Classification of Security Properties *

(Part II: Network Security)

Riccardo Focardi[1], Roberto Gorrieri[2], and Fabio Martinelli[3]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy
focardi@dsi.unive.it
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy
gorrieri@cs.unibo.it
[3] Istituto di Informatica e Telematica, C.N.R., Pisa, Italy
Fabio.Martinelli@iit.cnr.it

Abstract. Many security properties of cryptographic protocols can be all formalized as specific instances of a general scheme, called Generalized Non Deducibility on Composition (*GNDC*). This scheme derives from the *NDC* property we proposed a few years ago for studying information flow in computer systems. The theory is formulated for CryptoSPA, a process algebra we introduced for the specification of cryptographic protocols. One of the advantages of our unifying *GNDC*-based theory is that that formal comparison among security properties become easier, being them all instances of a unique general property. Moreover, the full generality of the approach has helped us in finding a few undocumented attacks on cryptographic protocols.

This paper is based on the results of [20, 22–25] and covers the second part of the course "Classification of Security Properties" given by Roberto Gorrieri and Riccardo Focardi at the FOSAD'00 and FOSAD'01 schools.

1 Introduction

Many security properties of cryptographic protocols have been identified in recent years, such as *secrecy* (confidential information should be available only to the partners of the communication), *authentication* (capability of identifying the other partner engaged in a communication), *integrity* (assurance of no alteration of message content), *non repudiation* (assurance that a signed document cannot be repudiated by the signer), *fairness* (in a contract, no party can obtain advantage by ending the protocol first), and some others.

Even if there is a widespread agreement on what is the intended meaning of these properties, under a closer scrutiny one realizes that they are very slippery

* Work partially supported by the EU Contract IST-2001-32617 MyThs, the MIUR project "Formal Methods for Security" (MEFISTO) and Microsoft Research Europe. The third author is also supported by the project "Model based design/validation for Web Services (MbDWS)".

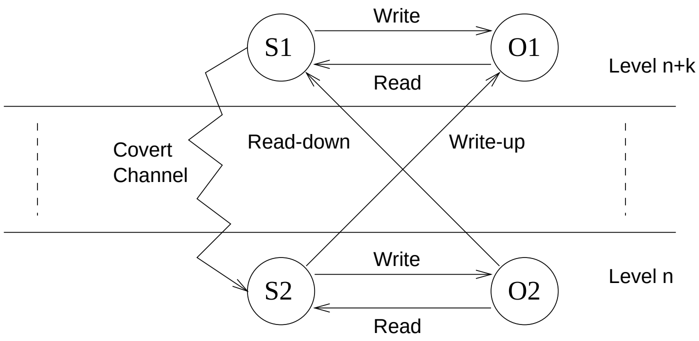

Fig. 1. Multilevel security: a high subject S_1 cannot write a low object O_2 and a low subject S_2 cannot read a high object O_1.

properties, especially authentication. As a matter of fact, formal definitions of, e.g. authentication, have been given rarely, not widely agreed upon, usually not compared and only recently proposed in the literature (see, e.g., [5, 27, 33, 42]). This is sometimes due to the fact that we first need a formal model on which the problem is defined (and this is often a source of possible proliferation of different proposals) and then a formal definition with respect to the chosen model. Moreover, even when a formal definition is given, usually this is not (easily) comparable to others, due to different mathematical assumptions of the model.

Our claim is that a classic approach to security, used to study information flow in multilevel [6] computer systems, can be profitably used also for the analysis of security properties in network protocols.

1.1 Multilevel security and non interference

In a multilevel systems, processes/users and objects are bound to a specific security level (e.g., in the military jargon, unclassified, classified, secret and top secret) and information can only flow from low levels to higher ones. This is usually implemented by constraining the possible actions of processes according to the rules of *no read-up* and *no write-down* (see Fig. 1).

The advantage of this approach with respect to conventional approaches used in commercially available operating systems (e.g., Unix) is that the possible information disclosures caused by the inadvertent execution of a Trojan Horse program is confined inside the level of the user that executed it. However, these two rules are not enough as indirect information flows, usually called *covert channels*, may be possible when using some shared resource. For instance, it is not difficult to build a Trojan Horse program that, once executed by a high level user, is able to downgrade information by synchronizing with a low level process on the system side-effects generated by repeatedly filling the shared hard disk: the high process can transmit a bit 0 by causing a disk-full error on the low level attempt to write, or a bit 1 by allowing the low process to write.

To solve the problem of preventing unauthorized information flows, be they direct or indirect, in the last two decades many proposals have been presented, starting from the seminal idea of *non interference* proposed in [26] for deterministic state machines. In recent work [17–19], two of the authors have studied the many non interference-like definitions in the literature, by defining all of them uniformly in a common process algebraic setting, producing the first taxonomy of these security properties reported in the literature.

In [17–19], we use a CCS-like process algebra [38], called Security Process Algebra (SPA for short), where the set of actions is partitioned into two sets L and H of low actions and high ones, respectively. Processes built by using only actions in H (L) are called high (low) level processes. These processes are secure by construction, because their activities (expressed by the actions they perform) are confined inside the high (low) level. More interesting is the case of *mixed* (built with actions from both L and H) processes, because only for them information can flow between the two levels. Of course, we are interested in detecting if information flow in the wrong direction (from high to low). Among the many non interference-like properties, we advocate a particular one, called *Non Deducibility on Composition* (*NDC* for short), that can be expressed as follows:

$$P \in NDC \text{ iff } \forall \Pi \in \mathcal{E}_H : (P \,\|\, \Pi)\backslash H \approx P\backslash H$$

where $\mathcal{E}_H$ is the set of all high level processes, $\approx$ is a behavioural equivalence [1] relation, $\|$ is the CCS parallel composition and $\backslash$ is the CCS restriction operator. Hence, on the one hand, $P\backslash H$ is able to exhibit only the low level behaviour of P, while $(P \,\|\, \Pi)\backslash H$ is the low level behaviour of $P \,\|\, \Pi$. The basic intuition is that the requirement of

No information flow from high to low

is expressed by the extensional, behavioural condition

No high level process can change the low behaviour.

1.2 Non interference for security protocols

When considering network security, other kinds of problems become relevant. In particular, communication protocols are executed on unreliable networks (hence, messages can be lost or duplicated) which are also insecure (hence, messages can be intercepted, fabricated and possibly modified by third, malicious parties that can control the network traffic). In order to achieve reasonable security services (e.g., secrecy of the transmitted data), network protocols typically exploits cryptographic primitives. Such enhanced protocols are usually called *cryptographic protocols.*

In the analysis of cryptographic protocols, one has to cope with the insecurity of the network. So, it is assumed that an external, possibly malicious, *attacker* (sometimes called *enemy* or *intruder*) of the protocol has complete control over

[1] Actually, *NDC* in [17, 18] is this property when $\approx$ is trace equivalence; variations on the theme are obtained by changing the relation, e.g., $BNDC$ is as above where $\approx$ is weak bisimulation, as we will see in the sequel.

the communication medium. On the other hand, it is also usually assumed *perfect cryptography*, i.e., such an enemy is not able to perform cryptanalytic attacks: an encrypted message can be decrypted by the enemy only if he knows (or is able to learn) the relevant decryption key. Such an analysis scenario is often referred to as the Dolev-Yao approach [12].

In order to model cryptographic protocols, the SPA process algebra is enriched with primitives for handling messages and cryptographic functions. The resulting process algebra is called CryptoSPA [25].

Correspondingly, the *NDC*-based analysis technique for security in computer systems has to be adapted. *NDC* essentially says that a system P is secure if its low behaviour in isolation is the same as its low behaviour when exposed to the interaction with any high level process Π. Analogously, we may think that a protocol P is secure if its (low) behaviour is the same as its (low) behaviour when exposed to the possible attacks of any intruder X. To set up the correspondence, this analogy forces to consider the enemies as the high processes. Since the enemy has complete control over the communication medium, the CryptoSPA *public* channels in set C (i.e., the names used for message exchange) are the high level actions. On the other hand, as a protocol specification is usually completely given by message exchanges, it may be not obvious what are the low level actions. In our approach, the low level actions are extra observable actions that are included into the protocol specification to observe properties of the protocol. Of course, the choice of these extra actions (and the place into the specification where they are to be inserted) is property dependent. For instance, we will see that to model some form of authentication as in [32], it is enough to include special start/commit actions for all the honest participants.

Furthermore, enemies should not be allowed to know secret information in advance: as we assume perfect cryptography, the initial knowledge of an enemy must be limited to include only publicly available information, such as names of entities and public keys, and its own private data (e.g., enemy's private key). Hence, by following [25, 24, 22], the set $\mathcal{E}_C^{\phi_I}$ of all the admissible attackers is as follows: $\mathcal{E}_C^{\phi_I} = \{X \mid sort(X) \subseteq C \text{ and } ID(X) \subseteq \mathcal{D}(\phi_I)\}$, where C is the set of public channel names, $sort(X)$ is the set of channel names syntactically occurring in X, $ID(X)$ is the set of messages that syntactically appear in X, ϕ_I is the initial knowledge given to any enemy X, and $\mathcal{D}$ is a deduction system that manipulates (blocks of) messages in the obvious way (e.g., a crypted information can be disclosed if the decryption key is known). By requiring that all the messages in $ID(X)$ are deducible from ϕ_I we are stating that the enemy cannot know in advance messages that are not derivable by the explicitly given set ϕ_I. The *NDC* property for a protocol P can hence be reformulated as:

$$P \in NDC_C^{\Phi_I} \text{ iff } \forall X \in \mathcal{E}_C^{\phi_I} : (P \parallel X) \setminus C \approx P \setminus C$$

On the one hand, $P \setminus C$ represents the secure specification of the protocol P running in isolation on perfectly secure channels. The visible behaviour of P is given by the property dependent, extra observable actions included into the specification. Hence, the behaviour of $P \setminus C$ describes the protocol in isolation, where the security property of interest holds. On the other hand, if $P \setminus C$ is equivalent to $(P \,\|\, X) \setminus C$, then this clearly means that X is not able to modify in any way the observable execution of P, i.e., the security property holds even when P is executed in any possible hostile environment with initial knowledge Φ_I. Intuitively, in the setting of cryptographic protocols, interferences may represent possible attacks. As a consequence, *non interference* guarantees the absence of attacks.

Interestingly enough, when the observational equivalence $\approx$ is trace equivalence (two systems are equivalent if they perform the same set of traces), then *NDC* can be characterized in a simpler way, by finding a canonical, most general enemy that can be used in place of all. By removing the universal quantification, *NDC* can be verified by one single, albeit huge, check. The most general enemy is an attacker that can eavesdrop/intercept any message (adding the intercepted information to its knowledge set), as well as produce new messages with pieces of information he knows.

1.3 The *GNDC* scheme

The Generalized *NDC* (*GNDC*) scheme [25, 23] relaxes, in some aspects, the *NDC* property presented above, in order to express uniformly many security properties. As we will see in the next sections, it is general enough to capture many different properties of cryptographic protocols. To give the flavour of how *GNDC* works, we present a very simple example of a key-exchange protocol. For such a protocol, we consider two authentication properties and we show how they can be formalized in terms of *observable events*. Finally, we show how such formalizations can be seen as instances of *GNDC*.

The simple key-exchange protocol we consider has the aim of distributing a fresh session-key K_s from *Alice* (A) to *Bob* (B) by using a long-term key K shared between such users. The new key K_s will be used by Alice and Bob to communicate inside the current session. Session-key distribution is a (standard) way to reduce the risk of cryptanalytic attacks on cryptographic keys, as it allows two parties to establish a new key for each new session.

$$A \to B : \{K_s, A\}_K$$

The notation $A \to B : msg$ above represents A sending message msg to B; moreover $\{M\}_K$ is used to denote the encryption of M with the key K.

In order for this protocol to be effective, it must intuitively provide both authentication and secrecy of the session key K_s. In particular, B should be guaranteed that K_s really comes from A and that nobody else knows it. Both of these properties should be guaranteed by the encryption of K_s with K. Here we focus on authentication and we consider two variants of it:

(i) *key authenticity* requires that key K_s is authentic from A; this should be guaranteed by the fact that only A and B know the long-term key K: when B decrypts the message he concludes that only A may have generated it.

In case this property is required on a generic message M (not necessarily a key), we refer to it as *message authenticity*;

(ii) *entity authentication* requires that, at the end of the protocol, B is convinced that A has been running the protocol with him; again, this should be guaranteed by the fact that only A and B know the long-term key K.

In the following we show that the two properties above do not hold on the simple protocol we are considering, when multiple sessions are considered.

Message Authenticity A simple way to check message authenticity is to fix in advance the key K_s that A is willing to distribute to B and require that in all the possible (concurrent) runs of the protocol B always receives the correct key K_s, i.e. the key that A wanted to send to B. If this is true, we can conclude that the protocol guarantees that no one is able to force B accepting a fake session key K'_s. This notion of message authenticity is due to Abadi and Gordon [2].

A first important thing to observe is that considering all the possible runs is not enough. At least we need to be more precise about what we mean by run. As a matter of fact, we have to consider all the possible executions of the protocol in every possible (potentially hostile) environment. It is certainly different if we consider the protocol execution with or without the presence of some malicious enemy which tries to send a fake K'_s. We can thus rephrase the message (key) authenticity property as follows:

> "Whatever hostile environment is considered, B will never receive (as part of Message 1), during all of his possible runs, a key which is different from K_s".

Notice that the property above looks really ad-hoc for the specific protocol we are considering. As a matter of fact, we are requiring that a particular piece of information sent inside a particular message differs from a certain fixed key K_s. This specificity is a quite typical aspect of precise definitions of security properties, since they often depend on the structure of the analyzed system or protocol. We can improve the generality of the property by assuming to have an event *received*(m) corresponding to the reception of message m by B and by denoting with $P(k)$ the protocol specification in which *Alice* is willing to send the session-key k to *Bob*. We can now state that

> "$P(K_s)$ guarantees message (key) authenticity if whatever hostile environment is considered, an event *received*(K'_s) with $K'_s \neq K_s$ can never occur".

We can now show that the simple protocol we have considered so far does not guarantee message authenticity, when more than one protocol session is possible. The flaw is due to a well-known *replay attack*, that allows the enemy to make B accept an old session key. This is very dangerous as the enemy could have time to break an old session key and force B using such a broken key again. Knowing the key, the enemy can completely impersonate A in the just established session. The attack sequence follows:

message 1a	$A \to B : \{K_s, A\}_K$	*The enemy stores this message*
...	...	...
message 1b	$\mathbf{E}(A) \to B : \{K_s, A\}_K$	*The enemy replays the old message*

$\mathbf{E}(A)$ denotes the enemy impersonating A.

To see why this attack is revealed by the definition of message authenticity we have given, it is sufficient to consider the situation in which two (sequential) instances of P are considered, with two different keys: $P(K_s^1), P(K_s^2)$. Now it is easy to see that $P(K_s^2)$ does not guarantee message authenticity. Indeed, the enemy may exploit the attack sequence above to make the B of $P(K_s^2)$ accept the session key K_s^1 exchanged in the first session $P(K_s^1)$. As a consequence B will produce the event $received(K_s^1)$ instead of the expected one $received(K_s^2)$. This may be depicted as follows:

message 1a	$A \to B : \{K_s, A\}_K$	$received(K_s^1)$
...	...	...
message 1b	$\mathbf{E}(A) \to B : \{K_s, A\}_K$	$received(K_s^1)$

Entity Authentication We now consider another security property: entity authentication. This property is more subtle. Informally, entity authentication should allow the verification of an entity's claimed identity, by another entity. There are several attempts in the literature to formalize this notion. Here, we follow the ones based on *correspondence* between actions of the participants (see, e.g., [28, 33, 45]).

As an example, in our protocol we would like that whenever B receives the protocol message (encrypted with the correct key) then A has indeed executed the protocol with him. To formalize this idea, we consider two events $commit(B, A)$ and $run(A, B)$ representing the fact that B has successfully terminated the protocol apparently with A and A has at least started the protocol with B. It is now sufficient to require that each event $commit(B, A)$ is always matched by an event $run(A, B)$ [33]. In other words $commit(B, A)$ should never happen if A has not started the protocol. We state entity authentication as follows:

> "P guarantees entity authentication of B with respect to A if, whatever hostile environment is considered, an event $commit(B, A)$ can never occur without a matching event $run(A, B)$".

Note that the same attack considered for message authenticity is also an attack for entity authentication:

message 1a	$A \to B : \{K_s, A\}_K$	$run(A, B)$, $commit(B, A)$
...	...	...
message 1b	$\mathbf{E}(A) \to B : \{K_s, A\}_K$	$commit(B, A)$

Notice that in the second message we have an event $commit(B, A)$ without the matching event $run(A, B)$, representing the fact that A is not running the protocol in the second session. This captures the replay attack performed by E, in which E impersonates A. We conclude that B cannot be guaranteed of the identity of the claimant, i.e., P does not guarantee entity authentication.

Remark 1. Entity authentication and message authenticity are different properties. As an example, if we do not have a message "to be sent" by the entity that we want to authenticate, message authenticity property becomes senseless. To see this point, consider the following (faulty) authentication protocol:

$$\begin{array}{lll} \text{message } 1 & A \to B : \{N_A\}_{K_{AB}} & run(B, A) \\ \text{message } 2 & B \to A : N_A & commit(A, B) \end{array}$$

In order to verify the identity of B, A sends a challenge N_A (typically a random number used only once, called nonce) to B encrypted with the symmetric key K_{AB} which is only known by A and B. Only B should be able to decrypt N_A and send it back to A. If A may play both the initiator and responder role in parallel sessions of the protocol, the following well-known *reflection attack* becomes possible:

$$\begin{array}{lrll} \text{message } 1 & A \to \mathbf{E}(B) & : \{N_A\}_{K_{AB}} & \\ \text{message } 1' & \mathbf{E}(B) \to A & : \{N_A\}_{K_{AB}} & \\ \text{message } 2' & A \to \mathbf{E}(B) & : N_A & \\ \text{message } 2 & \mathbf{E}(B) \to A & : N_A & commit(A, B) \end{array}$$

The enemy intercepts the first message and starts a new session of the protocol with A. Basically, the enemy uses this second session to obtain from A the value N_A. Finally the enemy can conclude the first session successfully masking as B. Note that, again, we have a $commit(A, B)$ event with no $run(B, A)$.

