A Denotational Approach to the Static Analysis of Cryptographic Processes

Benjamin Aziz [1,2]

Computer Science Department
University College Cork
Cork, Ireland

Geoff W. Hamilton[3], David Gray [4]

School of Computing
Dublin City University
Dublin, Ireland

Abstract

We present in this paper, a non-uniform static analysis for detecting the term-substitution property in processes specified in the spi calculus. The property is essential in defining security breaches, like secrecy and authenticity. The analysis is fully denotational, preserving compositionality and facilitating implementations in functional programming.

Key words: Static Analysis, Cryptographic Protocols, Denotational Semantics

1 Introduction

The spi calculus [3] offers a process algebraic theory for the modelling of mobile cryptographic systems that essentially extends the *value-passing* theory of the π-calculus [18] by the addition of primitives for performing cryptographic operations. We view the cryptographic operations as the *value-processing* behaviour of processes. One aspect of the term-passing and term-processing behaviours that has significance when detecting security breaches in processes is *term substitution*. Term substitutions occur whenever communications,

[1] This work was partially funded by project IMPROVE (Enterprise Ireland Strategic Grant ST/2000/94)
[2] Email: b.aziz@cs.ucc.ie
[3] Email: hamilton@computing.dcu.ie
[4] Email: dgray@computing.dcu.ie

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

successful decryption, signature verification and tuple splitting take place. Security implications arise in scenarios where a process classified at a low secrecy level obtains high-level data, or a process classified at a high trust level obtains low-level data. We call the former *information leakage* and the latter *authenticity breach*.

In this paper, we propose a non-uniform static analysis that captures the property of term substitution in the spi calculus, and that is capable of distinguishing between the different instances of these substitutions. The analysis is based on a closed denotational model of the language derived from Stark's domain-theoretic equations for the π-calculus [20]. Apart from facilitating the use of computationally important mathematical concepts, like fixed-points, the denotational approach has the advantage that it results in an implementation that is straightforward in functional programming.

The abstraction that we adopt limits, in a safe manner, the number of new names and the depth of data structures, both of which can grow, in concrete semantics, to an infinite level as a result of the use of the replication operator. Since the abstract semantic domain is kept finite, least fixed-point calculations are guaranteed to terminate for our monotonic semantic functions.

2 Related Work

The area of the static analysis of cryptographic processes has been researched intensively in recent years using a variety of techniques, which one can only mention here a few examples. These include types [2,1,7], symbolic methods [5,16,12,13,17], abstract interpretation [19,6,15,21] and control flow analysis [8,9,11,10]. The current work expands on previous work presented in [6] for mobile systems modelled by the π-calculus. In [6], names substituting variables were captured directly in the analysis, and limiting the number of copies of these names was sufficient as an abstraction. In the spi calculus, the difficulty with this abstraction is that complex data structures also have infinite depths, as well as containing infinite number of names. This problem is solved by the use of tags. Also, in [6], the main property of interest was privacy, whereas here, we also deal with authenticity. Finally, the current analysis, unlike [6], is capable of distinguishing between the different instances of variables, which is necessary for future definitions of the properties like freshness and resource exhaustion.

3 A Domain-Theoretic Model

We extend Stark's domain-theoretic model [20] by introducing the following predomain equations, which underlie the primitive behaviour of input, output and silent actions as well as termination (deadlock) in the spi calculus:

$$Spi \cong 1 + \mathbb{P}(Spi_\perp + In + Out) \tag{1}$$
$$In \cong N \times (T \to Spi_\perp) \tag{2}$$
$$Out \cong N \times (T \times Spi_\perp + N \to \ldots N \to (T \times Spi_\perp)) \tag{3}$$
$$T \cong N + Sec + Pub + Sig + Tup \tag{4}$$
$$Sec \cong T \times N \tag{5}$$
$$Pub \cong T \times N \tag{6}$$
$$Sig \cong T \times N \tag{7}$$
$$Tup \cong T \times \ldots \times T \tag{8}$$

Where $Spi_\perp$ is the domain of processes, In and Out are the predomains of input and output actions, respectively. Input actions are modelled as pairs; a name, N (the channel), and a function, $T \to Spi_\perp$, that can be instantiated with a term, T, yielding a process in $Spi_\perp$. Output actions are divided into free and bound output actions. These are pairs consisting of the channel, N, and either another pair, $T \times Spi_\perp$, denoting the message, T, and the residue $Spi_\perp$ (free outputs), or composed functions, $N \to \ldots N \to (T \times Spi_\perp)$, that introduce new names to the message, T, and the residue, $Spi_\perp$ (bound outputs). $\mathbb{P}(-)$ is Plotkin's powerdomain applied to the disjoint union of input, output and silent actions (the latter represented by $Spi_\perp$) to construct Spi. The one-element predomain, 1, representing terminated (deadlocked) processes is adjoined as in [4, Def. 3.4]. The flat predomain of closed terms, T, is defined as the disjoint union of the predomains of names, N, secret-key ciphers, Sec, public-key ciphers, Pub, digital signatures, Sig and finite tuples, Tup. The predomains Sec, Pub and Sig can be expressed as pairs, where a term, T, is encrypted/signed with a key, N.

The following functions are defined as usual, leading to $Spi_\perp$ [4, Def. 3.3]:

$$\emptyset : 1 \to Spi_\perp \tag{9}$$
$$\{\!|-|\!\} : (Spi_\perp + In + Out)_\perp \to Spi_\perp \tag{10}$$
$$\uplus : (Spi_\perp \times Spi_\perp) \to Spi_\perp \tag{11}$$
$$new : (N \multimap Spi_\perp) \to Spi_\perp \tag{12}$$

The empty set, $\emptyset$, is required to represent inactive processes. The singleton map, $\{\!|-|\!\}$, creates elements of $Spi_\perp$ from elements of input, output and silent actions. new is used to interpret the effects of restriction. Finally, $\uplus$, is the standard multiset union operator representing non-determinism.

Concrete elements of $t \in T$ include names, a, b, c, k, secret-key ciphers, $sec(t, k)$, public-key ciphers, $pub(t, k)$, digital signatures, $sig(t, k)$ and tuples, $(t_1, \ldots, t_n)$. Elements $p \in Spi_\perp$ include the bottom element, $\{\!|\perp|\!\}$, the empty set, $\emptyset$ (where $\{\!|\perp|\!\} \sqsubseteq \emptyset$), input actions, $in(a, \lambda y.p)$, free output actions, $out(a, t, p)$, bound output actions, $out(a, \lambda n_1 \ldots \lambda n_m.(t, p))$ and silent actions, $tau(p)$. The effects of restriction are interpreted by defining new concretely as in Figure 1.

These effects lead to the blocking of processes attempting to communicate over fresh, non-extruded channels and the transformation of free outputs to

$$
\begin{array}{lll}
new(\lambda n.\emptyset) & = & \emptyset \\
new(\lambda n.\{\!|\bot|\!\}) & = & \{\!|\bot|\!\} \\
new(\lambda n.\{\!|in(a,\lambda x.p)|\!\}) & = & \\
\left\{\begin{array}{ll} \emptyset, & \text{if } a = n \\ \{\!|in(a,\lambda x.new(\lambda n.p))|\!\}, & \text{otherwise} \end{array}\right. & & \\
new(\lambda n.\{\!|out(a,t,p)|\!\}) & = & \\
\left\{\begin{array}{ll} \emptyset, & \text{if } a = n \\ \{\!|out(a,\lambda n.(t,p))|\!\}, & \text{if } n \in n(t) \text{ and } n \neq a \\ \{\!|out(a,t,new(\lambda n.p))|\!\}, & \text{otherwise} \end{array}\right. & & \\
new(\lambda n.\{\!|out(a,\lambda m_1 \ldots \lambda m_k.(t,p))|\!\}) & = & \\
\left\{\begin{array}{ll} \emptyset, & \text{if } a = n \\ \{\!|out(a,\lambda n.\lambda m_1 \ldots \lambda m_k.(t,p))|\!\}, & \text{if } n \in n(t) \text{ and } n \neq a \\ \{\!|out(a,\lambda m_1 \ldots \lambda m_k.(t,new(\lambda n.p)))|\!\}, & \text{otherwise} \end{array}\right. & & \\
new(\lambda n.\{\!|tau(p)|\!\}) & = & \{\!|tau(new(\lambda n.p))|\!\} \\
new(\lambda n.(p_1 \uplus p_2)) & = & new(\lambda n.p_1) \uplus new(\lambda n.p_2)
\end{array}
$$

