Static Analysis for Secrecy and Non-Interference in Networks of Processes *

C. Bodei[1] P. Degano[1] F. Nielson[2] H. Riis Nielson[2]

[1] Dipartimento di Informatica, Università di Pisa
Corso Italia 40, I-56125 Pisa, Italy – {chiara,degano}@di.unipi.it
[2] Informatics & Mathematical Modelling, The Technical University of Denmark
Richard Petersens Plads bld 321,
DK-2800 Kongens Lyngby, Denmark – {nielson,riis}@imm.dtu.dk

Abstract. We introduce the νSPI-calculus that strengthens the notion of "perfect symmetric cryptography" of the spi-calculus by taking time into account. This involves defining an operational semantics, defining a control flow analysis (CFA) in the form of a flow logic, and proving semantic correctness. Our first result is that secrecy in the sense of Dolev-Yao can be expressed in terms of the CFA. Our second result is that also non-interference in the sense of Abadi can be expressed in terms of the CFA; unlike Abadi we find the non-interference property to be an extension of the Dolev-Yao property.

1 Introduction

The widespread usage of distributed systems and networks has furnished a great number of interesting scenarios in which security plays a significant role. Well established and well founded process algebraic theories offer a fertile ground to express distributed and concurrent systems in pure form, and to study their properties. In particular, protocols and security protocols can be conveniently written in the spi-calculus [1,5], an extension of the π-calculus with primitives for encryption and decryption. These are based on symmetric cryptography that is assumed to be perfect; as usual this is formulated in an algebraic manner: that encryption and decryption are inverses of one another. This facilitates expressing cryptographic protocols and one can reason on them exploiting the rich variety of techniques and tools, developed for calculi of computation and programming languages.

As observed in [1] the notion of perfect encryption embodied in the spi-calculus is too weak to guard against certain attacks based on comparing ciphertexts. As an example, consider a process that first communicates *true* encrypted under some key, then *false* encrypted under the same

* The first two authors have been partially supported by the Progetti MURST TOSCA and AI, TS & CFA.

key, and finally a secret boolean b encrypted under the same key; then confidentiality of b is not guaranteed because its value can be obtained by comparison of ciphertexts. To guard against this form of attack a type system is developed that enforces placement of so-called confounders in all encryptions [1].

By contrast our approach is based on the observation that many symmetric cryptosystems, e.g. DES operating in a suitable chaining mode, are *always* initialised with a random initialisation vector, thereby dealing with a notion of confounders dynamically. To be closer to real implementations, we therefore use a slight modification of the spi-calculus, called νSPI. We semantically model the randomization of the encryption function, by adding to each plaintext M a new and fresh value r, making any two encryptions of M different from each other. In other words, we obtain a notion of history dependent cryptography. Recent and independent developments along a similar line of thought may be found in [20,3]. Indeed, it seems unlikely that any approach only based on algebraic identities (and consideration of the free theory generated) will be able to mimic our semantics-based development.

In preparation for the applications to security we then develop in Section 3 a Control Flow Analysis (CFA) in the form of a Flow Logic [21]. Its specification is in line with previous developments for the π-calculus [8,7] and the same goes for its semantic correctness by means of a subject-reduction result and the existence of least solutions. However, the techniques needed for obtaining the least solution in polynomial time (actually $O(n^3)$) are more involved than before because the specification operates over an infinite universe [23,25].

Our first application to security in Section 4 is to show that CFA helps in showing that a protocol has no *direct* flows that violate confidentiality. The static condition, called *confinement*, merely inspects the CFA information to make sure that only public messages flow along public channels. The dynamic condition, called *carefulness*, then guarantees for all executions that no secrets are output on public channels. Correctness of the static analysis then follows from the subject-reduction result. This notion of security essentially says that no attacker, not even an active saboteur, can decipher a secret message sent on the network; actually, we show that if a process is careful then it preserves the secrecy of messages according to the notion originally advocated by Dolev and Yao [16,2]. A similar result has independently been achieved by [4] using a type system on a slightly different calculus.

Our second application to security in Section 5 is to show that CFA also helps in checking that a protocol has no *indirect* flows that violate confidentiality. In the formulation of Abadi [1] the static condition is formulated using a type system and the dynamic condition then compares executions using testing equivalence [13,10]. In our formulation the static condition, called *invariance*, is formulated as yet another check on the CFA information, and we retain the dynamic notion, which we prefer to call *message independence*. (Both our and Abadi's dynamic notions say that the active attacker cannot detect whatsoever information about the message sent, even by inspecting and changing the behaviour of a secure protocol; but this does not quite amount to non-interference in the sense of [17].) While inspired by [24, Section 5.3] this represents the first use of CFA for establishing testing equivalence for processes allowing cryptography. In our approach, confinement is a prerequisite for invariance, thus suggesting a deeper connection between Dolev-Yao and Non-Interference than reported by Abadi [2].

A more widely used alternative approach for calculi of computation and security is based on Type Systems [1,32,31,29,30,19,27,28,12,11]. Here security requirements are seen as static information about the objects of a system [1,32,12,11]. Our approach builds on the more "classical" approaches to static analysis and thus links up with the pioneering approach taken in the very early studies by Denning [14,15]; it also features a very good computational complexity.

Because of lack of space, we dispense with the proofs, which often use techniques similar to those of [7] and that can in part be found in the extended version of the paper.

2 History Dependent Cryptography

Syntax. We define the νSPI-calculus by modifying the spi-calculus [5] (we consider here its monadic form, for simplicity) so that the encryption primitive becomes history dependent. Roughly, this amounts to saying that every time we encrypt a message we get a different ciphertext, even if the message is the same and the key is the same. We do so by changing the semantics: each encryption necessarily generates a fresh confounder that is part of the message (corresponding to the random initialisation vector used when running DES in an appropriate chaining mode); therefore our analysis does not need to enforce this property (unlike the type system in [1]). This naturally leads to modifying the semantics to evaluate a message before it is actually sent; in other words we define a call-by-value programming language. — To aid the intuitions of the reader familiar with the spi-calculus we have also changed the syntax by letting each encryption contain a construct for generating the confounder; however this syntactic change is in no way essential (quite unlike the semantic change).

