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 Synchronous Calculus for Secrecy Analysis

Siva ANANTHARAMAN, LIFO, Orléans (Fr.)
Jing CHEN, LIFO, Orléans (Fr.)
Gaétan HAINS, LIFO, Orléans (Fr.)
Paliath NARENDRAN, SUNY at Albany-NY (USA)

Rapport N^o **2003-05**

A Synchronous Calculus for Secrecy Analysis

Siva Anantharaman[1], Jing Chen[1], Gaétan Hains[1], Paliath Narendran[2]

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

[2] University at Albany–SUNY (USA),
e-mail: dran@cs.albany.edu

Abstract. Protocols are often modeled as value-passing processes in parallel composition, and their analysis based on CCS's interleaving semantics. The synchronous branches are the ones which are crucial in these models for analyzing issues of confidentiality. It turns out that process algebras themselves can be designed with synchronous parallel composition as the central notion. In such a vision, the choice operator '+' is an $ACUID$-operator, parallel synchronous composition will be distributive over choice, and process algebras can be specified over a $PACUID$ signature obtained by augmenting $ACUID$ with action prefixes. Bisimulation between processes reduces then to congruence over the $PACUID$-equational theory. Information flow analysis is finer when based on bisimulation on such a synchronous algebra, than when it is based on trace or weak bisimulation equivalence. Issues such as confidentiality can be formulated *exclusively* in terms of synchronous composition, in the $PACUID$-syntax. We present a DEXPTIME algorithm in this synchronous syntax for deciding such problems.

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. [8],[9],[13] –, 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 undoubtedly 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 defined 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, eliminating thereby the observational problems caused by the divergence of processes.

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 have also proposed in [2] a method for the analysis of confidentiality via a notion of *robust non-interference*, based on strong bisimulation in our synchronous setup. It was shown there that information flow analysis via this method is finer than via the methods based on trace or weak bisimulation equivalence proposed in [8],[9],[13].

In the current paper we establish formally the viewpoint that the issue of determining whether a given protocol P modeled as a (value-passing) process algebra has an 'information leak' for some intruder process I, 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, its principal algebraic properties, and the notion of robust non-interference ('RNI') of any given process P with respect to an 'intruder' I, as presented in [2]. In Sections 4 and 5, we extend and refine this algebraic formalism and the notion of 'RNI' to the framework of value-passing process algebras, well-suited for modeling and analyzing protocols. And an algorithm is presented, based on this extended RNI, to decide the issue of confidentiality of information flow on value-passing processes modeling – e.g. cryptographic – protocols; only the synchronization operator of the algebra is used in this algorithm, which is of complexity DEXPTIME w.r.t. the height of the modeling process, plus an upper bound for the size of the messages transmitted.

2 A Synchronous Process Algebra

The process algebra is constructed here as usual over a given set $\texttt{Act}$ of action symbols. Let $\texttt{Act} = \{\theta\} \biguplus \{a, \bar{a}, b, \bar{b}, \ldots\}$ where θ is a special 'internalizing' action symbol 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; '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. We will use variables $x, y, \ldots \in \texttt{Act}$, $\alpha, \beta, \ldots \in \texttt{Act}^*$ and $P, Q, R, \ldots \in \texttt{Proc}$. The grammar of ground process terms is

$$\texttt{Proc} ::= 0 \mid \texttt{Act.Proc} \mid \texttt{Proc} + \texttt{Proc} \mid \texttt{Proc} * \texttt{Proc} \mid P \setminus x.$$

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. The synchronization of actions is defined as a binary operation denoted '$*$', as follows:

$$x * y = \begin{cases} \theta & \text{if } x = \bar{y} \\ x & \text{if } y = \theta \\ y & \text{if } x = \theta \\ 0 & \text{otherwise} \end{cases}$$

(We agree here to denote a null or undefined action, as well as the null process, by 0.) The difference between CCS's τ and our θ must be already perceptible: CCS's τ would not have synchronized with any non-internal action, nor with itself. The operational semantics of our processes is a labeled transition system, corresponding to a relation on $\mathtt{Proc} \times \mathtt{Act} \times \mathtt{Proc}$, that we shall denote as $P \xrightarrow{x} Q$ and define 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 only be concerned with finite processes.

Definition 1 *For any process term P the set of possible actions for P is defined as the set:* $\mathtt{Act}(P) = \{x \in \mathtt{Act} \mid \exists P'.\ P \xrightarrow{x} P'\}$ *and the set of its direct successor processes as:* $\mathtt{Next}(P) = \{P' \in \mathtt{Proc} \mid \exists x.\ P \xrightarrow{x} P'\}$. *If $x \in \mathtt{Act}(P)$ then we shall denote by $P \xrightarrow{x}$* the set *of all processes P' such that $P \xrightarrow{x} P'$.*

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

ii) The semantics behind rule (Sync) is that θ is meant as a passive clock step to be absorbed by non-empty events in a synchronous composition. A process emitting θ is "counting" non-θ events in its parallel sibling; in particular $\theta * \theta = \theta$, i.e. two 'counting' processes synchronize.

iii) 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:

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

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

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

(4) 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.*

(5) Synchronous composition is *non-associative*: in general we do not have $P * (Q * R) \not\lesssim (P * Q) * R$. Here is an example: Let $P = a$, $Q = \bar{a}$ and $R = b$, with $b \not\in \{a, \theta\}$. 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 $R = b \xrightarrow{b} 0$.