Notice that B has no private messages to send to A, thus it is not possible to express the attack above as an attack on message authenticity. Indeed, B does not send any message to A that should be authentic from him.

The general scheme We have seen that the two different properties of message authenticity and entity authentication, can be both specified by requiring that whatever hostile environment is considered, the protocol never shows some particular *bad behaviour*. We have also observed that this set of bad behaviours, in general, depends on the particular property and sometimes may also depend on the protocol P. For example, for message authenticity we need the parameter m of P in order to require that m is the message to be delivered. We take a complementary approach (that is more manageable), and we denote with $\alpha_S(P)$ the set of all possible good behaviour of P with respect to the security property S. Moreover we denote with $\gamma_S(P)$ the protocol P decorated with the suitable events of property S. Informally, our general scheme becomes the following:

> "P guarantees a security property S if, whatever hostile environment is considered, $\gamma_S(P)$ always shows behaviours in $\alpha_S(P)$".

The considerations above are at the base of the proposal of a uniform formal framework where security properties can be defined. The proposed schema, called Generalized *NDC* (*GNDC* for short), is as follows:

$$P \text{ is } GNDC_{\lhd}^{\gamma_S,\alpha_S} \text{ iff } \forall X \in \mathcal{E}_C^{\phi_I} : (\gamma_S(P) \parallel X) \setminus C \lhd \alpha_S(P)$$

where $\lhd$ is a behavioural preorder, i.e., $P \lhd P'$ means that all the behaviours shown by P are also shown by P'.

$GNDC_{\lhd}^{\gamma_S,\alpha_S}$ corresponds to the informal scheme sketched above. Indeed, $X \in \mathcal{E}_C^{\phi_I}$ selects enemies with a knowledge limited by ϕ_I. Then, the decorated protocol $\gamma_S(P)$ executed with X is required to show behaviours inside the set $\alpha_S(P)$ of good ones.

Given the *GNDC* scheme, we can define a specific property S by suitably instantiating the function $\alpha_S(P)$, the preorder $\lhd$ and the decoration function γ_S. We will see that many (network) security properties can be instantiated in the *GNDC* scheme. Here, we informally show how this is achieved for non interference (formalized as NDC), message authenticity and entity authentication:

- *Non-interference (NDC)*: $\alpha_{NI}(P) = P \setminus C$, $\lhd$ is $\approx$ and $\gamma_{NI}(P) = P$. We obtain the exact definition of *NDC*.
- *Message authenticity*: $\alpha_{MA}(P(M))$ is the process where event *received*(M') may only occur with $M' = M$. $\lhd$ is the trace-preorder, i.e., $P \lhd P'$ if all the execution sequences (i.e., trace) of P are execution sequences of P' and $\gamma_{MA}(P)$ is the process that suitably performs *received*(m), whenever the supposedly authentic message m is delivered by the protocol.
- *Entity Authentication*: $\alpha_{EA}(P)$ is the process where events *commit*(B, A) are always preceded by matching events *run*(A, B); $\lhd$ is the trace-preorder, as above, and $\gamma_{EA}(P)$ is the process that suitably performs *run*(A, B) and *commit*(B, A), whenever A starts the protocol with B and B concludes the protocol convinced to communicate with A, respectively.

1.4 Plan of the paper

The primary goal of this paper is to substantiate our claim that most (maybe all) security properties proposed for the analysis of cryptographic protocols are expressible as suitable instances of the *GNDC* schema above, by suitably choosing the property dependent (low) observable actions, the position where they are to be inserted into the specification, as well as a suitable behavioural equivalence. One interesting result that holds for trace semantics is that, once fixed the function γ, *NDC* is the strongest property in the $GNDC^{\alpha}$ family, for any choice of the function α.

We think that the advantages of our approach include at least the following:

- *formal comparison*: as the definitions are now given in a uniform style, it should be easier to compare the relative merits; this is especially true for slippery properties such as the many varieties of authentication.

- *one check for all*: As all the properties are defined in the same *NDC* style and *NDC* is the strongest property, it is possible to put into the specification the extra actions for all the properties of interest, hence obtaining that one successful *NDC* check for this rich case implies that all the properties are satisfied.
- *accuracy*: So far we have analyzed about 40 protocols (with the help of an automatic tool [13, 15]) of a well-known library of crypto-protocols [10]. Two supposedly correct protocols have been shown incorrect and for a few additional flawed protocols some new attacks have been found (see [14] for details). Our experience hence supports our claim that a protocol passing the *NDC* test is more likely to be flaw free.

The paper is organized as follows. Section 2 introduces the process algebra CryptoSPA that we use to formally define cryptographic protocols. Section 3 describes how the non interference approach, developed for SPA in [17–19], is rephrased over CryptoSPA. In particular, we study how to describe security properties by decorating the protocol specification with specific low level actions; we also study when an attacker (or enemy) is admissible (by imposing restriction on its initial knowledge) and, finally, the behavioural semantics for CryptoSPA we are interested in (trace, bisimulation), over which the *NDC* theory is parametrized. Section 4 addresses some technical problems we have to face in order to perform cryptographic protocol verification. For instance, the problem of finding the most general enemy, in order to remove the universal quantification in the *NDC* definition, as well as the problem of compositionality of secure cryptographic protocols. In Section 5 the general *GNDC* scheme is presented, as well as its instantiation to several security properties, including secrecy, message authenticity, entity authentication, non repudiation and fairness. Section 6 provides some formal comparison among security properties. Among the results reported there, we mention the proof that *NDC* is the strongest property in the *GNDC* family. Section 7 concludes the paper with some comparison with related work and some suggestions for future research.

2 The Model

In this section we describe the language we use for the specification of cryptographic protocols and their security properties. It is called *Cryptographic Security Process Algebra* (CryptoSPA for short), and is a variant of value-passing CCS [38], where the processes are provided with some primitives for manipulating messages [36]. In particular, processes can perform message encryption and decryption, and also construct complex messages by composing together simpler ones.

2.1 The CryptoSPA Syntax

The CryptoSPA syntax is based on the following elements:

- A set $I = \{a, b, \ldots\}$ of *input* channels and a set $O = \{\bar{a}, \bar{b}, \ldots\}$ of *output* channels, related through a function $\bar{\cdot} : I \cup O \to I \cup O$ which given an input $a \in I$ returns the *corresponding* output $\bar{a} \in O$ and vice-versa, i.e., $\bar{\bar{a}} = a$.
- A set M of basic messages. The set $\mathcal{M}$ of all messages is defined as the least set such that $M \subseteq \mathcal{M}$ and $\forall m, m', k \in \mathcal{M}$ we have that (m, m') (pairs) and $\{m\}_k$ (encryptions) also belong to $\mathcal{M}$.
- A set $C \subseteq I \cup O$ of channels, ranged over by c, such that $c \in C$ iff $\bar{c} \in C$; these *public* channels represent the insecure network on which the enemy can intercept and fake messages. Channels in $(I \cup O) \setminus C$ are the *private* channels.
- A function $Msg : I \cup O \longrightarrow \mathcal{P}(\mathcal{M})$ which maps every channel c into the set of possible messages that can be sent and received along such a channel. Msg is such that $Msg(c) = Msg(\bar{c})$.
- A set $Act = \{c(m) \mid c \in I, m \in Msg(c)\} \cup \{\bar{c}\, m \mid \bar{c} \in O, m \in Msg(c)\} \cup \{\tau\}$ of actions (τ is the internal, invisible action), ranged over by a (with abuse of notation).
- A set Var of *variables*, raged over by x.
- A set $Const$ of constants, ranged over by A.

The set $\mathcal{L}$ of CryptoSPA terms (or processes) is defined as follows:

$$\begin{aligned} P ::= \underline{0} \;\big|\; c(x).P \;\big|\; \bar{c}\,e.P \;\big|\; \tau.P \;\big|\; P + P \;\big|\; P \,\|\, P \;\big|\; P \setminus L \;\big|\; P[f] \;\big| \\ \big|\; [e = e']P; P \;\big|\; [\langle e_1 \ldots e_r \rangle \vdash_{rule} x]P; P \;\big|\; A(e_1, \ldots, e_n) \end{aligned}$$

where $e, e', e_1, \ldots, e_r$ are messages or variables, L is a set of input channels and $f : Act \mapsto Act$ is a function that relabels channel names inside actions[2] and may also relabel actions into the internal (invisible) action τ. Both the operators $c(x).P$ and $[\langle e_1 \ldots e_r \rangle \vdash_{rule} x]P; P'$ bind the variable x in P.

Most of the CryptoSPA operators are the same as in value-passing CCS. Intuitively, $\underline{0}$ is the empty process, which cannot do any action; $c(x).P$ reads a value v from channel c and then behaves as process P in which all the free occurrences of x are replaced by v; symmetrically, $\bar{c}\,e.P$ sends e as output on channel c then behaving as P; $\tau.P$ can do an internal action τ and then behaves like P; $P_1 + P_2$ can alternatively choose [3] to behave like P_1 or P_2 ; $P_1 \,\|\, P_2$ is the parallel composition of P_1 and P_2, where the executions of the two systems are interleaved, possibly synchronized on complementary input/output actions, producing an internal τ; $P \setminus L$ can execute all the actions P is able to do, provided that inputs and outputs are not performed over channels belonging to L; if P can execute action a, then $P[f]$ performs $f(a)$; finally, $[e = e']P_1; P_2$ behaves like P_1 if $e = e'$ and like P_2 otherwise.

Besides the above described standard value-passing CCS operators, we have an additional one that has been introduced in order to model message handling and cryptography. Informally, the $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]P_1; P_2$ process tries to

[2] The relabeling functions map channels in C to channels in C and channels in $(I \cup O) \setminus C$ to channels in $(I \cup O) \setminus C$.

[3] For notational convenience, we use sometimes the $\sum$ operator (indexed on a set) to represent a general n-ary (or even infinitary) sum operator.

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

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

Fig. 2. Inference System for message manipulation, where $m, m', k, k^{-1} \in \mathcal{M}$.

deduce a piece of information z from the tuple of messages $\langle m_1 \ldots m_r \rangle$ through one application of rule $\vdash_{rule}$; if it succeeds, then it behaves like $P_1[z/x]$, otherwise it behaves like P_2. See the next subsection for a more detailed explanation of derivation rules.

Finally, let Def : $Const \mapsto \mathcal{L}$ be a set of defining equations of the form $A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P$, where P may contain no free variables except $x_1, \ldots, x_n$, which must be distinct. Constants permit us to define recursive processes, but we have to be a bit careful in using them. A term P is *closed* with respect to Def if all the constants occurring in P are defined in Def (and, recursively, for their defining terms). A term P is *guarded* with respect to Def if all the constants occurring in P (and, recursively, for their defining terms) occur in a prefix context [38]. A term P is *finite* with respect to Def if the set of constants occurring in P or, recursively, in their defining terms is finite. We call $\mathcal{E}$ the set of all the CryptoSPA *closed* terms (i.e., with no free variables), that are closed, guarded and finite with respect to Def. Terms in $\mathcal{E}$ are usually called CryptoSPA *processes*. Let $sort(P)$ denote the *sort* of P, i.e., the set of channels occurring syntactically in P. With $\mathcal{E}_{\mathcal{C}}$ we denote the processes P such that $sort(P) \subseteq C$.

For the sake of readability, we always omit the termination $\underline{0}$ at the end of process specifications, e.g., we write a in place of $a.\underline{0}$. We also write $[m = m']P$ in place of $[m = m']P; \underline{0}$ and analogously for $[\langle m_1 \ldots m_r \rangle \vdash_{rule} x]P; \underline{0}$. Finally, we often replace constructive rules (encryption and pairing) with the resulting messages, e.g., we use $\overline{c}\,\{m\}_k$ as a shorthand for $[\langle m, k \rangle \vdash_{enc} x]\overline{c}\,x$.

2.2 The Operational Semantics of CryptoSPA

In order to model message handling and cryptography, CryptoSPA may be equipped with a set of suitable inference rules (inference system). Note that CryptoSPA syntax, its semantics and the results obtained herein are completely parametric with respect to the chosen inference system. For explanatory purposes, in Figure 2 we provide a simple inference system which is quite similar to those used by many authors (see, e.g., [32, 34]). However, it is possible (and easy) to adopt other rules, e.g., for modeling different kinds of cryptographic approaches as well as cryptographic weaknesses. We consider a function $\cdot^{-1} : \mathcal{M} \to \mathcal{M}$ which denotes, for each key k (i.e., a message possibly used as encryption key), the corresponding decryption key. Note that there are no rules

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

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

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

$$(\backslash L)\frac{P \xrightarrow{c(m)} P' \quad c \notin L}{P\backslash L \xrightarrow{c(m)} P'\backslash L} \quad (\mathcal{D}_1)\frac{\langle m_1 \dots m_r \rangle \vdash_{rule} m \qquad P_1[m/x] \xrightarrow{a} P_1'}{[\langle m_1 \dots m_r \rangle \vdash_{rule} x]P_1; P_2 \xrightarrow{a} P_1'}$$

$$(\mathcal{D}_2)\frac{\not\exists m : \langle m_1 \dots m_r \rangle \vdash_{rule} m \qquad P_2 \xrightarrow{a} P_2'}{[\langle m_1 \dots m_r \rangle \vdash_{rule} x]P_1; P_2 \xrightarrow{a} P_2'}$$

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

Fig. 3. Operational semantics (symmetric rules for $+_1$, $\backslash L$, $\|_1$ and $\|_2$ are omitted).

to obtain the message k^{-1} from k (and *vice-versa*). In particular the inference system can combine two messages obtaining a pair (rule $\vdash_{pair}$); it can extract one message from a pair (rules $\vdash_{fst}$ and $\vdash_{snd}$); it can encrypt a message m with a key k obtaining $\{m\}_k$ and, finally, decrypt a message of the form $\{m\}_k$ only if it has the corresponding (inverse) key k^{-1} (rules $\vdash_{enc}$ and $\vdash_{dec}$). As an example, process $[\langle \{m\}_k, k^{-1} \rangle \vdash_{dec} x]P_1; P_2$ decrypts message $\{m\}_k$ through the inverse key k^{-1} and behaves like $P_1[m/x]$, while $[\langle \{m\}_k, k' \rangle \vdash_{dec} x]P_1; P_2$ (with $k' \neq k^{-1}$) tries to decrypt the same message with the wrong inverse key k' and (since it is not permitted by $\vdash_{dec}$) it behaves like P_2.

Given an inference system, we say that a message m can be deduced from a set of messages ϕ whenever there exists a proof tree whose nodes are messages, such that the root is m, the leaves are contained in ϕ and each message in the tree may be obtained by applying a rule instance of the inference system whose premises are the descendants of the message in the tree. We consider a function $\mathcal{D}$, from finite sets of messages to sets of messages, such that $\mathcal{D}(\phi)$ is the set of messages that can be deduced from ϕ. We assume that $\mathcal{D}(\phi)$ is a decidable set.

Note that, in our model, we are assuming encryption as completely reliable. Thus, we do not allow any kind of cryptographic attack, e.g., the guessing of secret keys. Nevertheless, the goal of the theory we are going to present is to capture those attacks that can be carried out even if cryptography is completely reliable. The behavior of a CryptoSPA term is formally described by means of the *labeled transition system* $\langle \mathcal{E}, Act, \{\xrightarrow{a}\}_{a \in Act} \rangle$, where $\xrightarrow{a}_{a \in Act}$ is the least relation between CryptoSPA terms induced by the axioms and inference rules of Figure 3.

The behaviour of a specific term P is the portion of the above labelled transition system, which is reachable starting from the state P. We write $P \equiv P'$ if P and P' show exactly the same behaviour, i.e., if they have isomorphic labelled transition systems, up to state renaming.

Example 1. We give a simple example of CryptoSPA protocol specification. Consider again the (flawed) key exchange protocol we presented in section 1.3:

$$A \to B : \{K_s, A\}_K$$

It can be specified as the following CryptoSPA process P:

$$P = \overline{c}\,\{K_s, A\}_K.A'(K_s) \parallel c(y).[\langle y, K\rangle \vdash_{dec} w][w \vdash_{fst} z].B'(z)$$

where $K^{-1} = K$ (symmetric encryption) and $Msg(c) = \mathcal{M}$. Moreover, $A'(K_s)$ and $B'(z)$ represent the *continuations* of A and B, respectively, that will presumably communicate using the just established session key K_s. Notice that in $B'(z)$, variable z is bound to the received session key.

We want to analyze the execution of P with no intrusion; we thus consider $P \setminus \{c\}$, since the restriction guarantees that c is now a secure channel between A and B (no external attacker can access channel c). We obtain a process whose only possible execution is the correct one where A sends to B message $\{K_s, A\}_K$:

$$\begin{aligned} P \setminus \{c\} &\xrightarrow{\tau} (A'(K_s) \;\parallel [\langle \{K_s, A\}_K, K\rangle \vdash_{dec} w][w \vdash_{fst} z]B'(z)) \setminus \{c\} \\ &\equiv \; (A'(K_s) \;\parallel\; B'(K_s)) \setminus \{c\} \end{aligned}$$

Notice that, after the message exchange, the two continuations $A'(K_s)$ and $B'(K_s)$ share the correct session key K_s.

3 Non-interference for CryptoSPA

In this section we want to rephrase the non-interference theory developed for SPA to the richer setting of CryptoSPA. The following subsections are devoted (i) to illustrate how security properties can be specified by decorating suitably protocol specification, then (ii) to discuss the actual definition of admissible attackers and, finally, (iii) to study behavioural semantics for CryptoSPA over which the *GNDC* theory is parametrized.