The formulation of the CFA of the νSPI-calculus, in Section 3, is facilitated by making a few assumptions. Mainly, we slightly extend the standard syntax by mechanically assigning "labels" to the occurrences of terms; these are nothing but explicit notations for program points and in an actual implementation can be taken to be pointers into the syntax tree. Furthermore, to deal with the α-renaming of bound names in a simple and "implicit" way, we assume that names are "stable", i.e. that each name a is the canonical representative for its class of α-convertible names. To this aim, we define the set of names $\mathcal{N}'$ as the disjoint union of sets of indexed names, $\mathcal{N}' = \uplus_{a\in\mathcal{N}}\{a, a_0, a_1, \cdots\}$, and we write $\lfloor a_i \rfloor = a$ for the canonical name a associated to each actual name a_i. Then we restrict α-conversion so that we only allow a name a_i to be substituted for the name b_j, if $\lfloor a_i \rfloor = \lfloor b_j \rfloor$. In this way, we statically maintain the identity of names that may be lost by freely applying α-conversions. (For a more "explicit" approach using marker environments see [8,6].)

Definition 1. Let $l, l', l_i \in \mathcal{L}$ be *labels*, $n, m, \cdots \in \mathcal{N}'$ be *names* and $x, y, \cdots \in \mathcal{V}$ be *variables*. Then (labelled) *expressions*, $E, V \in \mathcal{E}$, (unlabelled) *terms*, $M, N \in \mathcal{M}$, *values* $w, v \in Val'$, and *processes*, $P, P_i, Q, R, \cdots \in \mathcal{P}$, are all built according to the following syntax:

$$
\begin{aligned}
E, V &::= M^l \\
M, N &::= n \mid x \mid (E, E') \mid 0 \mid suc(E) \mid \{E_1, \cdots, E_k, (\nu r)\ r\}_{E_0} \mid w \\
w, v &::= n \mid pair(w, w') \mid 0 \mid suc(w) \mid enc\{w_1, \cdots, w_k, r\}_{w_0} \\
P, Q &::= \mathbf{0} \mid \overline{E}\langle V\rangle.P \mid E(x).P \mid P|P \mid (\nu n)P \mid \\
&\qquad [E\ is\ V]P \mid !P \mid let\ (x, y) = E\ in\ P \mid \\
&\qquad case\ E\ of\ 0 : P\ suc(x) : Q \mid case\ E\ of\ \{x_1, \cdots, x_k\}_V\ in\ P
\end{aligned}
$$

Here $E(x).P$ binds the variable x in P, while $(\nu n)P$ binds the name n in P. We dispense with defining the standard notions of free and bound names (*fn* and resp. *bn*) and of free and bound variables *fv* (resp. *bv*). We often omit the trailing $\mathbf{0}$ and write $\tilde{\cdot}$ to denote tuples of objects.

The νSPI-calculus slightly extends the spi-calculus, that in turn extends the π-calculus (with which we assume the reader to be familiar) with more structured terms (numbers, pairs and encryptions) and process constructs dealing with them. Moreover, our term $\{E_1, \cdots, E_k, (\nu r)\ r\}_{E_0}$

represents the unevaluated encryption of $E_1, \cdots, E_k$ under the symmetric key E_0. Its evaluation results in the actual value $enc\{w_1, \cdots, w_k, r\}_{w_0}$, where w_i is the value of E_i and the restriction (νr) will make sure that the confounder (or initialisation vector) r is fresh (see below). The process $case\ E\ of\ \{x_1, \cdots, x_k\}_V\ in\ P$ attempts to decrypt E with the key V: if E is on the form $\{E_1, \cdots, E_k\}_V$ then the process behaves as $P[\tilde{E_i}/\tilde{x_i}]$, otherwise the process is stuck. Similarly, $let\ (x, y) = E\ in\ P$ attempts to split the pair E and $case\ E\ of\ 0 : P\ suc(x) : Q$ tries to establish if E is either 0 or a successor of some term.

Note that, unlike the π-calculus, names and variables are considered distinct. Finally we extend $\lfloor \cdots \rfloor$ to operate on values by the straightforward structural definition. We write Val for the set of canonical values, i.e. those values v such that $\lfloor v \rfloor = v$.

Entities are considered equal whenever they are α-convertible; so $P = Q$ means that P is α-convertible to Q. Substitution of terms, $\cdots[M/x]$, is standard; substitution of expressions, $\cdots[E/x]$, really denotes substitution of terms, so it preserves labels, hence $x^{l_x}[M^l/x]$ is M^{l_x}; finally, substitution of restricted values, $\cdots[(\nu r)w/x]$, acts as substitution of values, $\cdots[w/x]$, with the restriction moved out of any expressions, e.g. $\overline{n}\langle x\rangle[(\nu r)r/x] = (\nu r)\overline{n}\langle r\rangle$. We shall write $P \equiv Q$ to mean that P and Q are equal except that restriction operators may be placed differently as long as their effect is the same, e.g. $(\nu r)\overline{n}\langle s\rangle.\overline{m}\langle r\rangle \equiv \overline{n}\langle s\rangle.(\nu r)\overline{m}\langle r\rangle$.

Semantics. The semantics is built out of three relations: the evaluation, the reduction and the commitment relations. In all of them we will apply our disciplined α-conversion when needed. They all operate on *closed* entities, i.e. entities without free variables.

The evaluation relation $\gg$ in the upper part of Tab. 1 reduces an expression E to a value w. Although it is not part of the standard semantics of the spi-calculus, it is quite natural from a programming language point of view, and it is crucial in specifying history dependent encryption. As it will be clear soon, a term has to be fully evaluated before it is used either in a reduction (e.g. when matching or a decryption takes place) or as a message. So to speak, our variant of the calculus is a call-by-value one. The central rule is that for encryption: the restriction (νr) acting on the confounder r is pushed in the outermost position, so that every other name in the process is and will be different from r. Two different occurrences, M^l and $M^{l'}$ (with $l \neq l'$), of a term containing an unevaluated encryption operator, never evaluate to the same values, $(\nu\tilde{r})w$ and

$$1.\ \ n^l \gg n \qquad 2.\ \ 0^l \gg 0 \qquad 3.\ \ \frac{E_i \gg (\nu\tilde{r_i})\ w_i,\ \ i=1,2, \tilde{r_1}\tilde{r_2}\ \text{w.o. duplicates}}{(E_1,E_2) \gg (\nu\tilde{r_1}\tilde{r_2})\ pair(w_1,w_2)}$$

$$4.\ \ \frac{E \gg (\nu\tilde{r})\ w}{suc(E) \gg (\nu\tilde{r})\ suc(w)} \qquad 5.\ \ \frac{E_i \gg (\nu\tilde{r_i})\ w_i,\ \ i=0\cdots k, \tilde{r_1}\cdots\tilde{r_k}\tilde{r_0}r\ \text{w.o. duplicates}}{\{E_1,\cdots,E_k,(\nu r)\ r\}_{E_0} \gg (\nu\tilde{r_1}\cdots\tilde{r_k}\tilde{r_0}r)\ enc\{w_1,\cdots,w_k,r\}_{w_0}}$$