2.2 Asynchronous Parallel Composition

Although not formally part of the setup, the notion of asynchronous parallel composition can be defined uniquely up to bisimulation in our formalism, as follows. Let us agree to call a process P *shallow* iff its additive normal form is a sum of single actions; e.g. $a + b$ and $c + d$ are shallow. And for any such process P, associate two other processes as follows:

i) $\hat{P} = \theta.P$ ('count unit time before doing any action of P');

ii) if $P = \sum_i a_i.0$, then $\tilde{P} = \sum_i a_i.\theta.0$

('count unit time after any action of P before stopping').

The *parallel asynchronous composition* of any two finite processes P, Q – that we shall denote $P \parallel Q$, or more simply $P \mid Q$ – is then defined by induction on the height of the processes:

case 1) both P, Q are *shallow*: here $P \mid Q = Q \mid P = (\hat{P} * \tilde{Q}) + (\tilde{P} * \hat{Q}) + (P * Q)$

case 2) (general case): 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 (recursively) defined uniquely up to bisimulation, as: $P \mid Q = \sum_{a_i} a_i.(P_i \mid Q) \;+\; \sum_{b_j} b_j.(P \mid Q_j) \;+ \sum_{i,j} (a_i * b_j).(P_i \mid Q_j)$.

We indeed get the asynchronous composition of P, Q à la CCS, once every occurrence of θ 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.

2.3 Bisimulation as an Equational Congruence

Let $\equiv$ be the smallest congruence on the set `Proc` of all finite 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) $a.X * b.Y \doteq (a * b).(X * Y)$.

Then: '$\equiv$' is the same as the congruence defined by bisimulation (cf. [2] for the proof).

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

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. In particular, bisimulation between finite process terms is decidable by normalization under this convergent rewrite system.

3 The Notion of 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. [4]).

Although it is possible to present the basic ideas on a semi-complete lattice of levels of secrecy, 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 `Act` 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 [4], 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.

Remarks 2. i) It is not true that $P/\mathcal{O}_L$ is bisimilar to the process obtained from P by replacing any occurrence of any high level action symbol hi in P by an occurrence of θ. For instance for any P, any $lo \neq \theta$ and any hi, we have by our definitions that: $lo * hi.P \approx 0$, but $(lo * \theta.P) \not\approx 0$. (The reason is that a '$*$'-composition of actions is a *single* action.)

ii) It is easily seen however that if the synchronization operator '$*$' does not appear explicitly in P, then $P/\mathcal{O}_L$ is bisimilar to the process obtained from P by replacing any occurrence of any high level action symbol hi in P by an occurrence of θ. In particular, $P/\mathcal{O}_L$ is (bisimilar to) the process obtained by such a replacement *in the additive normal form* of P.

iii) $\mathcal{O}_L$ is compatible with bisimulation (cf. the appendix in [2]).

Definition 2 *A process P is said to satisfy* non-interference *w.r.t. $\mathcal{O}_L$ iff the following property is satisfied:*

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

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. Following usual convention, for any process Q the process $Q\backslash H$ is that 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 1 *A process P satisfies RNI if and only if: $P\backslash H \approx P/\mathcal{O}_L$.*

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. □

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 $BNNI$ of [8]. However, a process satisfying $BNNI$ does *not* in general satisfy RNI, cf. the example below where h is hi and a is lo, and R is the null process.

$$\begin{tikzcd}
\mathrm{P} \arrow[d, "\mathrm{h}"'] \arrow[dr, "\mathrm{a}"] & \\
\mathrm{Q} \arrow[r, "\mathrm{a}"'] & \mathrm{R}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\mathrm{P} \arrow[r, "\theta"] \arrow[d, "\mathrm{h}"'] & \mathrm{P'} \arrow[d, "\mathrm{a}"] \\
\mathrm{Q} \arrow[r, "\mathrm{a}"'] & \mathrm{R}
\end{tikzcd}$$

P = ha.R + a.R is BNNI (not RNI) P = ha.R + θa.R is RNI (and BNNI)

In the process P to the left a low level user can only get information a, but the time he spends to get this information from its two different branches is *not* the same; whereas in the process P to the right, the low level user spends the same amount of time to retrieve the same information, whatever be the branch observed.