Fig. 1. The concrete definition of *new* over elements $p \in Spi_\bot$.

bound outputs whenever the message of communication is a fresh name. The denotational semantics for the spi calculus can now be given as a semantic function, $\mathcal{S}(\![P]\!)\ \rho\ \phi_\mathcal{S} \in Spi_\bot$, defined by the set of rules of Figure 2. The multiset, ρ, is used to hold processes composed in parallel with the analysed process, where rule $(\mathcal{R}0)$ is used to interpret the contents of ρ. The environment, $\phi_\mathcal{S} : V \to T_\bot$, where V is the predomain of variables, captures any term substitutions that occur in the semantics. The special function, $\varphi_\mathcal{S} : (V \to T_\bot) \times Term \to T$, returns the semantic value of a term, given substitutions recorded by $\phi_\mathcal{S}$ as follows:

$$
\varphi_\mathcal{S}(\phi_\mathcal{S}, M) = \left\{\begin{array}{ll}
\phi_\mathcal{S}(M), & \text{if } M \in V \\
M, & \text{if } M \in N \\
sec(\varphi_\mathcal{S}(\phi_\mathcal{S}, M'), \varphi_\mathcal{S}(\phi_\mathcal{S}, N)), & \text{if } M = \{M'\}_N \\
pub(\varphi_\mathcal{S}(\phi_\mathcal{S}, M'), \varphi_\mathcal{S}(\phi_\mathcal{S}, N)), & \text{if } M = \{\!|M'|\!\}_N \\
sig(\varphi_\mathcal{S}(\phi_\mathcal{S}, M'), \varphi_\mathcal{S}(\phi_\mathcal{S}, N)), & \text{if } M = [\!\{M'\}\!]_N \\
(\varphi_\mathcal{S}(\phi_\mathcal{S}, M_1), \ldots, \varphi_\mathcal{S}(\phi_\mathcal{S}, M_n)), & \text{if } M = (M_1, \ldots, M_n)
\end{array}\right.
$$

$$
\begin{array}{ll}
(\mathcal{S}1) & \mathcal{S}(\!|\mathbf{0}|\!)\ \rho\ \phi_{\mathcal{S}} = \emptyset \\
(\mathcal{S}2) & \mathcal{S}(\!|M(y).P|\!)\ \rho\ \phi_{\mathcal{S}} = \{\!|in(\varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M), \lambda y.\mathcal{R}(\!|\{\!|P|\!\}_{\rho}|\!)\ \phi_{\mathcal{S}})|\!\} \text{ where, } \varphi(\phi_{\mathcal{E}}, M) \in N \\
(\mathcal{S}3) & \mathcal{S}(\!|\overline{M}\langle L\rangle.P|\!)\ \rho\ \phi_{\mathcal{S}} = \\
 & \quad \biguplus_{M'(z).P' \in \rho} \{\!|tau(\mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho[P'/M'(z).P']|\!)\ \phi_{\mathcal{S}}[z \mapsto \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, L)]|\!\} \\
 & \quad \uplus \{\!|out(\varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M), \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, L), \mathcal{R}(\!|\{\!|P|\!\}_{\rho}|\!)\ \phi_{\mathcal{S}})|\!\} \\
 & \quad \text{where, } \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M) = \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M') \in N \\
(\mathcal{S}4) & \mathcal{S}(\!|(\nu a)P|\!)\ \rho\ \phi_{\mathcal{S}} = new(\lambda a.\mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{S}}) \\
(\mathcal{S}5) & \mathcal{S}(\!|P \mid Q|\!)\ \rho\ \phi_{\mathcal{S}} = \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{S}} \\
(\mathcal{S}6) & \mathcal{S}(\!|!P|\!)\ \rho\ \phi_{\mathcal{S}} = \mathcal{F}(-1) \\
 & \text{where, } \mathcal{F}(n) = \mathit{let}\ p_1 = \mathcal{S}(\!|\prod_{i=1}^{n} P[bnv_i(P)/bnv(P)]\ |\!)\ \rho\ \phi_{\mathcal{S}}\ \mathit{in} \\
 & \qquad \mathit{let}\ p_2 = \mathcal{S}(\!|\prod_{i=1}^{n+2} P[bnv_i(P)/bnv(P)]\ |\!)\ \rho\ \phi_{\mathcal{S}} =\ \mathit{in} \\
 & \qquad \mathit{if}\ p_1 = p_2\ \mathit{then}\ p_1\ \mathit{else}\ \mathcal{F}(n+1) \\
 & \text{and, } bnv_i(P) = \{x_i \mid x \in bnv(P)\} \\
(\mathcal{S}7) & \mathcal{S}(\!|\mathit{if}\ M = L\ \mathit{then}\ P\ \mathit{else}\ Q|\!)\ \rho\ \phi_{\mathcal{S}} = \\
 & \quad \begin{cases} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{S}}, \text{ if } \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M) = \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, L) \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{S}}, \text{ otherwise} \end{cases} \\
(\mathcal{S}8) & \mathcal{S}(\!|\mathit{let}\ (x_1, \ldots, x_n) = M\ \mathit{in}\ P\ \mathit{else}\ Q|\!)\ \rho\ \phi_{\mathcal{S}} = \\
 & \quad \begin{cases} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi'_{\mathcal{S}}, \text{ if } \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, M) = (t_1, \ldots, t_n) \\ \text{where, } \phi'_{\mathcal{S}} = \phi_{\mathcal{S}}[x_1 \mapsto t_1, \ldots, x_n \mapsto t_n] \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{S}}, \text{ otherwise} \end{cases} \\
(\mathcal{S}9) & \mathcal{S}(\!|\mathit{case}\ L\ \mathit{of}\ \{x\}_N\ \mathit{in}\ P\ \mathit{else}\ Q|\!)\ \rho\ \phi_{\mathcal{S}} = \\
 & \quad \begin{cases} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi'_{\mathcal{S}}, & \text{if } \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, L) = sec(t, k) \text{ and } \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, N) = k \\ \text{where, } \phi'_{\mathcal{S}} = \phi_{\mathcal{S}}[x \mapsto t] \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{S}}, & \text{otherwise} \end{cases} \\
(\mathcal{S}10) & \mathcal{S}(\!|\mathit{case}\ L\ \mathit{of}\ \{\!|x|\!\}_N\ \mathit{in}\ P\ \mathit{else}\ Q|\!)\ \rho\ \phi_{\mathcal{S}} = \\
 & \quad \begin{cases} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi'_{\mathcal{S}}, \text{ if } \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, L) = pub(t, k^+) \text{ and } \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, N) = k^- \\ \text{where, } \phi'_{\mathcal{S}} = \phi_{\mathcal{S}}[x \mapsto t] \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{S}}, \text{ otherwise} \end{cases} \\
(\mathcal{S}11) & \mathcal{S}(\!|\mathit{case}\ L\ \mathit{of}\ [\![x]\!]_N\ \mathit{in}\ P\ \mathit{else}\ Q|\!)\ \rho\ \phi_{\mathcal{S}} = \\
 & \quad \begin{cases} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi'_{\mathcal{S}}, \text{ if } \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, L) = sig(t, k^-) \text{ and } \varphi_{\mathcal{S}}(\phi_{\mathcal{S}}, N) = k^+ \\ \text{where, } \phi'_{\mathcal{S}} = \phi_{\mathcal{S}}[x \mapsto t] \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{S}}, \text{ otherwise} \end{cases} \\
(\mathcal{R}0) & \mathcal{R}(\!|\rho|\!)\ \phi_{\mathcal{S}} = \biguplus_{P \in \rho} \mathcal{S}(\!|P|\!)\ (\rho \backslash \{\!|P|\!\}_{\rho})\ \phi_{\mathcal{S}}
\end{array}
$$

Fig. 2. The denotational semantics of the spi calculus.

Note that since we only deal with closed terms, the case where $M = x \in V$ and $\phi_{\mathcal{S}}(x) = \perp$ will never be encountered (the case of open terms).