$$Match:\ \frac{E_i \gg (\nu\tilde{r_i})\ w_i,\ \ i=1,2}{[E_1\ is\ E_2]P > (\nu\tilde{r_1}\tilde{r_2})\ P} \qquad (\nu\tilde{r_1}\tilde{r_2})\ w_1 = (\nu\tilde{r_1}\tilde{r_2})\ w_2;\ \tilde{r_1}\tilde{r_2}fn(P)\ \text{w.o. duplicates}$$

$$Let:\ \frac{E \gg (\nu\tilde{r})\ pair(w_1,w_2)}{let\ (x,y)=E\ in\ P > (\nu\tilde{r})\ P[w_1/x, w_2/y]} \qquad \tilde{r}fn(P)\ \text{w.o. duplicates}$$

$$Zero:\ \frac{E \gg 0}{case\ E\ of\ 0: P\ suc(x): Q > P} \qquad Rep:\ \frac{}{!P > P\ |!P}$$

$$Suc:\ \frac{E \gg (\nu\tilde{r})\ suc(w)}{case\ E\ of\ 0: P\ suc(x): Q > (\nu\tilde{r})\ Q[w/x]} \qquad \tilde{r}fn(Q)\ \text{w.o. duplicates}$$

$$Enc:\ \frac{E \gg (\nu\tilde{r_0})\ enc\{w_1,\cdots,w_k,s\}_{w_0}, V \gg (\nu\tilde{r_1})\ v}{case\ E\ of\ \{x_1,\cdots,x_k\}_V\ in\ P > (\nu\tilde{r_0})\ P[w_1/x_1,\cdots,w_k/x_k]}$$

$$\tilde{r_0}\tilde{r_1}fn(P)\ \text{w.o. duplicates}; (\nu\tilde{r_0}\tilde{r_1})\ w_0 = (\nu\tilde{r_0}\tilde{r_1})\ v$$

$$In:\ \ m(x).P \xrightarrow{m} (x)P \qquad Out:\ \frac{M^l \gg (\nu\tilde{r})\ w}{\overline{m}\langle M^l\rangle.P \xrightarrow{\overline{m}} (\nu\tilde{r})\langle w^l\rangle P} \quad \tilde{r}fn(P)\ \text{w.o. duplicates}$$

$$Inter:\ \frac{P \xrightarrow{m} F \quad Q \xrightarrow{\overline{m}} C}{P|Q \xrightarrow{\tau} F@C} \qquad Par:\ \frac{P \xrightarrow{\alpha} A}{P|Q \xrightarrow{\alpha} A|Q}$$

$$Red:\ \frac{P > Q \quad Q \xrightarrow{\alpha} A}{P \xrightarrow{\alpha} A} \qquad Res:\ \frac{P \xrightarrow{\alpha} A}{(\nu m)P \xrightarrow{\alpha} (\nu m)A} \quad \alpha \notin \{m, \overline{m}\}$$

$$Congr:\ \frac{P \equiv Q \quad Q \xrightarrow{\alpha} A \quad A \equiv B}{P \xrightarrow{\alpha} B}$$

Table 1. The semantics of νSPI: the evaluation relation, $\gg$; the reduction relation, $>$; and the commitment relation, $\xrightarrow{\alpha}$ (without symmetric rules).

$(\nu\tilde{r}')w'$, where $w = w'$ and the concatenation of vectors of names $\tilde{r}$ and $\tilde{r}'$ has no name occurring more than once (abbreviated $\widetilde{\tilde{r}r'}$ w.o. duplicates).

This is crucial for matching; indeed, $[\{0,(\nu r)r\}^l_w\ is\ \{0,(\nu r)r\}^{l'}_w]P$ never reduces to P, because every time we encrypt 0, even if under exactly the same evaluated key, we get a different value. The *reduction* rules in the central part of Tab. 1 govern the evaluation of guards. They only differ from the standard spi-calculus ones because, as mentioned above, the terms occurring in P that drive the reduction $P > Q$ have to be evaluated. This step may introduce some new restricted names, in particular when the terms include an encryption to be evaluated. This restriction is placed around Q, so as to make sure that the new names are indeed fresh and that there will be no captures. The side condition "$\tilde{r_1}\tilde{r_2}fn(P)$ w.o. duplicates" in the rule *Match* ensures that the scopes are preserved even though the restrictions are placed differently; similarly for the other side conditions. Finally, note that after a decryption, the process P has no access to the confounder s.

To define the *commitment relation*, we need the usual notions of abstraction $F = (x)P$ and of concretion $C = (\nu\tilde{n})\langle w^l\rangle Q$ (assuming that $(x)P \mid Q = (x)(P \mid Q)$, if $x \notin fv(Q)$, that $(\nu\tilde{n})\langle w^l\rangle Q \mid R = (\nu\tilde{n})\langle w^l\rangle(Q \mid R)$, if $\tilde{n} \cap fn(R) = \emptyset$, and the symmetric rules). Note that the message sent must be an actual value. The interaction $F@C$ (and symmetrically for $C@F$) is then the following, provided that $\{\tilde{n}\} \cap fn(P) = \emptyset$:

$$F@C = (\nu\tilde{n})(P[w^l/x] \mid Q)$$

The structural operational semantic rules for the commitment relation are in the lower part of Tab. 1; they are standard apart from rule *Out* that requires the evaluation of the message sent, and introduces the new restricted names $\tilde{r}$ (possibly causing also some α-conversions).

3 Control Flow Analysis (CFA)

Writing $\widehat{Val} = \wp(Val)$ the result of our analysis for a process P is a triple (ρ, κ, ζ), where:

- $\rho : \mathcal{V} \to \widehat{Val}$ is the *abstract environment* that associates variables with the values that they can be bound to; more precisely, $\rho(x)$ must include the set of values that x could assume at run-time.
- $\kappa : \mathcal{N} \to \widehat{Val}$ is the *abstract channel environment* that associates canonical names with the values that can be communicated over them; more precisely, $\kappa(n)$ must include the set of values that can be communicated over the channel n.
- $\zeta : \mathcal{L} \to \widehat{Val}$ is the *abstract cache* that associates labels with the values that can arise there; more precisely $\zeta(l)$ must include the set of the possible actual values of the term labelled l.

Acceptability. To define the *acceptability* of a proposed estimate (ρ, κ, ζ) we state a set of clauses operating upon flow logic judgments on the forms $(\rho, \kappa, \zeta) \models M$ and $(\rho, \kappa, \zeta) \models P$.

