * D) * C) * B) \setminus S \\
&=\; \theta^3.\theta_{\{x:=v\}}.\theta_{\{z:=x\}}.\theta^2.Sys;
\end{aligned}$$

and

$$\begin{aligned}
Y \;&=\; (((A_3(v) * (\overline{mess^{ur}_{db}}(y).D)) * C) * B) \setminus S \\
&=\; \theta_{\{z:=y\}}.(((A_2(v) * D) * C) * (\overline{ack_{bc}}.B)) \setminus S \\
&=\; \theta_{\{z:=y\}}.\theta.(((A_1(v) * D) * (\overline{ack_{ca}}.C)) * B) \setminus S \\
&=\; \theta_{\{z:=y\}}.\theta^2.(((A(v) * D) * C) * B) \setminus S \\
&=\; \theta_{\{z:=y\}}.\theta^2.Sys.
\end{aligned}$$

So we finally get:

$$Sys = \theta_{\{y:=v\}}.(\theta^3.\theta_{\{x:=v\}}.\theta_{\{z:=x\}}.\theta^2.Sys + \theta_{\{z:=y\}}.\theta^2.Sys)$$

In the above equations, there are two branches after A has transmitted the message v to D (represented by $\theta_{\{y:=v\}}$): the left branch, denoted by X, means that after 3 units of time, the sender reaches the timeout bound, so retransmits the message via the reliable medium C (represented by: $\theta_{\{x:=v\}}.\theta_{\{z:=x\}}$), and the right branch, Y, models the successful transmission via the unreliable medium D.

A few comments on how the calculations go, for instance on the branch X. For each $i \geq 1$, $(((A_i(v) * D) * C) * B) \setminus S$ can only do θ and then evolve to $(((A_{i-1}(v) * D) * C) * B) \setminus S$: this is so, because for each action $\alpha (\neq \theta)$ starting from $A_i(v), B, C$, or D, there is no action $\overline{\alpha}$ in the system that can synchronize with it; therefore the only possible action is θ (i.e., the system has to 'idle'). We thus reach $(((A_0(v) * D) * C) * B) \setminus S$; but now we have:

$$A_0(v) \xrightarrow{\overline{mess^r_{ac}}(v)} (\theta^2.ack_{ca}.A(v)), \quad \text{and} \quad C \xrightarrow{mess^r_{ac\{x:=v\}}(v)} (\overline{mess^r}_{cb\{x:=v\}}(x).C),$$

while $D \xrightarrow{\theta} D$ and $B \xrightarrow{\theta} B$; therefore the the only action that $(((A_0(v) * D) * C) * B) \setminus S$ can do, reduces to:

$$\begin{array}{rcl}((\overline{mess^r_{ac}}(v) * \theta) * mess^r_{ac\{x:=v\}}(v)) * \theta & = & (\overline{mess^r_{ac}}(v) * mess^r_{ac\{x:=v\}}(v)) * \theta \\ & = & \theta_{\{x:=v\}} * \theta \\ & = & \theta_{\{x:=v\}}\end{array}$$

and we get:

$$(((A_0(v) * D) * C) * B) \setminus S \;=\; \theta_{\{x:=v\}}.((((\theta^2.ack_{ca}.A(v)) * D) * (\overline{mess^r}_{cb\{x:=v\}}(x).C)) * B) \setminus S$$

Following the same lines of reasoning, we finally derive the equations above for X and Y. $\square$

8.2 An Object Manager

In the approach [11], the security analysis of any process P with high and low level actions denoted respectively H and L, is based on verifying the following property:

(BNDC): For all Π using actions in H, one has: $(P \mid \Pi) \setminus H \;\approx\; P \setminus H$,

where '$\approx$' is taken as weak bisimulation equivalence.