The description of rules ($\mathcal{S}1$)–($\mathcal{S}11$) is as follows. Rule ($\mathcal{S}1$) interprets the meaning of a null process as the empty set mapping, $\emptyset$. Rules ($\mathcal{S}2$) and ($\mathcal{S}3$) deal with processes guarded with input and output actions, respectively. The rule for output actions, ($\mathcal{S}3$), considers communications between the output channel and appropriate input channels guarding processes in ρ. The $\phi_{\mathcal{S}}$ is updated appropriately with semantic elements. Rule ($\mathcal{S}4$) uses the *new* mapping to interpret the meaning of a restriction. Rule ($\mathcal{S}5$) interprets directly parallel composition by the addition of the parallel subprocesses to ρ.

Finally, rule ($\mathcal{S}6$) interprets a replicated process, $!P$, by calculating the higher-order function, $\mathcal{F} : \mathbb{N} \rightarrow \mathit{Spi}_{\perp}$, starting from the bottom number, $n = -1$, which computes the bottom semantic element, $\mathcal{F}(-1) = \{\!|\perp|\!\}$. The value of n is increased by 2 in each iteration to allow for any interactions between the copies of the replication to take place. This continues until the least fixed-point is reached. Due to the fact that the semantic domain, $\mathit{Spi}_{\perp}$, is infinite, this calculation may not terminate within finite limits. The rule also uses the labelling mechanism to rename all the bound names and variables, $bnv(P)$, of the spawned processes by subscripting those variables and names with a number signifying each spawned copy. Since this renaming of bound variables and names is, in fact, α-conversion, the resulting process on the right side of the rule is structurally equivalent to a subprocess of the replication on the left side. This preserves the compositionality of the denotational semantics.

The rest of the rules rely on the meaning of terms as held by the $\phi_{\mathcal{S}}$ environment before resolving the analysed process. In rule ($\mathcal{S}7$), the meaning of two terms is compared, and depending on the result, one of two processes is chosen and added to ρ. Rules ($\mathcal{S}8$)–($\mathcal{S}11$) deal with tuple splitting and cryptographic processes. The result of the tuple splitting and cryptographic operations are used to update the $\phi_{\mathcal{S}}$ with the appropriate semantic terms. A residual process, P, signifying the success of the operation is also added to ρ. In case an operation fails, an alternative process, Q, is chosen and added to ρ.

4 Non-Standard Semantics

The non-standard semantics of the spi calculus extends the standard denotational semantics introduced in Section 3, where term substitutions are recorded in a special environment $\phi_{\mathcal{E}} : V \rightarrow \wp(T)$ that maps each variable of a closed process to the set of semantic terms that may substitute that variable during the evaluation of the meaning of a process. Since the non-standard semantics is precise (copies of bound names and variables are always distinct), each variable will be mapped to a singleton set at most per choice of control flow, representing the term that substitutes the variable.

A domain, $D_{\perp} = V \rightarrow \wp(T)$, can be constructed, ordered by subset inclu-

sion:

$$\forall \phi_{\mathcal{E}1}, \phi_{\mathcal{E}2} \in D_\perp : \phi_{\mathcal{E}1} \sqsubseteq_{D_\perp} \phi_{\mathcal{E}2} \Leftrightarrow \forall x \in V : \phi_{\mathcal{E}1}(x) \subseteq \phi_{\mathcal{E}2}(x)$$

With the bottom element, $\perp_{D_\perp}$, being the null environment, $\phi_{\mathcal{E}0}$, that maps each variable to the empty set. The union of environments operation, $\cup_\phi$, can also be defined as follows:

$$\forall \phi_{\mathcal{E}1}, \phi_{\mathcal{E}2} \in D_\perp, x \in V : (\phi_{\mathcal{E}1} \cup_\phi \phi_{\mathcal{E}2})(x) = \phi_{\mathcal{E}1}(x) \cup \phi_{\mathcal{E}2}(x)$$

The non-standard semantic domain is formed by pairing $D_\perp$ with the standard semantic domain, $Spi_\perp$, resulting in $Spi_\perp \times D_\perp$. The bottom element of this domain is the pair $(\perp_{Spi_\perp}, \perp_{D_\perp})$. The non-standard semantics for the spi calculus can now be defined by the semantic function, $\mathcal{E}(\llbracket P \rrbracket)\ \rho\ \phi_\mathcal{E} \in (Spi_\perp \times D_\perp)$, on the structure of P as in Figure 3.

The ρ multiset holds all the processes in parallel with the process under interpretation. The definition of the $\varphi_\mathcal{E} : (V \to \wp(T)) \times Term \to T$ function allows for the meaning of a term to be computed under a particular $\phi_\mathcal{E}$ environment:

$$\varphi_\mathcal{E}(\phi_\mathcal{E}, M) = \begin{cases} t, & \text{if } M \in V \ \wedge\ \phi_\mathcal{E}(M) = \{t\} \\ M, & \text{if } M \in N \\ sec(\varphi_\mathcal{E}(\phi_\mathcal{E}, M'), \varphi_\mathcal{E}(\phi_\mathcal{E}, N)), & \text{if } M = \{M'\}_N \\ pub(\varphi_\mathcal{E}(\phi_\mathcal{E}, M'), \varphi_\mathcal{E}(\phi_\mathcal{E}, N)), & \text{if } M = \{\!|M'|\!\}_N \\ sig(\varphi_\mathcal{E}(\phi_\mathcal{E}, M'), \varphi_\mathcal{E}(\phi_\mathcal{E}, N)), & \text{if } M = [\!\{M'\}\!]_N \\ (\varphi_\mathcal{E}(\phi_\mathcal{E}, M_1), \ldots, \varphi_\mathcal{E}(\phi_\mathcal{E}, M_n)), & \text{if } M = (M_1, \ldots, M_n) \end{cases}$$

The semantic rules are described as follows. In ($\mathcal{E}1$), the meaning of a null process is described as the pair $(\emptyset, \phi_\mathcal{E})$, where $\phi_\mathcal{E}$ is the environment supplied to the rule initially. Rules ($\mathcal{E}2$) and ($\mathcal{E}3$) deal with the cases of input and output actions, respectively. Communications are dealt with in ($\mathcal{E}3$) for output actions, therefore, $\phi_\mathcal{E}$ remains unchanged in ($\mathcal{E}2$) for input actions. The rule for output actions requires that terms used as channels should evaluate to names, and communications occur whenever an input channel is matched in ρ. The value of $\phi_\mathcal{E}$ is updated with the message term substituting the input parameter. Rule ($\mathcal{E}4$) interprets the meaning of a restriction using the *new* operation on the first element of the resulting pair, whereas the second element reflects the environment resulting from the residue. This is justified as internal communications are preserved by restriction. Rule ($\mathcal{E}5$) composes two parallel processes in ρ.

The replication of processes is dealt with in rule ($\mathcal{E}6$) by computing a special function, $\mathcal{F} : \mathbb{N} \to Spi_\perp \times D_\perp$, starting at the bottom number, $n = -1$, and incrementing n by 2 until we reach a least fixed-point for $v_1 \in Spi_\perp \times D_\perp$. Such a computation is not guaranteed to terminate due to the infinite nature