The analysis of expressions and of processes are in Tab. 2. Our rules make use of canonical names and values and of the following abbreviations, where $W \in \widehat{Val}$:

- $\textsc{suc}(W)$ for $\{suc(w) | w \in W\}$;
- $\textsc{pair}(W, W')$ for $\{pair(w, w') | w \in W, w' \in W'\}$;
- $\textsc{enc}\{W_1, \cdots, W_k, r\}_{W_0}$ for $\{enc\{w_1, \cdots, w_k, r\}_{w_0} | \forall i : \ w_i \in W_i\}$.

All the rules for validating a compound term or a process require that the components are validated. The rules for an expression M^l demand that $\zeta(l)$ contains all the values associated with its components. Moreover, the rule for output requires that the set of values associated with the message N can be passed on each channel associated with M. Symmetrically, the rule for input requires that each value passing along M is contained in the set of possible values of x, i.e. $\rho(x)$. The last three rules check the i^{th} sub-components of each value associated with the expression to split, compare or decrypt. Each sub-component must be contained in the corresponding $\rho(x_i)$.

Finally, the analysis is extended to concretions and abstractions in the last part of Tab. 2.

Correctness. To establish the semantic correctness of our analysis we establish subject-reduction results for the evaluation, the reduction and the commitment relations of the previous section.

Theorem 2 (Subject Reduction for $\gg, >$ and $\xrightarrow{\alpha}$).
Let $M^l \in \mathcal{E}$; if $(\rho, \kappa, \zeta) \models M^l$ and $M^l \gg (\nu \tilde{r})\ w$ then $\lfloor w \rfloor \in \zeta(l)$.

Let P be a closed process such that $(\rho, \kappa, \zeta) \models P$;

(1) if $P > Q$ then $(\rho, \kappa, \zeta) \models Q$.
(2) if $P \xrightarrow{\tau} Q$ then $(\rho, \kappa, \zeta) \models Q$;
(3) if $P \xrightarrow{\overline{m}} (\nu \tilde{n})\langle w^l \rangle Q$ then $(\rho, \kappa, \zeta) \models (\nu \tilde{n})\langle w^l \rangle Q$ and $\zeta(l) \subseteq \kappa(\lfloor m \rfloor)$;
(4) if $P \xrightarrow{m} (\nu \tilde{n})(x)Q$ then $(\rho, \kappa, \zeta) \models (\nu \tilde{n})(x)Q$ and $\kappa(\lfloor m \rfloor) \subseteq \rho(x)$.

Existence. So far we have only considered a procedure for validating whether or not a proposed estimate (ρ, κ, ζ) is in fact acceptable. Now, we show that there always exists a least choice of (ρ, κ, ζ) acceptable in the manner of Tab. 2.

It is quite standard to partially order the set of proposed estimates by setting $(\rho, \kappa, \zeta) \sqsubseteq (\rho', \kappa', \zeta')$ if and only if $\forall x \in \mathcal{V} : \rho(x) \subseteq \rho'(x)$, $\forall n \in \mathcal{N} : \kappa(n) \subseteq \kappa'(n)$ and $\forall l \in \mathcal{L} : \zeta(l) \subseteq \zeta'(l)$. Furthermore, a *Moore family* $\mathcal{I}$ is a set that contains $\sqcap \mathcal{J}$ for all $\mathcal{J} \subseteq \mathcal{I}$, where $\sqcap$ is the greatest lower bound operator (defined pointwise). One important property of a

$(\rho,\kappa,\zeta) \models n^l$	iff	$\{\lfloor n \rfloor\} \subseteq \zeta(l)$
$(\rho,\kappa,\zeta) \models x^l$	iff	$\rho(x) \subseteq \zeta(l)$
$(\rho,\kappa,\zeta) \models (M^{l_1}, N^{l_2})^l$	iff	$(\rho,\kappa,\zeta) \models M^{l_1} \wedge (\rho,\kappa,\zeta) \models N^{l_2} \wedge$ $\mathrm{PAIR}(\zeta(l_1),\zeta(l_2)) \subseteq \zeta(l)$
$(\rho,\kappa,\zeta) \models 0^l$	iff	$\{0\} \subseteq \zeta(l)$
$(\rho,\kappa,\zeta) \models suc(M^{l_M})^l$	iff	$(\rho,\kappa,\zeta) \models M^{l_M} \wedge \mathrm{SUC}(\zeta(l_M)) \subseteq \zeta(l)$
$(\rho,\kappa,\zeta) \models \{M_1^{l_1},\cdots,M_k^{l_k},(\nu r)\ r\}^l_{M_0^{l_0}}$	iff	$\forall i = 0,..,k : (\rho,\kappa,\zeta) \models M_i^{l_i} \wedge$ $\mathrm{ENC}\{\zeta(l_1),\cdots,\zeta(l_k),\{\lfloor r \rfloor\}\}_{\zeta(l_0)} \subseteq \zeta(l)$
$(\rho,\kappa,\zeta) \models w^l$	iff	$\{\lfloor w \rfloor\} \subseteq \zeta(l)$
$(\rho,\kappa,\zeta) \models \mathbf{0}$	iff	$true$
$(\rho,\kappa,\zeta) \models \overline{M^l}\langle N^{l'}\rangle.P$	iff	$(\rho,\kappa,\zeta) \models M^l \wedge (\rho,\kappa,\zeta) \models N^{l'} \wedge$ $(\rho,\kappa,\zeta) \models P \wedge \forall n \in \zeta(l) : \zeta(l') \subseteq \kappa(n)$
$(\rho,\kappa,\zeta) \models M^l(x).P$	iff	$(\rho,\kappa,\zeta) \models M^l \wedge (\rho,\kappa,\zeta) \models P \wedge \forall n \in \zeta(l) : \kappa(n) \subseteq \rho(x)$
$(\rho,\kappa,\zeta) \models P_1 \vert P_2$	iff	$(\rho,\kappa,\zeta) \models P_1 \wedge (\rho,\kappa,\zeta) \models P_2$
$(\rho,\kappa,\zeta) \models (\nu n)P$	iff	$(\rho,\kappa,\zeta) \models P$
$(\rho,\kappa,\zeta) \models\ !P$	iff	$(\rho,\kappa,\zeta) \models P$
$(\rho,\kappa,\zeta) \models [M^l\ is\ N^{l'}]P$	iff	$(\rho,\kappa,\zeta) \models M^l \wedge (\rho,\kappa,\zeta) \models N^{l'} \wedge (\rho,\kappa,\zeta) \models P$
$(\rho,\kappa,\zeta) \models let\ (x_1,x_2) = M^l in\ P$	iff	$(\rho,\kappa,\zeta) \models M^l \wedge (\rho,\kappa,\zeta) \models P \wedge$ $\forall pair(v,w) \in \zeta(l) : \{v\} \subseteq \rho(x_1) \wedge \{w\} \subseteq \rho(x_2)$
$(\rho,\kappa,\zeta) \models case\ M^l\ of\ 0 : P\ suc(x) : Q$	iff	$(\rho,\kappa,\zeta) \models M^l \wedge (\rho,\kappa,\zeta) \models P \wedge$ $(\rho,\kappa,\zeta) \models Q \wedge \forall\ suc(w) \in \zeta(l) : \{w\} \subseteq \rho(x)$
$(\rho,\kappa,\zeta) \models case\ M^l\ of\ \{x_1,\cdots,x_k\}_{N^{l'}}\ in\ P$	iff	$(\rho,\kappa,\zeta) \models M^l \wedge (\rho,\kappa,\zeta) \models N^{l'} \wedge (\rho,\kappa,\zeta) \models P \wedge$ $\forall\ enc\{w_1,\cdots,w_m,r\}_w \in \zeta(l)$: if $m = k \wedge w \in \zeta(l')$ then $\forall i = 1,\cdots,k :\ \{w_i\} \subseteq \rho(x_i)$
$(\rho,\kappa,\zeta) \models (\nu\tilde{m})\langle w^l \rangle P$	iff	$(\rho,\kappa,\zeta) \models P\ \wedge\ (\rho,\kappa,\zeta) \models w^l$
$(\rho,\kappa,\zeta) \models (x)P$	iff	$(\rho,\kappa,\zeta) \models P$