The example of *Object Manager*, as developed in [11], is a process that has to manage, following a mutual exclusion policy, the access to a shared variable. It implements the *no-write-down/no-read-up* policy specified as follows: High level users can only read the variable, Low level users can only write it; meaning that the information written should only flow from low level users to high level users. The 'preliminary' model as suggested in [11], is the following untimed process:

$$C \;\stackrel{def}{=}\; req_H.\overline{read_H}.C + req_L.write_L.C$$

with the set of high level actions defined as: $H = \{req_H, \overline{req_H}, read_H, \overline{read_H}\}$, and the set of low level actions as: $L = \{req_L, \overline{req_L}, read_L, \overline{read_L}\}$; and it was observed that this untimed model does not satisfy the BNDC property since $\Pi = \overline{req_H}.0$ can block the monitor process, as is e.g. revealed by (weak) bisimulation equivalence.

Later on, the authors modified the specification by adding 'timed' actions; the objective was to eliminate the blocking of the monitor with the help of a 'timeout' mechanism. The timed model they proposed was not only very complex but also created a problem of liveness: one of the branches of the model was the loop $C \xrightarrow{\tau}\xrightarrow{tick}\xrightarrow{tick} C$, along which the system can go into a livelock: an infinite *tick* sequence with no other action.

Our timed model is very different; it uses our 'idling' constructor $i(-,-,-)$:

$$C \;\stackrel{def}{=}\; req_H.i(\overline{read_H}.C, C, 1) + req_L.write_L.C$$

With this definition of C and $\Pi = \overline{req_H}.0$, let $P_1 = (C \mid \Pi) \setminus H$ and $P_2 = C \setminus H$; it is then easy to see that:

$$P_1 \;=\; \theta.\theta.C' + req_L.write_L.P_1, \;\text{ where }\; C' \;=\; req_L.write_L.C'$$

while $P_2 \;=\; req_L.write_L.P_2$. So we get, *in our model*, that P_1 is *weakly bisimilar* to P_2 (which was not the case in the setup proposed in [11]). Besides, it must also be emphasized that our timed model has no problem of liveness, since it contains no infinite θ loops.

Remark 6. Here are a few observations of comparison between the timed algebra proposed in [11], [12] or [22], with the one that we propose in this paper. In the latter two works, the following is one of the axioms for 'time determinacy': $tick.(X + Y) \approx tick.X + tick.Y$. We do *not* have such an axiom here, nor is it a consequence of our approach, although it can be added on to our algebra if needed. In the timed algebra proposed by [11], the property mentioned is first given as an axiom, but at a later stage the authors propose to drop it because of the problems encountered in applying their algebra to secrecy analysis; they also point out that some of these problems arise due to the 'negative' premises in some of their time determinacy axioms. Such problems seem absent in our timed algebra; it seems then that our algebra could be a better basis for the approach of 'partial evaluation' (or 'partial model checking') proposed in [11] as a tool for secrecy analysis. This is our concern below.

8.3 The Timed Interleaving Semantics via '$*$'

Recall that for any two processes P, Q the process $P \mid Q$ giving the interleaving (CCS) semantics of P acting concurrently with Q was defined in Section 2.3, uniquely up to bisimulation, for example by:

$$P \mid Q \;\; = \;\; \sum_{a_i} a_i.(P_i \mid Q) \; + \; \sum_{b_j} b_j.(P \mid Q_j) \; + \sum_{a_i \neq \theta \neq b_j} (a_i * b_j).(P_i \mid Q_j)$$

The restriction in the last summand of a_i, b_j to the 'standard' actions of P and Q was only needed for getting the CCS semantics. For timed processes, it is only natural that we drop such a restriction, and define the 'timed asynchronous composition' $P \mid Q$ as:

$$P \mid Q \;\; = \;\; \sum_{a_i} a_i.(P_i \mid Q) \; + \; \sum_{b_j} b_j.(P \mid Q_j) \; + \sum_{a_i, b_j} (a_i * b_j).(P_i \mid Q_j)$$

where the a_i, b_j run over $\mathtt{EAct}(P), \mathtt{EAct}(Q)$ respectively. This $P \mid Q$ obviously has some additional branches, since the last summand is taken over all actions of P and Q, including θ.

