Non-Interference Proof Techniques for the Analysis of Cryptographic Protocols*

Michele Bugliesi, Ambra Ceccato, and Sabina Rossi

Dipartimento di Informatica, Università Ca' Foscari di Venezia,
via Torino 155, 30172 Venezia, Italy
e-mail:{bugliesi,ceccato,srossi}@dsi.unive.it

Abstract. We develop new proof techniques, based on Non-Interference, for the analysis of safety and liveness properties of cryptographic protocols expressed as terms of the process algebra CtyptoSPA. Our approach draws on new notions of behavioral equivalence, built on top of a context-sensitive labeled transition system that allows us to characterize the behavior of a process in the presence of any attacker with a given initial knowledge. We demonstrate the effectiveness of the approach with an example of a protocol of fair exchange.

1 Introduction

Non-Interference has been advocated by various authors [1, 5] as a powerful method for the analysis of cryptographic protocols. In [5], Focardi *et al.* propose a general schema for specifying security properties in a uniform and concise definition. The approach draws on earlier work by the same authors on characterizing information-flow security in terms of Non-Interference for the Security Process Algebra (SPA, for short). We briefly review the main ideas below.

SPA is a variant of CCS in which the set of actions is partitioned into two sets: L, for low, and H for high. A Non-Interference property $\mathcal{P}$ for a process E is expressed as follows:

$$E \in \mathcal{P} \text{ if } \forall \Pi \in \mathcal{E}_H : (E||\Pi) \setminus H \approx_{\mathcal{P}} E \setminus H \tag{1}$$

where $\mathcal{E}_H$ is the set of all high-level processes, $\approx_{\mathcal{P}}$ is an observation equivalence (dependent on $\mathcal{P}$), $||$ is parallel composition, and $\setminus$ is restriction. The processes $E \setminus H$ and $(E||\Pi) \setminus H$ represent the low-level views of E and of $E||\Pi$, respectively. The basic intuition is expressed by the slogan "if no high-level process can change the low behavior, then no flow of information from high to low is possible".

In [5] this idea is refined to provide a general definition of security properties for cryptographic protocols described as terms of CryptoSPA, a process algebra that extends SPA with cryptographic primitives. Intuitively, the refinement amounts to viewing the participants to a protocol as low-level processes, while

* This work has been partially supported by the MURST project "Modelli formali per la sicurezza (MEFISTO)" and the EU project IST-2001-32617 "(MyThS)".

the high-level processes represent the external attackers. Then, Non-Interference implies that the attackers have no way to change the low behavior of the protocol.

There are two problems that need to be addressed to formalize this idea. A first problem is that the intruder should be assumed to have complete control over the public components of the network. Consequently, any step in a protocol involving a public channel should be classified as a high-level action. However, since a protocol specification is usually entirely determined by the exchange of messages over public channels, a characterization like (1) becomes trivial, as $(E||\Pi) \setminus H$ and $E \setminus H$ are simply the null processes. This is easily rectified by extending the protocol specification with low-level actions that are used to specify the desired security property.

A further problem arises from the formalization of the *perfect cryptography* assumption that is usually made in the analysis of logical properties of cryptographic protocols. In [5] this assumption is expressed by making the definition of Non-Interference dependent on the initial knowledge of the attacker and on a deduction system by which the attacker may compute new information. The initial knowledge ϕ includes private data (e.g., the enemy's private keys) as well as, of course, any piece of publicly available information, such as names of entities and public keys. Property (1) is thus reformulated for a protocol P as follows:

$$P \in \mathcal{P} \text{ if } \forall \Pi \in \mathcal{E}_H^\phi : (P||\Pi) \setminus H \approx_\mathcal{P} P \setminus H. \tag{2}$$

where $\mathcal{E}_H^\phi$ is the set of the high-level processes Π which can perform only actions using the public channel names and whose messages can be deduced from ϕ.

This framework is very general, and lends itself to the characterization of various security properties, obtained by instantiating the equivalence $\approx_\mathcal{P}$ in the schema above. Instead, it is less effective as a proof method, due to the universal quantification over the possible intruders Π in the class $\mathcal{E}_H^\phi$. In [5], the problem is circumvented by analyzing the protocol in presence of the "hardest attacker". However, the hardest attacker does not exist in general: in [5], this characterization is proved correct only for the class of relationships $\approx_\mathcal{P}$ that are behavioral preorders. Unfortunately, this class is somewhat restrictive, and does not include equivalences based on bisimulations. As a consequence, the proof method is not applicable for the analysis of liveness properties, such as *fairness*.