Table 2. CFA for expressions, processes, concretions and abstractions.

Moore family is that it always contains a least element. The following theorem then guarantees that there is always a least estimate to the specification in Tab. 2. Its statement concerns processes and the proof relies on analogous statements for expressions; this also holds for some of the following results.

Theorem 3. *The set* $\{(\rho,\kappa,\zeta) \mid (\rho,\kappa,\zeta) \models P\}$ *is a Moore family for all* P.

Polynomial time construction. In [7] we developed a polynomial time procedure for calculating least solutions. This development does *not* immediately carry over because we now operate over an *infinite* universe of values due to the expressions present in the calculus. Therefore the specification in Tab. 2 needs to be interpreted as defining a regular tree grammar whose least solution can be computed in polynomial time. A recent result [25] in fact shows that the time complexity can be reduced to cubic time.

4 CFA and Dolev-Yao Secrecy

In this section, we extend to the νSPI-calculus the static property of confinement, studied in [8] for the π-calculus. We then show that our notion corresponds to that of Dolev and Yao [16,9,26,2].

The dynamic notion. The names, $\mathcal{N}'$, are partitioned into the public ones, $\mathcal{P}$, and the secret ones, $\mathcal{S}$, in such a way that $n \in \mathcal{S}$ iff $\mathcal{N}_n \subseteq \mathcal{S}$. We demand that the free names of processes under analysis are all public; it follows that the secret names either do not occur at all or are restricted within a process. This partition is used as a basis for partitioning (also non canonical) values according to the two kinds S (for secret) and P (for public). The intention is that a single "drop" of secret makes the entire value secret except for what is encrypted with a secret key, which is anyway public. We do not consider confounders as they are discarded by decryptions.

Definition 4. The operator $kind : Val' \to \{\text{S}, \text{P}\}$ is defined as

- $kind(n) = \begin{cases} \text{S} & \text{if } n \in \mathcal{S} \\ \text{P} & \text{if } n \in \mathcal{P} \end{cases}$
- $kind(0) = \text{P}$; – $kind(suc(w)) = kind(w)$;
- $kind(pair(w, w')) = \begin{cases} \text{S} & \text{if } \ (kind(w) = \text{S} \ \vee \ kind(w') = \text{S}) \\ \text{P} & \text{otherwise;} \end{cases}$
- $kind(enc\{w_1, \cdots, w_k, r\}_{w_0}) = \begin{cases} \text{P} & \text{if } kind(w_0) = \text{S} \ \vee k = 0 \\ kind(\{w_1, \cdots, w_k\}) & \text{otherwise,} \end{cases}$

where, by abuse of notation, $kind(W) = \begin{cases} \text{S} \text{ if } \exists w \in W : kind(w) = \text{S} \\ \text{P} \text{ if } \forall w \in W : kind(w) = \text{P}. \end{cases}$
We shall write ValP for the set of canonical values of kind P.

To define the dynamic notion of secrecy we write $P \to^* Q$ to mean that $P \xrightarrow{\tau} \cdots \xrightarrow{\tau} Q$. Then carefulness means that no secrets are sent in clear on public channels:

Definition 5. A process P is *careful* w.r.t. $\mathcal{S}$ iff whenever $P \rightarrow^{*} P' \xrightarrow{\alpha} P''$, with the last step deduced with the premise $R \xrightarrow{\overline{m}} (\nu \tilde{r})\langle w^{l}\rangle R'$, then $m \in \mathcal{P}$ implies $kind(w) = \textsc{p}$.

The static notion. We now define the confinement property for the νSPI-calculus. It predicts at compile time that a process is careful. A check suffices on the κ component of a solution: the set of values that can flow on each public name n must be all the ones that have kind $\textsc{p}$.

Definition 6. A process P is *confined* w.r.t. $\mathcal{S}$ and (ρ, κ, ζ) if and only if $(\rho, \kappa, \zeta) \models P$ and $\forall n \in \mathcal{P} : \kappa(n) = \mathsf{ValP}$.

The subject reduction theorem extends trivially to confined processes thereby paving the way for showing that the static notion implies the dynamic one.