4 Value-Passing Process Algebras (VPPA’s)

In the applications that we shall be interested in, e.g. analysis of cryptographic protocols, processes are more 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 $c(x)$ or $\bar{c}(v)$ where c is any *channel*, and v a value from an externally defined domain (e.g. integers) and x is a 'value-variable' on such a domain, not to be confused with the upper-case 'process-variable' X. The '$*$'-product of actions is defined so that any $\bar{c}(v)$ is the conjugate of $c(x)$: $c(x) * \bar{c}(v) = \theta$ ("channel names must be conjugate; if so the value v is passed on to variable x"); $c(x) * \theta = \theta * c(x) = c(x)$, $\bar{c}(v) * \theta = \theta * \bar{c}(v) = \bar{c}(v)$, $\theta * \theta = \theta$; all the *other* products are deemed null. The (`Sync`) transition rule defined above is expressed in the value-passing setup as follows:

(`Sync-Absorb`) $$\frac{P \xrightarrow{\alpha} P' \quad Q \xrightarrow{\beta} Q' \quad \alpha = \theta \textit{ or } \beta = \theta,\ \alpha * \beta \neq 0}{P * Q \xrightarrow{\alpha * \beta} P' * Q'}$$

(`Sync-Comm`) $$\frac{P \xrightarrow{\bar{c}(v)} P' \quad Q \xrightarrow{c(x)} Q'}{P * Q \xrightarrow{\theta} P' * Q'[x/v]}$$

plus a symmetrical rule where P is exchanged with Q. It will also be useful to have terms of the form " `let` $x = e$ `in` P" which are defined as synonyms of $P[x/e]$; the issue of defining expression e's syntax and evaluating it to a value is assumed settled.

Value-passing processes are easily compiled into standard processes, while preserving their semantics (cf. e.g. [12]). We shall be assuming below 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 3. 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:

$$\begin{array}{ll} (1) & \pi_1\ (x,y) = x \\ (2) & \pi_2\ (x,y) = y \\ (3) & \texttt{dec}(\,\texttt{enc}(x,k), k^{-1}\,) = x \end{array}$$

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. [6]). 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. Then the process $(P \mid Q) \setminus C$ can have 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. This explains why the process $(P \mid Q) \setminus C$ is said to be the one deduced from P, Q under asynchronous composition with forced synchronization along C. We shall be making use of this notion below.

5 Non-Interference in Value-Passing Process Algebras

It is generally accepted that secrecy analysis of the information flow in any process P modeled as VPPA, can be done under the following assumption: the intruder I has access to all the channels of P, and has at his disposal one private output channel (inaccessible to the agents of P), on which he can transmit ('to the outside world') some information exchanged between the agents in P that was intended secret, and which he has intercepted as 'clear text'. For instance, the secrecy analysis methods developed in [9] are built on such a vision, w.r.t. a notion of non-interference based on weak trace equivalence between processes. The method that we propose to build is also based on similar lines of reasoning, the difference being that equivalence between processes, in our vision, is based on strong bisimulation; so our approach will be built on the notion of robust non-interference.

In the setup of value-passing process algebras, we need to refine further the notion of RNI. Indeed, for a meaningful secrecy analysis it does not suffice to just attribute secrecy levels to channel names; we will also need to consider the values ('data') getting transmitted along them. We propose to do the work in two steps for better comprehension.

5.1 Non-Interference in VPPA's: A Preliminary Definition

Let Q be any value-passing process and $\mathcal{X}$ any given set of channels. We can then define an observation criterion w.r.t. $\mathcal{X}$ on process Q – that we shall denote $\mathcal{O}_{\mathcal{X}}$ – as follows: all communication units along channels **not in** $\mathcal{X}$ are seen as θ. We then define process Q to satisfy $RNI_{\mathcal{X}}$ if and only if:

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

where $\mathcal{X}^c$ is the set of all channels of Q **not in** $\mathcal{X}$. The observation criterion $\mathcal{O}_L$ on Q defined earlier is just the special case where $\mathcal{X}$ is taken as the set of all low level channels of Q.

Let us try to apply this notion of non-interference for the analysis of information flow between processes.

Definition 3 *Let P be any process, I a second process (the 'intruder'), and let $\mathcal{X}$ be the set of all channels of P, all assumed 'public' (in particular shared with I). Denote by P_I be the process $(P \mid I) \setminus \mathcal{X}$, where the intruder I is in asynchronous composition with P, with forced synchronization on the (shared) channels in $\mathcal{X}$. and let $\mathcal{O}_{\mathcal{X}}$ be the corresponding observation criterion on P_I.*

Then P is said to be `confidentiality preserving` *for I, iff P_I satisfies the property $RNI_{\mathcal{X}}$. Otherwise, P is said to present an* `information leak` *for I.*

The following example illustrates and justifies (in part) the above definition.

Example 1. Consider a trivial 'protocol' $P = \bar{a}(1).0 \mid a(x).0$ where two principals simply exchange message-value 1. The channel set of P is $\{a\}$. Let us consider two possible intruders $I_0 = a(y).0$ and $I_1 = a(z).\bar{c}(z).0$. Intruder I_0 shares his unique channel a with P, intruder I_1 also shares a with P, but has an additional 'private' channel c.

Case of I_0: Following the notation of the definition above, the shared channel set between P and I_0 is $\mathcal{X} = \{a\}$. So we get: $P_{I_0} = (\bar{a}(1).0 \mid a(x).0 \mid a(y).0) \setminus a \approx \theta.0 + \theta.0 \approx \theta.0$. Now the set $\mathcal{X}^c$ is empty here; so $P_{I_0}/\mathcal{O}_{\mathcal{X}}$ is the same as P_{I_0}, and $P_{I_0} \setminus \mathcal{X}^c$ is also P_{I_0}. So the condition $RNI_{\mathcal{X}}$ is satisfied in this case.