We partially rectify the problem by developing a technique which does not require us to exhibit an explicit attacker (nor, in particular, it requires the existence of a hardest attacker). Our approach draws on ideas from [3] to represent the attacker indirectly, in terms of a context-sensitive labeled transition system. The labeled transitions take the form $\phi \triangleright P \xrightarrow{a} \phi' \triangleright P'$, where ϕ represents the context's knowledge prior to the transition, and ϕ' is the new knowledge resulting from P performing the action a. Building on this labeled transition system we provide quantification-free characterizations for various instantiations of (2), specifically when $\approx_\mathcal{P}$ is instantiated to trace equivalence, and to weak bisimulation equivalence. This allows us to apply our technique to the analysis of security properties based on safety as well as liveness properties. We demonstrate the latter with an example of a protocol of *fair exchange*.

$$\frac{m \quad m'}{(m,m')}\ (\vdash_{pair}) \qquad \frac{(m,m')}{m}\ (\vdash_{fst}) \qquad \frac{(m,m')}{m'}\ (\vdash_{snd})$$

$$\frac{m \quad k}{\{m\}_k}\ (\vdash_{enc}) \qquad \frac{\{m\}_k \quad k^{-1}}{m}\ (\vdash_{dec})$$

Table 1. Inference system for message manipulation where $m, m' \in \mathcal{M}$ and $k, k^{-1} \in K$

2 The CryptoSPA Language

The *Cryptographic Security Process Algebra* (CryptoSPA) [5] is an extension of SPA [4] with cryptographic primitives and constructs for value passing. The syntax is based on the following elements: a set M of basic messages and a set K of encryption keys with a function $\cdot^{-1} : K \longrightarrow K$ such that $(k^{-1})^{-1} = k$; a set $\mathcal{M}$ of all messages, defined as the least set containing $M \cup K$ and closed under the deduction rules in Table 1; a set $\mathcal{C}$ of channels partitioned into two sets H and L of high and low channels, respectively; a function *Msg* which maps every channel c into the set of messages that can be sent and received on c; a set $\mathcal{L} = \{c(m) \mid m \in Msg(c)\} \cup \{\overline{c}m \mid m \in Msg(c)\}$ of visible actions and the set $Act = \mathcal{L} \cup \{\tau\}$ of all actions; a function $chan(a)$ which returns c if a is either $c(m)$ or $\overline{c}m$ and *void* otherwise; a set *Const* of constants. By an abuse of notation, we write $c(m), \overline{c}\,m \in H$ whenever $c, \overline{c} \in H$, and similarly for the set L.

The structure of CryptoSPA *terms* (or *processes*) is defined as follows:

$$\begin{aligned} P ::= \mathbf{0} \mid c(x).P \mid \overline{c}m.P \mid \tau.P \mid P+P \mid P|P \mid P \setminus C \mid P[f] \mid \\ \mid A(m_1,...,m_n) \mid [m=m']P;P \mid [\langle m_1...m_n\rangle \vdash_{rule} x]P;P \end{aligned}$$

where $x, m, m', m_1, \ldots, m_n$ are messages or variables that will be instantiated to messages. Both $c(x).P$ and $[\langle m_1...m_n\rangle \vdash_{rule} x]P;P'$ bind the variable x in P. Constants are defined as: $A(x_1,...,x_n) \stackrel{def}{=} P$, where P is a CryptoSPA process that may contain no free variables except $x_1, \ldots, x_n$, which must be distinct.

