In *Proceedings of ICALP'00, Springer LNCS 1853, July 2000.*

Non Interference for the Analysis of Cryptographic Protocols *

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

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy.
e_mail: `focardi@dsi.unive.it`
[2] Dipartimento di Scienze dell'Informazione, Università di Bologna, Italy.
e_mail: `gorrieri@cs.unibo.it`
[3] Istituto per le Applicazioni Telematiche C.N.R., Pisa, Italy.
e_mail: `Fabio.Martinelli@iat.cnr.it`

Abstract. Many security properties of cryptographic protocols can be all seen as specific instances of a general property, we called Non Deducibility on Composition (NDC), that we proposed a few years ago for studying information flow properties in computer systems. The advantage of our unifying theory is that formal comparison among these properties is now easier and that the full generality of NDC has helped us in finding a few new attacks on cryptographic protocols.

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 properties, especially authentication. As a matter of fact, formal definitions, e.g. of authentication, have rarely been given, not widely agreed upon, usually not compared and only recently proposed in the literature (see, e.g., [5, 17, 21, 26]). 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 w.r.t. 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.

* Work partially supported by MURST Progetto TOSCA and Progetto "Certificazione automatica di programmi mediante interpretazione astratta"; CNR Progetto "Modelli e Metodi per la Matematica e l'Ingegneria"; CSP Progetto "ISA: Isp Secured trAnsactions".

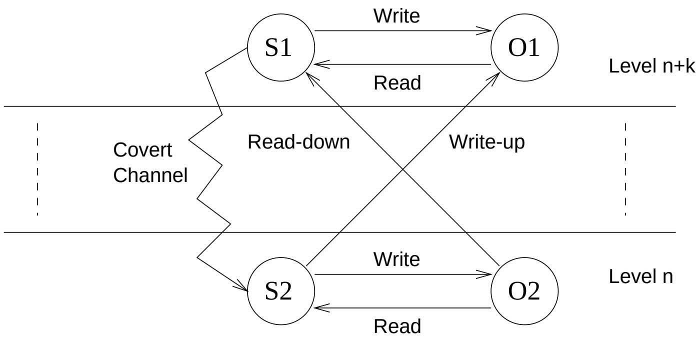

Fig. 1. Multilevel security: a high subject S_1 cannot write a low object O_2 and a low subject S_2 cannot read a high object O_1.

Our claim is that a classic approach to security, used to study information flow in multilevel [4] 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 w.r.t. 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: at a predefined initial time, the high process can trasmit 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, hence one bit each two clock cycles.

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 [16] for deterministic state machines. In recent work [10, 11], 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 [10, 11], we use a CCS-like process algebra [23], 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 by construction high (low) level processes. These processes are secure because their activities (expressed by the actions they perform) are confined inside the high (low) level. More interesting is the case of mixed (with actions from both L and H) processes, i.e., of those processes that even if belonging to the high (low) level may perform interactions with low (high) objects, because for them we want to know if they allow information to flow in the wrong direction. Among the many non interference-like properties, we advocate one special property, called *Non Deducibility on Composition* (NDC for short), that can be expressed as follows:

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

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

No information flow from high to low

is expressed by

No high level process can change the low behaviour.

1.2 Non interference for security protocols

NDC essentially says that, given two groups of users H and L, there is no information flow from H to L iff there is no way for H to modify the behaviour of L. Analogously, we may think that L is the set of the honest participants to a protocol and H is the external, possibly malicious, environment, i.e. the set of possible intruders (or enemies). Following the analogy, no information flow from high to low means that the intruders have no way to change the low behaviour of the protocol.

To set up this correspondence more precisely, we have to single out the high level actions and the low level ones in this setting. We should assume that an intruder may have complete control of the network, and so it is reasonable to assume that the public channels (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 is not clear 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 is property dependent. For instance,

[1] Actually, NDC in [10, 11] is this property when $\approx$ is trace equivalence; other similar properties have been proposed by changing the relation, e.g., $BNDC$ is as above where $\approx$ is weak bisimulation.

we will see that to model some form of authentication as in [20], 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* (a crypted information can be known by an enemy only if he knows the decryption key), the initial knowledge of an ebemy 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 [15, 14, 13], the set $\mathcal{E}_C^{\phi_I}$ of all the possible high level processes 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, $ID(X)$ 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 explicitly given. The NDC property for a protocol P can hence be reformulated as:

$$P \in NDC \text{ iff } \forall X \in \mathcal{E}_C^{\phi_I} : (P \,\|\, X) \setminus C \approx P \setminus C$$

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 observables included in 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 $(P \,\|\, X) \setminus C$, then this clearly means that X is not able to modify in any way the observable execution of P, i.e., the security property hold.