This asynchronous composition process can also be defined via induction on '$*$' as follows. We begin by adding a special action Θ to $\mathtt{EAct}(P)$ for any P, with the following properties:

$$\forall\, \alpha \in \mathtt{EAct}(P),\, \alpha \neq \Theta : \quad \Theta * \alpha = \alpha; \quad \text{and} \quad \Theta * \Theta = 0 \; (= \mathtt{undef})$$

i.e. to say, the added Θ cannot synchronize with itself, but it can with any action $\alpha \in \mathtt{EAct}(P)$. For any positive integer $k \geq 0$, we then define a process:

$$\hat{E}_k \;\; = \;\; \Theta.\hat{E}_{k-1} \; + \sum_{E \xrightarrow{\alpha} E'} \alpha.\hat{E}'_k$$

where we set $\hat{E}_0 = E$ by definition, and the sum to the right is taken over all the actions $\alpha \in EAct(E), \alpha \neq \Theta$.

For any finite process E we also define its depth $|E|$ as the depth of its additive $PACUID$-normal form; $|E|$ is determined uniquely up to bisimulation by induction, with the following formulas:

$$|0| = 0, |\alpha.E| = |E| + 1, |E + F| = \mathtt{max}\{|E|, |F|\}.$$

We then have the following result:

Theorem 4 *For all finite processes P and Q, we have: $P \mid Q \;\approx\; \hat{P}_{|Q|} * \hat{Q}_{|P|}$,*

Proof: Let us define a symmetric relation S on processes, as:

$$S = \{(P \mid Q,\;\; \hat{P}_m * \hat{Q}_n);\; n \geq |P| \textit{ and } m \geq |Q|, \text{ for all processes } P, Q\}.$$

In particular: $(P|Q,\; \hat{P}_{|Q|} * \hat{Q}_{|P|}) \in S$ for all P, Q. We then show that S is a strong bisimulation.

Step I: First we show, by case analysis, that whenever $P \mid Q \xrightarrow{\alpha} R_1$, then there is a $\hat{P}_m * \hat{Q}_n \xrightarrow{\alpha} R_2$ such that $(R_1, R_2) \in S$.

1. $P \mid Q \xrightarrow{\alpha} P' \mid Q$ is generated by $P \xrightarrow{\alpha} P'$. So $\hat{P}_m \xrightarrow{\alpha} \hat{P}'_m$, and by $\hat{Q}_n \xrightarrow{\Theta} \hat{Q}_{n-1}$, therefore $\hat{P}_m * \hat{Q}_n \xrightarrow{\alpha} \hat{P}'_m * \hat{Q}_{n-1}$. Since P is assumed finite and $P \xrightarrow{\alpha} P'$, we have $|P'| \leq |P| - 1$ and $n - 1 \geq |P| - 1 \geq |P'|$, thus $(P'|Q,\ \hat{P}'_m * \hat{Q}_{n-1}) \in S$.

   The case $P \mid Q \xrightarrow{\alpha} P \mid Q'$ generated by $Q \xrightarrow{\alpha} Q'$ is similar, since $\mid$ and $*$ are all symmetric.

2. $P \mid Q \xrightarrow{\gamma} P' \mid Q'$ is generated by $P \xrightarrow{\alpha} P'$ and $Q \xrightarrow{\beta} Q'$ where $\alpha * \beta = \gamma$. We have $\hat{P}_m \xrightarrow{\alpha} \hat{P}'_m$ and $\hat{Q}_n \xrightarrow{\beta} \hat{Q}'_n$, therefore $\hat{P}_m * \hat{Q}_n \xrightarrow{\gamma} \hat{P}'_m * \hat{Q}'_n$. Thus by $m \geq |Q| > |Q'|$ and $n \geq |P| > |P'|$, $(P'|Q',\ \hat{P}'_m * \hat{Q}'_n) \in S$.

Step II: In the opposite direction, we need to show that if $\hat{P}_m * \hat{Q}_n \xrightarrow{\alpha} R_2$, then we also have $P \mid Q \xrightarrow{\alpha} R_1$ with $(R_1, R_2) \in S$. We go again by case analysis:

1. $\hat{P}_m * \hat{Q}_n \xrightarrow{\alpha} \hat{P}'_m * \hat{Q}_{n-1}$ is generated by $\hat{P}_m \xrightarrow{\alpha} \hat{P}'_m$ and $\hat{Q}_n \xrightarrow{\Theta} \hat{Q}_{n-1}$. Then by the structure of $\hat{P}_m$, we have $P \xrightarrow{\alpha} P'$ and so $P \mid Q \xrightarrow{\alpha} P' \mid Q$, with $(P'|Q,\ \hat{P}'_m * \hat{Q}_{n-1}) \in S$, since $n - 1 \geq |P| - 1 \geq |P'|$.