Theorem 7. *If P is confined w.r.t. $\mathcal{S}$ then P is careful w.r.t. $\mathcal{S}$.*

Example 8. We consider here an adaptation of the Wide Mouthed Frog key exchange protocol as presented in [5]. The two processes A and B share keys K_{AS} and K_{BS} with a trusted server S. In order to establish a secure channel with B, A sends a fresh key K_{AB} encrypted with K_{AS} to S. Then, S decrypts the key and forwards it to B, this time encrypted with K_{BS}. Now A can send a message M encrypted with K_{AB} to B (for simplicity, M is a name). The analysis guarantees that M is kept secret. The protocol and its specification are as follows:

$$\begin{array}{lll} \text{Message 1} & A \rightarrow S & : \{K_{AB}\}_{K_{AS}} \\ \text{Message 2} & S \rightarrow B & : \{K_{AB}\}_{K_{BS}} \\ \text{Message 3} & A \rightarrow B & : \{M\}_{K_{AB}} \end{array}$$

$$P = (\nu K_{AS})(\nu K_{BS})(\ (A|B) \mid S)$$

$$A = (\nu K_{AB})(\overline{c^{l_{c1}}_{AS}}\langle\{K^{l_{k3}}_{AB}, (\nu r_1) r_1\}_{K^{l_{k1}}_{AS}}\rangle.\overline{c^{l_{c3}}_{AB}}\langle\{M^{l_M}, (\nu r_2) r_2\}_{K^{l_{k3}}_{AB}}\rangle)$$

$$S = c^{l_{c1}}_{AS}(x).case\ x^{l_x}\ of\ \{s\}_{K^{l_{k1}}_{AS}}\ in\ \overline{c_{BS}}\langle\{s^{l_s}, (\nu r_3) r_3\}_{K^{l_{k2}}_{BS}}\rangle$$

$$B = c^{l_{c2}}_{BS}(t).case\ t^{l_t}\ of\ \{y\}_{K^{l_{k2}}_{BS}}\ in\ c^{l_{c3}}_{AB}(z).case\ z^{l_z}\ of\ \{q\}_{y^{l_y}}\ in\ B'(q)$$

Let $\mathcal{S} = \{K_{AS}, K_{BS}, K_{AB}, M\}$ and $\mathcal{P} = \{c_{AS}, c_{BS}, c_{AB}\}$; the relevant part of an estimate for P (disregarding $B'(q)$) is:

$$\rho(bv) = \begin{cases} \mathsf{ValP} & \text{if } bv \in \{x, s, t, y, z, q\} \\ \emptyset & \text{otherwise} \end{cases} \qquad \kappa(c) = \begin{cases} \mathsf{ValP} & \text{if } c \in \{c_{AS}, c_{BS}, c_{AB}\} \\ \emptyset & \text{otherwise} \end{cases}$$

Moreover, $\zeta(l_{bv}) = \rho(bv)$ for $bv \in \{x, s, t, y, z, q\}$ and $\zeta(l) = \{n\}$. for all the names n^{l} occurring in P. It is now easy to check that P is confined, hence the secrecy of M is guaranteed. ■

The formulation of Dolev and Yao. We now show that our notion of confinement enforces secrecy in the sense of Dolev and Yao [16,9,26,2]. Its inductive definition simulates the placement of a process P in a hostile environment that initially has some public knowledge, and thus knows all the values computable from it. Then, the environment may increase its knowledge by communicating with P. The secrecy requirement is that a message M is never revealed by P if the environment cannot reconstruct M from the initial knowledge and the knowledge it has acquired by interacting with the process P.

In our case the initial knowledge is given by the numbers and by all the names that are not considered secret, among which those free in the process under consideration. In other words, we are interested in keeping secrets of honest parties, only. The values whose secrecy should be preserved are composed of at least a secret name, except for secret terms, when encrypted (and therefore protected) under a secret key.

We first make precise which messages are computable from a given set of canonical messages $W \subseteq Val$. The function $C : \widehat{Val} \to \widehat{Val}$ is specified as the closure operator (meaning that C is idempotent and extensive: $C(C(W)) = C(W) \supseteq W$) associated with the following inductive definition (where "iff" is short for a rule with "if" and one with "only if"):

- $0 \in C(W)$; $\quad - W \subseteq C(W)$; $\quad - w \in C(W)$ iff $suc(w) \in C(W)$;
- $pair(w, w') \in C(W)$ iff $w \in C(W)$ and $w' \in C(W)$;
- if $\forall i : w_i \in C(W)$ then $\forall r \in W : enc\{w_1, \cdots, w_k, r\}_{w_0} \in C(W)$;
- if $enc\{w_1, \cdots, w_k, r\}_{w_0} \in C(W)$, $w_0 \in C(W)$ then $w_1, \cdots, w_k \in C(W)$.

The following relation $\mathcal{R}$ (or $\mathcal{R}_{K_0,P_0}$ to be pedantic) specifies how the environment, which knows a set of names K_0, can acquire some additional knowledge by interacting with a process P_0:

- $\mathcal{R}(P_0, C(K_0))$;
- if $\mathcal{R}(P, W)$ and $P \xrightarrow{\tau} Q$ then $\mathcal{R}(Q, W)$;
- if $\mathcal{R}(P, W)$, $P \xrightarrow{m} (x)Q$, $\lfloor m \rfloor \in W$ and $\lfloor w \rfloor \in W$ then $\mathcal{R}(Q[w/x], W)$;
- if $\mathcal{R}(P, W)$, $P \xrightarrow{\overline{m}} (\nu\tilde{n})\langle w^l \rangle Q$ and $\lfloor m \rfloor \in W$ then $\mathcal{R}((\nu\tilde{n})Q,$ $C(W \cup \{\lfloor w \rfloor\}))$.

The notion of secrecy put forward by Dolev and Yao can now be phrased as follows. (Recall that $fn(P_0) \subseteq \mathcal{P}$; P_0 is closed; and that the names $\mathcal{N}'$ are partitioned in $\mathcal{S}$ and $\mathcal{P}$.)

Definition 9. The process P_0 *may reveal* M from $K_0 \subseteq \mathcal{P}$, with $M \gg (\nu\tilde{r})w$ and $kind(w) = s$, if $\exists P', W'$ s.t. $\mathcal{R}(P', W')$ and $\lfloor w \rfloor \in W'$.

The comparison. Next, we consider the *most powerful* attacker or saboteur S, and define the format of its estimate, which therefore will be an estimate for any other attacker. From this estimate and one confining P, we can construct an estimate showing that $P \mid S$ is also confined. In other words, P can be placed in any context without disclosing its secrets. Typically, the estimate for S will involve expressions of kind P, only. This and the following lemma deeply depend on the Moore family property (Theorem 3). To state this succinctly define the restrictions $\rho_{|B}$, $\kappa_{|C}$, $\zeta_{|L}$ $(B \subseteq \mathcal{V}, C \subseteq \mathcal{N}, L \subseteq \mathcal{L})$ as follows:

$$(\rho_{|B})(x) = \begin{cases} \rho(x) & \text{if } x \in B \\ \emptyset & \text{o.w.} \end{cases} \quad (\kappa_{|C})(n) = \begin{cases} \kappa(n) & \text{if } n \in C \\ \emptyset & \text{o.w.} \end{cases} \quad (\zeta_{|L})(l) = \begin{cases} \zeta(l) & \text{if } l \in L \\ \emptyset & \text{o.w.} \end{cases}$$

We now characterize the shape of estimates for an attacker Q.