3.1 Decorating protocol specifications

The first problem is to understand what are the high level actions and the low level ones in this setting. *NDC* essentially says that a system P is secure if its low behaviour in isolation is the same as its low behaviour when exposed to the interaction with any high level process Π. Analogously, we may think that a protocol P is secure if its (low) behaviour is the same as its (low) behaviour when exposed to the possible attacks of any intruder X.

To set up the correspondence, this analogy forces to consider the enemies as the high processes. Since the enemy has complete control over the communication medium, the CryptoSPA *public* channels in set C (i.e., the names used for message exchange) are the high level actions while the *private channels* in set $(I \cup O) \setminus C$ are the low level ones. As a protocol specification is usually completely given by message exchanges, it may be not obvious what are the low level actions. In our approach, they are extra observable actions that are included into the protocol specification to observe properties of the protocol. Of course, the choice of these extra actions (and the place into the specification where they are to be inserted) is property dependent, as we will see in the next section.

This idea of decorating a protocol with extra low level actions is formalized by considering a function γ, such that $\gamma(P)$ represents protocol P decorated with suitable low level actions. The decoration function γ should not affect the protocol behaviour, as its only purpose is to allow observing such a behaviour. This is achieved by explicitly requiring that $\gamma(P)$ behaves as P once the low level extra actions have been hidden (i.e., transformed into internal non-observable actions).

Let $f_{((I \cup O) \setminus C)}$ be the relabelling function that maps all the actions in $((I \cup O) \setminus C)$ into τ and let $\approx$ denote a suitable behavioural equivalence such that $P \approx P'$ only if P and P' behaves the same (equivalences of this kind will be formally defined in section 3.3).

Definition 1. *A function $\gamma : \mathcal{E}_C \rightarrow \mathcal{E}$ is a* decoration function *if for every CryptoSPA protocol P such that $sort(P) \subseteq C$, we have $P \approx \gamma(P)[f_{((I \cup O) \setminus C)}]$.*

We will see that the requirement above for γ functions may be easily achieved by considering γ's that decorate the protocols by only adding output actions, i.e., actions that simply make public some internal protocol values and do not modify the protocol execution by reading new values.

Example 2. Consider again the CryptoSPA protocol specification of example 1:

$$P = \overline{c}\,\{K_s, A\}_K.A'(K_s) \parallel c(y).[\langle y, K\rangle \vdash_{dec} w][w \vdash_{fst} z].B'(z)$$

Recall that it models the (flawed) key exchange protocol we presented in section 1.3:

$$A \rightarrow B : \{K_s, A\}_K$$

In such a section we discussed how both message authenticity and entity authentication might be verified by observing the suitable events $received(K_s)$ and $run(A, B), commit(B, A)$, respectively. Here we show how such events may be added to the specification P. We consider two decoration functions γ_{MA} and γ_{EA} such that:

$$\begin{aligned}
\gamma_{MA}(P) &= \overline{c}\,\{K_s, A\}_K.A'(K_s) \parallel c(y).[\langle y, K\rangle \vdash_{dec} w][w \vdash_{fst} z].\overline{received}(z).B'(z) \\
\gamma_{EA}(P) &= \overline{run}(A, B).\overline{c}\,\{K_s, A\}_K.A'(K_s) \\
&\quad \parallel\ c(y).[\langle y, K\rangle \vdash_{dec} w][w \vdash_{fst} z].\overline{commit}(B, A).B'(z)
\end{aligned}$$

$\gamma_{MA}(P)$ behaves as P apart from the fact that it sends the session key received by B on the low level channel *received*; $\gamma_{EA}(P)$ only modifies P by performing the two low level outputs $\overline{run}(A, B)$ and $\overline{commit}(B, A)$ when A starts the protocol and when B concludes it, respectively. Notice that these decorations do not change the protocol behaviour observed on channel c. As an example, $\gamma_{MA}(P)[f_{((I \cup O) \setminus C)}]$ corresponds to the protocol

$$\overline{c}\,\{K_s, A\}_K.A'(K_s) \parallel c(y).[\langle y, K\rangle \vdash_{dec} w][w \vdash_{fst} z].\tau.B'(z)$$

which is equivalent to P for any reasonable weak (i.e., ignoring τ's) behavioural equivalence.

3.2 The Enemy

Intuitively, an enemy can be thought of as a process which tries to attack a protocol by stealing and faking the information which is transmitted on the CryptoSPA *public* channels in set C. In principle, such a process could be modeled as a generic process X which can communicate only through the channels belonging to C. However, in this way we obtain that X is a too powerful attacker. A peculiar feature of the enemies is that they should not be allowed to know secret information in advance: as we assume perfect cryptography, the initial knowledge of an enemy must be limited to include only publicly available information, such as names of entities and public keys, and its own private data (e.g., enemy's private key). If we do not impose such a limitation, the attacker would be able to "guess" every secret information, as illustrated in the following example.

Example 3. Consider again the protocol P of Examples 1 and 2. Since only A and B know k_{AB}, this protocol should guarantee the authenticity of m_A even in the presence of an enemy, when only one protocol session is considered (we have seen that authenticity is not guaranteed for multiple sessions). We assume that $c \in C$ is a public channel and we consider the following attacker that belongs to $\mathcal{E}_C$:

$$X(m, a, k) \stackrel{\text{def}}{=} \overline{c}\,\{m, a\}_k$$

Notice that this process may only communicate over the public channel c, i.e., $sort(X(m, a, k)) = \{c\}$. Consider now $X(K_X, A, k_{AB})$, which knows k_{AB} and can consequently send a faked message $\{K_X, A\}_{k_{AB}}$ to B. In order to observe this, we consider the decorated protocol $\gamma_{MA}(P)$ "under the attack" of X (note that we put X inside the scope of restriction):

$$(\gamma_{MA}(P) \parallel X(K_X, A, k_{AB})) \setminus C$$

After one τ communication step, the process above can perform $\overline{received}\,(K_X, A)$ which represents the fact that B has received K_X instead of m_A. This happens since we are considering an attacker $X(K_X, A, k_{AB})$ which is in some sense "guessing" k_{AB} (X knows k_{AB} in advance). As we are interested in attacks that can be carried out even when cryptography is completely reliable, we would like to forbid such infeasible attacks by restricting the set of admissible enemies. ■

$$
\begin{aligned}
I(\underline{0}, V) &= \emptyset \\
I(c(x).P, V) &= I(P, V) \\
I(\overline{c}\, e.P, V) &= \mathit{get\text{–}msg}(e) \cup I(P, V) \\
I(\tau.P, V) &= I(P, V) \\
I(P_1 + P_2, V) &= I(P_1, V) \cup I(P_2, V) \\
I(P_1 \parallel P_2, V) &= I(P_1, V) \cup I(P_2, V) \\
I(P \setminus L, V) &= I(P, V) \\
I(P[f], V) &= I(P, V) \\
I(A(e_1, \ldots, e_n), V) &= \begin{cases} \bigcup_{i \in \{1,\ldots,n\}} \mathit{get\text{–}msg}(e_i) \text{ if } A \in V \\ I(P, V \cup \{A\}) \cup \bigcup_{i \in \{1,\ldots,n\}} \mathit{get\text{–}msg}(e_i) \\ \qquad\qquad\qquad\qquad\qquad\qquad \mathit{otherwise} \end{cases} \\
&\quad \text{where } A(x_1, \ldots, x_n) \stackrel{\text{def}}{=} P \\
I([e = e']P_1; P_2, V) &= \mathit{get\text{–}msg}(e) \cup \mathit{get\text{–}msg}(e') \cup I(P_1, V) \cup I(P_2, V) \\
I([\langle e_1 \ldots e_r \rangle \vdash_{rule} x]P_1; P_2, V) &= (\bigcup_{i \in \{1,\ldots,r\}} \mathit{get\text{–}msg}(e_i)) \cup I(P_1, V) \cup I(P_2, V)
\end{aligned}
$$

where

$$
\mathit{get\text{–}msg}(e) = \begin{cases} \{e\} & \text{if } e \text{ is a message} \\ \emptyset & \text{if } e \text{ is a variable} \end{cases}
$$

Fig. 4. Definition of $I(P, V)$.

The problem of guessing secret values can be solved by imposing some constraints on the initial data known by the intruders. Given a process P, we call $ID(P)$ the set of messages that occur syntactically in P. More formally, we define $ID(P)$ as $I(P, \emptyset)$, where $I : \mathcal{E} \times \mathcal{P}(Const) \longrightarrow \mathcal{P}(M)$ is given in Figure 4. Informally, $I(P, V)$ is a function that recursively visits the sub-terms of P and the body of the constants used. The argument V is used to check that the unwinding of a constant definition is performed only once.

Example 4. Consider $A(m_1)$, where $A(x) \stackrel{\text{def}}{=} \overline{c}\, x.\underline{0} \parallel \overline{c}\, m_2.A(m_3)$. Note that:

$$
\begin{array}{lll}
I(A(m_3), \{A\}) & = \{m_3\} & \\
I(\overline{c}\, x.\underline{0}, \{A\}) & = I(\underline{0}, \{A\}) & = \emptyset \\
I(\overline{c}\, m_2.A(m_3), \{A\}) & = \{m_2\} \cup I(A(m_3), \{A\}) & = \{m_2, m_3\} \\
I(A(x), \{A\}) & = I(\overline{c}\, x.\underline{0}, \{A\}) \cup I(\overline{c}\, m_2.A(m_3), \{A\}) & = \{m_2, m_3\} \\
I(A(m_1), \emptyset) & = \{m_1\} \cup I(A(x), \{A\}) & = \{m_1, m_2, m_3\}
\end{array}
$$

Thus, we have $ID(A(m_1)) = I(A(m_1), \emptyset) = \{m_1, m_2, m_3\}$. ■

Now, let $\phi_I \subseteq \mathcal{M}$ be the *finite*, initial knowledge that we would like to give to the intruders, i.e., the public information such as the names of the entities and the public keys, plus some possible private data of the intruders (e.g., their private keys or nonces). For a certain intruder X, we want that all the messages in $ID(X)$ are deducible from ϕ_I.

Definition 2. *Given a finite set $\phi_I \subseteq \mathcal{M}$, called the* initial knowledge*, we define the set $\mathcal{E}_C^{\phi_I}$ of* admissible enemies *as $\mathcal{E}_C^{\phi_I} = \{X \in \mathcal{E} \mid sort(X) \subseteq C$ and $ID(X) \subseteq \mathcal{D}(\phi_I)\}$.*

To see how $\mathcal{E}_C^{\phi_I}$ prevents the problem presented in Example 3, consider again the enemy $X(K_X, A, k_{AB})$ of that example. To indicate that k_{AB} is secret, we can now require that $k_{AB} \notin \mathcal{D}(\phi_I)$. Since $ID(X(K_X, A, k_{AB})) = \{K_X, A, k_{AB}\}$, we finally have that $X(K_X, A, k_{AB}) \notin \mathcal{E}_C^{\phi_I}$.

3.3 Behavioural semantics

In this section we define some well-known behavioural semantics that we will use in order to formalize security properties. As mentioned in the Introduction, *NDC* as well as our general *GNDC* scheme is parametric with respect to the chosen semantic preorder (or equivalence). The relevant ones for this paper are essentially two: trace and weak bisimulation [38].

Trace Semantics Most of the security properties that have been proposed for the analysis of security protocols are based on the simple notion of *trace*: two processes are equivalent if they show exactly the same execution sequences (called *traces*). Let us consider the following relations between CryptoSPA terms: $P \stackrel{\tau}{\Longrightarrow} P'$ (or $P \Longrightarrow P'$) if $P \stackrel{\tau}{\longrightarrow}^* P'$, where $\stackrel{\tau}{\longrightarrow}^*$ is the reflexive (hence $P \Longrightarrow P$ always holds) and transitive closure of the $\stackrel{\tau}{\longrightarrow}$ relation; and then $P \stackrel{a}{\Longrightarrow} P'$ if $P \stackrel{\tau}{\Longrightarrow} \stackrel{a}{\longrightarrow} \stackrel{\tau}{\Longrightarrow} P'$.

For a trace $\sigma = a_1 \dots a_n$ we write $P \stackrel{\sigma}{\Longrightarrow} P'$ if $P \stackrel{a_1}{\Longrightarrow} P_1 \stackrel{a_2}{\Longrightarrow} \cdots \stackrel{a_{n-1}}{\Longrightarrow} P_{n-1} \stackrel{a_n}{\Longrightarrow} P'$ for some $P_1, \dots, P_{n-1}$. We say that P' is a *derivative* of P if there exists a trace σ such that $P \stackrel{\sigma}{\Longrightarrow} P'$.

The set $Tr(P)$ of *traces associated with* P is then defined as $Tr(P) = \{\sigma \in (Act \setminus \{\tau\})^* \mid \exists P' : P \stackrel{\sigma}{\Longrightarrow} P'\}$.

Definition 3. *Let $P, Q \in \mathcal{E}$. We write $P \leq_{trace} Q$ iff $Tr(P) \subseteq Tr(Q)$. We also say that P and Q are* trace equivalent *(notation $P \approx_{trace} Q$) iff $P \leq_{trace} Q$ and $Q \leq_{trace} P$.* ■

Example 5. Consider again the protocol P and the (too powerful) enemy process $X(K_X, A, k_{AB})$, both discussed in Example 3. It is easy to see that

$$(\gamma_{MA}(P) \,\|\, X(K_X, A, k_{AB})) \setminus C \not\approx_{trace} \gamma_{MA}(P) \setminus C$$

Indeed,

$$Tr((\gamma_{MA}(P) \,\|\, X(K_X, A, k_{AB})) \setminus C) = \{\overline{received}\,(K_X, A), \overline{received}\,(K_s, A)\}$$
$$Tr(\gamma_{MA}(P) \setminus C) = \{\overline{received}\,(K_s, A)\}$$

This proves that X is able to attack the protocol, by guessing the key k_{AB}. ■

We can also prove that the trace preorder is preserved by the parallel composition and by the restriction operator.

Proposition 1. *Let $P, Q, R \in \mathcal{E}$ be three processes and L a set of input channels. If $P \leq_{trace} Q$ then: $P \,\|\, R \leq_{trace} Q \,\|\, R$ and $P \setminus L \leq_{trace} Q \setminus L$.* ■

Weak Bisimulation The general notion of *bisimulation* [38] consists of a mutual step-by-step simulation, i.e., given two processes E and F, when E executes a certain action moving to E' then F must be able to simulate this single step by executing the same action and moving to a term F' which is again bisimilar to E', and vice-versa. A weak bisimulation is a bisimulation which does not care about internal τ actions, i.e., when F simulates an action of E, it can also execute some τ actions before or after that action.

We write $E \stackrel{\hat{a}}{\Longrightarrow} E'$ for $E \stackrel{a}{\Longrightarrow} E'$ if $a \in \mathcal{L}$, and for $E(\stackrel{\tau}{\rightarrow})^* E'$ if $a = \tau$ (note that $\stackrel{\tau}{\Longrightarrow}$ requires at least one τ labelled transition while $\stackrel{\hat{\tau}}{\Longrightarrow}$ corresponds to $(\stackrel{\tau}{\rightarrow})^*$ and means zero or more τ labelled transitions).

The notion of *weak bisimulation* is defined as follows.

Definition 4 (Weak Bisimulation). *A binary relation $\mathcal{S} \subseteq \mathcal{E} \times \mathcal{E}$ over processes is a* weak bisimulation *if $(E, F) \in \mathcal{S}$ implies, for all $a \in Act$,*

- *whenever $E \stackrel{a}{\rightarrow} E'$, then there exists F' such that $F \stackrel{\hat{a}}{\Longrightarrow} F'$ and $(E', F') \in \mathcal{S}$;*
- *whenever $F \stackrel{a}{\rightarrow} F'$, then there exists E' such that $E \stackrel{\hat{a}}{\Longrightarrow} E'$ and $(E', F') \in \mathcal{S}$.*

Two processes $E, F \in \mathcal{E}$ are (weakly) bisimulation equivalent*, denoted by $E \approx_{bis} F$, if there exists a weak bisimulation $\mathcal{S}$ containing the pair (E, F).*

In [38] it is proved that $\approx_{bis}$ is the largest weak bisimulation and it is an equivalence relation.

NDC for CryptoSPA The family of security properties $NDC_C^{\Phi_I}$ for CryptoSPA, for any choice of the initial knowledge Φ_I and for any choice of the behavioural semantics $\approx$ is defined as follows. To simplify the notation, we will write $P \,\|_C\, X$ as a shorthand for $(P \,\|\, X) \backslash C$.

Definition 5. *Let $P \in \mathcal{E}$ be a decorated CryptoSPA process and $\approx$ one of the behavioural equivalences presented in the previous section. We say that*

$$P \in NDC_C^{\Phi_I} \text{ iff } \forall X \in \mathcal{E}_C^{\phi_I} : P \underset{C}{\|} X \approx P \setminus C$$

On the one hand, $P \setminus C$ represents the secure specification of the protocol P running in isolation on perfectly secure channels. The visible behaviour of P is given by the property dependent, extra observable actions included into the specification. Hence, the behaviour of $P \setminus C$ should describe the security property of interest. On the other hand, if $P \setminus C$ is equivalent to $P \,\|_C\, X$, then this clearly means that X is not able to modify in any way the observable execution of P, i.e., the security property holds.

Strictly speaking, *NDC* is used to denote the specific instance in the family when $\approx$ is $\approx_{trace}$, and $BNDC$ when $\approx$ is $\approx_{bis}$. Moreover, for notational convenience, if the behavioural semantics is a preorder $\lhd$, then the associated NDC variant is denoted with $NDC_{\lhd}$.

4 Verification and Compositionality

In this section we want to address the issue of formal verification of the *NDC* properties on cryptographic protocols.

First we will address the issue of the existence, in dependence of the behavioural semantics, of a *most powerful enemy* (*mpe*, for short), in such a way that the universal quantification over all possible enemies can be removed in favour of a single check against such a *mpe*.

We will also study the related problem of how to circumvent the universal quantification in the absence of a *mpe* (this is the case when the semantics is weak bisimulation) by, e.g., resorting to properties stronger than *NDC* (based on bisimulation).

