Electronic Notes in Theoretical Computer Science 62 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume62.html LastPage pages

Techniques for Security Checking: Non-Interference vs Control Flow Analysis [1]

Chiara Bodei [a], Pierpaolo Degano [a], Riccardo Focardi [b]
Roberto Gorrieri [c], Fabio Martinelli [d],

[a] *Dipartimento di Informatica, Università di Pisa, Italy*

[b] *Dipartimento di Informatica, Università Ca' Foscari Venezia, Italy*

[c] *Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy*

[d] *Istituto per le Applicazioni Telematiche, C.N.R. Pisa, Italy*

Abstract

We model, in a process algebra framework, a variant of the well known Wide Mouthed Frog security protocol. Its relevant security properties are addressed both from a dynamic and static point of view, having operational semantics as a common starting point. In one case, we exploit techniques based on Non-Interference, while in the other one we rely on Control Flow Analysis. We then compare these techniques.

1 Introduction

Security has become an essential requirement for many applications, especially in the last decade, due to the widespread diffusion of distributed systems and networks. This notion appears to be fairly obvious, but elusively refuses to assume a precise meaning in our field. The scientific community cannot help trying to catch this notion in thousands of slightly different properties. Formal methods and techniques can offer a way of generalizing and comparing these different formulations, driving this delicate translation of intuitive notions into formal specifications, essential for a careful design and analysis of secure computer systems. Below, we limit ourselves to consider security of network protocols, expressed in a process algebra framework. Their study can consequently exploit all the results and the analysis tools developed in the concurrency community.

In the last years, encouraging results have been obtained by the use of static techniques exploiting notions of information flow to establish various forms of

[1] Research partially supported by MURST Progetto Cofinanziato TOSCA.

secrecy and of integrity of data. In this approach, systems or protocols are specified as expressions of some (possibly idealized) programming language, like the spi-calculus [3,1]. Security properties are checked through static tests, that, if passed, guarantee that there is no violation of the property under consideration, at run-time. We use here a specific static technique: Control Flow Analysis (CFA), based on Flow Logic.

It is built around the more "classical" approaches to static analysis and at the same time, it offers a way to exploit the quite advanced state-of-the-art in static analysis also for the analysis of security, as shown in some recent proposals [6,5,7,20]. In particular, CFA offers static techniques for predicting safe and computable approximations to the set of values that the objects of a program may assume during its execution.

Information flow is, in general, a key issue in security. Its analysis is mainly carried out through dynamic techniques. A classic approach used to study information flow in multilevel [4] computer systems is Non-Interference (NI).

Roughly, NI requires that a set H of users, devised as high-level principals, is not able to interfere with a set L of low-level users in a certain system S. Thus, the low-level view of the system S should always be the same regardless of the high-level activities. The check whether a protocol is secure can be rephrased as an information flow one. Indeed, consider the high-level principals as attackers on a network protocol. Furthermore, assume that honest participants emit "control" actions, depending on a specific security property $\mathcal{A}$, that denote their beliefs about the current execution of the protocol.

These actions are considered as the low-level view of the protocol. Then, the protocol satisfies the security property $\mathcal{A}$ if its low-level, observable behaviour cannot be "significantly" altered when executed in parallel with any attacker. In other words, an attack is revealed as an interference to the normal (or intended) behaviour of honest participants. This way of studying security properties [13,16,15] is a direct generalization of *Non-deducibility on composition* (NDC) proposed in [14] for studying NI in a process algebra setting. Interestingly enough, NI can help bridging the gap between the analyzes of two different aspects like information flow and network protocols. Therefore NI offers a uniform framework for studying both aspects, that usually require rather different techniques.

The purpose of this note is to compare the static CFA with the dynamic NI. As a benchmark for our techniques, we use a simplified version of the Wide Mouthed Frog protocol [8]. The notion of security we shall exploit is based on the so-called secrecy à la Dolev-Yao [11]. This model postulates the existence of an intruder, or enemy, who has full control on the network. Initially, the intruder has some knowledge about some values, that can be increased by eavesdropping. Also, the enemy can start attacking a protocol, by sending messages resulting from suitable combination of pieces of her/his current knowledge. E.g., the intruder can pair values or encrypt/decrypt them with compromised or initially known keys. Of course, the enemy's goal is to discover secret information or to induce an honest participant to a protocol to misbehave. Both our approaches formalize the above and show whether given specifications of the WMF are resistant to attacks of the intruder or not.

Our working example is intentionally very simple, so that we can focus on the basic features of both approaches and start understanding their differences and their interplay. On this basis, we can try to single out some of the pro and cons of both. We shall discuss them in more detail in the concluding remarks; here we concisely mention the most relevant. On the one hand, CFA offers approximated, yet sound, answers, i.e. it may "err on the safe side". This approximation permits to have more efficient analysis tools. On the other hand, NI is precise and thus much less efficient in general; actually, it is complete for a class of (finite) systems.

As they share a large commonalty of goals, we hope that the comparison of two so different techniques can further a cross-fertilization between them, and can also give some useful insights on the security properties themselves.

Our Case-study: the Wide Mouthed Frog protocol

We recall a version of the WMF protocol (Protocol 1), simplified in that the two first messages do not contain time-stamps. This makes the protocol sensitive to replay attacks. The WMF is intended to establish a session key for A and B with the help of a server S. In the first message A sends to S its name, and then a fresh key K_{AB} and the name of the intended receiver B, encrypted under the key K_{AS}, shared between A and S. In the second one, S forwards the key and the sender name A to B, encrypted under the key K_{BS}, shared between B and S. Finally, A sends B the message M encrypted under K_{AB}.

In fact, we shall mainly focus on a slightly modified version (Protocol 2), where in the first message the name of the receiver is not included in the encryption. As we will show, this little modification leads to new and subtle attacks: the first one on secrecy (Attack 1) and the other two on authentication (Attacks 2 and 3), as reported in [12].

Protocol 1 *Wide Mouthed Frog Protocol (without time-stamps)*

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

Protocol 2 *Wide Mouthed Frog Protocol modified*

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

We show now the main attacks to Protocol 2. Note that we discard the reported parallel session attacks in [12], since we disallow A to commit with herself.