Intuitively, $\mathbf{0}$ is the empty process; $c(x).P$ waits for input m on channel c, and then behaves as $P[m/x]$ (i.e., P with all the occurrences of x substituted by m); $\overline{c}(m).P$ outputs m on channel c and continues as P; $P_1 + P_2$ represents the nondeterministic choice between P_1 and P_2; $P_1|P_2$ is the parallel composition of P_1 and P_2; $P \setminus C$ is like P but prevented from using channels in $C \subseteq \mathcal{C}$; in $P[f]$ every channel c is relabelled into $f(c)$; $A(m_1,...,m_n)$ behaves like the respective definition where the variables $x_1, \cdots, x_n$ are substituted with messages $m_1, \cdots, m_n$; $[m=m']P_1;P_2$ behaves as P_1 if $m=m'$ and as P_2 otherwise; $[\langle m_1...m_n\rangle \vdash_{rule} x]P_1;P_2$ tries to deduce an information z from the tuple $\langle m_1...m_n\rangle$ through rule $\vdash_{rule}$; if it succeeds then it behaves as $P_1[z/x]$, otherwise it behaves as P_2.

In formalizing the security properties of interest, we will find it convenient to rely on (an equivalent of) the *hiding* operator of CSP, noted P/C with P process and $C \subseteq \mathcal{C}$, which turns all actions using channels in C into internal τ's. This operator can be defined in CryptoSPA as follows: given any set $C \subseteq \mathcal{C}$, $P/C \stackrel{\mathrm{def}}{=} P[f_C]$ where $f_C(a) = a$ if $chan(a) \notin C$ and $f_C(a) = \tau$ if $chan(a) \in C$.

We denote by $\mathcal{E}$ the set of all CryptoSPA processes and by $\mathcal{E}_H$ the set of all high level processes, i.e., those constructed only using actions in $H \cup \{\tau\}$.

The operational semantics of CryptoSPA is defined in terms of the labelled transition system (LTS) in Table 2. Most of the transitions are standard, and sim-

$$(input)\quad \frac{m \in Msg(c)}{c(x).P \xrightarrow{c(m)} P[m/x]} \qquad (output)\quad \frac{m \in Msg(c)}{\overline{c}m.P \xrightarrow{\overline{c}(m)} P}$$

$$(tau)\quad \frac{}{\tau.P \xrightarrow{\tau} P} \qquad (+_1)\quad \frac{P_1 \xrightarrow{a} P_1'}{P_1 + P_2 \xrightarrow{a} P_1'}$$

$$(|_1)\quad \frac{P_1 \xrightarrow{a} P_1'}{P_1|P_2 \xrightarrow{a} P_1'|P_2} \qquad (|_2)\quad \frac{P_1 \xrightarrow{c(m)} P_1' \quad P_2 \xrightarrow{\overline{c}(m)} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'}$$

$$(=_1)\quad \frac{m \neq m' \quad P_2 \xrightarrow{a} P_2'}{[m = m']P_1; P_2 \xrightarrow{a} P_2'} \qquad (=_2)\quad \frac{m = m' \quad P_1 \xrightarrow{a} P_1'}{[m = m']P_1; P_2 \xrightarrow{a} P_1'}$$

$$([f])\quad \frac{P \xrightarrow{a} P'}{P[f] \xrightarrow{f(a)} P'[f]} \qquad (\backslash C)\quad \frac{P \xrightarrow{a} P' \quad chan(a) \notin C}{P \setminus C \xrightarrow{a} P' \setminus C}$$

$$(constant)\quad \frac{P[m_1/x_1, \ldots, m_n/x_n] \xrightarrow{a} P' \quad A(x_1, \ldots, x_n) \stackrel{def}{=} P}{A(m_1, \ldots, m_n) \xrightarrow{a} P'}$$

$$(\vdash_1)\quad \frac{\langle m_1, \ldots, m_n \rangle \vdash_{rule} m \quad P_1[m/x] \xrightarrow{a} P_1'}{[\langle m_1, \ldots, m_n \rangle \vdash_{rule} x]P_1; P_2 \xrightarrow{a} P_1'}$$

$$(\vdash_2)\quad \frac{\nexists m : \langle m_1, \ldots, m_n \rangle \vdash_{rule} m \quad P_2 \xrightarrow{a} P_2'}{[\langle m_1, \ldots, m_n \rangle \vdash_{rule} x]P_1; P_2 \xrightarrow{a} P_2'}$$

Table 2. The operational rules for CryptoSPA

ply formalize the intuitive semantics of the process constructs discussed above. The two transition rules ($\vdash_i$) rely on the deduction rules in Table 1 that provide the operational import of the cryptographic primitives. Specifically, the deduction system will be used to model the ability of the attacker to deduce new information from its initial knowledge: note, in particular, that secret keys, not initially known to the attacker, may not be deduced (hence we disregard cryptographic attacks, based on guessing secret keys). We say that m is *deducible* from a set of messages ϕ (and write $\phi \vdash m$) if m can be obtain from ϕ by applying the inference rules in Table 1. As in [5] we assume that $\vdash$ is decidable.