Then we will address the issue of composition of secure cryptographic protocols: given two subprotocols that are *NDC* secure, under which conditions can we compose them to get a *NDC* secure protocol?

4.1 Most powerful enemy

A serious obstacle to the widespread use of *NDC* is the universal quantification over all admissible enemies. While the proof that a protocol is not *NDC* can be naturally given by exhibiting an enemy that breaks the semantic equality, much harder seems the proof that a protocol is indeed *NDC*, as it requires an infinity of equivalence checks, one for each admissible enemy. One reasonable way out could be to study if there is an attacker that is more powerful (with respect to the chosen behavioural semantics) than all the others, so that one can reduce the infinity of checks to just one, albeit huge, check with respect to such a most powerful enemy.

We will say that a preorder $\lhd$ is a *pre-congruence* (w.r.t. the operator $\|_C$) if for every $P, Q, R \in \mathcal{E}$ if $Q \lhd R$ then $P \|_C Q \lhd P \|_C R$. Thus it is easy to prove the following [37].

Proposition 2. *If $\lhd$ is a pre-congruence and if there exists a process $Top \in \mathcal{E}_C^{\phi_I}$ such that for every process $X \in \mathcal{E}_C^{\phi_I}$ we have $X \lhd Top$, then:*

$$P \in NDC_{\lhd} \qquad \textit{iff} \qquad P \underset{C}{\|} Top \ \lhd \ P \backslash C \qquad \blacksquare$$

Proof. ($\Leftarrow$) *By the hypothesis that $\lhd$ is a pre-congruence, we have that $X \lhd X'$ implies $P \|_C X \lhd P \|_C X'$. Thus, if for every process $X \in \mathcal{E}_C^{\phi_I}$ we have $X \lhd Top$,*

then we will also have $P \parallel_C X \lhd P \parallel_C Top$ for every process $X \in \mathcal{E}_C^{\phi_I}$, and so, as by hypothesis $P \parallel_C Top \lhd P \backslash C$, we obtain $P \parallel_C X \lhd P \parallel_C Top \lhd P \backslash C$, i.e., P is $NDC_{\lhd}$.
($\Rightarrow$) *By definition of $NDC_{\lhd}$, since $Top \in \mathcal{E}_C^{\phi_I}$.* ■

If the hypotheses of the proposition above hold, then it is sufficient to check that $P \backslash C$ is equivalent to P composed to the most powerful enemy Top.

We also have the following corollary for the congruence induced by $\lhd$, which essentially clarifies that *NDC* is requiring that P under the attack of the most powerful enemy is equivalent to P under the attack of the less powerful enemy, as well as to P in isolation.

Corollary 1. *Let $\lhd$ be a pre-congruence and let $\approx = \lhd \cap \lhd^{-1}$. If there exist two processes $Bot, Top \in \mathcal{E}_C^{\phi_I}$ such that for every process $X \in \mathcal{E}_C^{\phi_I}$ we have $Bot \lhd X \lhd Top$ then*

$$P \in NDC_{\approx} \qquad \textit{iff} \qquad P \underset{C}{\parallel} Bot \approx P \underset{C}{\parallel} Top \approx P \backslash C$$ ■

Given these very general results, one may wonder if it is possible to apply them to some of the semantics we have described in Section 3.3. Indeed, this is the case, at least for the trace preorder $\leq_{trace}$, which is a pre-congruence (cf Proposition 1).

The easy part is to identify the minimal element Bot in $\mathcal{E}_C^{\phi_I}$ w.r.t. $\leq_{trace}$: it is clear that the minimum set of traces is the empty-set, that is generated, e.g., by process $\underline{0}$. As a matter of fact, $P \parallel_C \underline{0} \approx P \backslash C$, for most equivalences $\approx$.

Let us now try to identify the top element Top in $\mathcal{E}_C^{\phi_I}$ w.r.t. $\leq_{trace}$. A "most powerful enemy" can be defined by using a family of processes $Top_{trace}^{C,\phi}$ each representing the instance of the enemy with knowledge ϕ:

$$Top_{trace}^{C,\phi} = \sum_{\substack{c \,\in\, C \\ m \,\in\, Msg(c)}} c(m).Top_{trace}^{C,\phi\cup\{m\}} + \sum_{\substack{c \,\in\, C \\ m \,\in\, \mathcal{D}(\phi) \,\cap\, Msg(c)}} \overline{c}\, m.Top_{trace}^{C,\phi}$$

The "initial element" of the family is Top_{trace}^{C,ϕ_I} as ϕ_I is the initial knowledge. Note that it may accept any input message, to be bound to the variable x which is then added to the knowledge set $\phi \cup \{x\}$, and may output only messages that can transit on the channel c and that are deducible from the current knowledge set ϕ via the deduction function $\mathcal{D}$. Note also that τ summands are not considered, as inessential for the trace preorder. The following holds.

Proposition 3. *If $X \in \mathcal{E}_C^{\phi}$ then $X \leq_{trace} Top_{trace}^{C,\phi}$.*

Proof. *We prove that $\mathcal{R} = \{(X', Top_{trace}^{C,\phi'}) \mid X'$ is a derivative of $X, X' \in \mathcal{E}_C^{\phi'}\}$ is a (weak) simulation relation [38] containing the pair $(X, Top_{trace}^{C,\phi})$. As the simulation preorder is finer than the trace preorder, the thesis follows. There are three possible cases.*

- *$X' \xrightarrow{c(m)} X''$ with $c \in C$, $m \in Msg(c)$, $X'' \in \mathcal{E}_C^{\phi''}$ and $\phi'' = \phi' \cup \{m\}$. Then, $Top_{trace}^{C,\phi'} \xrightarrow{c(m)} Top_{trace}^{C,\phi''}$ is derivable, and the pair $(X'', Top_{trace}^{C,\phi''}) \in \mathcal{R}$.*

- $X' \xrightarrow{\overline{c}\,m} X''$ *with* $c \in C$, $m \in Msg(c) \cup \mathcal{D}(\phi)$ *and so* $X'' \in \mathcal{E}_C^{\phi'}$. *Then,* $Top_{trace}^{C,\phi'} \xrightarrow{\overline{c}\,m} Top_{trace}^{C,\phi'}$ *is derivable, and the pair* $(X'', Top_{trace}^{C,\phi'}) \in \mathcal{R}$.
- $X' \xrightarrow{\tau} X''$, *hence* $X'' \in \mathcal{E}_C^{\phi'}$. *Then,* $Top_{trace}^{C,\phi'} \stackrel{\hat{\tau}}{\Longrightarrow} Top_{trace}^{C,\phi'}$ *is derivable, and the pair* $(X'', Top_{trace}^{C,\phi'}) \in \mathcal{R}$. ■

So, we have proved that there exists a top of the set $\mathcal{E}_C^{\phi_I}$ with respect to $\leq_{trace}$ and it is indeed Top_{trace}^{C,ϕ_I}. Now we want to prove that the single check against the top element is enough to ensure *NDC*.

Corollary 2. *Let* P *be a CryptoSPA protocol. We have that* $P \in NDC_C^{\Phi_I}$ *iff* $P \,\|_C\, Top_{trace}^{C,\phi_I} \approx_{trace} P \setminus C$.

Proof. By Corollary 1, it is enough to observe that $\approx_{trace}$ *is a precongruence (cf Proposition 1) and that the top element is* Top_{trace}^{C,ϕ_I}. ■

It is interesting to observe that Top_{trace}^{C,ϕ_I} is the top element also for other behavioural preorders, namely the simulation preorder [38] and the barbed precongruence [40]. Hence the theory above can be rephrased also for these observational semantics.

On the other hand, for weak bisimulation semantics it seems that no (manageable) *mpe* exist. Hence, in order to avoid the universal quantification we have to follow an alternative strategy, that is the subject of the next subsection.

Finally, observe that even if we have removed the universal quantification, we are not yet in the condition of easy verification of protocols because the *mpe* is in general an infinite state system. In order to perform an efficient verification, we need to adopt the strategy illustrated in the next subsection, using however a symbolic semantics (this is outside the scope of the current presentation).

4.2 Static Characterization of NDC

In this section we develop for *NDC* an alternative strategy to avoid the universal quantification over all the admissible enemies. The idea is based on a *static characterization* for *NDC* in the setting of SPA [17, 19], according to which *NDC* is proved to coincide with another non interference property called *Strong Nondeterministic Non-Interference* (*SNNI* for short).

A SPA process P is $SNNI$ if $P \backslash H$, where no high level activity is allowed, behaves like system P where all the high level activities are *hidden* (i.e., transformed into internal τ actions). To express this second system, we need to introduce first the *hiding* operator P/L (where L is an arbitrary subset of $\mathcal{L}$), which is defined by means of the following rules.

$$\frac{P \xrightarrow{a} P'}{P/L \xrightarrow{a} P'/L}\,(a \notin L \cup \overline{L}) \qquad \frac{P \xrightarrow{a} P' \quad a \in L \cup \overline{L}}{P/L \xrightarrow{\tau} P'/L} \tag{1}$$

Now we are ready to define the property for SPA as follows: $P \in SNNI$ if and only if we have $P \backslash H \approx_{trace} P/H$. It is rather intuitive to see that P/H can be seen as $(P \,\|\, Top) \backslash H$, where Top is the top element of the trace preorder for SPA; hence, such a static characterization is somehow a corollary of the existence of a top element in the trace preorder (together with the fact that the trace preorder is a precongruence).

The main goal of this section is to show that such a result is translatable to CryptoSPA, with the proviso of handling the (initial) secrets properly. To this aim we first introduce a properly enhanced definition of the hiding operator $/^{\phi}$ for CryptoSPA that will allow us to define the non interference property $SNNI_C^{\phi}$, and then to prove that NDC_C^{ϕ} is the same as $SNNI_C^{\phi}$.

$$
\begin{gathered}
\frac{P \xrightarrow{a} P'}{P/^{\phi}C \xrightarrow{a} P'/^{\phi}C} \, (a \notin C \cup \overline{C}) \\
\frac{P \xrightarrow{\overline{c}\, m} P' \quad c \in C \cup \overline{C} \quad m \in Msg(c) \quad \phi' = \phi \cup \{m\}}{P/^{\phi}C \xrightarrow{\tau} P'/^{\phi'}C} \\
\frac{P \xrightarrow{c(m)} P' \quad c \in C \cup \overline{C} \quad m \in \mathcal{D}(\phi) \quad m \in Msg(c)}{P/^{\phi}C \xrightarrow{\tau} P'/^{\phi}C}
\end{gathered}
\qquad (2)
$$

We are now ready to define the family of $SNNI_C^{\phi}$ properties, parametrized on the behavioural semantics $\approx$.

Definition 6. *A process P is $SNNI_C^{\phi}$ if $P \backslash C \approx P/^{\phi}C$.*

As for *NDC*, we will prefix $SNNI$ with a letter characterizing the actual equivalence $\approx$ we are considering; so, we will have $SNNI$ for trace, but $BSNNI$ for weak bisimulation. When the chosen behavioural semantics $\approx$ is actually $\approx_{trace}$, we have the following results.

Proposition 4. *If $\phi' \subseteq \phi$ then $SNNI_C^{\phi} \subseteq SNNI_C^{\phi'}$.*

Proof. Let P be a $SNNI_C^{\phi}$ process. In order to prove that $P \in SNNI_C^{\phi'}$, we have to check that $P \backslash C \approx_{trace} P/^{\phi'}C$, if $P \backslash C \approx_{trace} P/^{\phi}C$. The thesis follows by observing that $Tr(P \backslash C) \subseteq Tr(P/^{\phi'}C) \subseteq Tr(P/^{\phi}C)$. ■

Proposition 5. *$P \in NDC_C^{\phi}$ iff $P \in SNNI_C^{\phi}$.*

Proof. If $P \in NDC_C^{\phi}$, then by Corollary 2 we have $P \,\|_C\, Top_{trace}^{C,\phi} \approx_{trace} P \backslash C$. Hence, the thesis holds if $P \,\|_C\, Top_{trace}^{C,\phi} \approx_{trace} P/^{\phi}C$. We actually prove such an equality for the simulation equivalence $\asymp$ [38], an equivalence which is finer than trace equivalence. To this aim, we prove that the following two relations are (weak) simulations.

$$
\begin{aligned}
\mathcal{R}_1 = \{(P' \underset{C}{\|} Top_{trace}^{C,\phi'}, P'/^{\phi'}C) \mid \\
P' \underset{C}{\|} Top_{trace}^{C,\phi'} \text{ is a derivative of } P \underset{C}{\|} Top_{trace}^{C,\phi}\}
\end{aligned}
$$

and

$$\mathcal{R}_2 = \{(P'/^{\phi'}C, P' \underset{C}{\|} Top^{C,\phi'}_{trace}) \mid P'/^{\phi'}C \textit{ is a derivative of } P/^{\phi}C\}$$

Let us start with $\mathcal{R}_1$. *We have the following three cases.*

- $P' \|_C Top^{C,\phi'}_{trace} \xrightarrow{a} P'' \|_C Top^{C,\phi'}_{trace}$ *with* $a \notin C \cup \overline{C}$. *Then,* $P'/^{\phi'}C \xrightarrow{a} P''/^{\phi'}C$ *and* $(P'' \|_C Top^{C,\phi'}_{trace}, P''/^{\phi'}C) \in \mathcal{R}_1$.
- $P' \|_C Top^{C,\phi'}_{trace} \xrightarrow{\tau} P'' \|_C Top^{C,\phi''}_{trace}$ *as* $P' \xrightarrow{\bar{c}\, m} P''$ *and* $Top^{C,\phi'}_{trace} \xrightarrow{c(m)} Top^{C,\phi''}_{trace}$. *Hence,* $c \in C$, $m \in Msg(c)$, $\phi'' = \phi' \cup \{m\}$. *Then, it is derivable* $P'/^{\phi'}C \xrightarrow{\tau} P''/^{\phi''}C$ *and* $(P'' \|_C Top^{C,\phi''}_{trace}, P''/^{\phi''}C) \in \mathcal{R}_1$.
- $P' \|_C Top^{C,\phi'}_{trace} \xrightarrow{\tau} P'' \|_C Top^{C,\phi'}_{trace}$ *as* $P' \xrightarrow{c(m)} P''$ *and* $Top^{C,\phi'}_{trace} \xrightarrow{\bar{c}\, m} Top^{C,\phi'}_{trace}$. *Hence,* $c \in C$, $m \in Msg(c)$, *and* $m \in \mathcal{D}(\phi')$. *Then, it is derivable* $P'/^{\phi'}C \xrightarrow{\tau} P''/^{\phi'}C$ *and* $(P'' \|_C Top^{C,\phi'}_{trace}, P''/^{\phi'}C) \in \mathcal{R}_1$.

Analogously for relation $\mathcal{R}_2$. ■

A consequence of the proposition above is that one may wonder why we should be interested to $SNNI^{\Phi}_{C}$ as, after all, it is equivalent to NDC^{Φ}_{C}. There are some good reasons of interest for $SNNI^{\Phi}_{C}$. First, even in the case of trace semantics, $SNNI^{\Phi}_{C}$ is of interest because it may lead to more efficient analysis, e.g., it is easier to formulate in a symbolic way. Second, it helps the formulation of compositionality principle, as we will see in the following. Third, the new characterization does not consider explicitly the notion of (admissible) enemy, while it considers only the set of (initial) secrets. Hence, it is definable also for those preorders that do not admit the top element (most powerful enemy *mpe*).

This last comment deserves more explanation. Assume we want to prove that P satisfies $BNDC$; this may be useful, e.g., for security properties that need some form of liveness (e.g., non repudiation). Hence in order to avoid the universal quantification, we can try to prove that P satisfies the finer property of $SBSNNI^{\Phi}_{C}$, that requires that $BSNNI^{\Phi}_{C}$ holds for all the derivatives of P. In this way, as the state space of P is (usually) finite, we have removed the infinity of checks that $BNDC$ requires in favour of a finite number (one for each derivative of P) of $BSNNI^{\Phi}_{C}$ checks. A recent paper [7] has independently proposed this approach and shown how to efficiently prove $SBSNNI^{\Phi}_{C}$ by means of a new, equivalent property (called P_BNDC) that needs not to be verified on all the derivatives of P.

4.3 Compositionality

In this section we illustrate a compositional proof method for $SNNI^{\phi}_{C}$, originally introduced in [29]. Within the SPA theory, $SNNI$ is compositional, i.e. if $P, Q \in SNNI$ then $P \| Q \in SNNI$. Unfortunately, the same does not hold when considering enemies with limited knowledge, as for $SNNI^{\phi}_{C}$. For instance, consider the processes:

$$P = \overline{c_1}m_1.c_2(x)[x = m_2].\overline{c_3}m_2$$
$$Q = \overline{c_1}m_2.c_2(x)[x = m_1].\overline{c_3}m_1$$

Now, assuming $C = \{c_1, c_2\}$ and $\phi = \emptyset$, we have that $P, Q \in SNNI_C^{\phi}$; indeed, $P \backslash C$ is equivalent to $\underline{0}$, as well as $P/^{\phi}C$ (it can perform two τ's and then stop). However, $P \,\|\, Q \notin SNNI^{\phi}$. As a matter of fact, $P \,\|_C\, Q$ is equivalent to $\underline{0}$, while $(P \,\|\, Q)/^{\phi}C$ may perform both $\overline{c_3}m_1$ and $\overline{c_3}m_2$.

However, if we strengthen the assumptions we can get a compositional rule for establishing that a process belongs to $SNNI_C^{\phi}$. The *stability* assumption we make is that the process cannot increment its knowledge.

Definition 7. *We say that a process P is stable w.r.t. ϕ, whenever if $P/^{\phi}C \Longrightarrow P'/^{\phi'}C$ then $\mathcal{D}(\phi) = \mathcal{D}(\phi')$.*

Thus, the following proposition holds.