Attack 1 *Secrecy Attack*

$$\begin{array}{lll}
\textit{Message } 1 & A \rightarrow E(S) & : A, B, \{K_{AB}\}_{K_{AS}} \\
\textit{Message } 1' & E(A) \rightarrow S & : A, E, \{K_{AB}\}_{K_{AS}} \\
\textit{Message } 2' & S \rightarrow E : & \{A, K_{AB}\}_{K_{ES}} \\
\textit{Message } 3 & A \rightarrow E(B) & : \{M\}_{K_{AB}}
\end{array}$$

Here the first message $\langle A, B, \{K_{AB}\}_{K_{AS}}\rangle$ is intercepted by E. Since B is passed in clear on the net, E can manipulate it in order to obtain $\langle A, E, \{K_{AB}\}_{K_{AS}}\rangle$. As a result, the secret message M for B arrives to E. This attack is not possible in Protocol 1, where B is inside the encryption and therefore the misleading message cannot be forged.

Attack 2 *Authentication Attack*

$$\begin{array}{lll}
\textit{Message } 1 & A \rightarrow E(S) & : A, E, \{K_{AE}\}_{K_{AS}} \\
\textit{Message } 1' & E(A) \rightarrow S & : A, B, \{K_{AE}\}_{K_{AS}} \\
\textit{Message } 2' & S \rightarrow B : & \{A, K_{AE}\}_{K_{BS}} \\
\textit{Message } 3 & A \rightarrow E : & \{M'\}_{K_{AE}} \\
\textit{Message } 3' & E(A) \rightarrow B & : \{M'\}_{K_{AE}}
\end{array}$$

Similarly, here the first message $\langle A, E, \{K_{AE}\}_{K_{AS}}\rangle$ received by E is manipulated in order to obtain $\langle A, B, \{K_{AE}\}_{K_{AS}}\rangle$. As a result, B is at the end convinced it is communicating with A, while A is instead communicating with E. Moreover, the message M' for E arrives to B.

Attack 3 *Replay Attack*

$$\begin{array}{lll}
\textit{Message } 1 & A \rightarrow S : & A, B, \{K_{AB}\}_{K_{AS}} \\
\textit{Message } 2 & S \rightarrow B : & \{A, K_{AB}\}_{K_{BS}} \ \textit{E eavesdrops and replays it} \\
\textit{Message } 2' & E(S) \rightarrow B & : \{A, K_{AB}\}_{K_{BS}} \\
\textit{Message } 3 & A \rightarrow B : & \{M\}_{K_{AB}} \ \textit{E eavesdrops and replays it} \\
\textit{Message } 3' & E(A) \rightarrow B & : \{M\}_{K_{AB}}
\end{array}$$

Here, E is able to replay the first session getting B to commit twice on the same message. Note that this attack is only possible because we assume B to play the role of responder in two parallel *sessions. The preceding attacks, instead, occur already in* single *sessions, where the principals play their role only once.*

2 The Calculus

Our main goal here is to compare the CFA static approach and the NI dynamic approach. Originally the two have been defined on two different process algebras, yet related: the spi-calculus [1] and Crypto-SPA [16]. So we need to tailor a common subset, that of course includes primitives for constructing messages, in particular encryption and decryption, and primitives for sending and receiving messages. Below, we intuitively introduce it, following to some extent the spi-calculus [3,1] notation, of which our calculus can be seen as a fragment. This calculus is enough for describing the cryptographic protocols usually reported in the literature, e.g., the case study discussed in this paper.

The terms can be names or variables and can also be structured as encryptions $\{M_1, \ldots, M_l\}_N$. An encryption $\{M_1, \ldots, M_l\}_N$ represents the ciphertext obtained by encrypting $M_1, \ldots, M_l$ under the key N (a variable x to be instantiated by a key K or a key itself), using a *shared-key* cryptosystem.

The processes are built as in earlier calculi: there are operators for input and output, parallel composition, restriction and matching.

Terms and processes are defined according to the following grammars.

$L, M, N ::=$	**terms**
a, b, c, K, m, n	*names*
x, y, z, w	*variables*
$\{M_1, \ldots, M_l\}_N$	*encryption*

$P, Q, R ::=$	**processes**
$\mathbf{0}$	*nil*
$\overline{c}\langle N_1, \ldots, N_l\rangle.P$	*output*
$c(x_1, \ldots, x_l).P$	*input*
$(\nu m)P$	*restriction*
$P \mid P$	*parallel composition*
$[M = N]P$	*matching*
case L *of* $\{x_1, \ldots, x_l\}_K$ *in* P	*decryption*

- The null process $\mathbf{0}$ does nothing.
- The process $\overline{c}\langle N_1, \ldots, N_l\rangle.P$ sends the terms $N_1, \ldots, N_l$ on channel c, then it behaves like P.
- The process $c(x_1, \ldots, x_l).P$ is ready to receive an input $N_1, \ldots, N_l$ on channel c and to behave like $P\{N_1/x_1, \ldots, N_l/x_l\}$, where each term N_i is bound to the variable x_i. Note that, unlike the spi-calculus, we do not allow to use received names as channels, as variables cannot be used as channel names.
- The operator $(\nu m)P$ acts as a static declaration (i.e. a binder for) the name

m in the process P that it prefixes, so stating that m is local to P. The agent $(\nu m)P$ behaves as P except that actions on m are prohibited.

- The operator | describes parallel composition of processes. The components of $P|Q$ may act independently; also, an output action of P (resp. Q) at any output port $\overline{c}$ may synchronize with an input action of Q (resp. P) at c. In this case, a non observable action τ results.
- Matching $[M = N]P$ is an `if-then` operator: the process P is activated only if $M = N$.
- The process *case* L *of* $\{x_1, \ldots, x_l\}_K$ *in* P tries to decrypt L with the key K. If L has the form $\{M_1, \ldots, M_l\}_K$, then the process behaves as $P\{M_1/x_1, \ldots, M_l/x_l\}$, otherwise the process is stuck.

We omit here the formal definition of the semantics of our calculus. It can be defined in the standard way, and could be either a reduction semantics or an SOS one. We also omit the definition of what is observable, assuming the standard notions, upon which suitable, standard equivalences are to be built, e.g., trace equivalence or bisimulation.