   The case $\hat{P}_m * \hat{Q}_n \xrightarrow{\alpha} \hat{P}_{m-1} * \hat{Q}'_n$, generated by $\hat{Q}_n \xrightarrow{\alpha} \hat{Q}'_n$ and $\hat{P}_m \xrightarrow{\Theta} \hat{P}_{m-1}$, is symmetric.

2. $\hat{P}_m * \hat{Q}_n \xrightarrow{\gamma} \hat{P}'_m * \hat{Q}'_n$ is generated by $\hat{P}_m \xrightarrow{\alpha} \hat{P}'_m$ and $\hat{Q}_n \xrightarrow{\beta} \hat{Q}'_n$ where $\alpha * \beta = \gamma$. Then $P \xrightarrow{\alpha} P'$ and $Q \xrightarrow{\beta} Q'$, so $P \mid Q \xrightarrow{\gamma} P' \mid Q'$. Since $m \geq |Q| > |Q'|$ and $n \geq |P| > |P'|$, we have $(P'|Q',\ \hat{P}'_m * \hat{Q}'_n) \in S$. □

Remark 7. For finite processes P, Q, the 'untimed' asynchronous composition $P \mid Q$ with CCS semantics can also be recovered via such an inductive reasoning on '$*$'. For this it suffices to stipulate that: $\theta * \theta = 0$. The construction of the processes $\hat{E}_k$ for all $k \geq 0$, as well as the lines of reasoning the above theorem, then hold verbatim.

8.4 Partial Evaluation Techniques

As in [19, 24], we consider modal formulas as given by the following BNF grammar:

$$\phi ::= \ \texttt{tt} \mid \texttt{ff} \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle \alpha \rangle \phi \mid [\alpha]\phi$$

Following usual convention, every process (tree) E is seen as a labeled state machine. We then use the standard notation $E \models \phi$ for model checking, meaning: iff the property ϕ is true (or satisfied) at (the root of the state machine) E. In the above, `tt` and `ff` are two logical constants, while $\langle\,\rangle$ and $[\,]$ are two modal operators. For example, $\langle \alpha \rangle \phi$ is true at E iff from E the system can make an α-transition to an E' such that ϕ is true at E'. While $[\alpha]\phi$ is true at E iff ϕ is true at E' for every $E \xrightarrow{\alpha} E'$.

The formal modal verification semantics is defined as follows:

$$\begin{array}{l}
E \models \texttt{tt} \\
E \not\models \texttt{ff} \\
E \models \phi \wedge \psi \ \texttt{iff}\ E \models \phi \ \textit{and}\ E \models \psi \\
E \models \phi \vee \psi \ \texttt{iff}\ E \models \phi \ \textit{or}\ E \models \psi \\
E \models \langle \alpha \rangle \phi \ \ \texttt{iff}\ \exists E'.\ E \xrightarrow{\alpha} E' \ \textit{with}\ E' \models \phi \\
E \models [\alpha]\phi \ \ \texttt{iff}\ \forall E'.\ E \xrightarrow{\alpha} E' \ \textit{with}\ E' \models \phi
\end{array}$$

We next define partial evaluation $\|\cdot\|$ for our synchronization operator '$*$' as follows:

$$
\begin{aligned}
&\|\mathtt{tt}\|_E &&= \mathtt{tt}\\
&\|\mathtt{ff}\|_E &&= \mathtt{ff}\\
&\|\phi \vee \psi\|_E &&= \|\phi\|_E \vee \|\psi\|_E\\
&\|\phi \wedge \psi\|_E &&= \|\phi\|_E \wedge \|\psi\|_E\\[1ex]
&\|\langle\gamma\rangle\phi\|_E &&= \bigvee_{E \xrightarrow{\alpha} E'} \langle\beta\rangle\|\phi\|_{E'} \textit{ where } \alpha * \beta = \gamma\\[1ex]
&\|[\gamma]\phi\|_E &&= \bigwedge_{E \xrightarrow{\alpha} E'} [\beta]\|\phi\|_{E'} \textit{ where } \alpha * \beta = \gamma
\end{aligned}
$$

We then have the following result, which will be the basis for our method of using partial model checking techniques, on processes of finite depth:

Theorem 5 $E * F \models \phi$ `if and only if` $F \models \|\phi\|_E$.