$$
\begin{array}{ll}
(\mathcal{E}1) & \mathcal{E}(\![\mathbf{0}]\!)\ \rho\ \phi_{\mathcal{E}} = (\emptyset, \phi_{\mathcal{E}}) \\
(\mathcal{E}2) & \mathcal{E}(\![M(x).P]\!)\ \rho\ \phi_{\mathcal{E}} = (\{\!|in(\varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M), \lambda x.p')|\!\}, \phi_{\mathcal{E}}) \\
 & \text{where, } (p', \phi'_{\mathcal{E}}) = \mathcal{R}(\![\{\!|P|\!\}_\rho]\!)\ \phi_{\mathcal{E}} \text{ and, } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) \in N \\
(\mathcal{E}3) & \mathcal{E}(\![\overline{M}\langle L\rangle.P]\!)\ \rho\ \phi_{\mathcal{E}} = \\
 & (\biguplus_{M'(z).P' \in \rho} \{\!|tau(p')|\!\} \uplus \{\!|out(\varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M), \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L), p'')|\!\}, \bigcup_{\phi}_{M'(z).P' \in \rho} \phi'_{\mathcal{E}} \cup_\phi \phi_{\mathcal{E}}) \\
 & \text{if, } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) = \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M') \in N \\
 & \text{where, } (p', \phi'_{\mathcal{E}}) = \mathcal{R}(\![\{\!|P|\!\}_\rho \uplus_\rho \rho[P'/M'(z).P']]\!)\ \phi_{\mathcal{E}}[z \mapsto \{\varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L)\}] \\
 & \text{and, } (p'', \phi''_{\mathcal{E}}) = \mathcal{R}(\![\{\!|P|\!\}_\rho]\!)\ \phi_{\mathcal{E}} \\
(\mathcal{E}4) & \mathcal{E}(\![(\nu a)P]\!)\ \rho\ \phi_{\mathcal{E}} = (new(\lambda a.p'), \phi'_{\mathcal{E}}) \text{ where, } (p', \phi'_{\mathcal{E}}) = \mathcal{R}(\![\{\!|P|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}} \\
(\mathcal{E}5) & \mathcal{E}(\![P \mid Q]\!)\ \rho\ \phi_{\mathcal{E}} = \mathcal{R}(\![\{\!|P|\!\}_\rho \uplus_\rho \{\!|Q|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}} \\
(\mathcal{E}6) & \mathcal{E}(\![!P]\!)\ \rho\ \phi_{\mathcal{E}} = \mathcal{F}(-1) \\
 & \text{where, } \mathcal{F}(n) = \mathit{let}\ v_1 = \mathcal{E}(\![\prod_{i=1}^{n} P[bnv_i(P)/bnv(P)]\ ]\!)\ \rho\ \phi_{\mathcal{E}}\ \mathit{in} \\
 & \mathit{let}\ v_2 = \mathcal{E}(\![\prod_{i=1}^{n+2} P[bnv_i(P)/bnv(P)]\ ]\!)\ \rho\ \phi_{\mathcal{E}}\ \mathit{in}\ \mathit{if}\ v_1 = v_2\ \mathit{then}\ v_1\ \mathit{else}\ \mathcal{F}(n+1) \\
 & \text{and, } bnv_i(P) = \{x_i \mid x \in bnv(P)\} \\
(\mathcal{E}7) & \mathcal{E}(\![\mathit{if}\ M = L\ \mathit{then}\ P\ \mathit{else}\ Q]\!)\ \rho\ \phi_{\mathcal{E}} = \begin{cases} \mathcal{R}(\![\{\!|P|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}, \text{ if } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) = \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L) \\ \mathcal{R}(\![\{\!|Q|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}, \text{ otherwise} \end{cases} \\
(\mathcal{E}8) & \mathcal{E}(\![\mathit{let}\ (x_1, \ldots, x_n) = M\ \mathit{in}\ P\ \mathit{else}\ Q]\!)\ \rho\ \phi_{\mathcal{E}} = \begin{cases} \mathcal{R}(\![\{\!|P|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}[x_1 \mapsto \{t_1\} \ldots \\ \qquad\qquad x_n \mapsto \{t_n\}], \\ \text{if } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) = (t_1, \ldots, t_n) \\ \mathcal{R}(\![\{\!|Q|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}, \text{ otherwise} \end{cases} \\
(\mathcal{E}9) & \mathcal{E}(\![\mathit{case}\ L\ \mathit{of}\ \{x\}_N\ \mathit{in}\ P\ \mathit{else}\ Q]\!)\ \rho\ \phi_{\mathcal{E}} = \begin{cases} \mathcal{R}(\![\{\!|P|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}[x \mapsto \{t\}, \\ \text{if } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L) = sec(t, k) \\ \text{and } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, N) = k \\ \mathcal{R}(\![\{\!|Q|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}, \text{ otherwise} \end{cases} \\
(\mathcal{E}10) & \mathcal{E}(\![\mathit{case}\ L\ \mathit{of}\ \{\![x]\!\}_N\ \mathit{in}\ P\ \mathit{else}\ Q]\!)\ \rho\ \phi_{\mathcal{E}} = \begin{cases} \mathcal{R}(\![\{\!|P|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}[x \mapsto \{t\}], \\ \text{if } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L) = pub(t, k^+) \\ \text{and } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, N) = k^- \\ \mathcal{R}(\![\{\!|Q|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}, \text{ otherwise} \end{cases} \\
(\mathcal{E}11) & \mathcal{E}(\![\mathit{case}\ L\ \mathit{of}\ [\![x]\!]_N\ \mathit{in}\ P\ \mathit{else}\ Q]\!)\ \rho\ \phi_{\mathcal{E}} = \begin{cases} \mathcal{R}(\![\{\!|P|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}[x \mapsto \{t\}], \\ \text{if } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, L) = sig(t, k^-) \\ \text{and } \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, N) = k^+ \\ \mathcal{R}(\![\{\!|Q|\!\}_\rho \uplus_\rho \rho]\!)\ \phi_{\mathcal{E}}, \text{ otherwise} \end{cases} \\
(\mathcal{R}0) & \mathcal{R}(\![\rho]\!)\ \phi_{\mathcal{E}} = (\biguplus_{P \in \rho} p', \bigcup_{\phi}_{P \in \rho} \phi'_{\mathcal{E}}), \text{ where, } (p', \phi'_{\mathcal{E}}) = \mathcal{E}(\![P]\!)\ (\rho \backslash \{\!|P|\!\}_\rho)\ \phi_{\mathcal{E}}
\end{array}
$$

Fig. 3. The non-standard semantics of the spi calculus.

of the non-standard semantic domain, $Spi_{\perp} \times D_{\perp}$. Also, α-conversion renames the set of bound names and variables of each process copy, while maintaining the compositionality of the semantics. Rule ($\mathcal{E}7$) deals with a conditional process, where the meaning of the overall process is chosen from the two branch processes based on the semantic equality of the compared terms. Pair splitting is dealt with in rule ($\mathcal{E}8$) where the $\phi_{\mathcal{E}}$ is updated to hold the result of the substitution of local variables by elements of a tuple. The rest of the rules ($\mathcal{E}9$)–($\mathcal{E}11$) deal with cryptographic processes performing secret-key decryption, public-key decryption and digital signature verification.

The correctness requirement for the non-standard semantics of the spi calculus, with respect to its standard semantics, is expressed in the following theorem, which states that the standard element of the non-standard semantics is equivalent to the value obtained from the standard semantics..

Theorem 1 (Correctness of the Non-Standard Semantics)
$\forall P \in \mathcal{P} : (\mathcal{S}(\llbracket P \rrbracket)\ \rho\ \phi_{\mathcal{S}} = p)\ \wedge\ (\mathcal{E}(\llbracket P \rrbracket)\ \rho\ \phi_{\mathcal{E}} = (p', \phi'_{\mathcal{E}})) \quad \Rightarrow \quad p = p'$

Proof. The proof is by induction on the standard and non-standard semantics. □

5 Abstract Semantics

We begin by assuming a finite predomain of tags, *Tag*, ranged over by $t, \dot{t}, \ddot{t}$, where t is the tag of a generic term, $\dot{t}$ is the tag of a primitive term (name, variable) and $\ddot{t}$ is the tag of a complex term (ciphertext, signature, tuple). Next we tag (sub)terms of the analysed process with unique tags. More precisely, we tag M in the constructs *let* $(x_1, \ldots, x_n) = (M_1, \ldots, M_n)$ *in* P *else* Q, *case* $\{M\}_L$ *of* $\{x\}_N$ *in* P *else* Q, *case* $\{\!|M|\!\}_L$ *of* $\{\!|x|\!\}_N$ *in* P *else* Q, *case* $[\![M]\!]_L$ *of* $[\![x]\!]_N$ *in* P *else* Q and $\overline{N}\langle M\rangle.P$.