The actual scheme we use, called $GNDC$, is a bit more general, where $\approx$ is any pre-order and $P \setminus C$ is replaced by a function on P, $\alpha(P)$, expressing the property as a set of processes having a special format that more directly and intutively recall the property of interest. Nonetheless, it is possible to show that NDC is the strongest property we can reasonably define over cryptographic protocols (for more details, see [15]).

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 (see [15]). By removing the universal quantification, NDC can be verified by one single, albeit huge, check. The most general intruder is an intruder 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 Plan of the paper

What do we want to show with this paper? Our primary goal 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, extra observable actions as well as a suitable behavioural equivalence. Some work in this direction has been reported in [14, 13, 15]. We think that the advantages of our approach include at least the following:

- *one check for all*: As all the properties are defined in the same NDC style, it is possible to put in the specification the extra actions for all the properties of interest, hence obtaining that one check for this rich case implies that all the properties are satisfied.
- *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 (e.g., see [13, 15] for some preliminary results in this direction).
- *accuracy*: So far we have analyzed about 40 protocols (with the help of an automatics tool [8, 9]) of a well-known library of crypto-protocols [6]. Two supposedly correct protocols have been shown incorrect and for a few additional flawed protocols some new attacks have been found. 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. In Section 2 we gently introduce the reader to the realm of security properties; by means of some simple examples, five security properties are informally described. Section 3 shows that all these properties are actually instances of the general scheme $GNDC$ and that NDC (when all the suitable extra actions have been inserted) implies them all. Section 4 shows one larger example, the Woo & Lam mutual authentication protocol as reported in Schneier's textbook [27]. This version of the protocol is flawed. Finally, Section 5 reports some final remarks and future work.

2 Security properties

In this section we present some typical security properties through some simple examples. All these properties have been defined for different aims and have been formalized using various models. Indeed, the examples will allow us to identify a *general common idea* behind all of these properties.

2.1 A simple key-exchange protocol

The first example we consider is a simple key-exchange protocol (see [11]) with public key cryptography. There is a process KDC (Key Distribution Center) on a remote host which is devoted to the distribution of the public keys. In particular, when *Alice* (A) need to send some secret information to *Bob* (B), A asks KDC for B's public key. Then, A can encode every secret message with this key and is guaranteed that only B will be able to read it. In particular, in the protocol we are going to analyze, A sends to B a session key encoded with the public key of B. Such session key will be used for every further communication, until the two users decide to establish a new session with a new session key.

The aim of the protocol is simply to distribute to B the session key generated by A. Since A uses the public key of B, she is assured that only B will know the session key and so there is a form of implicit one-side authentication of B for A. The opposite authentication is not valid since B has no guarantees about who he is talking to. Anyone can encrypt a session key with B's public key and send it pretending to be A. So, B will presumably check the identity of A during the session (if this is required). We could imagine a situation of a remote login on a machine B by a user A. First the user establishes a session key with the remote login server using the protocol above. Then the communication proceeds encrypted with the session key and the server checks the identity of A using, for instance, some login/password mechanism. However A is assured to communicate with B since only B can have received the session key.

In order to formalize the protocol we use the notation $A \to B : msg$ representing the sending of message msg from A to B. A possible definition of the protocol could be the following sequence of four message exchanges:

$$\begin{array}{lrl}
\text{Message 1} & A \to KDC : & A, B \\
\text{Message 2} & KDC \to A \quad\ : & PK_B \\
\text{Message 3} & A \to B \quad\ : & \{K_{sess}\}_{PK_B} \\
\text{Message 4} & B \to A \quad\ : & \{M\}_{K_{sess}}
\end{array}$$

where Message 1 is the request, sent from A to KDC, of B's public key; Message 2 is the reply from KDC to A, containing the public key PK_B of B; Message 3 contains the session key K_{sess} encoded using the public key PK_B and is sent to B; finally, in Message 4, B uses the session key to send a message M to A.