Note that in our calculus, neither the input nor the output capabilities can be transmitted, because our prefixes have the form $c(x_1, \ldots, x_l)$ and $\overline{c}\langle N_1, \ldots, N_l\rangle$ and c is a name and *not* a variable. Actually a variable x can be instantiated by a channel name, but it cannot occur as a channel on which communication takes place.

In order to facilitate the definition of our CFA, we assume that names are "stable", i.e. that each name a is the canonical representative for its class of α-convertible names [2]. As a consequence of assuming stable names, we are allowed to partition them all into two sets $\mathcal{S}$ and $\mathcal{P}$ of secret and public names, respectively.

3 Control Flow Analysis approach

For lack of space, we only intuitively introduce the Control Flow Analysis for our calculus. It refines the analysis in [7] that considers the full spi-calculus. Our analysis essentially focus on the use of channels, of values and of terms. Roughly, a result, or *estimate* $(\rho, \kappa, \zeta, \phi)$ of our CFA establishes a super-set of the set of (abstract) values to which variables can be bound (ρ), of the set of values that can flow over given channels (κ) and of the set of the possible actual values for a given term (ζ).

Additionally, an estimate establishes a super-set of all the values the environment can send and receive on public channels (for simplicity, we consider *all* the channels as public, but this is by no means a restriction). We are interested in analyzing the behaviour of a process P plugged in any hostile

[2] Technically this amounts to having a partition of names, each with a leader (see [7] for further details).

environment E, modeled as done by Dolev and Yao [11,2], that initially has some knowledge about public values, only. Then, E may increase its knowledge by communicating with P, directly or intercepting messages from or to P. Note that E can know all the values computable from its current knowledge (e.g. knowing K and $\{M\}_K$, E can compute M). In our new analysis, we want to directly analyze only the process P; at the same time, we need to take care of E; the last component ϕ is in charge for it. This fourth component refines the analysis in [7] and can easily carried over the full spi-calculus.

The formulation of the CFA is facilitated by making a few assumptions. Mainly, we slightly extend the standard syntax by mechanically assigning "labels" l to the occurrences of terms and "labels" π to particular program points; these are nothing but explicit notations for the analysis (they do not affect the dynamic semantics). Recall that our names are "stable". In details, the components are:

- ρ is the *abstract environment*. It associates variables with the values that they can be bound to; more precisely, $\rho(\pi)(x)$ must include the set of values that x could assume at run-time from the program point π on.
- κ is the *abstract channel environment*. It associates 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.
- ζ is the *abstract cache*. It 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 labeled l.
- ϕ is the *enemy environment*, and it is not present in [7]. This component approximates the enemy knowledge and therefore it includes all the values that can flow on the channels and *all* the values computable from them. For instance, if a key K is in ϕ, then also M is in ϕ if $\{M\}_K$ flows on the network.

To define the *acceptability* of a proposed estimate $(\rho, \kappa, \zeta, \phi)$ we state a set of clauses operating upon flow logic judgments on the forms $(\rho, \kappa, \zeta, \phi) \models_\pi M$ and $(\rho, \kappa, \zeta, \phi) \models_\pi P$. To have a look on how terms are validated, consider only the case for a variable:

$$(\rho, \kappa, \zeta, \phi) \models_\pi x^l \text{ iff } \rho(\pi)(x) \subseteq \zeta(l)$$

It says that all the values that the variable x can assume, at point π, should be collected in $\zeta(l)$.

To give the flavor of the other clauses, we just give an abstraction of the output and of the input ones, (for simplicity, we refer to monadic I/O actions).

$$\begin{gathered}(\rho, \kappa, \zeta, \phi) \models_\pi \overline{c^l}\langle N^{l'}\rangle.P \text{ iff } (\rho, \kappa, \zeta, \phi) \models_\pi c^l \wedge (\rho, \kappa, \zeta, \phi) \models_\pi N^{l'} \wedge \\ (\rho, \kappa, \zeta, \phi) \models_\pi P \wedge \zeta(l') \subseteq \kappa(c) \wedge \kappa(c) \subseteq \phi\end{gathered}$$

It says that the estimate is valid for $\overline{c^l}\langle N^{l'}\rangle.P$ if it is valid for c, N and for P and if the set of values associated with the message N can be passed on the channel c. More intuitively, whenever a value a is output on c, as in $\overline{c}\langle a\rangle$, it must be duly recorded in the κ and ϕ components, by ensuring that $a \in \kappa(c)$ and $a \in \phi$. The last inclusion amounts to saying that the enemy can acquire all the values that flow on the channel c.

$$(\rho, \kappa, \zeta, \phi) \models_\pi c^l(x).P \text{ iff } (\rho, \kappa, \zeta, \phi) \models_\pi c^l \wedge (\rho, \kappa, \zeta, \phi) \models_\pi P \wedge \\ \kappa(c) \subseteq \rho(\pi)(x) \wedge \phi \subseteq \kappa(c)$$

Similarly, each value passing along c must be contained in the set of possible values of x. Whenever a variable x inputs a value on c, as in $c(x)$, this must also be duly recorded in ρ, intuitively by ensuring that $a \in \rho(\pi)(x)$ for all $a \in \kappa(c)$ (the program point π indexes the satisfiability relation). Moreover, $\phi \subseteq \kappa(c)$, i.e. on c can flow all the values the enemy knows and can send.

From these clauses, it is possible to grasp how the component ϕ can model any hostile environment within the Dolev-Yao model, and consequently also the most powerful intruder.

We have seen that a CFA is formulated as a specification of the correctness of a candidate estimate. It is possible to show that least estimates always exist and to establish their semantic correctness with respect to the operational semantics, in the form of a subject reduction result (for a similar result see e.g. [6]).

Remarkably, there is also a constructive procedure for obtaining the least estimate. Essentially, establishing $(\rho, \kappa, \zeta, \phi) \models_\pi P$ amounts to checking a number of individual constraints.