For example, tagging the term $\{(\{a\}_c, \{b\}_e)\}_d$ yields $\{(\{a^{\dot{t}1}\}_c^{\ddot{t}1}, \{b^{\dot{t}2}\}_e^{\ddot{t}2})^{\ddot{t}3}\}_d^{\ddot{t}4}$.
The following functions are also defined over tags, terms and processes:
– $value\_of(\{t_1, \ldots, t_n\}) = \{M_1, \ldots, M_n\}$. This function can be applied to a set of tags, $\{t_1, \ldots, t_n\}$, returning the corresponding set of terms, $\{M_1, \ldots, M_n\}$. Hence, $value\_of(\{\dot{t}1, \ddot{t}4\}) = \{a^{\dot{t}1},\ \{(\{a^{\dot{t}1}\}_c^{\ddot{t}1}, \{b^{\dot{t}2}\}_e^{\ddot{t}2})^{\ddot{t}3}\}_d^{\ddot{t}4}\ \}$.
– $tags\_of(P) = \{t_1, \ldots, t_n\}$. This function returns the set, $\{t_1, \ldots, t_n\}$, of tags used in a process, P.
For example: $tags\_of(\overline{m}\langle a^{\dot{t}1}\rangle.\overline{m}\langle\{(b^{\dot{t}2}, c^{\dot{t}3})^{\ddot{t}1}\}_k^{\ddot{t}2}\rangle.\mathbf{0}) = \{\dot{t}1, \dot{t}2, \dot{t}3, \ddot{t}1, \ddot{t}2\}$.
– $untag(\{M'_1, \ldots, M'_n\}) = \{M_1, \ldots, M_n\}$. When applied to a set of tagged terms, $\{M'_1, \ldots, M'_n\}$, this function removes all associated tags yielding a set of untagged terms, $\{M_1, \ldots, M_n\}$. Hence:
$untag(\{a^{\dot{t}5},\ \{(a^{\dot{t}1}, \{b^{\dot{t}2}\}_e^{\ddot{t}2})^{\ddot{t}3}\}_d^{\ddot{t}4}\ \}) = \{a,\ \{(a, \{b\}_e)\}_d\ \}$. The function behaves as *id* if a term, M', has no tags.

We now introduce the $\alpha_{k,k'}$ abstraction function, which keeps to a finite level, the number of copies of bound variables, names and tags.

Definition 1 *Define* $\alpha_{k,k'} : \mathbb{N} \times \mathbb{N} \times (V + N + Tag) \rightarrow (V^\sharp + N^\sharp + Tag^\sharp)$*:*

$$\forall M \in (V+N+Tag), i, k, k' \in \mathbb{N} : \alpha_{k,k'}(M) = \begin{cases} \dot{t}_k, \ \textit{if } M = \dot{t}_i \in Tag \ \textit{and } i > k \\ \ddot{t}_{k'}, \ \textit{if } M = \ddot{t}_i \in Tag \ \textit{and } i > k' \\ x_k, \ \textit{if } M = x_i \in V \ \textit{and } i > k \\ a_k, \ \textit{if } M = a_i \in N \ \textit{and } i > k \\ M, \ \textit{otherwise} \end{cases}$$

The resulting abstract predomains, $V^\sharp$, $N^\sharp$ and $Tag^\sharp$, can be defined as $V^\sharp = V \backslash \{x_j \mid j > k\}$, $N^\sharp = N \backslash \{a_j \mid j > k\}$ and $Tag^\sharp = Tag \backslash (\{\dot{t}_j \mid j > k\} \cup \{\ddot{t}_i \mid i > k'\})$. Informally, k constrains the number of bound variables and names, and tags of primitive terms, whereas k' constrains the number of tags of complex terms. In effect, constraining the tags of primitive terms implies limiting the copies of bound names and variables carrying the tags, whereas constraining the number of tags of complex terms means limiting the depth of data structures.

For example, in the process $!(\nu\, n)\overline{a}\langle n^{\dot{t}}\rangle \mid\ !a(x)$, it is possible to spawn infinite copies of each replication, $(\nu n_1)\overline{a}\langle n_1^{\dot{t}_1}\rangle \mid a(x_1) \mid (\nu n_2)\overline{a}\langle n_2^{\dot{t}_2}\rangle \mid a(x_2) \mid \ldots$. It is clear that $\dot{t}$ is an indicator to the number of copies n has after spawning each process. On the other hand, the process $!a(x).\overline{a}\langle\{x\}_k^{\ddot{t}}\rangle \mid \overline{a}\langle b\rangle$, which also spawns $a(x_1).\overline{a}\langle\{x_1\}_k^{\ddot{t}_1}\rangle \mid a(x_2).\overline{a}\langle\{x_2\}_k^{\ddot{t}_2}\rangle \mid \overline{a}\langle b\rangle \mid \ldots$ demonstrates the role of $\ddot{t}$ as an indicator to the number of times the ciphertext, $\{x\}_k$, is applied to b.

Using the $\alpha_{k,k'}$ abstraction, we construct the abstract environment $\phi_\mathcal{A} : V^\sharp \rightarrow \wp(Tag^\sharp)$, which maps each abstract bound variable of the analysed process to a set of tags, representing terms that could substitute that variable during the abstract semantics. An abstract domain $D_\perp^\sharp = V^\sharp \rightarrow \wp(Tag^\sharp)$ is formed ordered by subset inclusion:

$$\forall \phi_{\mathcal{A}1}, \phi_{\mathcal{A}2} \in D_\perp^\sharp, x \in V^\sharp : \phi_{\mathcal{A}1} \sqsubseteq_{D_\perp^\sharp} \phi_{\mathcal{A}2} \Leftrightarrow \phi_{\mathcal{A}1}(x) \subseteq \phi_{\mathcal{A}2}(x)$$

The bottom element, $\perp_{D_\perp^\sharp}$, is the null environment, $\phi_{\mathcal{A}0}$, mapping each variable to $\{\}$. Taking $D_\perp^\sharp$ as the abstract semantic domain, we can define the abstract semantics of the spi calculus by the function $\mathcal{A}(\![P]\!)\ \rho\ \phi_\mathcal{A} \in D_\perp^\sharp$, as in Figure 4. ρ again is a multiset of processes in parallel with the analysed process. The special function, $\varphi_\mathcal{A} : (V^\sharp \rightarrow \wp(Tag^\sharp)) \times Term \rightarrow \wp(Term)$, returns a set of terms corresponding to a term, M, given substitutions captured by $\phi_\mathcal{A}$:
$\varphi_\mathcal{A}(\phi_\mathcal{A}, M) = \varphi'_\mathcal{A}(\phi_\mathcal{A}, M')_{\{\}}$,
where, $M' = M[\alpha_{k,k'}(t)/t][\alpha_{k,k'}(x)/x][\alpha_{k,k'}(n)/n]$
and $\varphi'_\mathcal{A}(\phi_\mathcal{A}, M)_s = \textit{if } M \in s \textit{ then } \{\} \textit{ else}$

$(\mathcal{A}1)$ $\mathcal{A}(\!|\mathbf{0}|\!)\ \rho\ \phi_{\mathcal{A}} = \phi_{\mathcal{A}}$ $\quad$ $(\mathcal{A}2)$ $\mathcal{A}(\!|M(x).P|\!)\ \rho\ \phi_{\mathcal{A}} = \phi_{\mathcal{A}}$

$(\mathcal{A}3)$ $\mathcal{A}(\!|\overline{M}\langle L^t\rangle.P|\!)\ \rho\ \phi_{\mathcal{A}} = (\bigcup_{M'(z).P' \in \rho}^{\phi}\ \phi'_{\mathcal{A}})\ \cup_{\phi}\ \phi_{\mathcal{A}}$

if, $untag(\varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, M)) \cap untag(\varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, M')) \cap \mathcal{N} \neq \{\}$

where, $\phi'_{\mathcal{A}} = \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho[P'/M'(z).P']|\!)\ \phi''_{\mathcal{A}}$

and, $\phi''_{\mathcal{A}} = \phi_{\mathcal{A}}[\alpha_{k,k'}(z) \mapsto\ \phi_{\mathcal{A}}(\alpha_{k,k'}(z)) \cup \{\alpha_{k,k'}(t)\}]$

$(\mathcal{A}4)$ $\mathcal{A}(\!|(\nu a)P|\!)\ \rho\ \phi_{\mathcal{A}} \quad = \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{A}}$

$(\mathcal{A}5)$ $\mathcal{A}(\!|P \mid Q|\!)\ \rho\ \phi_{\mathcal{A}} \quad = \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{A}}$

$(\mathcal{A}6)$ $\mathcal{A}(\!|!P|\!)\ \rho\ \phi_{\mathcal{A}} \quad = \mathcal{F}(-1)$ where, $\mathcal{F}(n) = let\ \phi_1 = \mathcal{A}(\!|\prod_{i=1}^{n} ren(P,i)\ |\!)\ \rho\ \phi_{\mathcal{A}}\ in$