We complement the definition of the semantics with a corresponding notion of *observation equivalence*, which is used to establish equalities among processes. The equivalences that are relevant to the present discussion are *trace equivalence*, noted $\approx_T$, and *weak bisimulation*, noted $\approx_B$. In the next section, we introduce coarser versions of these equivalences, noted $\approx_T^\phi$ and $\approx_B^\phi$, which distinguish processes in contexts with initial knowledge ϕ. These notions of equivalence are built on a refined version of the labeled transition system, which we introduce next.

3 Context-Sensitive Equivalences

Following [3], we characterize the behavior of processes in terms of "context-sensitive labelled transitions" where each process transition depends on the knowledge of the context. To motivate, consider a process P that produces and sends a message $\{m\}_k$ reaching the state P', and assume that m and P are known to P but not to the context. Under these hypotheses, the context will never be able to reply the message m to P' (or any continuation thereof). Hence, if P' waits for further input, we can safely leave any input transition involving m out of the LTS, as the P' will never receive m from the context.

The states of the new labeled transition system are *configurations* of the form $\phi \triangleright P$, where P is a process and ϕ is the current knowledge of the context, represented through a set of messages. Transitions represent interactions between the process and the context and now take the form

$$\phi \triangleright P \xrightarrow{a} \phi' \triangleright P'$$

where a is the action executed by the process P and ϕ' is the new knowledge at disposal to the context for further interactions with P'.

The transitions between configurations, in Table 3, are defined rather directly starting from the corresponding transitions between processes. In the rule (*output*), the context's knowledge is augmented with the information sent by the process. Dually, the rule (*input*) assumes that the context performs an output action synchronizing with the input of the process. The message sent by the context must be completely deducible from the context's knowledge ϕ, otherwise the corresponding transition is impossible. This is how the new transitions provide an explicit account of the attacker's knowledge. The remaining rules, (*tau*) and (*low*) state that internal actions of the protocol, and low actions do not contribute to the knowledge of the context in any way.

$$(output)\ \frac{P \xrightarrow{\overline{c}m} P' \quad \overline{c}m \in H}{\phi \rhd P \xrightarrow{\overline{c}(m)} \phi \cup \{m\} \rhd P'} \qquad (input)\ \frac{P \xrightarrow{c(m)} P' \quad \phi \vdash m \quad c(m) \in H}{\phi \rhd P \xrightarrow{c(m)} \phi \rhd P'}$$

$$(tau)\ \frac{P \xrightarrow{\tau} P'}{\phi \rhd P \xrightarrow{\tau} \phi \rhd P'} \qquad (low)\ \frac{P \xrightarrow{a} P' \quad a \in L}{\phi \rhd P \xrightarrow{a} \phi \rhd P'}$$

Table 3. Inference rules for the ELTS

In the rest of the presentation, we refer to the transition rules in Table 3 collectively as the *enriched LTS* (*ELTS*, for short). Also, we assume that the initial context's knowledge is restricted to the context's private variables and public ones. This is a reasonable condition, since it simply corresponds to assuming that each protocol run starts with fresh keys and nonces, a condition that is readily guaranteed by relying on time-dependent elements (e.g., time-stamps) and assuming that session keys are distinct for every executions.