Case of I_1: Here again the shared channel set between P and I_1 is $\mathcal{X} = \{a\}$. We get: $P_{I_1} = (P \mid I_1) \setminus a = (\ \bar{a}(1).0 \mid a(x).0 \mid a(z).\bar{c}(z).0\ ) \setminus a = \theta.0 + \theta.\bar{c}(1).0$. But now $\mathcal{X}^c = \{c\}$; so we have $P_{I_1}/\mathcal{O}_{\mathcal{X}} = \theta.0 + \theta.\theta.0$, while $P_{I_1} \setminus \mathcal{X}^c = \theta.0 + \theta.0 = \theta.0$. So the condition $RNI_{\mathcal{X}}$ is *not* satisfied in this case. This reflects the fact there has been information leak on P for I_1: Indeed the message-value 1 exchanged between the principals in P has been 'intercepted' by I_1, who then uses his 'private channel c to transmit it to the outside world'... □

Remarks 4. i) The reason why the preliminary definition as given above of non-interference on VPPA's is insufficient, must now be apparent. Indeed the above process $P = \bar{a}(1).0 \mid a(x).0$ would have presented information leak for the following two other 'intruders' too: $I_2 = a(z).\bar{c}(2).0$, and $I_3 = \bar{c}(1)$. Now I_2 does intercept the message-value 1 exchanged between the two agents in P, but is transmitting some other value to the outside world; whereas the value that I_3 is transmitting is one that he has not really intercepted. Thus, for $RNI_{\mathcal{X}}$ to be meaningful in information flow analysis on VPPA's, we need to incorporate some data flow information into it, by incorporating such an information into the observation criterion $\mathcal{O}_{\mathcal{X}}$. This will be done below via syntactical considerations on $\mathcal{O}_{\mathcal{X}}$.

ii) It might be argued, that semantically speaking there is no information leak on the above P for intruder I_1: since the value 1 that I_1 intercepts is actually transmitted in 'clear text' between the agents of P. It is not hard to distinguish such leaks as 'admissible' (or as 'admissible interferences'), from those where information transmitted as ciphered text between the principals of P is intercepted by an intruder as clear text; cf. the beginning of Section 7.

iii) The two processes considered (for bisimulation) in the definition of confidentiality preservation, namely: $(P \mid I) \setminus \mathcal{X} \setminus \mathcal{X}^c$ and $(\,(P \mid I) \setminus \mathcal{X}\,)/\mathcal{O}_{\mathcal{X}}$, have both only branches labeled with θ. It is immediate that the former is simulated by the latter: indeed 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 $(P \mid I) \setminus \mathcal{X} \setminus \mathcal{X}^c$.

Remarks 5. Suppose P is any (value-passing) process, $\mathcal{X}$ its set of channels. Then the process $P \setminus \mathcal{X}$ can be seen as reflecting the 'internal behavior' of P (note: only the synchronized actions along the channels in $\mathcal{X}$ remain visible on this restriction). In [9], 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, w.r.t. (strong) bisimulation, **NDC** is strictly stronger than $RNI_{\mathcal{X}}$.

Proposition 2 *Let P be any VPPA, $\mathcal{X}$ the set of its channels 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 that of confidentiality preservation: Let P be a simple 'ping-pong protocol' where two principals exchange a value back and forth:

$P = \bar{a}(1).a(y) \mid a(x).\bar{a}(x)$. Its channel set is $\mathcal{X} = \{a\}$, so the internal behavior of P is $P \setminus a \approx \theta.\theta.0$. Then P does *not* satisfy **NDC** for $I = a(z)$; indeed: $P_I = (P \mid I) \setminus a \approx \theta.0 + \theta.(\theta.0 + \theta.0) \approx \theta.0 + \theta.\theta.0$. However, $(P \mid I)$ has no channels outside $\mathcal{X} = \{a\}$, 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.) $\square$

5.2 Non-Interference in VPPA's: Observation to be Refined

We keep to the notation of the previous subsection. We propose here to refine the criterion $\mathcal{O}_{\mathcal{X}}$ into a criterion $\hat{\mathcal{O}}_{\mathcal{X}}$, by keeping track of the flow of data along the channels. Note that the branches of the process $(\,(P \mid I) \setminus \mathcal{X}\,)/\mathcal{O}_{\mathcal{X}}$ are all labeled with θ and that most of these θ's come from synchronized data transmission along channels of agents of P. The proposed refinement for $\mathcal{O}_{\mathcal{X}}$ will *cut* – or more precisely *shortcut* – the branches on this process whose θ-label was obtained earlier by hiding, but where the underlying action is along private output channels of I, transmitting data 'unconnected' to that accumulated earlier. This can be done by 'annotating' suitably each label θ. Here is an illustrative example.