The procedure that generates estimates explicitly extracts these constraints, proceeding by induction on the syntactic structure of processes. For instance, in case of output $\overline{c^l}\langle N^{l'}\rangle.P$, we add the constraint $\{\zeta(l') \subseteq \kappa(c)\}$ to the constraints already obtained for P. We argue that this procedure operates in low polynomial time with respect to the size of the process under analysis. Actually, we are confident that the time complexity is cubic, as a recent result for the spi-calculus [19] shows.

Static security properties

Applications of our analysis include establishing two simple security properties of processes.

We first recall the extension with respect to the new CFA of the static property of confinement, studied in [6,7]. It guarantees the dynamic property of secrecy put forward by Dolev and Yao, where intuitively a process P does not reveal its secrets if no secret is output on clear, or nothing that can help in computing it is output as well. The first notion depends on a partition of names in secret $\mathcal{S}$ and public $\mathcal{P}$ (recall that our names are "stable").

More precisely, we postulate as *public* everything built from $\mathcal{P}$, in particular *any* value, be it public or secret, when encrypted under a secret shared key (our cryptography is assumed to be *perfect*), as well as a *public* value when encrypted with a *public* key (there is nothing to protect). Instead, a *secret* value encrypted with a public key is vulnerable to attacks, and so we consider the whole encryption as public. Note that if a name happens to denote a key K, saying that K is "public" means that it is widely known, i.e. it is not secret. [3] In what follows, let ValP be the set of public values.

Definition 3.1 *The process P is* confined *(w.r.t.* S*) if*

(a) $(\rho, \kappa, \zeta, \phi) \models P$ *and* **(b)** $\mathcal{P} \subseteq \phi \subseteq \mathsf{ValP}$.

Note that, on least solutions, the condition (**b**) implies the condition (**b**$'$) $\forall c : \kappa(c) \subseteq \mathsf{ValP}$. The latter is in the style of [6], where the focus is on what flows on channels. Also, the intruder is *passive*, as it cannot build and send any messages on the network, but can only eavesdrop. On the contrary, here the focus is on what the enemy knows, intercepts and can deduce, allowing him/her to actively interact with the other principals.

The second property extends that of authorship introduced in [5]. It is a form of message authentication: it states when a message, or a part of it, is composed of names generated (i.e. declared via a restriction) by a given process P. In this way, the message is authentic from P.

Definition 3.2 *The variable x (occurring at program point π) is* booked *for a sub-process $P = (\nu n_1, \ldots, n_l)P'$ of a given process Q if and only if there exists an estimate $(\rho, \kappa, \zeta, \phi)$ such that*

(1) $(\rho, \kappa, \zeta, \phi) \models Q$ *and* **(2)** $ID(\rho(\pi)(x)) \subseteq \{n_1, \ldots, n_l\}$,

where $ID(I)$ returns the names occurring in the values of the set I.

The static notion requires that all the values that can be bound to x are composed only with names of P. We can use this property to guarantee that in x is possible to receive, dynamically, only messages authenticated by P.

CFA on WMF

We now apply our approach to the WMF protocol. We will give here two specifications: Sys_1 for Protocol 1 and Sys_2 for Protocol 2 and we will check whether they enjoy or not the confinement and being booked properties defined above. In Sys_1, both properties do hold (Sys_1 passes the corresponding static tests), while in Sys_2 does not. These failures show attacks 1 and 2, on Protocol 2. Remarkably, the same analysis is used to check either confinement or authorship: only the test changes.

This feature shows the generality of the CFA approach.

For the sake of simplicity, here we assume that only A can play the role of sender (each instance oriented to a different receiver) and that only B (the honest principal) and E (the enemy) can play the role of receivers. The server S is unique. In Sys_1 the first message is $\langle A, \{B, K_{AB}\}_{K_{AS}}\rangle$, while in Sys_2 it is $\langle A, B, \{K_{AB}\}_{K_{AS}})\rangle$. Since A is the only sender, we omit to send its name in the first message in both cases.

[3] Here, a public key is *not* the encryption key in an asymmetric or public-key cryptosystem.

$$
\begin{aligned}
Sys_1 &= (\nu K_{AS})(\nu K_{BS})(A_{B_1}|A_{E_1}|S_1|B)\\
A_{B_1} &= (\nu K_{AB})(\nu M)(\overline{c_{AS}}\langle\{B, K_{AB}\}_{K_{AS}}\rangle.\overline{c_{AB}}\langle\{M\}_{K_{AB}}\rangle)\\
A_{E_1} &= (\nu K_{AE})(\nu M')(\overline{c_{AS}}\langle\{E, K_{AE}\}_{K_{AS}}\rangle.\overline{c_{AE}}\langle\{M'\}_{K_{AE}}\rangle)\\
S_1 &= c_{AS}(y_{cipher}).case\ y_{cipher}\ of\ \ \{y_{rec}, y_{key}\}_{K_{AS}}\ in\\
&\quad {}^{\pi}([y_{rec} = B]\ {}^{\pi_B}\overline{c_B}\langle\{y_{key}\}_{K_{BS}}\rangle \mid [y_{rec} = E]\ {}^{\pi_E}\overline{c_E}\langle\{y_{key}\}_{K_{ES}}\rangle)\\
B &= c_{BS}(w_{cipher}).case\ w_{cipher}\ of\ \ \{w_{key}\}_{K_{BS}}\ in\\
&\quad {}^{\pi'}c_{AB}(z_{cipher})case\ z_{cipher}\ of\ \ \{z_{msg}\}_{w_{key}}\ in\ \ {}^{\pi''}\mathbf{0}
\end{aligned}
$$

$$
\begin{aligned}
Sys_2 &= (\nu K_{AS})(\nu K_{BS})(A_{B_2}|A_{E_2}|S_2|B)\\
A_{B_2} &= (\nu K_{AB})(\nu M)(\overline{c_{AS}}\langle B, \{K_{AB}\}_{K_{AS}}\rangle.\overline{c_{AB}}\langle\{M\}_{K_{AB}}\rangle)\\
A_{E_2} &= (\nu K_{AE})(\nu M')(\overline{c_{AS}}\langle E, \{K_{AE}\}_{K_{AS}}\rangle.\overline{c_{AE}}\langle\{M'\}_{K_{AE}}\rangle)\\
S_2 &= c_{AS}(y_{rec}, y_{cipher}).case\ y_{cipher}\ of\ \ \{y_{key}\}_{K_{AS}}\ in\\
&\quad {}^{\pi}([y_{rec} = B]\ {}^{\pi_B}\overline{c_{BS}}\langle\{y_{key}\}_{K_{BS}}\rangle \mid [y_{rec} = E]\ {}^{\pi_E}\overline{c_E}\langle\{y_{key}\}_{K_{ES}}\rangle)\\
B &= c_{BS}(w_{cipher}).case\ w_{cipher}\ of\ \ \{w_{key}\}_{K_{BS}}\ in\\
&\quad {}^{\pi'}c_{AB}(z_{cipher})case\ z_{cipher}\ of\ \ \{z_{msg}\}_{w_{key}}\ in\ \ {}^{\pi''}\mathbf{0}
\end{aligned}
$$