The notions of trace and weak bisimulation equivalences extend in the expected way from processes to ELTS configurations. In the following $\phi \rhd P \stackrel{a}{\Longrightarrow} \phi' \rhd P'$ is a short for $\phi \rhd P\ (\xrightarrow{\tau})^* \ \phi \rhd P_1 \ \xrightarrow{a} \ \phi' \rhd P_2\ (\xrightarrow{\tau})^* \ \phi' \rhd P'$, where $\phi = \phi'$ if $\xrightarrow{a}$ is an input, low or silent action. Suppose $\gamma = a_1 \dots a_n \in \mathcal{L}^*$ is a sequence of (non silent) actions, then $\phi \rhd P \stackrel{\gamma}{\Longrightarrow} \phi' \rhd P'$ if $\phi \rhd P \stackrel{a_1}{\Longrightarrow} \phi_1 \rhd P_1 \stackrel{a_2}{\Longrightarrow} \dots \stackrel{a_{n-1}}{\Longrightarrow} \phi_{n-1} \rhd P_{n-1} \stackrel{a_n}{\Longrightarrow} \phi' \rhd P'$. The notation $\phi \rhd P \stackrel{\hat{a}}{\Longrightarrow} \phi' \rhd P'$ stands for $\phi \rhd P \stackrel{a}{\Longrightarrow} \phi' \rhd P'$ if $a \in \mathcal{L}$ and for $\phi \rhd P\ (\xrightarrow{\tau})^* \ \phi \rhd P'$ if $a = \tau$.

Definition 1 (Trace Equivalence over configurations). $T(\phi \rhd P) = \{\gamma \in \mathcal{L}^* \mid \exists \phi', P' : \phi \rhd P \stackrel{\gamma}{\Longrightarrow} \phi' \rhd P'\}$ *is the set of* traces *associated with a configuration* $\phi \rhd P$*. Two configurations* $\phi_P \rhd P$ *and* $\phi_Q \rhd Q$ *are* trace equivalent, *denoted by* $\phi_P \rhd P \approx^c_T \phi_Q \rhd Q$*, if* $T(\phi_P \rhd P) = T(\phi \rhd_Q Q)$.

Based on the notion of trace equivalence over configurations we define a corresponding notion of process equivalence, for processes executing in an environment with initial knowledge ϕ. Formally, $P \approx^{\phi}_T Q$ whenever $\phi \rhd P \approx^c_T \phi \rhd Q$.

Definition 2 (Weak Bisimulation over configurations). *A binary relation* $\mathcal{R}$ *over configurations is a weak bisimulation if* $(\phi_P \rhd P, \phi_Q \rhd Q) \in \mathcal{R}$ *implies that, for all* $a \in Act$,

- *if* $\phi_P \rhd P \xrightarrow{a} \phi_{P'} \rhd P'$*, then there exists a configuration* $\phi_{Q'} \rhd Q'$ *such that* $\phi_Q \rhd Q \stackrel{\hat{a}}{\Longrightarrow} \phi_{Q'} \rhd Q'$ *and* $(\phi_{P'} \rhd P', \phi_{Q'} \rhd Q') \in \mathcal{R}$*;*
- *if* $\phi_Q \rhd Q \xrightarrow{a} \phi_{Q'} \rhd Q'$*, then there exists a configuration* $\phi_{P'} \rhd P'$ *such that* $\phi_P \rhd P \stackrel{\hat{a}}{\Longrightarrow} \phi_{P'} \rhd P'$ *and* $(\phi_{P'} \rhd P', \phi_{Q'} \rhd Q') \in \mathcal{R}$.

Two configurations $\phi_P \rhd P$ *and* $\phi_Q \rhd Q$ *are* weakly bisimilar, *noted* $\phi_P \rhd P \approx^c_B \phi_Q \rhd Q$*, if there exists a weak bisimulation containing* $(\phi_P \rhd P, \phi_Q \rhd Q)$*.*

It is not difficult to prove that relation $\approx^c_B$ is the largest weak bisimulation over configurations, and that it is an equivalence relation.

4 Non-Interference Proof Techniques

We show that the new definitions of behavioral equivalence may be used to construct effective proof methods, that are stated independently from the attacker itself, for various security properties within the general schema proposed in [5].

The first property we study, known as NDC, results from instantiating $\approx_{\mathcal{P}}$ in (2) (see the introduction) to the trace equivalence relation, $\approx_T$. As discussed in [5], this property is a generalization of the classical idea of Non-Interference to non-deterministic systems and can be used for analyzing different security properties of cryptographic protocols such as secrecy, authentication and integrity. NDC can readily be extended to account for the context's knowledge as follows:

Definition 3. $P \in NDC^{\phi}$ *if* $P \setminus H \approx_T (P||\Pi) \setminus H,\ \forall\ \Pi \in \mathcal{E}^{\phi}_H$.