Example 1 (*contd.*) We go back to the process $P = \bar{a}(1).0 \mid a(x).0$; and consider two possible intruders $I_3 = \bar{c}(1)$ and $I_4 = a(z).\bar{d}(2).\bar{c}(z).0$. For both, the channel set shared with P is $\mathcal{X} = \{a\}$; with the earlier definition of $\mathcal{O}_{\mathcal{X}}$, we would have respectively:

$$(\,(P \mid I_3) \setminus \mathcal{X}\,)/\mathcal{O}_{\mathcal{X}} = \theta.\theta \qquad \text{and} \qquad (\,(P \mid I_4) \setminus \mathcal{X}\,)/\mathcal{O}_{\mathcal{X}} = \theta + \theta.\theta.\theta.$$

Process P would have therefore presented an information leak for both I_3 and I_4. We can however distinguish between the various θ's by annotating them with data flow as subscripts; thus we may write:

$$(\,(P \mid I_3) \setminus \mathcal{X}\,)/\mathcal{O}_{\mathcal{X}} = \theta_{\{x:=1\}}.\theta_{\{\}},$$
$$(\,(P \mid I_4) \setminus \mathcal{X}\,)/\mathcal{O}_{\mathcal{X}} = \theta_{\{x:=1\}} + \theta_{\{z:=1\}}.\theta_{\{\}}.\theta_{\{z:=1\}}.$$

where x, z are bound or input variables (and z is also an output variable) in the processes $P \mid I_3,\ P \mid I_4$. The refined observation criterion $\hat{\mathcal{O}}_X$ will be so defined, that we shall have:

$$(\,(P \mid I_3) \setminus \mathcal{X}\,)/\hat{\mathcal{O}}_{\mathcal{X}} = \theta_{\{x,1;x:=1\}},$$
$$(\,(P \mid I_4) \setminus \mathcal{X}\,)/\hat{\mathcal{O}}_{\mathcal{X}} = \theta_{\{x,1;x:=1\}} + \theta_{\{z,1;z:=1\}}.\theta_{\{z\}}.$$

In other words, the θ-edges where the subscripts have only values/variables with no 'ancestors' in the subscripts of the preceding edges, will be cut or shortcut by $\hat{\mathcal{O}}_{\mathcal{X}}$. Following $\hat{\mathcal{O}}_{\mathcal{X}}$, process P will then have no leak for I_3, and the leak for I_4 will be of the same nature ('admissible' or otherwise) as for the intruder I_1 that we considered earlier. $\square$

5.3 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. [10]) 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 assignments event 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:

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

The operational semantics for the annotated actions on the processes remain as previously in essence; the (`Sync-Comm`) rule is now formulated w.r.t. the annotated actions, as follows:

$$(\mathtt{Sync-Comm}) \qquad \frac{E \xrightarrow{\bar{a}(v)_A} E' \quad F \xrightarrow{a(x)_B} F'}{E * F \xrightarrow{\gamma} E' * F'[x/v]} \quad \bar{a}(v)_A * a(x)_B = \gamma \neq 0$$

plus a symmetrical rule where E is exchanged with F.

The actions of any VPPA P 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 $a(x)$ has an empty annotation in general, except when it gets one under synchronization with a θ. Note that, by the above given rules for annotation, only synchronization accumulates annotations.

`Nota Bene`: *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, and consequently do *not* add up to one θ-action.

5.4 The Refined Observation Criterion and Non-Interference

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:

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

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, and up to α-conversion, we can make each input action use distinct bound variable, thus 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 $Parent_A$ will be operationally unambiguous.)

If $\Gamma = \alpha^1_{A_1} \alpha^2_{A_2} \ldots \alpha^n_{A_n}$ is any sequence of annotated actions in a process 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 shall refer to E' as the Γ-successor of E. We define then $Ancestor(E, \Gamma, x) = Parent_{A_1 \cup \cdots \cup A_n}(x)$, and set: $Assn(\Gamma) = A_1 \cup \cdots \cup A_n$. For each variable $x \in E'$, $Ancestor(E, \Gamma, x)$ is the set of value variables and Const_args of E on which x depends.

Given a root process G, a set $\mathcal{Y}$ of channel names, and a set S of value variables and value constants appearing in G. we define now a relation denoted $Out^S_{\mathcal{Y}} = Out^S_{\mathcal{Y}}(G)$ between value-passing successor processes of G, as the reflexive transitive closure of transitions along output action channels **not** in the given set $\mathcal{Y}$, *and such that the value variables of this output action have no ancestor in the set S.* In more formal terms:

- For any P, $P \xrightarrow{Out^S_{\mathcal{Y}}}_{\emptyset} P$;
- For any P with $G \xrightarrow{\Gamma} P$, we have $P \xrightarrow{Out^S_{\mathcal{Y}}}_{\Gamma'} P'$ for some annotation sequence Γ', iff:

- $P \xrightarrow{\Gamma'} P'$ where $\Gamma' = \bar{a}_1(e_1)_{A_1} \ldots \bar{a}_m(e_m)_{A_m}$, with $\bar{a}_i \notin \mathcal{Y}$ for $i = 1..m$; and
- for all $i = 1..m$, for all $x \in Vars(e_i)$, $Ancestor(G, \Gamma \cdot \bar{a}_1(e_1)_{A_1} \ldots \bar{a}_i(e_i)_{A_i}, x) \cap S = \emptyset$.