Proposition 6. *Assume that $P, Q \in SNNI_C^{\phi}$ and that P, Q are stable w.r.t. ϕ. Then $(P \,\|\, Q) \in SNNI_C^{\phi}$ and $P \,\|\, Q$ is stable w.r.t. ϕ.*

Example 6. We consider a simple example of the application of the principle above. Consider the processes $P = \overline{c}\{m\}_k.0$ and $Q = c?x.[x \quad k \vdash_{dec} z]\tau$. Consider also the process $KEN' = c?x.[x = m].\overline{public}\,m$. Assuming $\phi = \{\{m\}_k\}$ and $C = \{c\}$, we can establish that $P, Q, KEN' \in SNNI_C^{\phi}$ and are all stable; hence $P \,\|\, Q \,\|\, KEN' \in SNNI_C^{\phi}$, which means that $P \,\|\, Q$ keeps m secret. As a matter of fact, $(P \,\|\, Q \,\|\, KEN')\backslash C \approx_{trace} \underline{0}$.

Considering bisimulation semantics, it is not difficult to see that the compositionality principle above scales to $SBSNNI_C^{\phi}$.

Proposition 7. *Assume that $P, Q \in SBSNNI_C^{\phi}$ and that P, Q are stable w.r.t. ϕ. Then $P \,\|\, Q \in SBSNNI_C^{\phi}$ and $P \,\|\, Q$ is stable w.r.t. ϕ.*

5 A General Schema for Security Properties

In this section we formally define the $GNDC_{\lhd}^{\alpha}$ family of properties. The proposed family of security property is the following: [4]

Definition 8. *Let $P \in \mathcal{E}$ be a CryptoSPA process. We say that*

$$P \in GNDC_{\lhd}^{\gamma,\alpha} \text{ iff } \forall X \in \mathcal{E}_C^{\phi_I} : \gamma(P) \underset{C}{\|} X \lhd \alpha(P)$$

where $\lhd \in \mathcal{E} \times \mathcal{E}$ is a relation between processes, $\gamma : \mathcal{E}_C \rightarrow \mathcal{E}$ is a decoration function and $\alpha : \mathcal{E} \rightarrow \mathcal{E}$ is a function between processes. ■

[4] Indeed *GNDC* depends on the set $\mathcal{E}_C^{\phi_I}$, hence on ϕ_I and C, but we will omit them for the sake of readability.

We propose a sufficient criterion for a static characterization (i.e. not involving the universal quantifier $\forall$) of $GNDC_{\lhd}^{\gamma,\alpha}$ properties.

Proposition 8. *If $\lhd$ is a pre-congruence w.r.t. $\|_C$ and if there exists a process $Top \in \mathcal{E}_C^{\phi_I}$ such that for every process $X \in \mathcal{E}_C^{\phi_I}$ we have $X \lhd Top$, then:*

$$P \in GNDC_{\lhd}^{\gamma,\alpha} \qquad \textit{iff} \qquad \gamma(P) \underset{C}{\|} Top \lhd \alpha(P) \qquad \blacksquare$$

In particular, if the hypotheses of the proposition above hold then it is sufficient to check that $\alpha(P)$ is satisfied when P is composed with the most general (i.e., most powerful) environment Top. The proposition above is a generalization to *GNDC* of Proposition 3. The following corollary for the congruence induced by $\lhd$, generalizes Corollary 2:

Corollary 3. *Let $\lhd$ be a pre-congruence w.r.t. $\|_C$ and let $\approx = \lhd \cap \lhd^{-1}$. If there exist two processes $Bot, Top \in \mathcal{E}_C^{\phi_I}$ such that for every process $X \in \mathcal{E}_C^{\phi_I}$ we have $Bot \lhd X \lhd Top$ then*

$$P \in GNDC_{\approx}^{\gamma,\alpha} \qquad \textit{iff} \qquad \gamma(P) \underset{C}{\|} Bot \approx \gamma(P) \underset{C}{\|} Top \approx \alpha(P) \qquad \blacksquare$$

5.1 Compositionality

We have a compositionality principle for the $GNDC_{\leq_{trace}}^{\gamma,\alpha}$ schema, also in this case under the assumption that the involved processes are stable. It basically states that security properties, represented as specification processes $\alpha_i(P_i)$ $i = 1, 2$, can be composed resulting in the property represented by the composition of the specification processes $\alpha_1(P_1) \| \alpha_2(P_2)$. As far as decorations are concerned, we assume that the compound process $P_1 \| P_2$ is decorated componentwise, hence below we omit the explicit mention to γs functions.

Proposition 9. *Given the set of initial knowledge ϕ and the set of public channels C, assume $P_i \in GNDC_{\leq_{trace}}^{\alpha_i(P_i)}$, with $i = 1, 2$, and P_1, P_2 are stable w.r.t. ϕ. It follows that $(P_1 \| P_2) \in GNDC_{\leq_{trace}}^{\alpha_1(P_1) \| \alpha_2(P_2)}$ and $P_1 \| P_2$ is stable w.r.t. ϕ.*

Example 7. Consider the following family of processes $S(i)$, each sending a message (m_i, i) after every time unit:

$$S(i) = [(m_i, i) \quad pk_A \vdash_{enc} z].\overline{c}z.S(i+1))$$

Consider also a family of receivers of this kind:

$$R(i) = c(y)[y \quad pk_A^{-1} \vdash_{dec} t][t \vdash_{snd} t_2][t_2 = i][t \vdash_{fst} t_1]\overline{out}\, t_1.R(i+1)$$

Basically, we have that $(S(0) \| R(0)) \backslash C$, where $C = \{c\}$, is trace included into $Spec(0)$ where

$$Spec(i) = \overline{out}\, m_i.Spec(i+1)$$

Consider $\phi = \{\{(m_i, i)\}_{pk_A} \mid 0 \leq i\} \cup \{pk_A\}$ as the intruder's knowledge set. Then, we have that $S(0)$ and $R(0)$ are stable w.r.t. ϕ and $S(0) \in NDC_C^{\phi}$, with $S(0)\backslash C =_{trace} \underline{0}$, and $R(0) \in GNDC_{\leq_{trace}}^{Spec(0)}$. By Proposition 9, we have that

$$(S(0) \,\|\, R(0) \,\|\, Top_{trace}^{C,\phi})\backslash C \leq_{trace} \underline{0} \,\|\, Spec(0) \leq_{trace} Spec(0)$$

5.2 Non Deducibility on Compositions

Since $GNDC_{\lhd}^{\gamma,\alpha}$ is a generalization of *NDC* it can be instantiated in order to obtain *NDC* and also the *bisimulation* based *NDC*, called *BNDC*. Let Id_C be the identical decoration function. Then, *NDC* and *BNDC* correspond to $GNDC_{\approx_{trace}}^{Id_C,E\backslash C}$ and $GNDC_{\approx_{bisim}}^{Id_C,E\backslash C}$, respectively.

For *NDC* it is also possible to apply Corollary 3 obtaining an interesting static characterization.

Proposition 10. *P is NDC iff* $P \,\|_C\, Top_{trace}^{C,\phi_I} \approx_{trace} P \setminus C$. ■

This result is the analogous of the one in [17], for multilevel security. Note that here we have found it as a particular case of the more general result of Corollary 3.

5.3 The Agreement Property

In this section we show that also the approach proposed in [33] for the analysis of authentication properties, inside the framework of CSP [31] process algebra, can be rephrased in terms of our specification schema. *Agreement* is a formalization, in the CSP calculus, of the *correspondence* idea proposed in [45], and previously discussed in section 1.3. The basic idea of the *Agreement* property is the following:

> "A protocol guarantees to a responder B *Agreement* with an initiator A on a set of data items ds if, whenever B (acting as responder) completes a run of the protocol, apparently with initiator A, then A has previously been running the protocol, apparently with B, and A was acting as initiator in her run, and the two agents agreed on the data values corresponding to all the variables in ds, and each such run of B corresponds to a unique run of A".

What is technically done in the *Agreement* property is to have for each party an action representing the running of the protocol and another one representing the completion of it. For example, consider an action $commit\_res(B, A, d)$ representing a correct termination of B as a responder that is convinced to communicate with A and agrees on data in d. Moreover we have an action $running\_ini(A, B, d)$ that represents the fact that A is running the protocol as initiator, apparently with B and with data d. If we have these two actions specified in the protocol, the *Agreement* property requires that when B executes $commit\_res(B, A, d)$ then A has previously executed $running\_ini(A, B, d)$. This means that every time B completes the protocol with A convinced that the relevant data are the ones represented by d, then A must have been running the protocol with B using exactly the data in d.

We can see the actions representing the runs and the commits as output actions over two particular channels *running_ini* and *commit_res*. In [33], it is assumed that the actions representing the runs and the commits are correctly specified in the protocol. Here, we consider protocols of the form:

$$P \stackrel{\text{def}}{=} C_1(U_1, U_1', d_1) \parallel \ldots \parallel C_n(U_n, U_n', d_n) \parallel V_1^{y_1}(Z_1, Z_1') \parallel \ldots \parallel V_n^{y_n}(Z_n, Z_n')$$

Where each *claimant* process $C_i(U_i, U_i', d_i)$ and each *verifier* process $V_i^{y_i}(Z_i, Z_i')$ are all sequential processes, i.e., processes composed only of sequences of actions, and represent user U_i willing to authenticate with user U_i', agreeing on data d_i, and user Z_i verifying the identity of user Z_i', agreeing on data contained in variable y_i. Given this specific (but reasonable) protocol form, we can easily give a decoration $\gamma_{Agree}(P)$ as follows:

$$\begin{aligned}\gamma_{Agree}(P) \stackrel{\text{def}}{=} & \overline{running\_ini}\,(U_1, U_1', d_1).C_1(U_1, U_1', d_1) \parallel \ldots \\ & \parallel \overline{running\_ini}\,(U_n, U_n', d_n).C_n(U_n, U_n', d_n) \\ & \parallel \tilde{V}_1(Z_1, Z_1', d_1') \parallel \ldots \parallel \tilde{V}_n(Z_n, Z_n', d_n')\end{aligned}$$

where $\tilde{V}_i$ is the process V_i in which $\underline{0}$ is replaced by $\overline{commit\_res}\,(Z_i, Z_i', y_i)$, i.e., it terminates with action $\overline{commit\_res}\,(Z_i, Z_i', y_i)$.

Now that we have defined how to decorate a protocol, we need to specify which are the good behaviours $\alpha(P)$, of a process P. For simplicity, we only analyze the case where A is the initiator and B is the responder, and the set ds of variables is composed only by d which can assume values in a set D. However, the specification can be easily extended in order to cover all the cases studied in [33]. Function $\alpha(P)$ can be defined as follows:

$$\begin{aligned} P' &= Top_{trace}^{Sort(P)\setminus\{running\_ini, commit\_res\}, \mathcal{M}} \\ P''(x,y) &= \textstyle\sum_{d \in D} \overline{running\_ini}\,(x,y,d) \ \ . \ \ \overline{commit\_res}\,(y,x,d) \\ \alpha_{Agree}(P) &= P' \parallel P''(A,B)\end{aligned}$$

Given P, $\alpha(P)$ represents the most general system which satisfies the agreement property and has the same sort as P. As a matter of fact in $\alpha(P)$ the action $\overline{running\_ini}\,(A, B, d)$ always precedes $\overline{commit\_res}\,(B, A, d)$ for every datum d, and every combination of the other actions of P can be executed. In order to analyze more than one session, it is sufficient to consider an extended α which has several processes $P''(A,B)$ in parallel. For example, for n sessions we can consider the following:

$$\alpha_{Agree}(P) = P' \parallel \underbrace{P''(A,B) \parallel \ldots \parallel P''(A,B)}_{n}$$

We want that even in the presence of an hostile process X, P does not execute traces that are not in $\alpha(P)$,i.e., we require that $P \parallel_C X \leq_{trace} \alpha(P)$. So we can give the following definition:

Definition 9. *P satisfies Agreement iff P is* $GNDC^{\gamma_{Agree},\alpha_{Agree}}_{\leq_{trace}}$. ■

Example 8. Consider once more the protocol first presented in Example 1. It can be easily written in the form discussed above as follows:

$$\begin{aligned} P &= C(A,B,K_s) \,\|\, V^z(B,A) \\ C(a,b,k) &= \overline{c}\,\{k,a\}_K.\underline{0} \\ V^z(b,a) &= c(y).[\langle y,K\rangle \vdash_{dec} w][w \vdash_{fst} z].\underline{0} \end{aligned}$$

Notice that $V^z(B,A)$ specifies z as the variable which should contain the datum the processes are willing to agree on, i.e., the session key K_s. Now we have that

$$\begin{aligned} \gamma_{Agree}(P) = &\ \overline{running\_ini}\,(A,B,K_s).\overline{c}\,\{K_s,A\}_K.\underline{0} \\ &\| \ c(y).[\langle y,K\rangle \vdash_{dec} w][w \vdash_{fst} z].\overline{commit\_res}\,(B,A,z) \end{aligned}$$

It is possible to prove that $\forall X \in \mathcal{E}_C^{\phi_I} : \gamma_{Agree}(P) \,\|_C\, X \leq_{trace} \alpha_{Agree}(P)$, i.e., that $P \in GNDC^{\gamma_{Agree},\alpha_{Agree}}_{\leq_{trace}}$, if $K \notin \phi_I$. However, as noticed in Section 1.3, this protocol is flawed when more the one session is considered. Two sessions of P can be modelled by just replicating P twice, i.e., by considering $P \,\|\, P$. Now let $X = c(w).\overline{c}\,(w).\overline{c}\,(w)$ be the enemy that intercepts the message sent over c and replays it twice. It is easy to see that $Tr(\gamma_{Agree}(P \,\|\, P) \,\|_C\, X) = \{\overline{running\_ini}\,(A,B,K_s).\overline{commit\_res}\,(B,A,K_s).\overline{commit\_res}\,(B,A,K_s)\}$ that is not included in $Tr(\alpha_{Agree}(P \,\|\, P))$, as the second commit is not matched by any running action. ■

Note that in [33] it is only required that *Agreement* holds when the system is composed with a particular intruder, which turns out to be equivalent to the most general one. In the following we exploit Proposition 8 in order to formally prove that such a (static) requirement is indeed sufficient (and necessary) to guarantee our *GNDC*-based version of *Agreement*. As a matter of fact, by Propositions 3 and 8 we immediately have the following result:

Proposition 11. *P satisfies Agreement iff it holds* $\gamma_{Agree}(P) \,\|_C\, Top^{C,\phi_I}_{trace} \leq_{trace} \alpha_{Agree}(P)$. ■

In [33], other versions of *Agreement* are defined. We can rephrase all of them in our model by simply changing the α function. [5]

5.4 Message-Oriented Authentication

Now, we consider the message-based approach to authentication defined in [43, 42] using the CSP language. The idea is to observe when a set of messages T

[5] Indeed, *recentness* cannot be immediately rephrased in our CCS-based model, because of the difference in handling communication with respect to CSP. This could be overcome by extending our language with time as done in [21, 29]. This is only related to the differences in the model, and is not caused by a weakness of our schema.

authenticates another set of messages R. Informally, T authenticates R if the occurrence of some element of T implies the occurrence of some element of R (it is required that T and R are disjoint). When a system P satisfies this property we say that P satisfies T **authenticates** R.

In [43] the net is represented by a process $Medium$ which acts like a router by receiving and forwarding the messages to the correct process. In CSP, it is possible to observe the communication between the processes and the medium since they are not "internalized" as in CCS. However, we can simulate this by assuming that the $Medium$ echoes every routing of messages through particular output actions on two reserved channels *send* and *rcv* which do not belong to C. Action $\overline{send}\,(i,m)$ corresponds to the sending of message m performed by agent i and, symmetrically, $\overline{rcv}\,(i,m)$ represents the reception of it by agent i. (We assume to have a set $Agents$ of agent identities with X denoting the intruder identity). In this way we can observe communication as done in CSP. This message echoing is obtained by suitably decorating the specification as follows:

$$\begin{aligned}
&\gamma_{auth}(P) = (P[f] \parallel Medium) \setminus W \\
&Medium = \sum_{c \in C, i,j \in Agents} c^i_{send}(x).\overline{send}\,(i,x).(Medium \parallel \overline{c^j_{rcv}}\, x.\overline{rcv}\,(j,x).\underline{0})
\end{aligned}$$

where f is a relabelling function that maps all the output actions $\overline{c}\,(m)$, performed by agent i, into $\overline{c^i_{send}}\,(m)$ and all the input actions $c(m)$, performed by agent i, into $c^i_{rcv}(m)$. [6] Moreover, $W = \{c^i_* \mid i \neq X \text{ and } * = send, rcv\}$, i.e., W is the set of all the $Medium$ channels that are not used to communicate with the intruder X. This relabelling has the effect of forcing all the communication through the Medium, which can consequently make observable every message exchange through the special channels *send* and *rcv*. Notice that messages are buffered so that the Medium is always ready to "route" new messages.