$let\ \phi_2 = \mathcal{A}(\!|\prod_{i=1}^{n+2} ren(P,i)\ |\!)\ \rho\ \phi_{\mathcal{A}}\ in\ if\ \phi_1 = \phi_2\ then\ \phi_1\ else\ \mathcal{F}(n+1)$

and, $\forall x \in bnv(P), t \in tags\_of(P) : ren(P,i) = (P[x_i/x])[t_i/t]$

$(\mathcal{A}7)$ $\mathcal{A}(\!|if\ M = L\ then\ P\ else\ Q|\!)\ \rho\ \phi_{\mathcal{A}} =$

$$\begin{cases} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{A}}, \text{ if, } untag(\varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, M)) \cap untag(\varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, L)) \neq \{\} \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{A}}, \text{ otherwise} \end{cases}$$

$(\mathcal{A}8)$ $\mathcal{A}(\!|let\ (x_1, \ldots, x_n) = M\ in\ P\ else\ Q|\!)\ \rho\ \phi_{\mathcal{A}} =$

$$\begin{cases} \bigcup_{(M_1^{t_1}, \ldots, M_n^{t_n}) \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, M)}^{\phi} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi'_{\mathcal{A}} \ \text{ if, } \exists (M_1^{t_1}, \ldots, M_n^{t_n}) \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, M) \\ \text{where, } \phi'_{\mathcal{A}} = \phi_{\mathcal{A}}[\alpha_{k,k'}(x_1) \mapsto \phi_{\mathcal{A}}(\alpha_{k,k'}(x_1)) \cup \{\alpha_{k,k'}(t_1)\}, \ldots, \\ \qquad\qquad \alpha_{k,k'}(x_n) \mapsto \phi_{\mathcal{A}}(\alpha_{k,k'}(x_n)) \cup \{\alpha_{k,k'}(t_n)\}] \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{A}}, \qquad\qquad \text{otherwise} \end{cases}$$

$(\mathcal{A}9)$ $\mathcal{A}(\!|case\ L\ of\ \{x\}_N\ in\ P\ else\ Q|\!)\ \rho\ \phi_{\mathcal{A}} =$

$$\begin{cases} \bigcup_{\{M^t\}_n \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, L)}^{\phi} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi'_{\mathcal{A}}, \text{ if, } n \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, N) \\ \text{where, } \phi'_{\mathcal{A}} = \phi_{\mathcal{A}}[\alpha_{k,k'}(x) \mapsto \phi_{\mathcal{A}}(\alpha_{k,k'}(x)) \cup \{\alpha_{k,k'}(t)\}] \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{A}}, \qquad\qquad \text{otherwise} \end{cases}$$

$(\mathcal{A}10)$ $\mathcal{A}(\!|case\ L\ of\ \{\!|x|\!\}_N\ in\ P\ else\ Q|\!)\ \rho\ \phi_{\mathcal{A}} =$

$$\begin{cases} \bigcup_{\{\!|M^t|\!\}_{n^+} \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, L)}^{\phi} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi'_{\mathcal{A}}, \text{ if, } n^- \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, N) \\ \text{where, } \phi'_{\mathcal{A}} = \phi_{\mathcal{A}}[\alpha_{k,k'}(x) \mapsto \phi_{\mathcal{A}}(\alpha_{k,k'}(x)) \cup \{\alpha_{k,k'}(t)\}] \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{A}}, \qquad\qquad \text{otherwise} \end{cases}$$

$(\mathcal{A}11)$ $\mathcal{A}(\!|case\ L\ of\ [\![x]\!]_N\ in\ P\ else\ Q|\!)\ \rho\ \phi_{\mathcal{A}} =$

$$\begin{cases} \bigcup_{[\![M^t]\!]_{n^-} \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, L)}^{\phi} \mathcal{R}(\!|\{\!|P|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi'_{\mathcal{A}}, \text{ if, } n^+ \in \varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, N) \\ \text{where, } \phi'_{\mathcal{A}} = \phi_{\mathcal{A}}[\alpha_{k,k'}(x) \mapsto \phi_{\mathcal{A}}(\alpha_{k,k'}(x)) \cup \{\alpha_{k,k'}(t)\}] \\ \mathcal{R}(\!|\{\!|Q|\!\}_{\rho} \uplus_{\rho} \rho|\!)\ \phi_{\mathcal{A}}, \qquad\qquad \text{otherwise} \end{cases}$$

$(\mathcal{R}0)$ $\mathcal{R}(\!|\rho|\!)\ \phi_{\mathcal{A}} \quad = \bigcup_{P \in \rho}^{\phi} \mathcal{A}(\!|P|\!)\ (\rho \backslash \{\!|P|\!\}_{\rho})\ \phi_{\mathcal{A}}$

Fig. 4. The abstract semantics of the spi calculus.

$$\begin{cases} \bigcup\limits_{L \in value\_of(\phi_{\mathcal{A}}(untag(M)))} \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, L)_{s \cup \{M\}} & \text{if } M \in \mathcal{V} \\ \{M\}, & \text{if, } M \in \mathcal{N} \\ \{\forall N' \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, N)_{s \cup \{M\}}, L' \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, L)_{s \cup \{M\}} : \ \{N'\}^t_{L'}\}, & \text{if, } M = \{N\}^t_L \\ \{\forall N' \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, N)_{s \cup \{M\}}, L' \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, L)_{s \cup \{M\}} : \ \{\![N']\!\}^t_{L'}\}, & \text{if, } M = \{\![N]\!\}^t_L \\ \{\forall N' \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, N)_{s \cup \{M\}}, L' \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, L)_{s \cup \{M\}} : \ [\![N']\!]^t_{L'}\}, & \text{if, } M = [\![N]\!]^t_L \\ \{\forall M'_1 \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, M_1)_{s \cup \{M\}}, \ldots, M'_n \in \varphi'_{\mathcal{A}}(\phi_{\mathcal{A}}, M_n)_{s \cup \{M\}} : \\ \quad (M'_1, \ldots, M'_n)^t \ \}, & \text{if, } M = (M_1, \ldots, M_n)^t \end{cases}$$

The description of the rules is as follows. Rules ($\mathcal{A}1$) and ($\mathcal{A}2$) return the $\phi_{\mathcal{A}}$ environment unchanged. Communications are dealt with in rule ($\mathcal{A}3$) for output actions, where synchronising output and input channels yield a communication, in which the tag of the message is captured by $\phi_{\mathcal{A}}$. The semantics is imprecise, since $\phi_{\mathcal{A}}$ only captures an abstract tag as a value for an abstract variable. Rules ($\mathcal{A}4$) and ($\mathcal{A}5$) deal with the cases of restriction and parallel composition directly by placing the subprocesses with the rest in ρ. The rule for replication, ($\mathcal{A}6$), performs a least fixed-point calculation using a special function, $\mathcal{F} : \mathbb{N} \to D^\sharp_\perp$. This least fixed-point occurs at the minimum number, n, such that $\mathcal{A}(\![\prod_{i=1}^{n} ren(P, i)]\!]\ \rho\ \phi_{\mathcal{A}} = \mathcal{A}(\![\prod_{i=1}^{n+2} ren(P, i)]\!]\ \rho\ \phi_{\mathcal{A}}$. The termination property of this calculation is stated formally in the following theorem.

Theorem 2 (Termination of the least fixed-point calculation) *The calculation of rule* ($\mathcal{A}6$) *terminates.*

Proof. To prove the termination property, it is necessary to satisfy two requirements. First, the semantic domain must be finite. This is satisfied by the definition of $D^\sharp_\perp$. The second requirement is to prove the monotonicity of $\mathcal{A}(\![\prod^{n} P]\!]\ \rho\ \phi_{\mathcal{A}}$, i.e. $\mathcal{A}(\![\prod^{n} P]\!]\ \rho\ \phi_{\mathcal{A}} \sqsubseteq \mathcal{A}(\![\prod^{n+2} P]\!]\ \rho\ \phi_{\mathcal{A}}$. To prove this, we simplify the inequality into $\mathcal{A}(\![Q]\!]\ \rho\ \phi_{\mathcal{A}} \sqsubseteq \mathcal{A}^\pi(\![Q \mid P]\!]\ \rho\ \phi_{\mathcal{A}}$, where $Q = \prod^{n} P$. This is further simplified to become $\mathcal{A}(\![Q]\!]\ \rho\ \phi_{\mathcal{A}} \sqsubseteq \mathcal{A}^\pi(\![Q]\!]\ \rho'\ \phi_{\mathcal{A}}$, where $\rho' = \rho \uplus_\rho \{\!|P|\!\}_\rho$. This can be proven by induction over $\mathcal{A}(\![P]\!]\ \rho\ \phi_{\mathcal{A}}$. In particular, the most interesting cases are rules ($\mathcal{A}3$) and ($\mathcal{A}8$)–($\mathcal{A}11$), where $\phi_{\mathcal{A}}$ changes. For example, in rule ($\mathcal{A}3$), we have that since $\rho \subseteq \rho'$, then $M'(y).P' \in \rho \Rightarrow M'(y).P' \in \rho'$. From this we can conclude that $\mathcal{A}(\![Q]\!]\ \rho\ \phi_{\mathcal{A}} \sqsubseteq \mathcal{A}(\![Q]\!]\ \rho'\ \phi_{\mathcal{A}}$, since the environment resulting from $\mathcal{A}(\![Q]\!]\ \rho\ \phi_{\mathcal{A}}$ will necessarily be a subset of the environment resulting from $\mathcal{A}(\![Q]\!]\ \rho'\ \phi_{\mathcal{A}}$ (i.e. the larger system induces more term substitutions). □