Lemma 10. *Let Q be a closed process with all names in $\mathcal{P}$; then $(\rho', \kappa'_{|\mathcal{P}}, \zeta') \models Q$ where $\forall x, \forall n \in \mathcal{P}, \forall l :\ \rho'(x) = \kappa'_{|\mathcal{P}}(n) = \zeta'(l) = \mathsf{ValP}$.*

Given an estimate for P, we can reduce it to act on the variables and labels of P only.

Lemma 11. *Let B and L be the sets of variables and labels in P, then $(\rho, \kappa, \zeta) \models P$ if and only if $(\rho_{|B}, \kappa, \zeta_{|L}) \models P$.*

From the estimate confining P, we can construct an estimate confining $P \mid Q$, using the above estimate for Q and the above lemma.

Proposition 12. *Let P be confined w.r.t. $\mathcal{S}$; and let Q be a closed process with names all in $\mathcal{P}$, and such that all variables and labels occurring inside Q do not occur inside P. Then $P \mid Q$ is confined w.r.t. $\mathcal{S}$.*

Due to this proposition, there is *no need* to actually compute the estimate for the most powerful attacker S or for any attacker Q and more importantly that for $P \mid S$: the estimate for P as defined in Defn. 6 suffices for checking secrecy. Indeed, since P is confined, so is $P \mid Q$ which also is careful, by Theorem 7. This suffices for proving that P never reveals secret messages to an attacker knowing only public data.

It follows that our static notion of confinement suffices to guarantee Dolev and Yao's property of secrecy. Indeed, a confined (and thus careful) process never sends secrets in clear on public channels.

Theorem 13. *A process P confined w.r.t. $\mathcal{S}$, does not reveal any message M, with $M \gg (\nu\tilde{r})w$ and kind$(w) = s$, from any $K_0 \subseteq \mathcal{P}$.*

5 CFA and Message-Independence

The notion of secrecy seen above does not guarantee absence of implicit information flow, cf. [2] for more explanation. A typical case of implicit flow is when a protocol P behaves differently, according to the result of comparing a secret value against a public one. In this case, an attacker can detect some information about a message sent by noticing, e.g., that the message is *not* the number 0. In this section, we follow Abadi's approach [1], and consider the case in which a message received does not influence the overall behaviour of the protocol, even in presence of an active attacker Q. Note however that Q running in parallel with P *may* change the behaviour of P, e.g. by sending a message that permits to pass a matching. We shall show that our CFA can guarantee this form of non-interference, that we call *message independence.*

More precisely, we shall make sure that no attacker can detect whether a process $P(x)$ (where for simplicity, x is the only free variable) uses a message M or a different one M' in place of x. To interface with the developments of Section 4 we shall focus on a specific canonical channel $n_* \in \mathcal{S}$ not otherwise used; it will be used to track the places where the value of x may reach. Technically, we can either assume that all solutions (ρ, κ, ζ) considered have $\rho(x) = \{n_*\}$ or else substitute n_* for x in all instances where we invoke the analysis and the notion of confinement.

To cater for this development, we assign two sorts to values, according to whether they contain n_* or not (again, encryption is an exception). Intuitively, a value w has sort I if either n_* does not occur in w, or it appears encrypted; otherwise n_* is "visible" in w that then gets sort E. Also, note that if $kind(w) = \mathrm{P}$ then $sort(w) = \mathrm{I}$.

Definition 14. The operator $sort : Val' \to \{\mathrm{I}, \mathrm{E}\}$ is defined as

- $sort(n) = \begin{cases} \mathrm{I} & \text{if } n \neq \lfloor n_* \rfloor \\ \mathrm{E} & \text{if } n = \lfloor n_* \rfloor \end{cases}$
- $sort(0) = \mathrm{I}$; – $sort(suc(w)) = sort(w)$;
- $sort(pair(w, w')) = \begin{cases} \mathrm{I} & \text{if } sort(w) = sort(w') = \mathrm{I} \\ \mathrm{E} & \text{otherwise;} \end{cases}$
- $sort(enc\{w_1, \cdots, w_k, r\}_{w_0}) = \mathrm{I}$

Again, by abuse of notation, $sort(W) = \begin{cases} \mathrm{E} \text{ if } \exists w \in W : sort(w) = \mathrm{E} \\ \mathrm{I} \text{ if } \forall w \in W : sort(w) = \mathrm{I}. \end{cases}$

With our next definition, we statically check if a process uses (the value that will bind) x in points where an attacker can grasp it. More in detail, we consider as sensitive data those terms that are used as channels or as keys or in comparisons, and check that they will never depend on the message M. Otherwise, in the first case, the attacker may establish different communications with $P[M/x]$ and $P[M'/x]$; in the second case, the attacker may decrypt a message if M turns out to be public; in the last case, the attacker may detect some information about M (e.g. if it is not 0, see above). The static check controls that the special name n_* never belongs to the sets of values that are associated by the ζ component of estimates to each occurrence of these sensitive data. Note that we allow decomposing a term containing x; we only forbid, in a lazy way, that x is used to alter the flow of control.

Definition 15. The process $P(x)$ is *invariant* w.r.t. x and (ρ, κ, ζ) if and only if for all occurrences of

- terms $\{V_1, \cdots, V_k, (\nu r)r\}_{N^l}$, are s.t. $sort(\zeta(l)) = \mathrm{I}$;
- prefixes $\overline{M^l}\langle V\rangle.P$ and $M^l(y).P$ and constructs $let\ (y,z) = M^l\ in\ P$; $case\ M^l\ of\ 0 : P\ suc(y) : Q$; $case\ M^l\ of\ \{y_1, \cdots, y_k\}_{N^{l'}}\ in\ P$, are s.t. $n_* \notin \zeta(l)$ and $sort(\zeta(l')) = \mathrm{I}$;
- constructs $[M^l\ is\ N^{l'}]P$, are s.t. $sort(\zeta(l)) = sort(\zeta(l')) = \mathrm{I}$.

Before defining our notion of message independence we need to adapt testing equivalence. Basically two processes are testing equivalent [13,10] if they pass exactly the same set of tests, i.e. if one process is ready to communicate with any partner then so is the other, and viceversa.

Definition 16. Let P, P' and Q be closed processes and let β be m or $\overline{m}$. The process P *passes* a *public test* (Q, β) if and only if $fn(Q) \subseteq \mathcal{P}$ and $(P|Q) \xrightarrow{\tau} Q_1 \cdots \xrightarrow{\tau} Q_n \xrightarrow{\beta} A$, for some $n \geq 0$, some processes $Q_1, \cdots, Q_n$ and some agent A. The two processes P and P' are *public testing equivalent*, in symbols $P \sim P'$, if $\forall (Q, \beta)$, if P passes (Q, β) then P' passes (Q, β) and viceversa.