We define now a refined observation criterion $\hat{\mathcal{O}}_{\mathcal{Y},S} = \hat{\mathcal{O}}_{\mathcal{Y},S}(G)$, w.r.t. a given root process G, a given channel set $\mathcal{Y}$ and a given set S of value-variables and value-constants, as follows:

$$\frac{G \xrightarrow{\Gamma} E \xrightarrow{Out^S_{\mathcal{Y}}}_{\Gamma'} E',\ E' \xrightarrow{\alpha_B} E''}{E/\hat{\mathcal{O}}_{\mathcal{Y},S} \xrightarrow{\theta_{Assn(\Gamma')\cup B}} E''/\hat{\mathcal{O}}_{\mathcal{Y},S}} \quad \begin{array}{ll} if & \alpha = a(x) \ and \ a \notin \mathcal{Y}, \\ or\ if & \alpha = \bar{a}(e),\ \bar{a} \notin \mathcal{Y},\ and\ for\ all\ x \in Vars(e), \\ & Ancestor(G, \Gamma \cdot \Gamma' \cdot \alpha_B, x) \cap S \neq \emptyset; \end{array}$$

$$\frac{E \xrightarrow{\alpha_A} E'}{E/\hat{\mathcal{O}}_{\mathcal{Y},S} \xrightarrow{\alpha_A} E'/\hat{\mathcal{O}}_{\mathcal{Y},S}} \quad if \quad \alpha = a(x) \ or \ \bar{a}(e),\ with\ a \in \mathcal{Y},\ or\ if\ \alpha = \theta.$$

In more intuitive terms: the observation criterion $\hat{\mathcal{O}}_{\mathcal{Y},S}$ *shortcuts* all **output** actions along channels **not** in the given set $\mathcal{Y}$, *if the value variables of these output actions have no ancestors in the set S.* For example, let $G = a(x)|\bar{a}(1)|a(z).\bar{d}(2).\bar{d}(z)$, we have: $G \xrightarrow{\theta_{\{z:=1\}}} P \xrightarrow{\bar{d}(2)} P' \xrightarrow{\bar{d}(z)} P''$, but since we only care about whether the message 1 is intercepted and sent out, we may *shortcut* the useless output transition $\xrightarrow{\bar{d}(2)}$; we will thus have $P/\hat{\mathcal{O}}_{\{a,\bar{a}\},\{1\}} \xrightarrow{\bar{d}(z)} P''/\hat{\mathcal{O}}_{\{a,\bar{a}\},\{1\}}$; there will be no transition on $P/\hat{\mathcal{O}}_{\{a,\bar{a}\},\{1\}}$ under the action $\bar{d}(2)$.

Definition 4 *Let P be any process, I a second process (the 'intruder'), and let $\mathcal{X}$ be the set of channels of P, all assumed shared with I. (Without loss of generality) Assume P and I do not share value variables, and denote by S be the set of value variables and value constants in P. Let P_I be the process $(P \mid I) \setminus \mathcal{X}$, and denote by $\hat{\mathcal{O}}_{\mathcal{X}}$ the refined observation $\hat{\mathcal{O}}_{\mathcal{X},S}(P_I)$ as defined above. Then P is said to be* `confidentiality preserving` *for I, iff P_I satisfies the following (refined non-interference) property:*

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

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

As mentioned earlier (cf. end of Section 5.3) the above bisimulation relation on the two VPPA's has to be checked under the annotated actions. The following example shows (if needed) why this is necessary.

Example 3. Consider $P = a(x).0 \mid \bar{a}(1).0$ and $I = a(z).(\bar{c}(z).0 + \bar{a}(2).0)$, the shared channel set being $\mathcal{X} = \{a, \bar{a}\}$. So the processes to compare are:

$$\begin{aligned} P_1 &= (P \mid I) \setminus \mathcal{X} \setminus \mathcal{X}^c &&= \theta_{\{x:=1\}} + \theta_{\{z:=1\}}.\theta_{\{x:=2\}} \\ P_2 &= ((P \mid I) \setminus \mathcal{X})/\hat{\mathcal{O}}_{\mathcal{X}} &&= \theta_{\{x:=1\}} + \theta_{\{z:=1\}}.(\theta_{\{z:=1\}} + \theta_{\{x:=2\}}) \end{aligned}$$

If bisimulation does *not* take annotations into consideration, then we shall have $P_1 \approx P_2$. But obviously there is a leak, along the path $\theta_{\{z:=1\}}.\theta_{\{z:=1\}}$ in P_2, where the first θ comes from $\bar{a}(1) * a(z)$, and the second θ is the hiding of $\bar{c}(z)$ which outputs the intercepted message. In other words $\hat{\mathcal{O}}_{\mathcal{X}}$ alone cannot detect leaks, if bisimulation does not consider the annotations. (Note: P here is the same process as in Example 1, while our I here has the I_1 of Example 1 as a branch!) $\square$

6 Secrecy Analysis based on $\mathbf{R\hat{N}I}$

Given a VPPA P, our objective in this section is to propose an algorithm to find out if P presents an information leak for some intruder I, using only the syntax of our synchronous algebra. A process J will be said to be *linear* if it has no choice operator '+', i.e. to say as a term it is of the form $\alpha_1 \ldots \alpha_n.0$, where each α_i is an input or output action or θ (with annotations as suffixes).