Sets T and R range over these reserved actions. We can now define the $\alpha_{auth^T_R}(P)$ function as follows:

$$\begin{aligned}
\alpha_{auth^T_R}(P) &= P' \\
P' &= \Big(\sum_{\substack{a \in Act \\ a \notin R \cup T}} a.P'\Big) + \sum_{\substack{a \in Act \\ a \in R}} a.P'' \\
P'' &= Top^{Sort(P),\mathcal{M}}_{trace}
\end{aligned}$$

Process $\alpha_{auth^T_R}(P)$ can execute actions in T only after it has executed some actions in R. This can be seen by noticing that $\alpha_{auth^T_R}(P)$ moves to P'' (which can execute also actions in T) only after it performs at least one action in R. This is exactly what we require by our system P and $\alpha_{auth^T_R}(P)$ is indeed the

[6] We are implicitly assuming that channels with subscripts *send* and *rcv* are not used in protocol specification. This assumption is trivially met by a syntactical renaming of channels. We are also assuming that every agent has a unique name i. This is also done in [43] and will be exploited when considering non-repudiation.

most general system (with the same sort as P) satisfying T **authenticates** R. So we can give the following definition:

Definition 10. *P satisfies T* **authenticates** *R iff P is $GNDC^{\gamma_{auth},\alpha_{auth^T_R}}_{\leq_{trace}}$* ∎

As in the section above, we can prove that the approach followed in [43], where it is considered only the most powerful intruder, guarantees that the property holds in the presence of whatever hostile process. By Propositions 3 and 8 we obtain that:

Proposition 12. *P satisfies T* **authenticates** *R iff $\gamma_{auth}(P) \parallel_C Top^{C,\phi_I}_{trace} \leq_{trace} \alpha_{auth^T_R}(P)$.* ∎

5.5 Secrecy

In this section we show that *NDC* can be easily adapted for analysing secrecy in networks. Consider now a protocol $P(M)$ and assume that we want to verify if $P(M)$ preserves the secrecy of message M. This can be done by proving that every enemy which does not know message M, cannot learn it by interacting with $P(M)$. Thus, we need a mechanism that notifies whenever an enemy is learning M. We implement it through a simple process called *knowledge notifier* which reads from a public channel $c_k \in C \setminus sort(P(M))$ not used in $P(M)$ and executes a $\overline{learnt}\,M$ action if the read value is exactly equal to M. For a generic message m, it can be defined as follows:

$$KN(m) \stackrel{\text{def}}{=} c_k(y).[m = y]\overline{learnt}\,m$$

We assume that *learnt* is a special channel that is never used by protocols and is not public, i.e., $learnt \notin sort(P) \cup C$. We now decorate $P(m)$ as follows:

$$\gamma_{secret}(P(m)) = P(m) \parallel KN(m)$$

Intuitively, $\gamma_{secret}(P(m))$ is a modified protocol where the learning of m is now notified. To guarantee the secrecy of m, it is enough to require that for every secret M and for every enemy X, process $(\gamma_{secret}(P(M)) \parallel X)\setminus C$ never executes a $\overline{learnt}\,M$ action. On the one hand, if $(\gamma_{secret}(P(M)) \parallel X)\setminus C$ executes $\overline{learnt}\,M$ then M has been sent over the public channel c_k by either $P(M)$ or X. In both cases the message is not secret anymore. In the former situation $P(M)$ is making M public, while in the latter X has for sure learned M before sending it over c_k. On the other hand, if an enemy X is able to learn message M then there also exists an enemy X' that will send such a message over channel c_k and thus $(\gamma_{secret}(P(M)) \parallel X') \setminus C$ will eventually execute $\overline{learnt}\,M$.

This can be formalized by considering a simple function α_{secret} that can execute all the protocol actions but the special action *learnt*:

$$\alpha_{secret}(P) = Top^{Sort(P),\mathcal{M}}_{trace}$$

Definition 11. *$P(m)$ preserves the secrecy of m iff for all (secret) messages $M \in \mathcal{M} \setminus \mathcal{D}(\phi_I)$ $P(M)$ is $GNDC^{\gamma_{secret},\alpha_{secret}}_{\leq_{trace}}$.* ∎

Notice that $\alpha_{secret}(P(M))$ never executes action *learnt*, thus forbidding its execution even when every possible intruder is considered.

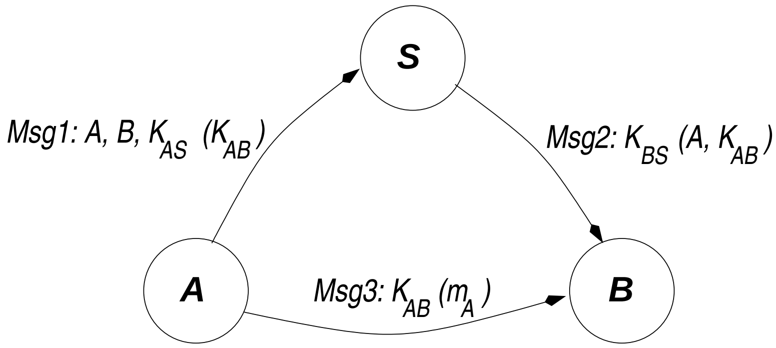

Fig. 5. Graphical description of the *WMF* protocol.

An Example In this section we show through a simple example how the *NDC*-based secrecy verification works. We consider a simplified version of the Wide Mouthed Frog Protocol [9].

Consider two processes A and B respectively sharing 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 the server S. Then, the server decrypts the key and forwards it to B, this time encrypted with k_{BS}. Now B has the key k_{AB} and A can send a message m_A encrypted with k_{AB} to B. The protocol is composed of the following three messages (see also Figure 5):

$$\begin{array}{ll} \text{message 1} & A \rightarrow S : A, B, \{k_{AB}\}_{k_{AS}} \\ \text{message 2} & S \rightarrow B : \{A, k_{AB}\}_{k_{BS}} \\ \text{message 3} & A \rightarrow B : \{m_A\}_{k_{AB}} \end{array}$$

The main differences with respect to the original protocol is that here messages 1 and 2 do not contain timestamps (as studied in [2] for authentication). Moreover, in message 1 the identifier B is sent as plaintext while in the original protocol it is encrypted with the session key (this modification generates, as we will show, a secrecy attack). We specify the protocol as follows: [7]

$$\begin{aligned} A(m,k) &\stackrel{\text{def}}{=} \overline{c_1}\,((A,B),\{k\}_{k_{AS}})\ .\ \overline{c_3}\,\{m\}_k \\ B &\stackrel{\text{def}}{=} c_2(y)\ .\ [\langle y, k_{BS}\rangle \vdash_{dec} z]\ [z \vdash_{snd} s]\ c_3(t)\ .\ [\langle t, s\rangle \vdash_{dec} w] \\ S &\stackrel{\text{def}}{=} c_1(x)\ .\ [x \vdash_{fst} i]\ [i \vdash_{fst} s]\ [i \vdash_{snd} r] \\ &\qquad [x \vdash_{snd} c]\ [\langle c, K(s)\rangle \vdash_{dec} z]\ \overline{c_2}\,\{(s,z)\}_{K(r)}\ .\ S \\ P(n) &\stackrel{\text{def}}{=} A(n, k_{AB}) \parallel B \parallel S \end{aligned}$$

where $K(id)$ is a function that returns the key shared between the server and entity id (e.g., $K(A)$ returns k_{AS}). Moreover we have that $\{c_1, c_2, c_3\} \subseteq C$.

[7] We encode tuples through a left associative canonical form, e.g., the first message $A, B, \{k_{AB}\}_{k_{AS}}$ becomes $((A,B),\{k_{AB}\}_{k_{AS}})$

Consider now the following enemy:

$$
\begin{array}{ll}
\tilde{X} \stackrel{\text{def}}{=} c_1(x)\ [x \vdash_{snd} y] & \texttt{\% intercepts message 1} \\
\quad\ \overline{c_1}\,((A,E),y). & \texttt{\% replaces B with E, sends it} \\
\quad\ c_2(z)\ [\langle z, k_{ES}\rangle \vdash_{dec} w]\ [w \vdash_{snd} k]. & \texttt{\% intercepts msg 2, obtains k} \\
\quad\ c_3(j)[\langle j, k\rangle \vdash_{dec} m] & \texttt{\% decrypts msg 3} \\
\quad\ \overline{c_k}\, m & \texttt{\% sends message to KN}
\end{array}
$$

It is easy to see that process $(\gamma_{secret}(P(M)) \parallel \tilde{X}) \setminus C \not\leq_{trace} \alpha_{secret}(P(M))$ as the former process can execute $\overline{learnt}\, M$. The attack performed by $\tilde{X}$ is the following:

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

By message $2'$ the enemy learns K_{AB} and, by message 3 which is addressed to B, it finally learns M.

Consider now the protocol where, in the first message, B is encrypted with the session key:

$$\text{message } 1 \qquad A \rightarrow S : A, \{B, K_{AB}\}_{K_{AS}}$$

Here the secrecy attack is not possible anymore since cryptography protects the identifier from being modified by the intruder.

5.6 Non-Repudiation

In this section, we show that also non-repudiation properties can be formulated within the *GNDC* schema. Non repudiation protocols aim at producing evidence about the execution of services, among parties that do not trust each other (see [46, 47]).

In [41] Schneider shows how to apply verification methods based on *CSP* process algebra to the analysis of a (fair) non repudiation protocol proposed in [46]. Among the non repudiation properties studied in [41, 47], we briefly recall:

- *Non Repudiation of Origin (NRO)* is intended to protect the receiver from the false denial of another party to have sent a message.
- *Non Repudiation of Receipt (NRR)* is intended to protect the sender form the false denial of another party to have received a message.

Intuitively, the analysis performed by Schneider is similar to his message based authentication (see section above). As an example, consider *NRO* verification: if the receiver is able to produce an evidence of the sending of a certain message m, then m should have been sent. In other words, such an evidence should "authenticate" m (in the sense of message-based authentication).

Non-repudiation protocols differ from the protocols discussed so far, which always involve communication among two or more trusted parties in an hostile environment. In non-repudiation protocols, parties do not trust each other, and in particular, one of them could try to act maliciously in order to obtain some advantage. As we will see, this can be modelled in the *GNDC* schema by considering the malicious party as a whole with the hostile environment.

In the verification of NRO (NRR) we assume that a Judge should be able to establish that a certain message has been sent (received) if he obtains some evidence of it from the receiver (sender). This verification should be carried out by only assuming that both the sender and the receiver have not sent on the net some secret information which could invalidate the evidence, like, e.g., their signature keys. In case of a dispute, the Judge cannot assume that both the parties have followed the protocol but she will always assume that none of them has compromised his own secret key.

Schneider models both the sender and the receiver similarly to the most general intruder with the constraint that long-term keys are never compromised. In order to apply the *GNDC* schema, we consider a weaker (but still reasonable) notion of NRO; in particular, we require that if the receiver B, after following the protocol, is able to give evidence of origin, then the sender A has actually sent that message. We call this NRO with honest receiver written NRO_{hr}. It can be encoded in the *GNDC* schema by considering a process P_B where only the receiver B and the fragment of A, T_A, related to encryption with long-term (secret) keys, are specified. P_B has the following form:

$$P_B \stackrel{\text{def}}{=} T_A \parallel \ldots \parallel T_A \parallel B_1^{y_1}(Z_1, Z_1') \parallel \ldots \parallel B_n^{y_n}(Z_n, Z_n')$$

Where each process $B_i^{y_i}(Z_i, Z_i')$ is sequential and represent Z_i getting the evidence of origin of the message contained in variable y_i sent by user Z_i'. Given this specific (but reasonable) protocol form, we can easily give a decoration $\gamma_{nro}(P_B)$ as follows:

$$\gamma_{nro}(P_B) \stackrel{\text{def}}{=} \gamma_{auth}(\tilde{P}_B)$$

where γ_{auth} is the decoration function defined in section 5.4 for message-oriented authentication, and $\tilde{P}_B$ is process P_B in which all the $\underline{0}$'s of processes $B_i^{y_i}(Z_i, Z_i')$ are replaced by $\overline{ev\_of\_or}\,(Z_i, Z_i', y_i)$. For instance, action $\overline{ev\_of\_or}\,(B, A, m)$ is the action which signals that B has evidence of origin of m from A. This latter decoration is similar to the one we used for adding commit actions in Section 5.3.

Now, recall that in process $Medium$ $send(A, m)$ represents the sending of message m by agent A. Then α_{nro} can be simply defined as $\alpha_{auth^{\{ev\_of\_or(B,A,m)\}}_{\{send(A,m)\}}}$.

Definition 12. *P_B guarantees NRO_{hr} iff P_B is $GNDC^{\gamma_{nro},\alpha_{nro}}_{\leq trace}$*

We show how this definition works through a simple example.

Example 9. A typical way of guaranteeing non-repudiation of origin, is to use digital signature. Consider the message exchange:

$$\text{message 1} \qquad A \to B : M, sign_A(M)$$

Since only A may generate $sign_A(M)$, then B is guaranteed that A has originated such a message. To see how the formalization of non-repudiation works, we consider a flawed version of the protocol above, in which shared key is used instead of signature:

$$\text{message } 1 \qquad A \rightarrow B : M, MAC_K(M)$$

K is a key shared between A and B. The protocol uses a *Message Authentication Code* (MAC), i.e., a keyed hash function such that $MAC_K(M)$ can be efficiently calculated if and only if the key K is known. A MAC can be modelled in SPA by making the inverse key $k^{-1} \neq k$ not available to both the principals and the enemy. This is a simple way of simulating the unidirectionality provided by the (one-way) hashing. This protocol does not guarantee non-repudiation since both A and B might have generated $MAC_K(M)$. We specify B as follows:

$$Responder^j(b, a, k) \stackrel{\text{def}}{=} c(x).[x \vdash_{fst} i]\ [x \vdash_{snd} j]\ [\{i\}_k = j].\underline{0}$$

As mentioned above, we also need to model the honest fragment of A, i.e., the part of A that deals with long-term keys. Indeed, A can never repudiate a message by declaring she has erroneously disclosed her long-term key. It is her responsibility to keep keys securely stored. The remaining part of Alice is not modelled in order to let her (possibly) behave dishonestly:

$$Initiator(a, k) \stackrel{\text{def}}{=} d(x).\overline{d}\,(\{x\}_k).Initiator(a, k)$$

Notice that this honest fragment of A uses a new channel d. This channel is used to communicate with the intruder which implicitly describes the dishonest fragment of A.

We first consider $P_B = Initiator(A, K) \parallel Responder^j(B, A, K)$, in which A and B only play the Initiator and Responder roles, respectively. Since A is the only entity which generates messages encrypted with key k, NRO is guaranteed, i.e., every action $\overline{ev\_of\_or}\,(B, A, j)$ executed by $\gamma_{nro}(P_B)$ will be preceded by a $\overline{send}\,(A, j)$ generated by the $Medium.$ Notice that $Responder^j(B, A, K)$, once decorated by γ_{nro}, executes a $\overline{ev\_of\_or}\,(B, A, j)$ only if j is of the form $\{i\}_K$, i.e., only if the MAC is valid.

To see a potential vulnerability of this protocol it is sufficient to consider two parallel sessions with exchanged roles:

$$\begin{aligned} P'_B = {} & Initiator(A, K) \parallel Responder^j(B, A, K) \parallel \\ & Initiator(B, K) \parallel Responder^j(A, B, K) \end{aligned}$$

Consider now the intruder $X_A = \overline{d^X_{send}}\,(M).d^X_{rcv}(x).\overline{c^X_{send}}\,(M, x).\underline{0}$, which asks one of the initiators to produce a MAC of M and then sends the pair $(M, MAC_K(M))$ to one of the responders. A non-repudiation problem arises when the same entity is producing the MAC and checking it. We show an execution sequence where B is convinced to receive a MAC from A but he is actually the creator of the MAC. Recall that $\gamma_{nro}(P'_B)$ introduces the medium $Medium$ of section 5.4 in between the communicating parties.

$$
\begin{array}{ll}
 & X_A \ \| \ \gamma_{nro}(P'_B) \\
\overset{\overline{send}(X,M)\ \overline{rcv}(B,M)\ \overline{send}(B,\{M\}_K)}{\Longrightarrow} & d^X_{rcv}(x).\overline{c^X_{send}}(M,x).\underline{0} \ \| \ \gamma_{nro}(P'_B) \\
 & \quad \| \ \overline{d^X_{rcv}}\{M\}_K.\overline{rcv}(X,\{M\}_K).\underline{0} \\
\overset{\overline{rcv}(X,\{M\}_K)}{\Longrightarrow} & \overline{c^X_{send}}(M,\{M\}_K).\underline{0} \ \| \ \gamma_{nro}(P'_B) \\
\overset{\overline{send}(X,(M,\{M\}_K))}{\Longrightarrow} & \gamma_{nro}(P'_B) \ \| \ \overline{c^B_{rcv}}(M,\{M\}_K).\overline{rcv}(B,(M,\{M\}_K)).\underline{0} \\
\overset{\overline{rcv}(B,(M,\{M\}_K))\ \overline{ev\_of\_or}(B,A,\{M\}_K)}{\Longrightarrow} & \gamma_{nro}(Initiator(A,K) \,\|\, Initiator(B,K) \\
 & \quad \| \ Responder^j(A,B,K))
\end{array}
$$

The execution trace is:

$$
\begin{array}{l}
\overline{send}(X,M), \overline{rcv}(B,M), \overline{send}(B,\{M\}_K), \overline{rcv}(X,\{M\}_K), \\
\overline{send}(X,(M,\{M\}_K)), \overline{rcv}(B,(M,\{M\}_K)), \overline{ev\_of\_or}(B,A,\{M\}_K)
\end{array}
$$

corresponding to the enemy X sending M to B playing the initiator role, B responding with $\{M\}_K$ and, finally, X forwarding message $\{M\}_K$ to B playing the responder role, who finally executes $\overline{ev\_of\_or}(B,A,\{M\}_K)$.

Notice that there is no $\overline{send}(A,\{M\}_K)$ matching the $\overline{ev\_of\_or}(B,A,\{M\}_K)$ event (indeed A is doing nothing). This is due to the fact that the encrypted message $\{M\}_K$ has been generated by B himself (in the Initiator role) at the execution step corresponding to $\overline{send}(B,\{M\}_K)$. Thus this trace is not a trace of $\alpha_{nro}(P'_B)$, proving that P'_B does not guarantee NRO. ■

Analogously, we define non-repudiation of origin with honest sender, i.e., NRO_{hs}. This property can be encoded in the *GNDC* schema by simply considering process P_A instead of P_B. *NRO* can be defined as the intersection of NRO_{hr} and NRO_{hs}, i.e., $P_A, P_B \in GNDC^{\alpha_{nro}}_{\leq_{trace}}$.

An analogous definition may be given for *weak-NRR*, even though the situation is slightly more complicated since, in this case, also liveness properties should be considered. In fact, it is not necessary that the message has been effectively received: it is sufficient to require that the message is “available” for reception, through, e.g. , a Trusted Third Party which makes the message downloadable, as proposed in [46]. We do not address this issue in details, since we prefer to focus our attention to another property which is also based on liveness. The property is *fairness* [47]:

- *Fairness*: At no point in the protocol run does either of participants have an advantage. In other words no one of the party can get his own evidence and avoid the other to get his corresponding evidence.