Proof: We proceed by induction on the size $|\phi|$ of the formula ϕ:

1. $\phi = \mathtt{tt}$ or $\mathtt{ff}$. It is trivial. Since both $E * F$ and F satisfy $\mathtt{tt}$, and both do not satisfy $\mathtt{ff}$.
2. $\phi = \langle\gamma\rangle\phi'$. Then $\|\phi\|_E = \bigvee_{E \xrightarrow{\alpha} E'} \langle\beta\rangle\|\phi'\|_{E'}$ where $\alpha * \beta = \gamma$.

   If $E * F \models \phi$, then there exists $E * F \xrightarrow{\gamma} E' * F'$, with $E \xrightarrow{\alpha} E'$, $F \xrightarrow{\beta} F'$, $\alpha * \beta = \gamma$ and $E' * F' \models \phi'$. By induction $F' \models \|\phi'\|_{E'}$, therefore $F \models \langle\beta\rangle\|\phi'\|_{E'}$ and so $F \models \|\phi\|_E$.

   If $F \models \|\phi\|_E$, then $\exists\alpha, \beta.\ E \xrightarrow{\alpha} E'$ and $F \models \langle\beta\rangle\|\phi'\|_{E'}$ with $\alpha * \beta = \gamma$, so $\exists F'.\ F \xrightarrow{\beta} F'$ and $F' \models \|\phi'\|_{E'}$. By induction $E' * F' \models \phi'$, therefore $E * F \models \phi$.
3. $\phi = [\gamma]\phi'$. Then $\|\phi\|_E = \bigwedge_{E \xrightarrow{\alpha} E'} [\beta]\|\phi'\|_{E'}$ where $\alpha * \beta = \gamma$. The reasoning here is similar.
4. $\phi = \phi_1 \vee \phi_2$ or $\phi_1 \wedge \phi_2$. Then $\|\phi\|_E = \|\phi_1\|_E \vee \|\phi_2\|_E$ or $\|\phi_1\|_E \wedge \|\phi_2\|_E$, and the proof follows by induction.

It is known (see [19] and [24]) that we have:

Theorem 6 $E \approx F$ `if and only if`*:* $\forall\phi.\ E \models \phi \Leftrightarrow F \models \phi$.

In the work [21, 25, 14], characteristic formulae were used to prove the above property in a different manner, by proceeding as follows. For each process E, define its *characteristic formula* X_E as the modal formula:

$$X_E = \bigwedge_{E \xrightarrow{\alpha} E'} \langle\alpha\rangle X_{E'} \wedge \bigwedge_{\alpha} \left([\alpha] \bigvee_{E \xrightarrow{\alpha} E'} X_{E'} \right)$$

This characterizes so to say all the potential properties of E, along all its branches (the set of "*actions that E can do as well as those it cannot do*") up to strong bisimulation. In [21, 25, 14], it was actually proved that the following holds, when Hennesy-Milner logic is extended with fixed-points:

Proposition 4 $E \approx F$ `if and only if` $F \models X_E$

Besides, it was also pointed out in [14], that $F \models X_E$ should be interpreted as: F belongs 'to the *maximal* solution for' X_E.