A process P is NDC^{ϕ} if for every high level process Π with initial knowledge ϕ a low level user cannot distinguish P from $(P||\Pi)$, i.e., if Π cannot interfere with the low level execution of the process P.

Focardi *et al.* in [5] show that when ϕ is finite it is possible to find a most general intruder Top^{ϕ} so that verifying NDC^{ϕ} reduces to checking $P \setminus H \approx_T (P||Top^{\phi}) \setminus H$. Here we provide an alternative[1], quantification-free characterization of NDC^{ϕ}.

Theorem 1 (NDC$^{\phi}$). $P \in NDC^{\phi}$ *if and only if* $P \setminus H \approx^{\phi}_T P/H$*.*

More interestingly, our approach allows us to find a sound, proof method for the $BNDC^{\phi}$ property, which results from instantiating (2) in the introduction with the equivalence $\approx_B$ as follows:

Definition 4. $P \in BNDC^{\phi}$ *if* $P \setminus H \approx_B (P||\Pi) \setminus H,\ \forall \Pi \in \mathcal{E}^{\phi}_H$.

Being based on bisimulation, $BNDC^{\phi}$ is adequate for liveness properties such us *fairness*. As for NDC^{ϕ}, the definition falls short of providing a proof method due to the quantification over Π. Here, however, the problem may not be circumvented by resorting to a hardest attacker, as the latter does not exist, being there no (known) preorder on processes corresponding to weak bisimilarity.

What we propose here is a solution that relies on providing a coinductive (and quantification free) characterization of a sound approximation of NDC^{ϕ}, based on the following *persistent* version of $BNDC^{\phi}$. This is analogous to what is done in [4, 6], which provide sufficient conditions for $BNDC$ in the SPA language.

[1] An analogous result has been recently presented by Gorrieri *et al.* in [7] for a *timed* extension of CryptoSPA. We discuss it in Section 6.

Definition 5. *$P \in P\_BNDC^{\phi}$ if $P' \in BNDC^{\phi}$, $\forall P'$ reachable from P.*

P_BNDC^{ϕ} is the context-sensitive version of the P_BNDC property studied in [6]. Following the technique in [6], one can show that P_BNDC^{ϕ} is a sound approximation of $BNDC^{\phi}$ which admits elegant quantification-free characterizations. Specifically, like P_BNDC, P_BNDC^{ϕ} can be characterized both in terms of a suitable weak bisimulation relation "up to high-level actions", noted $\approx^{\phi}_{\setminus H}$, and in terms of unwinding conditions, as discussed next.

We use the transition relation $\stackrel{\hat{a}}{\Longrightarrow}_{\setminus H}$ which stands for $\stackrel{a}{\Longrightarrow}$ if $a \notin H$ and for $\stackrel{\hat{a}}{\Longrightarrow}$ or $\stackrel{\tau}{\Longrightarrow}$ if $a \in H$. Now, weak bisimulations up to H, are defined as weak bisimulations except that they allow a high action to be matched by zero or more high-actions. Formally:

Definition 6 (Weak Bisimulation up to H over configurations). *A binary relation $\mathcal{R}$ over configurations is a* weak bisimulation up to H *if $(\phi_P \rhd P, \phi_Q \rhd Q) \in \mathcal{R}$ implies that, for all $a \in Act$,*

- *if $\phi_P \rhd P \stackrel{a}{\longrightarrow} \phi_{P'} \rhd P'$, then there exists a configuration $\phi_{Q'} \rhd Q'$ such that $\phi_Q \rhd Q \stackrel{\hat{a}}{\Longrightarrow}_{\setminus H} \phi_{Q'} \rhd Q'$ and $(\phi_{P'} \rhd P', \phi_{Q'} \rhd Q') \in \mathcal{R}$;*
- *if $\phi_Q \rhd Q \stackrel{a}{\longrightarrow} \phi_{Q'} \rhd Q'$, then there exists a configuration $\phi_{P'} \rhd P'$ such that $\phi_P \rhd P \stackrel{\hat{a}}{\Longrightarrow}_{\setminus H} \phi_{P'} \rhd P'$ and $(\phi_{P'} \rhd P', \phi_{Q'} \rhd Q') \in \mathcal{R}$.*