Different security properties can be now considered: (i) *message authenticity*, e.g., message M should be authentic from B (or as it would have been sent from B) since only A and B should know the session key at the end of the protocol; (ii) *entity authentication*, e.g., if A receives the last message encrypted with the correct key, then A should be guaranteed that B has run the protocol with her (or at least is "alive"); (iii) *secrecy*, e.g., at the end of the protocol, the session key and the message M should be known only to A and B. In the following, we consider all of these properties, using the protocol above as a running example.

2.2 Message Authenticity

Suppose that we can fix the message M that B is willing to send to A. This means that M does not represent a generic message, but it is a particular one. If we can do this, we can check message authenticity by just considering all the possible runs of the protocol and by requiring that, in such runs, A always receives the correct message M, i.e. the message B wanted to send to A. If this is true, we can conclude that the protocol is indeed guaranteeing that no one is able to force A accepting a faked message M'. This notion of message authenticity is due to Abadi and Gordon [2]. [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 faked M'. (As mentioned in the Introduction, we have to consider an initial knowledge ϕ_I but, for sake of readibility, we often omit it.) We can thus rephrase the message authenticity property as follows:

> "Whatever hostile environment is considered, A will never receive (as part of Message 4) during all her possible runs a message different from M".

A second important issue is now also evident. We are requiring that a particular piece of information sent inside a particular message differs from a certain fixed message M. Indeed, this property looks really ad-hoc for the key exchanged protocol considered here. This is quite typical when trying to define precisely security properties, since they often depend on the structure of the analyzed protocol/system. We can make more intuitive the specification by using an event *received*(m) corresponding to the fact that A is receiving message m. If $P(m)$ is the protocol where *Bob* is willing to send message m to *Alice* we can just state that

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

We will further generalize this idea in the following. Before that we show that the protocol we have considered until now does not guarantee message authenticity. The weakness is indeed in the second message. An enemy can easily intercept the public key of B and substitute it with its own key. This allows the enemy to learn the session key and to send a faked message M' as follows:[3]

Message 1	$A \rightarrow KDC$:	A, B	
Message 2	$KDC \rightarrow \mathbf{E}(A)$:	PK_B	E intercepts this
Message 2′	$\mathbf{E}(KDC) \rightarrow A$:	$\mathbf{PK_E}$	
Message 3	$A \rightarrow \mathbf{E}(B)$:	$\{K_{sess}\}_{\mathbf{PK_E}}$	
Message 4	$\mathbf{E}(B) \rightarrow A$:	$\{\mathbf{M'}\}_{K_{sess}}$	event *received*(M')

[2] Indeed, in [2] a universal quantification over all the possible messages to be sent is required. For the sake of simplicity we do not consider it here.

[3] We denote with $E(U)$ the enemy which is impersonating the entity U. So $E(U) \rightarrow A : M$ means that E sends M to A by simulating U, while $A \rightarrow E(U) : M$ that E intercepts the message M from A to U (so U receives nothing). However we must point out that this message sequences notation should be used only for the intuitive description of the protocols (and attacks) and not for their formal analysis as remarked by many authors (e.g., see [1]).

Thus, message authenticity does not hold since, in this particular execution, an event *received*(M') occurs and M' can be whatever message (different from M) the enemy is willing to send to A.

2.3 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 (e.g., see [18, 21, 30]).

As an example, in our protocol we would like that whenever A receives the last message then B has indeed executed the protocol. Consider two events $commit(A, B)$ and $run(B, A)$ representing the fact that A has successfully terminated the protocol apparently with B and B has at least started the protocol (i.e., he has received a session key) with A. It is now sufficient to require that event $commit(A, B)$ is always preceded by event $run(B, A)$ [21]. In other words $commit(A, B)$ should not happen if B has not started the protocol. Similarly to the previous property we can require that:

> "P guarantees entity authentication of B with respect to A if whatever hostile environment is considered, it can never occur an event $commit(A, B)$ when $run(B, A)$ has not occurred previously".