For any modal formula ϕ, let us denote by $\phi \setminus H$ the formula obtained from ϕ by deleting all the sub-formulae of the form $\langle a\rangle\phi'$ and $[a]\phi'$ such that $a \in H$. We then have the following:

Theorem 7 *For any given $\phi = X_{E\setminus H}$, we have: $P \setminus H \models \phi$* `iff` $P \models \phi \setminus H$

Proof: We shall reason by induction on $|\phi|$. By definition, $\phi = X_{E\setminus H}$, so of the form:

$$\phi = \bigwedge_{a_i \notin H} \langle a_i\rangle\phi_i \wedge \bigwedge_{b_j} [b_j]\phi_j$$

and therefore:

$$\phi \setminus H = \bigwedge_{a_i \notin H} (\langle a_i\rangle\phi_i \setminus H) \wedge \bigwedge_{b_j \notin H} [b_j](\phi_j \setminus H)$$

where $\phi_i = X_{E_i\setminus H}$, with $E \setminus H \xrightarrow{a_i} E_i \setminus H$; each ϕ_j is of the form: $\phi_j = \bigvee_{E\setminus H \xrightarrow{b_j} E_{jk}\setminus H} X_{E_{jk}\setminus H}$.

(Note: for each $b_j \in H$, since $b_j \notin \mathtt{EAct}(E \setminus H)$, we have $\phi_j = \mathtt{ff}$.)

The assertion of the theorem is trivial if $\phi = \mathtt{tt}$; so assume that $\phi \neq \mathtt{tt}$. We first prove the implication $\Rightarrow$. So assume $P \setminus H \models \phi$; then :

- we have $P \setminus H \models \langle a_i\rangle\phi_i$; So $P \setminus H \xrightarrow{a_i} P' \setminus H$ (i.e., $P \xrightarrow{a_i} P'$) with $P' \setminus H \models \phi_i$. By induction we deduce $P' \models \phi_i \setminus H$, therefore $P \models \langle a_i\rangle\phi_i \setminus H$.
- and we also have $P \setminus H \models [b_j]\phi_j$; now since for each $b_j \in H$, the 'branch' $[b_j]\phi_j$ (i.e. to say the 'branch' $[b_j]\mathtt{ff}$) has already been cut in $\phi \setminus H$, we only need to check that for each $b_j \notin H$, $P \setminus H \models [b_j]\phi_j$ implies $P \models [b_j]\phi_j \setminus H$:

  If $P \setminus H \models [b_j]\phi_j$, then we have: $\forall P'.\ P \setminus H \xrightarrow{b_j} P' \setminus H$ with $P' \setminus H \models \phi_j$. Now since $\phi_j = \bigvee_{E\setminus H \xrightarrow{b_j} E_{jk}\setminus H} X_{E_{jk}\setminus H}$, there must exist a k for which $P' \setminus H \models X_{E_{jk}\setminus H}$; and so by induction $P' \models X_{E_{jk}\setminus H} \setminus H$, therefore $P' \models \bigvee X_{E_{jk}\setminus H} \setminus H$, which is $\phi_j \setminus H$ by definition; we deduce therefore that: $P \models [b_j]\phi_j \setminus H$.

This completes the proof that $P \setminus H \models \phi$ implies $P \models \phi \setminus H$. Consider next the inverse implication ($\Leftarrow$); and assume $P \models \phi \setminus H$.

For each $a_i, b_j \notin H$, by induction we deduce:

$P \models \langle a_i\rangle\phi_i \setminus H$ implies $P \setminus H \models \langle a_i\rangle\phi_i$, and $\quad P \models [b_j]\phi_j \setminus H$ implies $P \setminus H \models [b_j]\phi_j$.

Remains to study the 'branches' $[b_j]\phi_j$ such that $b_j \in H$.

Now, $b_j \notin \mathtt{EAct}(E\setminus H)$, so for each b_j, ϕ contains only $[b_j]\mathtt{ff}$, and obviously $P\setminus H \models [b_j]\mathtt{ff}$. This completes the proof that $P \models \phi \setminus H$ implies $P \setminus H \models \phi$. $\square$

One can use the various characterizations deduced above, for the security analysis of processes, in our timed synchronous setup, in particular via the criteria **NDC** and **RNI** that we defined earlier. Recall that by definition, a process E is said to satisfy **NDC** iff $\forall I$ we have $(E \mid I) \setminus H \approx E \setminus H$. Now by the results of Section 8.3, this is equivalent (for finite processes) to showing that:

$$\forall I.\ (\hat{E}_{|I|} * \hat{I}_{|E|}) \setminus H \approx E \setminus H.$$