Message independence of a process $P(x)$ then merely says that no external observer can determine the term instantiating the variable x.

Definition 17. A process $P(x)$ is message independent iff $P[M/x] \sim P[M'/x]$ for all closed messages M and M'.

Finally, we establish that a confined and invariant process is message independent; our formulation offers an alternative to Abadi's approach, based on type systems. Moreover, our formulation sheds light on the role played by confidentiality in non interference. It is crucial to keep confidential secrets for not exposing, either directly or indirectly, the values that can be bound to the free variable x.

Theorem 18. *If $P(x)$ is confined (w.r.t. $\mathcal{S}$ containing n_*) and invariant (w.r.t. x and the same solution), then it is message independent.*

6 Conclusion

Control Flow Analysis has already been successfully used for studies of security in the π-calculus [8] (focusing on direct flows violating confidentiality) and for studies of mobility in the Mobile Ambients [22,18] (focusing on firewalls).

Here, we have proved that our overall approach to direct flows does scale up to a calculus with perfect cryptography, despite the need to use more advanced techniques for efficiently implementing the analysis. Prior to that, we have also overcome a weakness in previous formulations of perfect symmetric cryptography, usually formulated using algebraic identities, by defining its properties as part of the semantics of the νSPI-calculus.

Our second technical result was to show that our approach is also amenable to the treatment of indirect flows, in the form of non-interference results, thereby obtaining results similar to those obtained using type systems. Indeed, we have factored confidentiality out of non interference. This separation of concerns may clarify the relationship between the two properties and may help checking them separately.

References

1. M. Abadi. Secrecy by Typing In Security protocols. *Journal of the ACM*, 5(46):18–36, September 1999.
2. M. Abadi. Security protocols and specifications. In *FoSSaCS'99, LNCS 1578*, pages 1–13. Springer, 1999.
3. M. Abadi, C. Fournet. Mobile Values, New names, and Secure Communication. In *POPL'01*, ACM, 2001.
4. M. Abadi, B. Blanchet. Secrecy Types for Asymmetric Communication. In *FoSSaCS'01, LNCS 2030*, pages 25–41. Springer, 2001.
5. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols - The Spi calculus. *Information and Computation 148*, 1:1–70, January 1999.
6. C. Bodei. *Security Issues in Process Calculi.* PhD thesis, Dipartimento di Informatica, Università di Pisa. TD-2/00, March, 2000.
7. C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static analysis for the π-calculus with their application to security. To appear in *I&C*. Available at `http://www.di.unipi.it/~chiara/publ-40/BDNNi00.ps`.

8. C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Control flow analysis for the π-calculus. In *CONCUR'98, LNCS 1466*, pages 84–98. Springer, 1998.
9. D. Bolignano. An approach to the formal verification of cryptographic protocols. In *3rd ACM Conf. on Computer and Communications Security*, pages 106–118. ACM Press, 1996.
10. M. Boreale and R. De Nicola. Testing equivalence for mobile processes. *Information and Computation*, 120(2):279–303, August 1995.
11. L. Cardelli and A.D. Gordon. Types for mobile ambients. In *POPL'99*, pages 79–92. ACM Press, 1999.
12. R. De Nicola, G. Ferrari, and R. Pugliese, B. Venneri. Types for access control. *Theoretical Computer Science* 240(1): 215-254, June 2000.
13. R. De Nicola and M.C.B. Hennessy. Testing equivalence for processes. *Theoretical Computer Science*, 34:83–133, 1984.
14. D. E. Denning. A Lattice Model of Secure Information Flow. *Communications of the ACM*, pages 236–243, May 1976.
15. D. E. Denning and P. J. Denning. Certification of Programs for Secure Information Flow. *Communications of the ACM*, pages 504–513, July 1977.
16. D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, IT-29(12):198–208, March 1983.
17. J.A. Goguen and J. Meseguer. Security policy and security models. In *1982 IEEE Symposium on Research on Security and Privacy*, pages 11–20. IEEE Press, 1982.
18. R. R. Hansen, J. G. Jensen, F. Nielson, and H. R. Nielson. Abstract interpretation of mobile ambients. In *SAS'99, LNCS 1694*, pages 135–148, 1999.
19. N. Heintze and J.G Riecke. The SLam calculus: Programming with secrecy and integrity. In *POPL'98*, pages 365–377. ACM Press, 1998.
20. J. Jürjens. Bridging the Gap: Formal vs. Complexity-theoretical Reasoning about Cryptography. Security through Analysis and Verification, Dagstuhl Dec. 2000.
21. F. Nielson and H. R. Nielson. Flow logics and operational semantics. *Electronic Notes of Theoretical Computer Science*, 10, 1998.
22. F. Nielson, H. R. Nielson, R. R. Hansen, and J. G. Jensen. Validating firewalls in mobile ambients. In *CONCUR'99, LNCS 1664*, pages 463–477, 1999.
23. F. Nielson, H. Seidl. Control-Flow Analysis in Cubic Time. In *ESOP'01, LNCS 2028*, pages 252–268, 2001.
24. H. Riis Nielson and F. Nielson. *Semantics with Applications: A Formal Introduction.* Wiley Professional Computing. Wiley, 1992.
25. H. Riis Nielson, F. Nielson, H. Seidl. Cryptographic Analysis in Cubic Time. Manuscript, 2001.
26. L.C. Paulson. Proving properties of security protocols by induction. In *CSFW'97*, pages 70–83. IEEE, 1997.
27. J. Riely and M. Hennessy. A typed language for distributed mobile processes. In *POPL'98*, pages 378–390. ACM Press, 1998.
28. J. Riely and M. Hennessy. Trust and partial typing in open systems of mobile agents. In *POPL'99*, pages 93–104. ACM Press, 1999.
29. D. Volpano and G. Smith. Language Issues in Mobile Program Security. In *Mobile Agent Security*, LNCS 1419, pages 25–43. Springer, 1998.
30. D. Volpano and G. Smith. Probabilistic noninterference in a concurrent language. In *CSFW'98*, pages 34–43. IEEE, 1998.
31. D. Volpano and G. Smith. Secure information flow in a multi-threaded imperative language. In *POPL'98*, pages 355–364. ACM Press, 1998.
32. D. Volpano, G. Smith, and C. Irvine. A sound type system for secure flow analysis. *Journal of Computer Security*, 4:4–21, 1996.