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

Message 1	$A \rightarrow KDC$:	A, B	
Message 2	$KDC \rightarrow \mathbf{E}(A)$:	PK_B	E intercepts this
Message 2′	$\mathbf{E}(KDC) \rightarrow A$:	$\mathbf{PK_E}$	
Message 3	$A \rightarrow \mathbf{E}(B)$:	$\{K_{sess}\}_{\mathbf{PK_E}}$	
Message 4	$\mathbf{E}(B) \rightarrow A$:	$\{\mathbf{M'}\}_{K_{sess}}$	event $commit(A, B)$

Since B is doing nothing, no event $run(B, A)$ can happen and the entity authentication property does not hold. As a matter of fact, in the attack sequence the enemy is indeed able to mask as B. This means that A cannot be sure about the identity of the other party, i.e., no entity authentication is guaranteed.

Note that entity authentication and message authenticity are indeed 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 useless. Consider the following (faulty) authentication protocol:

Message 1	$A \rightarrow B$:	$\{N_A\}_{K_{AB}}$
Message 2	$B \rightarrow A$:	N_A

In order to verify the identity of B, A sends a challenge N_A (typically a random number) to B encrypted with symmetric key K_{AB} which is only known by A

and B. Only B will be able to decrypt N_A and send it back to A. Note that here B has no private messages to send to A. As a consequence, the following entity authentication attack does not represent a message authenticity attack:

Message 1	$A \rightarrow \mathbf{E}(B)$:	$\{N_A\}_{K_{AB}}$	
Message 1′	$\mathbf{E}(B) \rightarrow A$:	$\{N_A\}_{K_{AB}}$	
Message 2′	$A \rightarrow \mathbf{E}(B)$:	N_A	
Message 2	$\mathbf{E}(B) \rightarrow A$:	N_A	events $commit(A, B)$, $received(N_A)$

This is a typical parallel session attack: 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 $run(B, A)$. Note also that we have tried to detect a possible message authenticity attack by observing the $received(N_A)$ event. However N_A is exactly the expected message and, moreover, no other message would be accepted by A. In other words, we have no message here to authenticate and entity authentication should be guaranteed (indeed it is not) by the possibility for B of decrypting a message.

2.4 Secrecy

Let us now consider the third property: secrecy. This is quite intuitive and requires that messages declared to be secret should not be learnt by unauthorized users. We can consider a new event $learnt(M)$ that represents the fact that a certain (secret) message M has been learnt by the external environment (i.e., by the enemy). So, this new (low) event is performed by the enemy and not by the honest participants, as for the previously analysed properties. In our first example of attack we had such an event for K_{sess}:

Message 1	$A \rightarrow KDC$:	A, B	
Message 2	$KDC \rightarrow \mathbf{E}(A)$:	PK_B	E intercepts this
Message 2′	$\mathbf{E}(KDC) \rightarrow A$:	$\mathbf{PK_E}$	
Message 3	$A \rightarrow \mathbf{E}(B)$:	$\{K_{sess}\}_{\mathbf{PK_E}}$	event $learnt(\mathbf{K_{sess}})$
Message 4	$\mathbf{E}(B) \rightarrow A$:	$\{\mathbf{M'}\}_{\mathbf{K_{sess}}}$	

Indeed, secret key K_{sess} is learnt by the enemy when it is sent encrypted with enemy's public key. We can thus formulate secrecy in our usual style as follows:

> "P guarantees secrecy of m if whatever hostile environment is considered, the event $learnt(m)$ can never occur".

2.5 The example (partially) repaired

In this section we show how to repair the initial protocol in order to avoid the attacks reported above (even though other ones are possible). The attacks work since it is possible to fake Message 2 in order to provide a wrong public key, i.e., not the one associated to B. In actual implementations, the KDC sends the public key together with the name of the associated user, all signed with its own key. Hence, the resulting protocol is the following:

$$\begin{array}{lrl} \text{Message 1} & A \to KDC : & A, B \\ \text{Message 2} & KDC \to A \quad : & \{PK_B, B\}_{SK_{KDC}} \\ \text{Message 3} & A \to B \quad : & \{K_{sess}\}_{PK_B} \\ \text{Message 4} & B \to A \quad : & \{M\}_{K_{sess}} \end{array}$$

where Message 2 now consists of the pair (PK_B, B) encrypted with the private key of KDC (SK_{KDC}), in such a way that everyone can decrypt it and be sure that the associated public key PK_B has been originated by KDC. It is easy to see that the attacks previously shown are not possible anymore, since the enemy is not able to generate the block $\{PK_E, B\}_{SK_{KDC}}$.