Two configurations $\phi_P \rhd P$ and $\phi_Q \rhd Q$ are weakly bisimilar up to H, *denoted by $\phi_P \rhd P \approx^{c}_{\setminus H} \phi_Q \rhd Q$, if there exists a weak bisimulation up to H containing the pair $(\phi_P \rhd P, \phi_Q \rhd Q)$.*

Again, we can prove that the relation $\approx^{c}_{\setminus H}$ is the largest weak bisimulation up to H and it is an equivalence relation. Also, we can recover an associated relation over processes in a context with initial knowledge ϕ by defining $P \approx^{\phi}_{\setminus H} Q$ if and only if $\phi \rhd P \approx^{c}_{\setminus H} \phi \rhd Q$.

We can finally state the two characterizations of P_BNDC^{ϕ}. The former characterization is expressed in terms of $\approx^{\phi}_{\setminus H}$ as follows.

Theorem 2 (P_BNDC$^{\phi}$ 1). *$P \in P\_BNDC^{\phi}$ if and only if $P \setminus H \approx^{\phi}_{\setminus H} P$.*

The second characterization of P_BNDC^{ϕ} is given in terms of *unwinding conditions* which demand properties of individual actions.

Theorem 3 (P_BNDC$^{\phi}$ 2). *$P \in P\_BNDC^{\phi}$ if and only if for all $\phi_i \rhd P_i$ reachable from $\phi \rhd P$, if $\phi_i \rhd P_i \stackrel{h}{\longrightarrow} \phi'_i \rhd P'_i$ for $h \in H$, then $\phi_i \rhd P_i \stackrel{\hat{\tau}}{\Longrightarrow} \phi''_i \rhd P''_i$ such that $\phi'_i \rhd P'_i \setminus H \approx^{c}_{B} \phi''_i \rhd P''_i \setminus H$.*

Both the characterizations can be used for verifying the correctness of cryptographic protocols. We illustrate how that can be done in the next section.

5 An Example: the ASW fair exchange protocol

The ASW contract signing protocol [2] is used in electronic commerce to enable two parties, named O (originator) and R (responder), to obtain each other's commitment on a previously agreed contractual text M. To deal with unfair situations, each party may appeal to a trusted third party T.

Here, we focus on the main *exchange* sub-protocol that is specified by the following four messages, where M is the contractual text on which we assume the two parties previously agreed, while SK_O and SK_R (PK_O and PK_R) are the private (public) keys of O and R, respectively.

$$\begin{array}{l}
O \to R : me_1 = \{M, h(N_O)\}_{SK_O} \\
R \to O : me_2 = \{\{M, h(N_O)\}_{SK_O}, h(N_R)\}_{SK_R} \\
O \to R : me_3 = N_O \\
R \to O : me_4 = N_R
\end{array}$$

In the first step, O commits to the contractual text by hashing a random number N_O, and signing a message that contains both $h(N_O)$ and M. While O does not actually reveal the value of its contract authenticator N_O to the recipient of message me_1, O is committed to it. As in a standard commitment protocol, we assume that it is not computationally feasible for O to find a different number N'_O such that $h(N'_O) = h(N_O)$. In the second step, R replies with its own commitment. Finally, O and R exchange the actual contract authenticators.

We specify the sub-protocol in CryptoSPA (see the figure below), by introducing some low level actions to verify the correctness of protocol's executions. We can say that an execution is correct if we observe the sequence of low-level actions $\overline{received}\_me_1$, $\overline{received}\_me_2$, $\overline{received}\_N_O$, $\overline{received}\_N_R$ in this order.