Here, the server S should sort out session keys to B or to E, depending on the receiver name included in the previous message, i.e., in terms of our calculus, depending on a match. Note that E can act in a legitimate way as B can, using K_{AE} and M'.

Moreover, we restrict M and M', because we want to use them for the authorship property.

Just to give the flavor of how the estimates look like, we write down parts of them.

Sys₁ $\kappa(c_{AS}) \not\ni (E, \{K_{AB}\}_{K_{AS}}), (B, \{K_{AE}\}_{K_{AS}})$
$\kappa(c_{AB}), \rho(\pi')(z_{cipher}) \not\ni \{M'\}_{K_{AE}}$
$\kappa(c_{AE}) \not\ni \{M\}_{K_{AB}}$
$\kappa(c_{BS}) \not\ni \{K_{AE}\}_{K_{BS}}$
$\kappa(c_{ES}) \not\ni \{K_{AB}\}_{K_{BS}}$
$\rho(\pi'')(z_{msg}) \not\ni M'$

Sys₂ $\kappa(c_{AS}) \ni (B, \{K_{AB}\}_{K_{AS}}), (E, \{K_{AE}\}_{K_{AS}}), (E, \{K_{AB}\}_{K_{AS}}), (B, \{K_{AE}\}_{K_{AS}})$
$\kappa(c_{AB}), \rho(\pi')(z_{cipher}) \ni \{M\}_{K_{AB}}, \{M'\}_{K_{AE}}$
$\kappa(c_{AE}) \ni \{M'\}_{K_{AE}}, \{M\}_{K_{AB}}$
$\kappa(c_{BS}) \ni \{K_{AB}\}_{K_{BS}}, \{K_{AE}\}_{K_{BS}}$
$\kappa(c_{ES}) \ni \{K_{AB}\}_{K_{ES}}, \{K_{AE}\}_{K_{ES}}$
$\rho(\pi'')(z_{msg}) \ni M, M'$

Secrecy Property

Let $\mathcal{S}$ be $\{K_{AS}, K_{BS}, K_{AB}, M\}$ and $\mathcal{P}$ include all the other names, from which the public values in ValP can be built. In particular, $\mathcal{P}$ includes also the "private" names of the enemy, such as $\{K_{ES}, K_{AE}, M'\}$, i.e. those names s/he could use in a legitimate way, if A wants her/him as a receiver. In other words, we are only interested in preserving the secrets of honest parties, i.e. the terms built from the names that intruders do not initially know and that should not acquire. Recall that a term encrypted with a secret key is considered public and can be sent in clear, while a secret term encrypted with a public key is considered secret. Our second specification of the WMF Sys_2 is not confined, because obviously $K_{ES} \in \phi$, and $\{K_{AB}\}_{K_{ES}} \in \phi$, thanks to the condition $\kappa(n) \subseteq \phi$ for output. Therefore, $K_{AB} \in \phi$, as well, and everything encrypted with K_{AB}, such as M, belongs to ϕ. Such a secrecy leak depends on the fact that the server S_2 can also receive the forged message $(E, \{K_{AB}\}_{K_{AS}})$ on c_{AS}. On the current version of the WMF the condition **(b′)**: $\kappa(c) \subseteq \mathsf{ValP}$ does not hold, either. Indeed, K_{ES} is public and thus $\kappa(c_{ES})$ includes $\{K_{AB}\}_{K_{ES}}$, which in turn is public, according to the assumptions discussed above. So our estimate shows that a secret value may pass "in clear" on the network, for instance $\{M\}_{K_{AB}}$.

Instead, the specification Sys_1 is confined. Actually, the intruder cannot decrypt the message $\{B, K_{AB}\}_{K_{AS}}$, as K_{AS} is kept secret. Therefore s/he cannot forge the misleading message $\{E, K_{AB}\}_{K_{AS}}$ (that a fortiori cannot flow on c_{AS}).

An Authentication Property

Our second specification of the WMF Sys_2 does not guarantee authorship. In fact, z_{msg} is not booked for A_B, since $\rho(\pi'')(z_{msg})$ includes M', that is a name restricted in A_{E_1}. In our first specification Sys_1, the name z_{msg} is booked for A_B, instead, because $M' \notin \rho(\pi'')(z_{msg})$.

4 NI based approach

Recall that NI essentially says that, given two groups of high-level users H, and of low-level users L, there is no information flow from H to L if and only if there is no way for H to modify the behaviour of L. Alternatively, we may think of L as the set of the honest participants to a protocol and of H as the external environment hosting a number of possible intruders. Following the analogy, no information flow from high- to low-level means that the intruders have no way to change the behaviour of the honest principals.

To set up this correspondence more precisely, we single out the high-level and the low-level actions. As done before, we assume that an intruder has complete control of the network, and so it is sensible to assume that the high-level actions are those occurring on public channels. As a protocol specification is usually completely given by message exchanges on public channels, it may be unclear what are the low-level actions. In our approach, these actions are *ad hoc*, observable control actions that are included into the protocol specification. The choice of these extra actions depends on the property to be analyzed. For instance, we shall use a pair of special start/commit actions to be performed by all the honest participants, in order to study some form of authentication, as in [17].