The rule for replication also uses the labelling mechanism to α-convert the

set of bound names and variables of each copy of the replication, $bnv(P)$, as well as its set of tags. This renaming retains the compositionality of the semantics. The rule for conditional processes, ($\mathcal{A}7$), relies on the equality of two untagged terms under $\phi_{\mathcal{A}}$. If in the case that the equality does not hold, a different alternative process is chosen. The rule for tuple splitting, ($\mathcal{A}8$), attempts to split elements of a set of tuples corresponding to the value of $\varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, L)$ of a term, L. The $\phi_{\mathcal{A}}$ environment is updated with the tags of the elements of each tuple. In case no tuples exist in the set, an alternative process is chosen and $\phi_{\mathcal{A}}$ is left unchanged. The rest of the rules, ($\mathcal{A}9$)–($\mathcal{A}11$), deal with cryptographic processes. Again, a process attempts to decipher (verify) a term, L, closed by $\varphi_{\mathcal{A}}(\phi_{\mathcal{A}}, L)$. The tags of the deciphered terms are added to $\phi_{\mathcal{A}}$. Else a different process is chosen without affecting $\phi_{\mathcal{A}}$. Finally, rule ($\mathcal{R}0$) groups all the environments resulting from the interpretation of processes in ρ with the union of environments operation, $\cup_{\phi}$.

We can state the safety of the abstract semantics by the following theorem.

Theorem 3 (Safety of the abstract semantics for the spi calculus)
$(\mathcal{E}[\![P]\!]\ \rho\ \phi_{\mathcal{E}} = (p, \phi'_{\mathcal{E}}))\ \wedge\ (\mathcal{A}[\![P]\!]\ \rho\ \phi_{\mathcal{A}} = \phi'_{\mathcal{A}})\ \wedge$
$(\exists M \in \mathit{Term} : \varphi_{\mathcal{E}}(\phi_{\mathcal{E}}, M) \in \phi_{\mathcal{E}}(x)\ \Rightarrow\ \exists t \in \phi_{\mathcal{A}}(\alpha_{k,k'}(x)) : \mathit{value\_of}(\{t\}) = \{M'\}\ \wedge\ \mathit{untag}(M') = (\forall y \in bnv(M) : M[\alpha_{k,k'}(y)/y]))$

$\Rightarrow\ (\exists M \in \mathit{Term} : \varphi_{\mathcal{E}}(\phi'_{\mathcal{E}}, M) \in \phi'_{\mathcal{E}}(x)\ \Rightarrow\ \exists t \in \phi'_{\mathcal{A}}(\alpha_{k,k'}(x)) : \mathit{value\_of}(\{t\}) = \{M'\}\ \wedge\ \mathit{untag}(M') = (\forall y \in bnv(M) : M[\alpha_{k,k'}(y)/y]))$

Proof. The proof is by induction over the structure of the abstract semantics. □

The theorem states that for any term, M, captured in the non-standard semantics by including its $\varphi_{\mathcal{E}}(\phi'_{\mathcal{E}}, M)$ value in the value of a variable, $\phi'_{\mathcal{E}}(x)$, then that will correspond to capturing a tag, t, in the abstract semantics, by $\phi'_{\mathcal{A}}(\alpha_{k,k'}(x))$. The appropriateness of t is expressed by the ability to obtain an abstract form, $\forall x \in bnv(M) : M[\alpha_{k,k'}(x)/x]$, of the concrete term, M, by evaluating t using *value_of* and untagging the resulting term, M', using *untag*. More concisely, every concrete term, M, captured in the non-standard semantics is captured as the corresponding abstract tag, t, in the abstract semantics.

6 Secrecy and Authenticity

To reason about the secrecy and authenticity properties of a system, it is necessary to give formal definitions of these properties with respect to the result of the static analysis, i.e. $\phi_{\mathcal{A}}$. We start by assuming that S and A are finite chains of secrecy and trust levels, respectively, where $l, l' \in S$ and $a, a' \in A$. A well-defined security policy, controlled by the system administrators,

classifies (sub)processes of a system with their secrecy and trust levels using the S and A chains, and according to the security requirements of the system.

Furthermore, let $\xi_S : (\mathcal{N} \cup \mathcal{V}) \to S$ and $\xi_A : (\mathcal{N} \cup \mathcal{V}) \to A$ be two environments that map the new names and variables of *classified processes*, $\lceil P \rceil^l$ and $\lceil P \rceil^a$, to their secrecy and trust levels, respectively. The null environments are defined as: $\forall x \in \mathcal{N} \cup \mathcal{V} : \xi_{S0}(x) = \bot_S, \xi_{A0}(x) = \bot_A$. To construct a general environment ξ for some classified process, the function $\mathcal{Z}$ is defined over the structure of $\lceil P \rceil^z$ as in Figure 5, where $\xi = \xi_S$ whenever $z \in S$, and $\xi = \xi_A$ whenever $z \in A$.

$$
\begin{array}{lcl}
\mathcal{Z}(\lceil \mathbf{0} \rceil^z)\ \xi & = & \xi \\
\mathcal{Z}(\lceil M(y).P \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi[y \mapsto z] \\
\mathcal{Z}(\lceil \overline{M}\langle N \rangle.P \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi \\
\mathcal{Z}(\lceil (\nu\, a)P \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi[a \mapsto z] \\
\mathcal{Z}(\lceil P \mid Q \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi \cup_\xi \mathcal{Z}(Q)\ \xi \\
\mathcal{Z}(\lceil !P \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi \\
\mathcal{Z}(\lceil \mathit{if}\ M = N\ \mathit{then}\ P\ \mathit{else}\ Q \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi \cup_\xi \mathcal{Z}(Q)\ \xi \\
\mathcal{Z}(\lceil \mathit{let}\ (x_0, \ldots, x_{n-1}) = M\ \mathit{in}\ P\ \mathit{else}\ Q \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi[x_0 \mapsto z, \ldots, x_{n-1} \mapsto z] \cup_\xi \\
 & & \mathcal{Z}(Q)\ \xi[x_0 \mapsto z, \ldots, x_{n-1} \mapsto z] \\
\mathcal{Z}(\lceil \mathit{case}\ L\ \mathit{of}\ \{x\}_N\ \mathit{in}\ P\ \mathit{else}\ Q \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi[x \mapsto z]\ \cup_\xi\ \mathcal{Z}(Q)\ \xi[x \mapsto z] \\
\mathcal{Z}(\lceil \mathit{case}\ L\ \mathit{of}\ \{\!|x|\!\}_N\ \mathit{in}\ P\ \mathit{else}\ Q \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi[x \mapsto z]\ \cup_\xi\ \mathcal{Z}(Q)\ \xi[x \mapsto z] \\
\mathcal{Z}(\lceil \mathit{case}\ L\ \mathit{of}\ [\{x\}]_N\ \mathit{in}\ P\ \mathit{else}\ Q \rceil^z)\ \xi & = & \mathcal{Z}(P)\ \xi[x \mapsto z]\ \cup_\xi\ \mathcal{Z}(Q)\ \xi[x \mapsto z]
\end{array}
$$

Fig. 5. Definition of the $\mathcal{Z}$ function over classified processes.