This updated protocol may still be subject to some form of attacks, namely replay attacks. These attacks work since the enemy is able to re-use the information obtained in previous runs of the protocol in order to fake messages for A. Imagine the situation where the session key K_{sess} between A and B has been safely established and is used to send two different messages, say M, M', to A, by simply adding another step to the previous protocol:

$$\text{Message 5} \quad B \to A : \; \{M'\}_{K_{sess}}$$

The enemy could eavesdrop Message 4, intercept Message 5 and replay the Message 4 to A. Thus, A receives two times message M! For example, if M represents a bank transfer request this attack could result in a double transfer of money. This is a message authenticity attack and is revealed since we obtain two events $\mathit{received}(M)$ instead of $\mathit{received}(M)$, $\mathit{received}(M')$.

This kind of attacks may be prevented by inserting freshness in the messages (such as newly generated random numbers). It is not our intention to give here a complete spectrum of attacks on cryptographic protocols; we just show that these are very subtle and make the design of such protocols very challenging. (For some guidelines about cryptographic protocols design see [3].)

2.6 Non Repudiation and Fairness

There are other interesting properties that can be rephrased in our common style. For example, *non repudiation* and *fairness* (i.e., fair message exchange). The former is related to the possibility of considering a certain message as a signed contract, which is thus not repudiable by the sender. This is typically obtained through protocols which are based on some electronic signature mechanism. Non repudiation is of course very important for electronic commerce. The latter property requires that mutual information exchanges are performed in a fair way, i.e., no one of the parties involved in the exchange should get an advantage by obtaining the information before the other party is also able to obtain it. Indeed, if an exchange is not fair, the advantaged party could refuse to give its information once it has received the information from the other party. This property is sometimes required together with non repudiation when the parties want to simultaneously exchange two non repudiable messages (for example, one message could be the electronic payment and the other one could be the corresponding electronic receipt). Here we follow the treatment in [15, 25].

(Fair) Non repudiation protocols are often quite complex. Here, in order to illustrate these properties we consider the following very simple example (see also [31]):

$$\begin{array}{ll} \text{Message 1} & A \to B: \; M, \{M, B\}_{SK_A} \\ \text{Message 2} & B \to A: \; \{M, A\}_{SK_B} \end{array}$$

A sends a message M to B, signed with her secret key SK_A. This is a guarantee that only A could have produced such an encryption. This means that A cannot deny to have sent such a message to B (note that B is also contained into the signature). Then B sends back to A message M signed with its own secret key, in order to confirm the reception of M. After that, also B will not be able to deny that he has received M from A. We can imagine that M represents some form of electronic payment: A wants a proof that B has received from her the money (a receipt) and B wants a proof that A had indeed sent the payment M to him (for example, in case M is not a valid payment). These are non-repudiation properties and are guaranteed by the signatures (assuming that neither A nor B publicizes her/his secret key). For example, if A collects a certain evidence that B has sent a message (in our protocol the signature $\{M, A\}_{SK_B}$ represents such an evidence), then B should have indeed sent such a message. This is somehow similar to the entity authentication property we have discussed above. The main difference is that B can be malicious and may try to send a faked evidence.

Consider now the fairness property. Indeed, in the simple protocol we have presented, B has an evident advantage over A. He can just refuse to send the last message and A will never be able to prove that she has indeed sent the money to B. The exchange is clearly not fair. We can try to define non-repudiation with fairness as follows:

> "P guarantees non-repudiation with fairness to A on a message M if, whatever malicious B is considered, if B gets evidence that A has originated M than also A will eventually obtain the evidence that B has received M".