As mentioned above, what is in this section has been developed for a different calculus, namely Crypto-SPA [16]. Essentially, this is a value-passing CCS enriched with a set of constructs that can be parameterized to handle specific data manipulation, notably the cryptographic ones. The restriction operator of Crypto-SPA binds channels but not names. So, when we need to analyze a protocol $(\nu n)P$, where n should denote a secret value of P, against every possible enemy, we actually check P against possible enemies which does not know n.

Now, the set of all possible intruders $\mathcal{E}_C^{\mathcal{P}}$ can be formally characterized, following [16,15] [4]. This set is parameterized by two sets. The first is C and contains public channel names; it is used to take care of the restricted names of processes for the reason discussed above. The second set is $\mathcal{P}$ and, as above, it contains the *initial* knowledge of the intruders in $\mathcal{E}_C^{\mathcal{P}}$. More formally, we have the following:

$$\mathcal{E}_C^{\mathcal{P}} = \{E \mid sort(E) \subseteq C \text{ and } ID(E) \subseteq \mathcal{D}(\mathcal{P})\}$$

where $sort(E)$ and $ID(E)$ are the set of channels used by the process E and the set of names occurring in E, respectively. As the set of data constructors is parametric, also the way messages are computed from $\mathcal{P}$ is parametric. Above, this set is referred to $\mathcal{D}(\mathcal{P})$. The additional flexibility is implemented through the function $\mathcal{D}$, that depends on the specific operations allowed on data, in particular on the specific crypto-systems used.

We are ready to express NI as a property of processes, that is usually called NDC [14,16]. Its simplified formulation for the sub-calculus we consider here follows. Let $\approx$ be a trace equivalence between processes, then

$$P \text{ enjoys } NDC_C^{\mathcal{P}} \text{ iff } \forall E \in \mathcal{E}_C^{\mathcal{P}} : (\nu C)(P \mid E) \approx (\nu C)P$$

We briefly comment on the above. Recall first that the only observables of processes are the extra "control" actions that are to be included to reflect a specific property; additionally, these control actions are not performed on communication channels in C. Now, $(\nu C)P$ represents the protocol P intended to run in isolation on perfectly secure channels. Hence, its observable behaviour exhibits the security property of interest, as no attack is possible. If $(\nu C)P$ is trace equivalent (on the observables!) to $(\nu C)(P \mid E)$, where E is *any* possible enemy, then clearly E has no way to alter the observable execution of P. Thus, the security property implicitly holds.

As a matter of fact, one can use a slightly more general scheme, namely the Generalized NDC ($GNDC$). It results from replacing trace equivalence $\approx$ with any pre-order $\lhd$ between processes and by replacing $(\nu C)P$ with a function on P, $\alpha(P)$. This function is related to a specific security property $\mathcal{A}$ and specifies which are the behaviour of P that respect $\mathcal{A}$. Different instances of $\lhd$ and of the function α give uniform definitions of different security

[4] There, in the original notation, $\mathcal{P}$ was actually Φ.

properties. Therefore, $GNDC$ makes it easier to compare different properties. Also, NDC happens to be the strongest property one can define within the $GNDC$ scheme under the assumption that the protocol in isolation satisfies the security requirements, i.e. $(\nu C)P \lhd \alpha(P)$. Consequently, one can put in the specification of the protocol under analysis *all* the control actions characterizing the properties of interest: a single successful NDC check implies that all the properties are satisfied.

Interestingly enough, NDC can be characterized in a simpler way than made above. A canonical, most general enemy exists, that intuitively can eavesdrop/intercept any message, adding the intercepted information to the current knowledge set, as well as produce new messages with pieces of information the enemy knows. Its formulation follows, that uses indexed summation and recursion:

$$Top_{\mathcal{P}} = \sum_{c \in C; m \in \mathcal{D}(\mathcal{P})} \overline{c}\langle m \rangle . Top_{\mathcal{P}} + \sum_{c \in C} c(m) . Top_{\mathcal{P} \cup \{m\}}$$

Then, the universal quantification over the enemies $E \in \mathcal{E}_C^{\mathcal{P}}$ can be dropped, and the NDC check can be made by exploiting $Top_{\mathcal{P}}$, only.

Below, we analyze the WMF protocol within the NI approach, and check a secrecy and an authentication property. For the sake of clarity, we separately discuss the analysis of the two properties, instead of checking both at the same time.

Secrecy Property

As done in the previous section, we wish to check whether the specifications given in the Introduction do indeed keep secret the message M of A, according to the Dolev-Yao's notion of secrecy.

We start by inserting the needed control actions within the specification Sys_2, from which we removed all the labels π. First, we choose a distinguished channel, say c_{learnt}, not occurring in the resulting process (still denoted by Sys_2 below). Then build the set C as consisting of the just introduced name and of all the channel names that occur in Sys_2, i.e. $C = \{c_{learnt}, c_{AB}, c_{AS}, c_{BS}\}$. Also, define the initial knowledge $\mathcal{P}$ to be such that $\mathcal{P} \cap \{M, K_{AS}, K_{AB}, K_{BS}\} = \emptyset$. Note that in particular $M \notin \mathcal{P}$. On the contrary, the channel c_{learnt} must be included in $\mathcal{P}$, as well as the key K_{AE}, shared between the server and a whatsoever enemy E chosen from the set $\mathcal{E}_C^{\mathcal{P}}$. Now, we can define the process that we actually analyse:

$$Sys_2' = Sys_2 \,|(c_{learnt}(x).[x = M]\overline{learnt}\langle M \rangle)$$

Consider the system $Sys_2' \,|\, E$. Since Sys_2 cannot output on channel c_{learnt}, by construction Sys_2' can only input on that channel. Thus, a communication on c_{learnt} is possible only if E outputs a message on it. As a consequence, if $Sys_2' \,|\, E$ emits $\overline{learnt}\langle M \rangle$ then the intruder discovered M. Note that $(\nu C)Sys'_2$, is trace equivalent to $\mathbf{0}$ and $\overline{learnt}\langle M\rangle$ is the sole observable. So, we have that M remains a secret of honest participants in Sys_2 if and only if Sys'_2 enjoys the property $NDC^{\mathcal{P}}_C$. Also, note that $Sys'_2 \,|\, Top_{\mathcal{P}}$ emits $\overline{learnt}\langle M\rangle$ because $Top_{\mathcal{P}}$ can perform the following sequence of actions, showing the secrecy attack 1:

$$c_{AS}(B, \{K_{AB}\}_{K_{AS}}).\overline{c_{AS}}\langle E, \{K_{AB}\}_{K_{AS}}\rangle.$$
$$c_{AS}(\{K_{AB}\}_{K_{ES}}).c_{AB}(\{M\}_{K_{AB}}).\overline{c_{learnt}}\langle M\rangle$$

Authentication Property

We check now a form of entity authentication, called *agreement* in [17]. Essentially, it says that whenever a principal B ends a session with a partner A, then it was A who started that section with B.