Definition 5 *Let $\mathcal{X}$ be the channel set of P, $\bar{d}$ an output channel not element of $\mathcal{X}$ and with variables disjoint from P, and let S denote the set of value variables and value constants appearing in P. An* `intrusion witness` *for P is a linear process Q of the form $\alpha_1 \ldots \alpha_n.0$, 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:*

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

A linear process $\beta_1 \ldots \beta_n.0$ 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 an information leak from a process P can be traced down to a leak from some linear sub-process. More precisely:

Theorem 2 *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 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 channel set of P. We shall 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 $\bar{d}$; and the other θ-labels represent synchronization under '$*$'. Let us call *special* an edge on the tree P_2 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 below 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. Now we are assuming that 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 channels of P, say $\alpha_1(x), \bar{\alpha}_1(v)$. The θ-label of this edge gets indexed by an 'annotation' resulting from this synchronization, of the form: $x := v$. Assuming that x is an input variable in P and v an output variable (or value) in I, set: $P' = \alpha_1(x).P'^{(1)}$ and $J = \bar{\alpha}_1(v).J^{(1)}$. This means that we have guessed non-deterministically the first actions for the processes P' and J that we are looking for. (In case x comes from J and v from P, J will get $\alpha_1(x)$ and P' will get $\bar{\alpha}_1(v)$.)

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: $P' = \alpha_1(x_1) \ldots \alpha_m(x_m).P'^{(m)}$ and $J = \bar{\alpha}_1(v_1) \ldots \bar{\alpha}_m(v_m).J^{(m)}$. The $(m+1)$-st step goes then as follows:

case i): 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 $\hat{\mathcal{O}}_{\mathcal{X}}$ would have shortcut the action. 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.

case ii): 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. □

Remark 6. 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 Confidentiality Analysis in Cryptographic Protocols

The approach described above is applicable to the analysis of cryptographic protocols modeled as VPPA's, for analyzing confidentiality. 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 3.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 in general used for an information leak which is *not admissible* in the sense we just defined; see subsection 7.1 for a more formal definition. The following known attack on the Needham-Schroeder public key authentication protocol, illustrates well the reasoning behind the proof of Theorem 2.

Example 4. 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.

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

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:

1.1. $A \to I \quad : \{N_a, A\}_{K_i}$
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. $I \to A \quad : \{N_a, N_b\}_{K_a}$
1.3. $A \to I \quad : \{N_b\}_{K_i}$
2.3. $I(A) \to B : \{N_b\}_{K_b}$

The attack involves two concurrent runs of the protocol, where 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 intercepts in clear text –

modulo Dolev-Yao – the nonce N_b that B has intended for A (indeed it has been encrypted with 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; all these channels are public by assumption.

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

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 $Q = \theta_{A_1}.\theta_{A_2}.\theta_{A_3}.\theta_{A_4}.\theta_{A_5}.(\texttt{let } u = \texttt{dec}(z', K_i) \texttt{ in } \bar{d}(u)).0$

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

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 2, the intrusion witness $Q = \theta_{A_1}.\theta_{A_2}.\theta_{A_3}.\theta_{A_4}.\theta_{A_5}.(\texttt{let } u = \texttt{dec}(z', K_i) \texttt{ in } \bar{d}(u)).0$ can be obtained as $P' * J$, where:

$$P' = \bar{c}_1(v_1).c(y).\bar{c}(v_2).c_1(x).\bar{c}_1(v_3).\theta.0,$$

and $J = c_1(y').\bar{c}(v_1').c(x').\bar{c}_1(x').c_1(z').\bar{d}_{\{u:=\texttt{dec}(z', K_i)\}}(u).0.$

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. □

Our objective now is to present an algorithm based on the proof of Theorem 2 for detecting confidentiality attacks on a cryptographic protocol modeled as a VPPA.

7.1 A DEXPTIME-Algorithm for Deciding Attacks

We assume given a VPPA P of finite height h, and a finite set S of value-terms (those intended to remain secret between the agents acting in P), with respect to which possible 'attacks' are to be detected. 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 3.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. We define now in formal terms the notion of an attack.

Definition 6 *Let $\mathcal{X}$ be the set of channels of P, all assumed public. Suppose P presents an information leak for some process I, w.r.t. the observation criterion $\hat{\mathcal{O}}_{\mathcal{X},S}$. Then the leak is called an* `attack on` *P* `by` *I if and only if: 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* $x \in Parent_{A_1,\ldots A_n}(z) \cap Vars(P)$.

<u>**Algorithm for detecting attacks on the VPPA P**</u>:

N = Upper bound for the size of messages; `Chans` = Channel set of P.