For this, by Proposition 4, we only need to show that:

$$\forall I.\ (\hat{E}_{|I|} * \hat{I}_{|E|}) \setminus H \ \models\ X_{E\setminus H}$$

and by Theorem 7, which is equivalent to:

$$\forall I.\ (\hat{E}_{|I|} * \hat{I}_{|E|}) \ \models\ X_{E\setminus H} \setminus H$$

which is equivalent by Theorem 5 to showing that:

$$\forall I.\ \hat{I}_{|E|} \ \models\ \|X_{E\setminus H} \setminus H\|_{\hat{E}_{|I|}}$$

It seems possible to exploit the reductions of the form given in Theorem 5, or the one deduced above, to show that a given VPPA E does not satisfy **NDC**, or that it admits an attack of confidentiality. In order to perform such deductions efficiently, it seems necessary to formulate these reductions as transformations of the alternating tree automata that can be associated to a given modal μ-formula, see [26]. These are among the issues that we are currently investigating.

9 Conclusion

Cryptographic protocols have been modeled as value-passing process algebras by many authors in the recent past, and the resulting algebraic tools have been used in the analysis of information flow. Indeed the interleaving semantics of processes in parallel composition is a good basis for modeling protocols where several agents act concurrently. However, the key for the secrecy analysis in such models lies along the synchronous branches along which the information sent gets captured. As a consequence, our objective has been to show that the synchronous process algebra that we defined in [2] is an adequate algebraic basis on which secrecy analysis can be built. With such a purpose, we have extended and refined the notion of robust non-interference – introduced in [2] – to the setup of value-passing process algebras. We have shown that the interleaving semantics of CCS, for processes under parallel composition, can be expressed entirely in terms of our binary algebraic operator defining synchronization.

Given levels of secrecy, we have also shown how to define good observation criteria, capable of registering information on the flow of data between high and low level users, in such a way that these criteria are applicable to the secrecy analysis of cryptographic protocols; several examples of authentication and confidentiality analysis have been given. A by-product of our approach which is worth mentioning is the assertion proved in Theorem 3, namely: *if* there is an intruder, then there is a *linear* intruder. It is our belief that this could be used in building unification-like approaches, based on the $PACUID$-syntax which underlies our algebra.

We have shown how to extend our synchronous formalism into a timed process algebra. As we mentioned already, it is much simpler to obtain such a timed extension in our approach than in the earlier works [22, 12, 11], due principally to two reasons: (i) our algebra does not need an additional 'tick' action, and (ii) the synchronization operator '$*$' comes in handy for handling several aspects of 'time determinacy' in timed specifications. We have also shown how to recover algebraically and exclusively in terms of '$*$' the interleaving semantics of processes acting concurrently. Rules for partial model checking are also easier to define in our setup, based on the fact that synchronization is the central notion. The issue of how to use efficiently partial model checking techniques (expressed in our algebraic setup) for the purposes of formal verification, is part of our on-going research.

References

1. M. Abadi, A.D. Gordon. *Reasoning about Cryptographic Protocols in the Spi Calculus*, In. Proc. CONCUR'97, Concurrency Theory, LNCS 1243:59–73, Springer-Verlag, 1997.
2. S. Anantharaman, G. Hains. *A Synchronous Bisimulation Based Approach for Information Flow Analysis*, In Proc. of the Third Workshop AVOCS03, Southampton (UK), April 2003. (Long version: Research Report RR-2003-01, LIFO Université d'Orléans (Fr.) `http://www.univ-orleans.fr/SCIENCES/LIFO/prodsci/rapports/RR2003.htm.en`)
3. S. Anantharaman, J. Chen, G. Hains, P. Narendran. *A Synchronous Calculus for Secrecy Analysis*, Research Report RR-2003-05, LIFO Université d'Orléans (Fr.) `http://www.univ-orleans.fr/SCIENCES/LIFO/prodsci/rapports/RR2003.htm.en`)
4. H.R. Andersen. Partial Model Checking. In *Proc. of LICS'95*, pp. 398–407, IEEE Computer Scociety Press, 1995.
5. G. Boudol, *Notes on Algebraic Calculi of Processes*, In Logic and Models of Concurrent Systems, NATO ASI Series F-13, pp. 261–303, Springer, 1985.
6. G. Boudol and I. Castellani, *Non-Interference for Concurrent Programs and Thread Systems.* Technical Report 4254, Institut National de Recherche en Informatique et en Automatique (INRIA), September 2001. Accepted for publication in Theoretical Computer Science.
7. D. Dolev and A. C. Yao. *On Security of Public Key Protocols.* In IEEE Trans. on Information Theory, IT-30(1983):198–208.