As done before, we decorate the system Sys_2 with control actions specific to this property. The action $\overline{start}\langle A, M, B\rangle$ (resp. $\overline{commit}\langle B, M, A\rangle$) means that the agent A (resp. B) starts (resp. ends) a run with B (resp. A) for delivering (resp. receiving) the message M. We obtain the following:

$$\begin{aligned}
Sys''_2 &= (A''_{B_2} | S_2 | B'')\\
A''_{B_2} &= (\overline{start}\langle A, M, B\rangle.\overline{c_{AS}}\langle B, \{K_{AB}\}_{K_{AS}}\rangle.\overline{c_{AB}}\langle \{M\}_{K_{AB}}\rangle)\\
A''_{E_2} &= (\overline{start}\langle A, M', E\rangle.\overline{c_{AS}}\langle E, \{K_{AE}\}_{K_{AS}}\rangle.\overline{c_{AE}}\langle \{M'\}_{K_{AE}}\rangle)\\
S_2 &= c_{AS}(y_{rec}, y_{cipher}).case\ y_{cipher}\ \ of\ \ \{y_{key}\}_{K_{AS}}\ \ in\\
&\qquad ([y_{rec} = B]\ \overline{c_{BS}}\langle \{y_{key}\}_{K_{BS}}\rangle \mid [y_{rec} = E]\ \overline{c_E}\langle \{y_{key}\}_{K_{ES}}\rangle)\\
B'' &= c_{BS}(w_{cipher}).case\ w_{cipher}\ \ of\ \ \{w_{key}\}_{K_{BS}}\ \ in\\
&\qquad c_{AB}(z_{cipher}).case\ z_{cipher}\ \ of\ \ \{z_{msg}\}_{w_{key}}\ \ in\ \ \overline{commit}\langle B, z_{msg}, A\rangle.\mathbf{0}
\end{aligned}$$

The agreement property is satisfied by a computation, if, whenever an action $\overline{commit}\langle B, M, A\rangle$ is emitted, then an action $\overline{start}\langle A, M, B\rangle$ appeared previously. Note that the unique possible *start/commit* sequence allowed in the system in isolation $(\nu C)Sys''_2$ is

$$\overline{start}\langle A, M, B\rangle.\overline{commit}\langle B, M, A\rangle$$

Thus, if we require that Sys''_2 is NDC, i.e. $(\nu C)(Sys''_2 \,|\, E) \approx (\nu C)Sys''_2$, for all intruders E, we have that Sys''_2 enjoys the authentication property. We illustrate the role of the most powerful intruder, to which in this example we assign an empty initial knowledge. Even if perfectly blind, this intruder can significantly interfere with the protocol, because $(\nu C)(Sys''_2 \,|\, Top_{\emptyset})$ can perform the following sequence of observable actions:

$$\overline{start}\langle A, M', E\rangle.\overline{commit}\langle B, M', A\rangle$$

which shows an instance of the attack 2. Indeed, $\overline{commit}\langle B, M', A\rangle$ says that B just ended a run with A, who never started any! Actually, $\overline{start}\langle A, M', E\rangle$ denotes that A instead initiated the session with E.

If we add a further receiver B'' to the specification Sys_2'', obtaining a new specification Sys_2''', we also discover the attack 3, with the same $Top_\emptyset$ process. Indeed, in this case we find that Sys_2''' can perform the following sequence of observable actions:

$$\overline{start}\langle A, M, B\rangle.\overline{commit}\langle B, M, A\rangle.\overline{commit}\langle B, M, A\rangle$$

showing that the enemy convinced B that A started two sessions with him.

Note that also the CFA described in the previous section gives an estimate for Sys_2''' that shows this attack. Presently, neither technique is able to deal with protocols with an unbounded number of principals, unless replication is somehow constrained.

Finally, note also that the protocol 1 is not vulnerable to any of the attacks seen so far, except for the replay attack 3. This is possible to be modelled only if we add a further receiver B'', as done above to obtain the specification Sys_2'''.

5 Conclusions

We briefly surveyed two different techniques for analyzing security properties of cryptographic protocols, and applied them to a simple example, the Wide Mouthed Frog protocol.

The first approach, called CFA, is static and it based on Flow Logic. This technique collects, once and for all estimates, i.e. information about the objects occurring in the specification of a protocol, typically an approximation of the values that variables can assume at run time. Estimates are then checked, typically to test whether certain values may be bound to a specific variable. Different tests ensure that the protocol enjoys different *static* properties, that are proved to ensure their corresponding *dynamic* properties.

The second approach, called NI, is dynamic and it is based on equivalence checks. The main idea underlying NI is that proving most security properties can be seen as checking whether a protocol enjoys the Non Interference property: high-level users, the intruders, cannot change the behaviour of low-level, honest principals. Also here, a *single* check on the protocol under analysis is sufficient to detect possible flows, if any, or to guarantee that the protocol does indeed enjoy the various security properties of interest.

Our main goal here was to compare these two approaches. For making it possible, we tailored our presentation to stress their similarities and their differences. We refer the interested reader to the literature cited, to fill the gaps of our description, to clarify and to make precise some definitions and technical properties, only given here intuitively, mainly because of space limitations. Crucial has been the selection of the common subset of the process calculi on which the two techniques have originally been defined, the spi-calculus for CFA and Crypto-SPA for NI.

On the specific example, both techniques are able to reveal some delicate secrecy and authentication flaws. The main difference between them is that NI, being dynamic, can be and indeed *is* precise, while CFA, being static, gives approximate, yet *sound* answers, i.e. it may have false positives. Additionally, some attacks have an intrinsic dynamic nature and are quite hard and reluctant to be characterized within a static property.