As observed in [41], this property cannot be defined as a safety property (i.e. nothing bad happens). Indeed we have to prove that whenever one of the two participants obtains his own evidence, then the other must be in the position to get his own evidence too. This can be seen as a liveness property (i.e. something good happens). For this reason, in the analysis of this property it is used the *failure* model [31] instead of the trace one. As a matter of fact, failure equivalence is actually able to observe potential deadlocks in the executions, and so it permits to see if something can be executed or not (i.e., if an evidence can be obtained or not).

The verification technique for the fairness property proposed in [41] directly fits in the *GNDC* schema. Indeed, it is reasonable to assume that an agent can require fairness from the other one only in the case he behaves correctly, i.e., if it follows the protocol. For example, the *fairness* for the sender A of receiving evidence of message receipt can be defined in the *GNDC* schema by considering a suitable relation $\leq_{failure}$ which takes into account failures, and a function α_{fair} which models the fact that after the receiver gets evidence of the origin (ev_of_or) then the sender has the possibility to obtain his own evidence of receipt (ev_of_re). This is modeled in α_{fair} by making the action ev_of_re always executable after that ev_of_or has been engaged. We show how this formalization of fairness works, through a simple example.

Example 10. Consider the following non-repudiation protocol, where Bob requires non-repudiation of origin and Alice requires non-repudiation of receipt of the same message M:

$$\begin{array}{ll} \text{message 1} & A \to B : B, M, sign_A(B, M) \\ \text{message 2} & B \to A : A, sign_B(A, M) \end{array}$$

In order for this protocol to be fair, we need to guarantee that whenever Alice gets her evidence of receipt, then Bob eventually gets his evidence of origin, and vice-versa. If A and B execute the ev_of_re and ev_of_or actions just after checking the received signatures, we easily find out that the protocol does not guarantee fairness: it is sufficient for B to quit the protocol session after he has received his evidence of origin. This corresponds to an execution trace in which ev_of_or is executed but no ev_of_re is possible after it. This failure in executing the latter action is revealed by the failure preorder, and, consequently, the attack is captured. We leave the formalization of this example as an exercise to the interested reader. ■

In [7, 8], an extension of the bisimulation-based *BNDC* property is applied to the verification of non-repudiation protocols.

5.7 Authentication in the Spi-calculus

In [2, 1] an interesting notion of authentication is proposed. The basic idea is the following: consider a protocol $P(M)$, which tries to transmit message M from one party (say A) to another one (say B). The authentication of the message

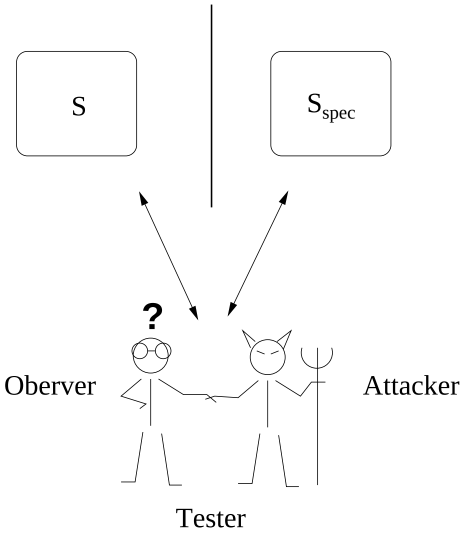

Fig. 6. Testers as "observers and attackers" in spi-authentication.

M is checked by verifying if $P(M)$ is equivalent to a specification $P_{spec}(M)$ where M is always delivered correctly. In $P_{spec}(M)$ the receiver B always knows M and whatever happens on the communication channel, B will continue its execution exactly as it had received the correct message M. In other words, $P_{spec}(M)$ represents the situation where M is always received and no enemy is able to replace it with a different message. If $P(M)$ is equivalent to $P_{spec}(M)$ then also $P(M)$ is clearly able to avoid any possible attack. The language used in [2, 1] is the spi-calculus. Moreover the may-testing equivalence [11], denoted with $\approx_{may}$, is used in order to check that $P(M)$ is equivalent to $P_{spec}(M)$ with respect to any possible interaction with the (hostile) environment. The definition of authentication in the spi-calculus has the following form: $P(M)$ guarantees spi-authentication if and only if for all M we have that

$$P(M) \approx_{may} P_{spec}(M)$$

There are many similarities between spi-calculus authentication and *NDC*: both properties are based on a notion of behavioral equivalence; moreover, they both check whether the "process under attack" behaves like a secure specification. It is however important to notice that this is done in a quite different way. In the spi-calculus the process is implicitly checked against all the possible interactions with the (hostile) environment through the use of the may-testing equivalence. There, the tester plays simultaneously both the role of the attacker and the role of the observer (see Fig. 6). On the other hand, the *NDC*-based approach performs an explicit quantification over all possible intruders, then observing the outcome of the attack (see Fig. 7).

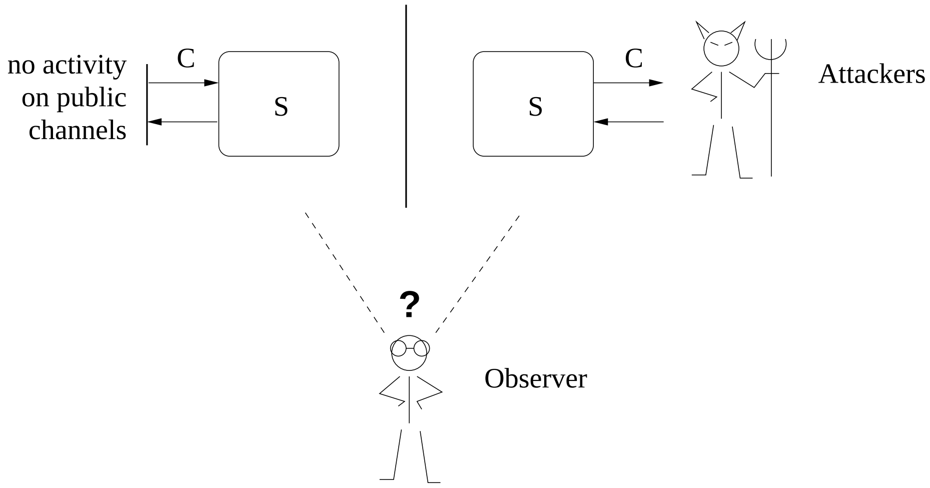

Fig. 7. Separated attackers and observers in *NDC*-authentication.

In [20] we have proved that spi-authentication can be equivalently expressed as an instance of *NDC*. To see how this is done it is useful to write a protocol in a particular style that we call *normal form*. We assume that, after delivering message M, protocol $P(M)$ executes a *continuation* process $F(M)$. In general, more than one continuation could be present. Given a protocol S, we denote all of its occurrences of continuations as $\{F_1(x_1), \ldots, F_n(x_n)\}$, where x_i represents the only free variable of F_i. From S we derive a process $S_{nf}(m_1, \ldots, m_n)$ in normal form as follows:

$$(S' \parallel \Pi_{i \in 1 \ldots n}\ p_{F_i}(x_i).F_i(x_i)) \setminus \boldsymbol{p} \tag{3}$$

where S' is the process S where every continuation $F_i(x_i)$ is replaced by $\overline{p_{F_i}}(x_i)$, and $\boldsymbol{p} = \{p_{F_1}, \ldots, p_{F_n}\}$ is a set of channels that are used neither in S nor in F_i and are not contained in C. Note that the channels in $\boldsymbol{p}$ are indexed with the continuations F_i. This is useful for managing multiple concurrent sessions between senders and receivers which can be modeled by considering n copies of the sender and n copies of the receiver in parallel. We assume that syntactically equal continuations (up to renaming of bound variables, i.e., α-conversion) correspond to the same protocol between two users but in different parallel sessions.

Given this particular form for protocols, it is quite natural to derive a secure specification. More precisely, given the normal form $S_{nf}(m_1, \ldots, m_n)$ as in (3), it is sufficient to define $S_{spec}(m_1, \ldots, m_n)$ as follows:

$$(S' \parallel \Pi_{i \in 1 \ldots n}\ p_{F_i}(x_i).[x_i = m_i]F_i(m_i)) \setminus \boldsymbol{p} \tag{4}$$

Note that every continuation is enabled only if the received message x_i is equal to the correct message m_i. Note also that in the case of multiple sessions, this simply requires that a “correct” multiset of messages is delivered from one process to another one, in whatever possible order (see the example below in this section).

We require some reasonable properties of the specification S_{spec}. First we require that it guarantees *NDC*, for all vector of messages $\mathbf{m}$. In [20] we prove that any specification S_{spec} guarantees *NDC*, under the following well-formedness condition:

WFC1 For every vector of messages $\mathbf{m}$, $S_{spec}(\mathbf{m}) \setminus C \approx_{may} \Pi_{k \in 1..n} F_k(\mathbf{m}_k)$

This condition is a very natural one as it requires that all the continuations of the specifications when there is no attacker at all, are eventually enabled. In other words, the specification is well-formed for not containing unreachable continuations. If it does, then some useless redundancy is present in S_{spec}.

We have an additional well-formedness condition:

WFC2 For every vector of messages $\mathbf{m}$, $S(\mathbf{m}) \setminus C \approx_{may} S_{spec}(\mathbf{m}) \setminus C$

The condition above is an obvious requisite for S_{spec}: the process S and its specification S_{spec} behave in the same way when the public channels are protected through the restriction (i.e., when no attack is possible).

Let $S^{\boldsymbol{F'}}$ denote the protocol S where continuations are replaced by $F'_i(x_i) = \overline{out_{F_i^*}}(x_i)$. These special continuations just send the received messages on the observable channels $out_{F_i^*}$. In [20] it is proved the following result:

Theorem 1. *Let S be a protocol which guarantees the well-formedness conditions* WFC1 *and* WFC2. *Then, S guarantees spi-authentication if and only if $S^{\boldsymbol{F'}}$ guarantees* NDC*, for all* $\mathbf{m}$.

An Example: The Wide Mouthed Frog Protocol In this section we show how to use *NDC* to analyze a simplified version (also studied in [2]) of the Wide Mouthed Frog Protocol [9], we already discussed in Section 5.5. We still consider a version of the protocol with no time-stamps, which makes the protocol sensitive to a replay attack (as already remarked in [1]). Notice that, differently from Section 5.5, in the first message B is encrypted, as required by the original protocol.

$$\begin{array}{ll} \text{message 1} & A \to S : A, \{B, k_{AB}\}_{k_{AS}} \\ \text{message 2} & S \to B : \{A, k_{AB}\}_{k_{BS}} \\ \text{message 3} & A \to B : \{m_A\}_{k_{AB}} \end{array}$$

We specify two sessions of the protocol as the following CryptoSPA normal form: [8]

$$\begin{aligned} A(m,k) &\stackrel{\text{def}}{=} \overline{c_1}\,(A, \{(B,k)\}_{k_{AS}})\ .\ \overline{c_3}\,\{m\}_k \\ B &\stackrel{\text{def}}{=} c_2(y)\ .\ [\langle y, k_{BS}\rangle \vdash_{dec} z]\ [z \vdash_{snd} s]\ c_3(t)\ .\ [\langle t, s\rangle \vdash_{dec} w]\ \overline{p_F}\,w \\ S &\stackrel{\text{def}}{=} c_1(u)\ .\ [\langle u\rangle \vdash_{snd} x]\ [\langle x, k_{AS}\rangle \vdash_{dec} y]\ [y \vdash_{snd} z]\ \overline{c_2}\,\{(A,z)\}_{k_{BS}}\ .\ S \\ P(m,m') &\stackrel{\text{def}}{=} (A(m,k_{AB}) \parallel A(m',k'_{AB}) \parallel B \parallel B \parallel S \parallel p_F(z).F(z) \parallel \\ &\qquad \parallel p_F(z).F(z)) \setminus p_F \end{aligned}$$

[8] This protocol specification is a bit simplified since there are only two users and the possible sessions are fixed in advance. However, this modelling is sufficient to show how the attack can be revealed.

where c_1, c_2 and c_3 are the three channels over which messages 1,2 and 3 are communicated, respectively. Note that we have considered two instances of A and B in order to observe the replay attack. If we consider only one instance of A and B no attack is possible here. Note also that A has different messages and session keys in the two sessions.

Let us see if the protocol guarantees the well-formedness conditions. First, it is easy to see that $P(\mathbf{m}, \mathbf{m}')$ with no enemy, i.e., $P(\mathbf{m}, \mathbf{m}') \setminus \{c_1, c_2, c_3\}$, is trace equivalent to $F(\mathbf{m}) \parallel F(\mathbf{m}')$. This represents the intended execution of the protocol. As a matter of fact the two sessions can be executed in any possible interleaving. It is also easy to prove that $P_{spec}(\mathbf{m}, \mathbf{m}') \setminus \{c_1, c_2, c_3\} \approx_{trace} F(\mathbf{m}) \parallel F(\mathbf{m}')$. Thus, we finally have $P(\mathbf{m}, \mathbf{m}') \setminus C \approx_{trace} P_{spec}(\mathbf{m}, \mathbf{m}') \setminus C \approx_{trace} F(\mathbf{m}) \parallel F(\mathbf{m}')$.

Since $P(\mathbf{m}, \mathbf{m}')$ is well-formed, in order to check spi-authentication on it, we can verify if $P^{\boldsymbol{F}'}(\mathbf{m}, \mathbf{m}')$ guarantees NDC-authentication. Note that $P^{\boldsymbol{F}'}(\mathbf{m}, \mathbf{m}')$ is obtained from $P(\mathbf{m}, \mathbf{m}')$ by replacing $F(w)$ with $\overline{out_F}\, w$. Note also that in the two instances of B the continuation is exactly the same (this allows to model multiple sessions).

We show that $P^{\boldsymbol{F}'}(\mathbf{m}, \mathbf{m}')$ does not guarantee such a property. Consider the enemy $X \stackrel{\text{def}}{=} c_2(x).\overline{c_2}\, x.\overline{c_2}\, x.c_3(y).\overline{c_3}\, y.\overline{c_3}\, y$. It is easy to see that $P \parallel_C X$ is able to execute the trace $\overline{out_F}\,\mathbf{m}.\overline{out_F}\,\mathbf{m}$ that is not a trace for process $\overline{out_F}\,\mathbf{m} \parallel \overline{out_F}\,\mathbf{m}'$. Hence, $P^{\boldsymbol{F}'}(\mathbf{m}, \mathbf{m}') \parallel_C X \;\not\leq_{trace}\; \overline{out_F}\,\mathbf{m} \parallel \overline{out_F}\,\mathbf{m}' \;\approx_{trace}\; P^{\boldsymbol{F}'}(\mathbf{m}, \mathbf{m}') \setminus C$ and NDC-authentication is not satisfied.

The enemy X intercepts messages 2 and 3 and replays them, inducing B to commit twice on message $\mathbf{m}$ (as shown by trace $\overline{out_F}\,\mathbf{m}.\overline{out_F}\,\mathbf{m}$). This attack is quite critical in some situations. As an example, $\mathbf{m}$ could be a request of money transfer that would be executed twice. In order to avoid this attack, it is possible to modify the protocol (as done in [2]) by adding nonce-based challenge responses.

6 Some Simple Comparison Results

In this section, we show that having a uniform treatment of security properties make it easier to study the relationships and the differences among them. First, we show that NDC may be seen as a sufficient condition for every property which is based on trace-preorder. This result is interesting since it relates the non-interference property NDC, originally proposed for modeling information flow security, to network security properties like, e.g., authentication. The result holds for what we will call *good candidates* for a pair of functions γ, α, i.e., processes P such that $\gamma(P) \setminus C \leq_{trace} \alpha(P)$. This condition is quite reasonable since we certainly want that at least the (decorated) protocol under no attacks, i.e., $\gamma(P) \setminus C$, "satisfies" $\alpha(P)$.

Proposition 13. *Let γ be a decoration function, let α be a function between processes and let P be a good candidate for γ, α, i.e., $\gamma(P) \setminus C \leq_{trace} \alpha(P)$. Then, $\gamma(P)$ is NDC implies that P is $GNDC^{\gamma,\alpha}_{\leq_{trace}}$.*

Proof. *The fact that $\gamma(P)$ is NDC can be equivalentely expressed by the fact that P is $GNDC^{Id_C,\gamma(P)\backslash C}_{\approx_{trace}}$. Then, for every $X \in \mathcal{E}^{\phi}_C$ we have $Id_C(\gamma(P)) \parallel_C X \leq_{trace} \gamma(P)\backslash C$, i.e., $\gamma(P) \parallel_C X \leq_{trace} \gamma(P)\backslash C$. By the hypothesis that P is a good candidate, i.e., $\gamma(P)\backslash C \leq_{trace} \alpha(P)$, we obtain $\gamma(P) \parallel_C X \leq_{trace} \gamma(P)\backslash C \leq_{trace} \alpha(P)$ and so P is $GNDC^{\gamma,\alpha}_{\leq_{trace}}$.* ■

Note that if a pair γ, α does not have good candidates then it represents an empty property (no process satisfies it).

The result above shows that NDC, required on a process $\gamma(P)$, is stronger than any $GNDC^{\gamma,\alpha}_{\leq_{trace}}$ property, for all the good candidates P. For example, $\gamma_{Agree}(P) \in NDC$ implies $GNDC^{\gamma_{Agree},\alpha_{Agree}}_{\leq_{trace}}$ and $\gamma_{auth}(P) \in NDC$ implies $GNDC^{\gamma_{auth},\alpha_{auth}{}^{T}_{R}}_{\leq_{trace}}$, for their respective good candidates. This is quite intuitive since NDC basically requires that the protocol behaviour is completely preserved even under attack. So if $\gamma(P)$ satisfies a certain property under no attacks (i.e., it is a good candidate), the fact that $\gamma(P)$ is NDC will preserve such a property under every possible attack.