$$\begin{array}{rl}
O(M, N_O) \stackrel{def}{=} & [\langle N_O, k_h\rangle \vdash_{enc} n][\langle (M, n), SK_O\rangle \vdash_{enc} p]\ \overline{c}p.\ c(v). \\
& [\langle v, PK_R\rangle \vdash_{dec} i][i \vdash_{fst} p'][i \vdash_{snd} r'][p' = p]\ \overline{received_v}. \\
& \overline{c}N_O.\ c(j).\ [\langle j, k_h\rangle \vdash_{enc} r''][r'' = r']\ \overline{received_j} \\
R(M, N_R) \stackrel{def}{=} & c(q).\ [\langle q, PK_O\rangle \vdash_{dec} s][s \vdash_{fst} m][s \vdash_{snd} n'][m = M]\ \overline{received_q}. \\
& [\langle N_R, k_h\rangle \vdash_{enc} r][\langle (q, r), SK_R\rangle \vdash_{enc} t]\ \overline{c}t.\ c(u). \\
& [\langle u, k_h\rangle \vdash_{enc} n''][n'' = n']\ \overline{received_u}.\ \overline{c}N_R \\
P \stackrel{def}{=} & O(M, N_O) \mid\mid R(M, N_R)
\end{array}$$

Fig. 1. The CryptoSPA specification of the *exchange* sub-protocol of ASW

The protocol does not satisfy property P_BNDC^{ϕ} when ϕ consists of public information and private data of possible attacker's. This can be easily checked through Theorem 3. However, the fact that, in this case, the ASW protocol does not satisfy property P_BNDC^{ϕ} does not represent a real attack to the protocol since such a situation is resolved by inching the trusted party T mentioned above.

More interestingly, we can analyze the protocol under the assumption that one of the participants is corrupt. This can be done by augmenting the knowledge ϕ with the corrupt party's private information such us its private key and its contract authenticator. We can show that the protocol does not satisfy P_BNDC^{ϕ} when O is corrupt, finding the attack already described in [8].

6 Conclusions and Related Work

We have studied context-sensitive equivalences and relative proof techniques within the process algebra CryptoSPA. Our approach builds on context-sensitive labelled transition systems, where transitions are constrained by the knowledge of the environment. We showed that our technique can be used to analyze both safety and liveness properties of cryptographic protocols.

In a recent paper Gorrieri *et al.* [7] prove results related to ours, for a real-time extension of CryptoSPA. They prove an equivalent of our Theorem 1, but with a different proof technique. More precisely, instead of using context-sensitive LTS's, [7] introduces a special hiding operator $/^{\phi}$ and prove that $P \in NDC^{\phi}$ iff $P \setminus H \approx_T P/^{\phi}H$. Process $P/^{\phi}H$ corresponds exactly to configuration $\phi \rhd P/H$, in the sense that their LTS's are isomorphic. However, the approach of [7] is still restricted to the class of observation equivalences that are behavioral preorders on processes and thus it does not extend to equivalences based on bisimulations.

References

1. M. Abadi. Security Protocols and Specifications. In *Proc. of the Second International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'99)*, vol. 1578 of *LNCS*, pages 1–13. Springer, 1999.
2. N. Asokan, V. Shoup, and M. Waidener. Asynchronuous Protocols for Optimistic Fair Exchange. In *Proc. of the IEEE Symposium on Research in Security and Privacy*, pages 86–99. IEEE Computer Society Press, 1998.
3. M. Boreale, R. De Nicola, and R. Pugliese. Proof Tecniques for Cryptographic Processes. In *Proc. of the 14th IEEE Symposium on Logic in Computer Science (LICS'99)*, pages 157–166. IEEE Computer Society Press, 1999.
4. R. Focardi and R. Gorrieri. Classification of Security Properties (Part I: information flow). In *Foundations of Security Analysis and Design*, vol. 2171 of *LNCS*. 2001.
5. R. Focardi, R. Gorrieri, and F. Martinelli. Non Interference for the Analysis of Cryptographic Protocols. In *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP'00)*, vol. 1853 of *LNCS*, pages 744–755. Springer, 2000.
6. R. Focardi and S. Rossi. Information Flow Security in Dynamic Contexts. In *Proc. of the 15th IEEE Computer Security Foundations Workshop*, pages 307–319. IEEE Computer Society Press, 2002.
7. R. Gorrieri, E. Locatelli, and F. Martinelli. A Simple Language for Real-time Cryptographic Protocol Analysis. In *Proc. of 12th European Symposium on Programming Languages and Systems*, LNCS. Springer, 2003. To appear.
8. V. Shmatikov and J. C. Mitchell. Analysis of a Fair Exchange Protocol. In *Proc. of 7th Annual Symposium on Network and Distributed System Security (NDSS 2000)*, pages 119–128. Internet Society, 2000.