A token in favor of CFA is that estimates are computable in low polynomial time in the size of the protocol under analysis, often in cubic time. Like other static analysis, CFA provides a repertoire of efficient automatic and decidable methods and tools for analyzing properties of systems. The NI approach requires instead exponential time, because the generation of the transition system, on which the equivalence checks are made, suffers from state explosion. Moreover, the NI tools (see, e.g. [12]) are also able to return the descriptions of successful attacks, on the form of sequences of observables. Instead, the CFA estimates [5] of a flawed protocol can only implicitly give an idea of which are the possible attacks the protocol seems to be prone to. It would be quite interesting to combine the static and the dynamic tools, so to use on need the cheapest or the most friendly or the most accurate tool within the two families.

The dynamic approach offers some decidability results for mostly common security properties, like secrecy and authentication when analyzing finite systems (roughly, systems with a bounded number of runs and parties). So far, about 40 protocols from [9] have been analyzed on which all the previously known attacks are detected, as well as new ones. Our experience with CFA is still limited, and so is the number of the security properties dealt with.

Both approaches can be further extended to deal with infinite systems. Some results in [21] show how one can deduce the correctness of certain classes of unbounded security protocols, from the analysis of bounded ones. See [18] for a CFA that defines estimates of infinite systems accurate enough — recall that the approximated nature of the CFA may help circumventing the undecidability issues generally arising in security, mainly concerned with replication.

In this note, we only considered a shared-key encryption schema, but other schemata can be used as well; similarly for different data and/or different manipulation operators. As a matter of fact, in its general formulation, the NDC property includes a parametric manner of computing the knowledge of intruders, namely via the function $\mathcal{D}(\mathcal{P})$; its use may seem irrelevant using our current sub-calculus, but it finds full justification in Crypto-SPA. Also the clauses defining our CFA do not require any changes when considering the public-key encryption. Indeed, these clauses follow a common pattern, and so we are confident that also different encryption schemata or different data manipulations can be easily accommodated in our approach. Some results in

[5] The reader may play with the on-line systems reachable at the page `http://www.daimi.au.dk/~fn/FlowLogic.html`

applying the CFA approach to other calculi can be found in [18,10].

We conclude this quick comparison by noting again that the NI approach is in a sense more explicit than the CFA one. As shown in the previous section, the most general enemy can be written as the process $Top_{\mathcal{P}}$, with the help of indexed summation. Instead, in the CFA approach the most general intruder is hidden in the way estimates are characterized, in particular in the definition of ValP and in the reverse inclusions between the κ and ϕ components of estimates, that are required for input and output clauses.

Acknowledgments.

We wish to thank Flemming Nielson and Hanne Riis Nielson for joint work on the CFA topics, and Antonio Durante for joint work on NI topics. The referees' comments have been extremely useful in reshaping our presentation.

References

[1] M. Abadi. Secrecy by Typing In Security protocols. *Journal of the ACM*, 5(46):18–36, Sept. 1999.

[2] M. Abadi. Security Protocols and Specifications. In *Proc. of FoSSaCS'99*, LNCS 1578, Springer, 1999.

[3] M. Abadi and A. D. Gordon. A Calculus for Cryptographic Protocols: The Spi Calculus. *Information and Computation* 148(1):1–70, Jan. 1999.

[4] D.E. Bell and L.J. LaPadula. Secure Computer Systems: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, Mitre C., 1976.

[5] C. Bodei. *Security Issues in Process Calculi.* PhD thesis, Dipartimento di Informatica, Università di Pisa. TD-2/00, March, 2000.

[6] C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static Analysis for the π-calculus with their Application to Security. *Information and Computation* 165: 68-92, 2001.

[7] C. Bodei, P. Degano, F. Nielson, and H. Riis Nielson. Static Analysis for Secrecy and Non Interference in Network of Processes. In *Proc. of PaCT'01*, LNCS 2127, pp. 27-41, Springer, 2001.

[8] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 1(8): 18–36, Feb. 1990.

[9] J. Clark and J. Jacob. A Survey of Authentication Protocol Literature. Unpublished, 1997.

[10] P. Degano, and F. Levi and C. Bodei. Control Flow Analysis for Mobile Safe Ambients. In *Proc. of ASIAN'00*, LNCS 1961, pp. 199-214, Springer, 2000.

[11] 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.

[12] A. Durante, R. Focardi and R. Gorrieri. A Compiler for Analysing Cryptographic Protocols using Non-Interference. *ACM Transactions on Software Engineering and Methodology*, 9(4): 488-528, Oct. 2000.

[13] R. Focardi and R. Gorrieri. The Compositional Security Checker: A Tool for the Verification of Information Flow Security Properties. *IEEE Transactions on Software Engineering* 27: 550–571, 1997.

[14] R. Focardi and R. Gorrieri. A Classification of Security Properties. *Journal of Computer Security* 1(3): 5–33,1995.

[15] R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In *Proc. of ICALP'00*, LNCS 1853, Springer, 2000.

[16] R. Focardi and F. Martinelli. A Uniform Approach for the Definition of Security Properties. In *Proc. of World Congress on Formal Methods in the Development of Computing Systems*, LNCS 1708, pp. 794–813, Springer-Verlag, 1999.

[17] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-key Protocol using FDR. In *Proc. of TACAS'96*, LNCS 1055, pp. 146–166, Springer, 1996.

[18] H. R. Nielson and F. Nielson. Shape Analysis for Mobile Ambients. In *Proc. of POPL'00*, pp.142–154, ACM Press, 2000.

[19] F. Nielson, H. Nielson and H. Seidl. Cryptographic Analysis in Cubic Time. This volume.

[20] F. Nielson, H. R. Nielson, R. R. Hansen, and J. G. Jensen. Validating Firewalls in Mobile Ambients. In *Proc. of CONCUR'99*, LNCS 1664, pp. 463–477, 1999.

[21] S. Stoller. A Reduction for Automated Verification of Authentication Protocols. In *Workshop on Formal Methods and Security Protocols (FMSP'99)*, Trento, Italy, 1999.