8. R. Focardi, R. Gorrieri, *A Classification of Security Properties for Process Algebras*, In J. of Computer Security, 3(1):5-33, 1994/1995.
9. R. Focardi, *Analysis and Automatic Detection of Information Flow in Systems and Networks*. Ph.D Thesis - Report UBLCS-99-16, Dept. of Comp. Sc., University of Bologna, July 1999.
10. R. Focardi, R. Gorrieri, F. Martinelli, *Secrecy in Security Protocols as Non Interference*, In Elect. Notes in Theor. Computer Sc.(ENTCS), Elsevier, Vol 32, 2000.
11. R. Focardi, R. Gorrieri, F. Martinelli, *Rael-time Information Flow Analysis*, In IEEE Journal on Selected Areas in Communication, Vol 21:1, January 2003.
12. M. Hennessy, T. Reagan. *A Process Algebra for Timed Systems* In Information and Computation, Vol. 117, 1995, pp. 221-239.
13. A. Ingolfsdottir, H. Lin. *A Symbolic Approach to Value-passing Processes*. Chapter 7 in Handbook of Processes Algebra, Elsevier. 2001.
14. A. Ingolfsdottir and K.G. Larsen. Hennessy-Milner Logic with Recursion (Draft, March 4, 2003) `http://www.cs.auc.dk/ luca/DAT4/TEACH-NOTES/logic-note.pdf`
15. G. Lowe. *An Attack on the Needham-Schroeder Public-Key Authentication Protocol*. In Information Processing Letters, 56(3):131–133, 1995.
16. G. Lowe. *Some New Attacks on Security Protocols*. In Proc. of the 9th IEEE Computer Security Foundations Workshop, 1996.
17. F. Martinelli. Formal Methods for the Analysis of Open Systems with Applications to Security Properties. *Ph.D. Thesis*, University of Siena, 1998.
18. F. Martinelli. Partial Model Checkingand Theorem Proving for ensuring Security Properties. In *Proc. of CSFW'98*, pages 44-52, IEEE Computer Society Press, 1888.
19. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
20. S. Lafrance and J. Mullins, *Bisimulation-based Non-deterministic Admissible Interference and its Applications to the Analysis of Cryptographic Protocols*. In Elect. Notes in Theor. Computer Sc. (ENTCS), Elsevier, Vol 61, 2001.
21. M. Muller-Olm. Derivation of Characteristic formulae. In *Electronic Notes in Theoretical Computer Science*, vol 18, 1998.
22. X. Nicollin, J. Sifakis. *The Algebra of Timed Processes, ATP: Theory and Application* In Information and Computation, Vol. 114, 1994, pp. 131-178.
23. M. Rusinowitch, M. Turuani. *Protocol Insecurity with Finite Number of Sessions is NP-Complete* In Theor. Computer Science, Elsevier, Vol. 299(2003):451-475.
24. C.S. Stirling. Modal and Temporal Logics for Processes. In *Texts in Computer Science*, ISBN: 0-387-98717-7, Ed. Springer, 2001.
25. B. Steffen and A. Ingolfsdottir. Characteristic formulae for processes with divergence. In *Information and Computation*, 110(1): 149-163, 1994.
26. T. Wilke. Alernating Tree Automata, Parity Games, and Modal *mu*-Calculus. In *Bull. Soc. Math. Belge*, 8(2), May 2001 (Also: Invited Talk, Journés Montoises, Marne-la-Vallée, March 2000)
27. T.Y.C. Woo, S.S. Lam. *A Lesson on Authentication Protocol Design* In *ACM SYGOPS* Operating Systems Review, 1994, 28(3):24-37.