Example 11. Consider once more the protocol first presented in Example 1 written in the form required by the agreement property (Example 8).

$$\begin{aligned} P &= C(A, B, K_s) \parallel V^z(B, A) \\ C(a, b, k) &= \overline{c}\,\{k, a\}_K.\underline{0} \\ V^z(b, a) &= c(y).[\langle y, K\rangle \vdash_{dec} w][w \vdash_{fst} z].\underline{0} \end{aligned}$$

Recall that

$$\begin{aligned} \gamma_{Agree}(P) = {} & \overline{running\_ini}\,(A, B, K_s).\overline{c}\,\{K_s, A\}_K.\underline{0} \\ & \parallel\ c(y).[\langle y, K\rangle \vdash_{dec} w][w \vdash_{fst} z].\overline{commit\_res}\,(B, A, z) \end{aligned}$$

It is trivial to prove that P is a good candidate for $\gamma_{Agree}, \alpha_{Agree}$, i.e., $\gamma_{Agree}(P)\backslash C \leq_{trace} \alpha_{Agree}(P)$. As a matter of fact the only trace of $\gamma_{Agree}(P) \setminus C$ is $\overline{running\_ini}\,(A, B, K_s), \overline{commit\_res}\,(B, A, K_s)$. Thus, we can use NDC to check the agreement property, i.e., if we prove that $\gamma(P)$ is NDC then, by Proposition 13, we also have that $P \in GNDC^{\gamma_{Agree},\alpha_{Agree}}_{\leq_{trace}}$.

It can be checked (e.g., using the tools described in [15]) that $\gamma_{Agree}(P)$ is NDC whenever $K \notin \phi_I$. However, as noticed in Section 1.3, this protocol is flawed when more then one session is considered. As a consequence we have that $\gamma_{Agree}(P \parallel P)$ cannot be NDC (otherwise we would get a contradiction). As done in Example 8 we consider an enemy $X = c(w).\overline{c}\,(w).\overline{c}\,(w)$. It is easy to see that trace $\overline{running\_ini}\,(A, B, K_s).\overline{commit\_res}\,(B, A, K_s).\ \overline{commit\_res}\,(B, A, K_s)$ is in the set $Tr(\gamma_{Agree}(P \parallel P) \parallel_C X)$ but not in $Tr(\gamma_{Agree}(P \parallel P)\backslash C)$, as the second commit is not matched by any running action. So, as expected, $\gamma_{Agree}(P \parallel P)$ is not NDC. ■

In general, we observe that if $\lhd \subseteq \lhd'$ then $GNDC^{\gamma,\alpha}_{\lhd} \subseteq GNDC^{\gamma,\alpha}_{\lhd'}$, furthermore if for all $P \in \mathcal{E}$ we have $\alpha(P) \lhd \alpha'(P)$ and $\gamma'(P) \lhd \gamma(P)$ [9] then $GNDC^{\gamma,\alpha}_{\lhd} \subseteq GNDC^{\gamma',\alpha'}_{\lhd}$.

[9] Notice that α and γ are counter-variant here.

As an example, we study the natural extension of secrecy to two messages. Consider a protocol $P(m_1, m_2)$ and let us define two different γ decoration functions: γ^1 requiring the secrecy of m_1 and γ^2 requiring the secrecy of both the messages:

$$\begin{aligned}\gamma^1_{secret}(P(m_1, m_2)) &= P(m_1, m_2) \parallel KN(m_1)\\ \gamma^2_{secret}(P(m_1, m_2)) &= P(m_1, m_2) \parallel KN(m_1) \parallel KN(m_2)\end{aligned}$$

It is trivial to see that, for all $P(m_1, m_2)$ we have $\gamma^1_{secret}(P(m_1, m_2)) \leq_{trace} \gamma^2_{secret}(P(m_1, m_2))$. Thus we obtain $GNDC^{\gamma^2_{secret},\alpha_{secret}}_{\leq_{trace}} \subseteq GNDC^{\gamma^1_{secret},\alpha_{secret}}_{\leq_{trace}}$, reflecting the intuition that requiring the secrecy of both the messages is stronger than requiring the secrecy of the first one only.

We give another simple example, by considering message-oriented authentication. It is easy to prove that $\alpha_{auth^T_R}(P) \leq_{trace} \alpha_{auth^{T'}_{R'}}(P)$ whenever $T \geq T'$ and $R \leq R'$. As a consequence $GNDC^{\gamma_{auth},\alpha_{auth^T_R}}_{\leq_{trace}} \subseteq GNDC^{\gamma_{auth},\alpha_{auth^{T'}_{R'}}}_{\leq_{trace}}$, reflecting the intuition that actions in T are executable only after at least one action in R has been executed. If we increment the set R, the number of processes satisfying message-oriented authentication increases. The same happens when we decrease set T, because the actions that are no more in T become direcly executable.

We end this section by showing how to apply the theory developed so far in order to check multiple properties with just one NDC check. We have observed that enlarging the decoration γ makes the resulting property stronger, i.e., if, for all P, $\gamma'(P) \lhd \gamma(P)$ then $GNDC^{\gamma,\alpha}_{\lhd} \subseteq GNDC^{\gamma',\alpha}_{\lhd}$. Thus, given two different properties characterized by the pairs (γ_1, α_1) and (γ_2, α_2), we can try to find a larger γ_3 that contains both the two decorations, i.e., such that $\gamma_1(P) \lhd \gamma_3(P)$ and $\gamma_2(P) \lhd \gamma_3(P)$, for all P. Intuitively, this corresponds to considering the sum of the two decorations. By Proposition 13 we directly obtain that if $\gamma_3(P) \setminus C \leq_{trace} \alpha_1(P)$ and $\gamma_3(P) \setminus C \leq_{trace} \alpha_2(P)$ then $\gamma_3(P)$ is NDC implies that P is $GNDC^{\gamma_1,\alpha_1}_{\leq_{trace}}$ and $GNDC^{\gamma_2,\alpha_2}_{\leq_{trace}}$. Thus, NDC may be used to simultaneously check both the two properties. Notice that, in general, the two conditions $\gamma_3(P) \setminus C \leq_{trace} \alpha_1(P)$ and $\gamma_3(P) \setminus C \leq_{trace} \alpha_2(P)$ are not restrictive, as $\alpha_1(P)$ and $\alpha_2(P)$ impose restrictions only on the actions added by their respective decorations. Thus, e.g., $\gamma_1(P) \setminus C \leq_{trace} \alpha_1(P)$ should imply that also $\gamma_3(P) \setminus C \leq_{trace} \alpha_1(P)$.

Example 12. Consider once more the protocol P discussed in Example 11 and let us write it as $P(K_s)$, i.e., parametric with respect to the session key.

$$\begin{aligned}P(K_s) &= C(A, B, K_s) \parallel V^z(B, A)\\ C(a, b, k) &= \overline{c}\,\{k, a\}_K.\underline{0}\\ V^z(b, a) &= c(y).[\langle y, K\rangle \vdash_{dec} w][w \vdash_{fst} z].\underline{0}\end{aligned}$$

Recall that

$$\begin{aligned}\gamma_{Agree}(P(K_s)) = {} & \overline{running\_ini}\,(A, B, K_s).\overline{c}\,\{K_s, A\}_K.\underline{0}\\ & \parallel\ c(y).[\langle y, K\rangle \vdash_{dec} w][w \vdash_{fst} z].\overline{commit\_res}\,(B, A, z)\end{aligned}$$

Suppose that we also want to check whether or not the secrecy of the session key is guaranteed. Recall that

$$\gamma_{secret}(P(K_s)) = P(K_s) \parallel KN(K_s)$$

Now if we simply consider the new decoration function

$$\gamma_{A\&S}(P(K_s)) = \gamma_{Agree}(P(K_s)) \parallel KN(K_s)$$

we obtain that $\gamma_{Agree}(P(K_s)) \leq_{trace} \gamma_{A\&S}(P(K_s))$ and $\gamma_{secret}(P(K_s)) \leq_{trace} \gamma_{A\&S}(P(K_s))$. Intuitively, $\gamma_{A\&S}(P(K_s))$ is the sum of the two decorations. Now, it is trivial to prove that $P(K_s)$ is still a good candidate for $\gamma_{A\&S}, \alpha_{Agree}$, i.e., $\gamma_{A\&S}(P) \setminus C \leq_{trace} \alpha_{Agree}(P)$. As a matter of fact the only trace of $\gamma_{A\&S}(P) \setminus C$ is $\overline{running\_ini}\,(A, B, K_s), \overline{commit\_res}\,(B, A, K_s)$. For the same reason, it is also trivial to see that $P(K_s)$ is a good candidate for $\gamma_{A\&S}, \alpha_{secret}$. Thus, we can use NDC to simultaneously check agreement and secrecy. It can be checked (e.g., using the tools described in [15]) that $\gamma_{A\&S}(P(K_s))$ is NDC whenever $K \notin \phi_I$. Thus, for a single session, $P(K_s)$ guarantees both agreement and the secrecy of the session key. We have seen that for multiple sessions agreement is not guaranteed. As a matter of fact, trying to check NDC on $\gamma_{A\&S}(P(K_s) \parallel P(K'_s))$ fails because of the attack on authentication discussed in Example 11. ■

7 Conclusion

In this paper we have proposed a general scheme, called GNDC, that allows us to specify different security properties in a uniform setting. Rephrasing different properties in our uniform scheme is interesting from different perspectives: (*i*) it allows to better understand and compare the properties of interest, (*ii*) it allows to reuse general results and proof techniques, (*iii*) it allows to check all the rephrased properties as just one NDC check, (*iv*) since NDC is supported by the automated tools CVS/CoSeC [15], the GNDC scheme allows us to apply such tools to all the rephrased properties.

This analysis technique has been applied to many case studies, sometimes finding new failures or variant of known failures [14].

There are many extensions to the model presented in this paper that have already been developed. Extensions with time [21] allow us to detect attacks due to timing covert-channels and to analyze real-time cryptographic protocols [29]. Extensions with probabilities [3, 4] allow us to refine process specification in order to detect flaws that are related to event probabilities. The extension of probabilistic models to deal with cryptographic protocols is currently under development. We are planning to compare our approach with other ones, based on different process calculi. Some preliminary results in this direction can be found in [30]. An alternative approach based on logical specification of the correct behavior and on partial model checking techniques has been proposed in [35–37].

References

1. M. Abadi and A. D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In *Proceedings of CONCUR'97*, pages 59–73. Lecture Notes in Computer Science 1243, 1997.
2. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.
3. A. Aldini, M. Bravetti, R. Gorrieri. A Process-algebraic Approach for the Analysis of Probabilistic Non-interference. To appear on *Journal of Computer Security*, 2003.
4. A. Aldini. Probabilistic Information Flow in a Process Algebra. In *12th Int. Conference on Concurrency Theory* (CONCUR'01), Springer LNCS 2154:152-168, Aalborg, Denmark, August 2001.
5. C. Bodei, P. Degano, R. Focardi, and C. Priami. Authentication via localized names. In *Proceedings of CSFW'99*, pages 98–110. IEEE press, 1999.
6. D. E. Bell, L. J. La Padula. Secure Computer Systems: Unified Exposition and Multics Interpretation. ESD-TR-75-306, MITRE MTR-2997, 1976.
7. M. Bugliesi, A. Ceccato, S. Rossi. Non-Interference Proof Techniques for the Analysis of Cryptographic Protocols. In *Proceedings of 2003 IFIP WG 1.7, ACM SIGPLAN and GI FoMSESS Workshop on Issues in the Theory of Security* (WITS'03), April 5 - 6, 2003, Warsaw, Poland.
8. M. Bugliesi, A. Ceccato, and S. Rossi. Context-Sensitive Equivalences for Non-Interference based Protocol Analysis. In *Proc. of the 14th International Symposium on Fundamentals of Computation Theory*, FCT 2003, LNCS 2751, pag. 364-375, Springer-Verlag, 2003.
9. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proceedings of the Royal Society of London*, 426:233–271, 1989.
10. J. Clark and J. Jacob. A Survey of Authentication Protocol Literature: Version 1.0. http://www.cs.york.ac.uk/∼jac/papers/drareview.ps.gz, November 1997.
11. R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.
12. D. Dolev, A.C. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2), 1983.
13. A. Durante, R. Focardi, and R. Gorrieri. CVS: A compiler for the analysis of cryptographic protocols. In *Proceedings of CSFW'99*, pages 203–212. IEEE press, 1999.
14. A. Durante, R. Focardi, and R. Gorrieri. CVS at Work: A Report on new Failures upon some Cryptographic Protocols. In *Procs. International Workshop Mathematical Methods, Models and Architectures for Computer Networks Security*, LNCS 2052, 287-299, Springer, St. Petersburg, Russia, 2001.
15. A. Durante, R. Focardi, and R. Gorrieri. A compiler for analysing cryptographic protocols using non-interference. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 9(4):488–528, October 2000.
16. R. Focardi, A. Ghelli, and R. Gorrieri. Using non interference for the analysis of security protocols. In *Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols*, 1997.
17. R. Focardi and R. Gorrieri. A classification of security properties for process algebras. *Journal of Computer Security*, 3(1):5–33, 1994/1995.
18. R. Focardi and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Transactions on Software Engineering*, 23(9):550–571, 1997.

19. R. Focardi, R. Gorrieri. Classification of Security Properties. Part I: Information Flow. in *Foundations of Security Analysis and Design* (R.Focardi, R.Gorrieri eds), LNCS 2171, 331-396, Springer, 2001.
20. R. Focardi, R. Gorrieri, and F. Martinelli. A Comparison of Three Authentication Properties. *Theoretical Computer Science*, Volume 291(3), Pages 285-327, January 2003, Elsevier Science.
21. R. Focardi, R. Gorrieri and F. Martinelli. Real-Time Information Flow Analysis. *IEEE Journal on Selected Areas in Communications*. January 2003, Volume 21, Number 1. IEEE press.
22. R. Focardi, R. Gorrieri, and F. Martinelli. Message authentication through non-interference. In *Proceedings of 8th International Conference in Algebraic Methodology and Software Technology* (AMAST 2000), pages 258–272. Springer Lecture Notes in Computer Science 1816, 2000.
23. R. Focardi, R. Gorrieri, and F. Martinelli. Non interference for the analysis of cryptographic protocols. In *proceedings of 27th International Colloquium on Automata, Languages and Programming (ICALP'00), (U. Montanari, ed.)*, pages 354–372. Lecture Notes in Computer Science 1853, July 2000.
24. R. Focardi, R. Gorrieri, and F. Martinelli. Secrecy in security protocols as non-interference. Workshop on secure architectures and information flow, ENTCS 32, 2000.
25. R. Focardi and F. Martinelli. A uniform approach for the definition of security properties. In *Proceedings of World Congress on Formal Methods (FM'99)*, pages 794–813. Springer, Lecture Notes in Computer Science 1708, 1999.
26. J. A. Goguen and J. Meseguer. Security policy and security models. In *Proceedings of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Press, 1982.
27. D. Gollman. What do we mean by entity authentication? In *Proceedings of Symposium in Research in Security and Privacy*, pages 46–54. IEEE Press, 1996.
28. D. Gollman. On the verification of cryptographic protocols - a tale of two committees. In *Workshop on secure architectures and information flow*, volume 32 of *ENTCS*, 2000.
29. R. Gorrieri, E. Locatelli, F. Martinelli. A Simple Language for Real-Time Cryptographic Protocol Analysis. In Proceedings of *12th European Symposium on Programming, (ESOP 2003)*, Springer LNCS 2618, pages 114-128, April 2003.
30. R. Gorrieri, F. Martinelli. Process Algebraic Frameworks for the Specification and Analysis of Cryptographic Protocols. In procs. 28th International Symposium on *Mathematical Foundations of Computer Science* (MFCS03), LNCS 2747, 46-67, Springer, Bratislava (SK), August 2003.
31. C.A.R. Hoare. Communicating Sequential Processes. Prentice-Hall, 1985.
32. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of TACAS'96*, pages 146–166. Lecture Notes in Computer Science 1055, 1996.
33. G. Lowe. A hierarchy of authentication specification. In *Proceedings of the 10th Computer Security Foundation Workshop*, pages 31–43. IEEE press, 1997.
34. W. Marrero, E. Clarke, and S. Jha. A model checker for authentication protocols. In *Proceedings of DIMACS Workshop on Design and Formal Verification of Security Protocols*. Rutgers University, Sep. 1997.
35. F. Martinelli. Analysis of security protocols as open systems. *Theoretical Computer Science* 290(1): 1057-1106 (2003)
36. F. Martinelli. Languages for description and analysis of authentication protocols. In *Proceedings of ICTCS'98*, pages 304–315. World Scientific, 1998.

37. F. Martinelli. Partial model checking and theorem proving for ensuring security properties. In *Proceedings of CSFW'98*, pages 44–52. IEEE press, 1998.
38. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
39. P. Y. A. Ryan and S. Schneider. Process algebra and non-interference. In *Proceedings of CSFW'99*, pages 214–227. IEEE press, 1999.
40. D. Sangiorgi. *Expressing Mobility in Process Algebra: First-Order and Higher-Order Paradigms*. PhD thesis, University of Edinburgh, 1992.
41. S. Schneider. Formal analysis of a non-repudiation protocol. In *Proceedings of CSFW'98*, pages 54–65. IEEE Press, 1998.
42. S. Schneider. Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9), September 1998.
43. P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley, 2001.
44. J. T. Wittbold and D. M. Johnson. Information flow in nondeterministic systems. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy*, pages 144–161. IEEE Computer Society Press, 1990.
45. T. Y. C. Woo and S. S. Lam. A semantic model for authentication protocols. In *Proceedings of the 1993 IEEE Computer Society Symposium on Security and Privacy (SSP '93)*, pages 178–195. IEEE Press, May 1993.
46. J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proc. of Symposium in Research in Security and Privacy*, pages 55–61. IEEE Press, 1996.
47. J. Zhou and D. Gollmann. Towards verification of non-repudiation protocols. In *International Refinement Workshop and Formal Methods Pacific*, 1998.