Set $\mathcal{X} = Vars(P)$ as a global variable; and call $CheckAttack(P, 0, 0, \emptyset)$,
    where the function *CheckAttack* is defined as:

$CheckAttack(T, P', J, Assn)$ {
  For each $T \xrightarrow{\alpha} T'$ do
    Case i) $\alpha = a_A(x)$: choose a data-expression e such that
        $size(e) \leq N$ and $Vars(e) \subseteq Vars(J)$:
          set $B = \{x := e\}$ and let $\bar{\alpha} = \bar{a}_B(e)$;
    Case ii) $\alpha = \bar{a}_A(e)$: choose a new value-variable y
        with $y \notin Vars(P) \cup Vars(J)$;
          set $B = \{y := e\}$ and let $\bar{\alpha} = a_B(y)$;
  Set $P' := P'.\alpha\,;\ J := J.\bar{\alpha}\,;\ Assn := Assn \cup A \cup B$;
  For all expressions e formed of variables and Const_args in J, and
      value-constants, such that: $Vars(e) \neq \emptyset$ and $size(e) \leq N$,
    do
      If $e\!\downarrow\in S$, `and` $Assn$ contains `no` assignment of the form:
        $x := v, e\!\downarrow, v'$, where $x \in Parent_{Assn}(e) \cap \mathcal{X}$,
      `then` choose channels $d, \bar{d} \notin$ `Chans`, a new variable z,
        and RETURN $J.\bar{d}_{\{z:=e\}}(z)$; EXIT; /* Found an Attack !! */
      `else` $CheckAttack(T', P', J, Assn)$; /* Try with the subtree below*/
    od /* End of loop `For .. ` e ` .. do` */
od /* End of loop `For each ..`$T \xrightarrow{\alpha} T'$` .. do` */
RETURN(NULL); /* all subtrees visited, no attack*/
}

This algorithm (which remains to be optimized) is of complexity DEXPTIME w.r.t. $(h + N)$, where h is the height of P: indeed the number of ways of choosing the subtrees in P is

$O(exp(h))$ and the number of ways of choosing the data-expressions e in Case i), or in the 'For all e ..' loop, is $O(exp(N))$. $\square$

8 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 analysis lies along the synchronous branches. As a consequence, our objective in this paper 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 the binary algebraic operator defining synchronization. And a natural algorithm is established entirely in the syntax of our synchronous algebra, for the secrecy analysis of protocols modeled as value-passing processes. It was shown in [2] that bisimulation in our setup reduces to the congruence modulo the $PACUID$-equational theory underlying the process algebra. It seems therefore likely that our algorithm for the detection of attacks can be expressed as one of semantic unification between value-passing processes, modulo $PACUID$ plus the inference rules formalizing the observation criterion presented in the current paper. Developing in detail such a vision over some of the ideas from [3] is part of our projected future work; we hope thereby to extend the approach of this paper to infinite processes; we also hope to define observation criteria which are more symbolic than the data-oriented $\hat{\mathcal{O}}_{\mathcal{X}}$ of this paper.

Finally a word of comparison with the work done in [8, 9], which was based on non-interference and weak trace equivalence. Our main theorem (Theorem 2) implies in essence that information leaks detected by our approach can also be detected via theirs. The essential differences between our approach and theirs are as follows: In [8, 9] the objective was to check that a process P satisfies the non-interference property w.r.t. *all intruders* I; to get around this universal quantification on I, the notion of a *most powerful intruder* was developed; and to construct effectively such an agent as a value-passing process, a finiteness hypothesis was made on the universe of messages. Our approach looks at the problem in its negated form: *does there exist* an I for whom P presents an information leak ? The algorithm we present gives a complete and natural answer for this question (under similar finiteness hypothesis); it is based on the assertion proved in Theorem 2, namely: *if* there is an intruder, then there is a *linear* intruder.

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, P. Narendran, M. Rusinowitch, *ACID-Unification is NEXPTIME-Decidable*, In Proc. **MFCS'03**, Bratislava, LNCS 2747:169–178, Springer-Verlag, August 2003. (Long version: Research Report RR-2003-02, LIFO, Université d'Orléans (Fr.) (`ftp://ftp-lifo.univ-orleans.fr/pub/Users/siva/RR2003-02.ps`)
4. G. Boudol, *Notes on Algebraic Cacluli of Processes*, In Logic and Models of Concurrent Systems, NATO ASI Series F-13, pp. 261–303, Springer, 1985.
5. 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.
6. D. Dolev and A. C. Yao. *On Security of Public Key Protocols*. In IEEE trans. on Information Theory, IT-30(1983):198–208.
7. R. Focardi, R. Gorrieri, *A Classification of Security Properties for Process Algebras*, In J. of Computer Security, 3(1):5-33, 1994/1995.
8. 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.
9. 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.
10. A. Ingolfsdottir, H. Lin. *A Symbolic Approach to Value-passing Processes*. Chapter 7 in Handbook of Processes Algebra, Elsevier. 2001.
11. G. Lowe. *An Attack on the Needham-Schroeder Public-Key Authentication Protocol*. In Information Processing Letters, 56(3):131–133, 1995.
12. R.Milner. *Communication and Concurrency*. Prentice Hall, 1989.
13. 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.