The rules use the $\cup_\xi$ operation defined as follows:

$$\forall x \in \mathcal{N} \cup \mathcal{V} :\ (\xi_1 \cup_\xi \xi_2)(x) = \xi_1(x) \sqcup \xi_2(x) \tag{13}$$

The use of $\sqcup$ is due to the fact that x can be defined in one environment at most, ξ_1 or ξ_2. One may now define the following secrecy and authenticity threats:

Property 1 (Information Leakage & Authenticity Breach) *Given a process P, the static analysis environment $\phi_{\mathcal{A}} = \mathcal{A}[\![P]\!]_s\ \rho_0\ \phi_{\mathcal{A}0}$, and the environments ξ_S and ξ_A, then the information leakage and authenticity breach threats occur whenever the following conditions hold true:*

$$\exists x \in \mathit{dom}(\phi_{\mathcal{A}}), y \in \phi_{\mathcal{A}}(x) : \xi_S(x) \sqsubseteq \xi_S(y) \qquad \textit{(Information Leakage)}$$
$$\exists x \in \mathit{dom}(\phi_{\mathcal{A}}), y \in \phi_{\mathcal{A}}(x) : \xi_A(x) \sqsupseteq \xi_A(y) \qquad \textit{(Authenticity Breach)}$$

The information leakage property captures instances where names created by high-secrecy-level processes are obtained by processes classified at low secrecy levels. The word *obtained* refers to substitutions occurring as a result of input

actions or the success of cryptographic and tuple-splitting operations. On the other hand, the direction of concern in the authenticity breach property is reversed. Here, the authenticity requirements of a high-trust-level process are breached whenever it inputs a name, or decrypts, verifies or splits a term resulting in a name, which was created by a less trusted process.

7 Conclusion and Future Work

We have presented a static analysis for security properties in cryptographic processes modelled by the spi calculus. The analysis is characterized as being fully denotational; this leads to implementations that are closely related to functional programming. The applicability of the analysis in detecting instances of information leakage and authenticity breaches was demonstrated for a number of cryptographic protocols, with the results shown in Figure 6. All of these protocols were analysed in the presence of an intruder process, I, such that I represents the most general attacker of Dolev-Yao [14]. The analysis is then performed by applying $\mathcal{A}(\llbracket Protocol \rrbracket)\ \{\!|I|\!\}\ \phi_{\mathcal{A}}$, where *Protocol* is the specification of the protocol.

Protocol	**Attack**	**Breach**
Needham-Schroeder public-key	man-in-the-middle	A,S
SPLICE/AS	impersonation attack	A,S
Otway-Rees	impersonation	A,S
Kerberos	none	–
Yahalom	none	–
Woo-Lam one-way authentication	impersonation	A,S
A: Authenticiy, S: Secrecy		

Fig. 6. Results of the analysis of some authentication protocols.

Future work is under way to expand the analysis to be able to use the non-uniformity of the domain of $\phi_{\mathcal{A}}$ in defining security properties, in particular, the freshness property. This non-uniformity also permits the definition of a resource exhaustion property based on a cost-resource relationship, where variables denote the available resources and the captured names, the cost.

References

[1] Martín Abadi. Secrecy by typing in security protocols. In Martín Abadi and Takayasu Ito, editors, *Proceedings of the* 3rd *International Symposium on Theoretical Aspects of Computer Software*, volume 1281 of *Lecture Notes on*

Computer Science, pages 611–638, Sendai, Japan, September 1997. Springer Verlag.

[2] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. In *Proceedings of the* 29th *Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 33–44, Portland, USA, January 2002. ACM Press.

[3] Martín Abadi and Andrew Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the* 4th *ACM Conference on Computer and Communications Security*, pages 36–47, Zurich, Switzerland, April 1997. ACM Press.

[4] Samson Abramsky. A domain equation for bisimulation. *Information and Computation*, 92(2):161–218, June 1991.

[5] Roberto Amadio and Denis Lugiez. On the reachability problem in cryptographic protocols. In Catuscia Palamidessi, editor, *Proceedings of the* 11th *International Conference on Concurrency Theory*, volume 1877 of *Lecture Notes on Computer Science*, pages 380–394, Pennsylvania, USA, August 2000. Springer Verlag.

[6] Benjamin Aziz and Geoff Hamilton. A privacy analysis for the π-calculus: The denotational approach. In *Proceedings of the* 2nd *Workshop on the Specification, Analysis and Validation for Emerging Technologies*, Copenhagen, Denmark, July 2002.

[7] Bruno Blanchet. From secrecy to authenticity in security protocols. In Manuel V. Hermenegildo and German Puebla, editors, *Proceedings of the* 9th *International Symposium in Static Analysis*, volume 2477 of *Lecture Notes on Computer Science*, pages 342–359, Madrid, Spain, September 2002. Springer Verlag.

[8] Chiara Bodei, Pierpaolo Dagano, Flemming Nielson, and Hanne Riis Nielson. Control flow analysis for the π-calculus. In *Proceedings of the* 9th *Conference on Concurrency Theory*, volume 1466 of *Lecture Notes on Computer Science*, pages 84–98, Nice, France, September 1998. Springer Verlag.

[9] Chiara Bodei, Pierpaolo Dagano, Flemming Nielson, and Hanne Riis Nielson. Static analysis of processes for no read-up and no write-down. In *Proceedings of the Conference on Foundations of Software Science and Computation Structures*, volume 1578 of *Lecture Notes on Computer Science*, pages 120–134, Lisbon, Portugal, March 1999. Springer Verlag.

[10] Chiara Bodei, Pierpaolo Dagano, Flemming Nielson, and Hanne Riis Nielson. Static analysis for secrecy and non-interference in networks of processes. In *Proceedings of the* 6th *International Conference in Parallel Computing Technologies*, volume 2127 of *Lecture Notes on Computer Science*, pages 27–41, Novosibirsk, Russia, September 2001. Springer Verlag.

[11] Chiara Bodei, Pierpaolo Dagano, Flemming Nielson, and Hanne Riis Nielson. Static analysis for the π-calculus with applications to security. *Information and Computation*, 168(1):68–92, July 2001.

[12] Michele Boreale. Symbolic trace analysis of cryptographic protocols. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Proceedings of the* 28^{th} *International Colloquium on Automata, Languages and Programming*, volume 2076 of *Lecture Notes on Computer Science*, pages 667–681, Crete, Greece, July 2001. Springer Verlag.

[13] Michele Boreale and Maria Grazia Buscemi. Experimenting with sta: A tool for automatic analysis of security protocols. In *Proceedings of the ACM Symposium on Applied Computing*, pages 281–285, Madrid, Spain, March 2002. ACM Press.

[14] Danny Dolev and A. Yao. On the security of public key protocols. In *Proceedings of the* 22^{nd} *Annual Symposium on Foundations of Computer Science*, pages 350–357, October 1981.

[15] Jérôme Feret. Confidentiality analysis of mobile systems. In *Proceedings of the* 7^{th} *International Static Analysis Symposium*, volume 1824 of *Lecture Notes on Computer Science*, pages 135–154, University of California, Santa Barbara, USA, June 2000. Springer Verlag.

[16] Antti Huima. Efficient infinite-state analysis of security protocols. In *Proceedings of the FLOC 1999 Formal Methods and Security Protocols Workshop*, pages 21–51, Trento, Italy, July 1999.

[17] Fabio Martinelli. Symbolic partial model checking for security analysis. In *Proceedings of the* 2^{nd} *Workshop on the Specification, Analysis and Validation for Emerging Technologies*, Copenhagen, Denmark, July 2002.

[18] Robin Milner, John Parrow, and David Walker. A calculus of mobile processes (parts i & ii). *Information and Computation*, 100(1):1–77, September 1992.

[19] David Monniaux. Abstracting cryptographic protocols with tree automata. In Agostino Cortesi and Gilberto Filé, editors, *Proceedings of the* 6^{th} *International Static Analysis Symposium*, volume 1694 of *Lecture Notes on Computer Science*, pages 149–163, Venice, Italy, September 1999. Springer Verlag.

[20] Ian Stark. A fully abstract domain model for the π-calculus. In *Proceedings of the* 11^{th} *Annual IEEE Symposium on Logic in Computer Science*, pages 36–42, New Brunswick, New Jersey, USA, July 1996. IEEE Computer Society.

[21] Arnaud Venet. Abstract interpretation of the π-calculus. In Mads Dam, editor, *Proceedings of the* 5^{th} *LOMAPS Workshop on Analysis and Verification of Multiple-Agent Languages*, volume 1192 of *Lecture Notes on Computer Science*, pages 51–75, Stockholm, Sweden, June 1996. Springer Verlag.