We observe two important points. First of all, in this definition one of the parties (B) takes the role of the hostile environment. This is intuitively correct, since we want to see if B is able to cheat A by keeping the payment without releasing the receipt. The second point is the most important: this property cannot be expressed as a safety property (i.e., nothing bad happens). As a matter of fact we are requiring that something good should happen if B gets his evidence, i.e., that also A should soon or later get her evidence. This leads us to relax our general scheme as follows:

"P guarantees non-repudiation with fairness to A on a message M if, whatever hostile environment is considered (with a certain initial knowledge ϕ_I), then P *satisfies* the specification $\alpha(P)$",

where B is the enemy (ϕ_I is then B's initial knowledge), the relation *satisfies* is a deadlock-sensitive process preorder (e.g., the testing preorder [7]) and $\alpha(P)$ is the process where every time B gets his evidence then also A gets her own evidence. The choice of a deadlock-sensitive preorder is justified by the fact that we want to check that no deadlock is possible after B evidence and this is what we want to require also in protocol P. In the example above, if B decides not to send the second message we clearly obtain such a deadlock and the property is not satisfied.

3 A general scheme for security properties

We have seen that several different properties (message authenticity, entity authentication, secrecy) can be (informally) written in a similar style which sounds like:

"P guarantees a security property S if, whatever hostile environment is considered, P never shows some particular *bad behaviour*".

In general, this set of bad behaviours 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 define what is a bad behaviour. It is sometimes easier to choose a complementary approach and describe which are the good behaviors (they just correspond to all the behaviour that are not bad). So, if we denote by $\alpha_S(P)$ the set of all possible good behaviour of P with respect to the security property S, then our general scheme becomes the following:

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

Indeed, cryptographic protocols typically rely on some secret values (keys or random numbers used for challenge-response). Moreover, if we want to analyze secrecy we certainly have to face the presence of some initially secret message. We can thus slightly refine our scheme as follows:

"P guarantees a security property S if, whatever hostile environment is considered with a certain initial knowledge ϕ_I, then P always shows behaviours in $\alpha_S(P)$".

Moreover, as discussed for fair non repudiation, it may be useful also to parameterize the previous notion w.r.t. the notion of behaviour, by considering *satisfaction* relations among processes, hence obtaining:

"P guarantees a security property S if, whatever hostile environment is considered with a certain initial knowledge ϕ_I, then P *satisfies* the specification $\alpha_S(P)$", The (informal) considerations above are at the base of the proposal, originally reported in [15], of a uniform formal framework where security properties can be defined. The proposed schema, called General NDC ($GNDC$ for short), is as follows: [4]

$$P \text{ is } GNDC^{\alpha}_{\approx} \text{ iff } \forall X \in \mathcal{E}_C^{\phi_I} : (P \,\|\, X) \setminus C \approx \alpha(P)$$

where $\approx$ is a behavioural preorder and α is a function from processes to processes. Now, we can just define a specific property by suitably instantiating the function $\alpha(P)$ and the preorder $\approx$. We reconsider the properties presented so far, showing informally their corresponding $\alpha(P)$ functions and $\approx$ relations:

- *Non-interference*: $\alpha_{NI}(P(M)) = P(M) \setminus C$. We obtain the exact definition of NDC if we use trace equivalence as $\approx$.
- *Message authenticity*: $\alpha_{MA}(P(M))$ is the process where $received(M)$ is the only event *received* which may occur. For a formal characterisation of this property in the $GNDC$ scheme for a large class of protocols, please see [13], where the $\approx$ relation is (a suitable) may testing or trace equivalence.
- *Entity authentication*: $\alpha_{EA}(P)$ is the process where $commit(A, B)$ is always preceded by $run(B, A)$. Please refer to [15] for other authentication properties based on the *correspondence* idea, such as the ones in the hierarchy of [21] or message authentication as proposed in [24]. The relation $\approx$ is in general trace inclusion.
- *Secrecy of* m: $\alpha_{Sec}(P(m))$ is the set of processes where the event $learnt(m)$ can never occur (for more details, see [14]). The relation $\approx$ is in general trace inclusion.
- *Non repudiation*: $\alpha_{nr}(P(M))$ is the process where whenever an evidence of a message M is obtained then that message has been effectively sent (see [15] for a deeper discussion). The relation $\approx$ is in general trace inclusion.
- *fairness*: $\alpha_{fair}(P(M))$ is the process such that if the event $B\_ev\_A\_or\_M$ (signaling that B has evidence that A originated M) then the event $A\_ev\_B\_rec\_M$ (signaling that A has evidence that B received M) will eventually happen (see [15] for a deeper discussion). The relation $\approx$ is in general a failure or testing preorder (see [7, 19]).

3.1 Security attacks as interferences

In the previous section we have shown how several security properties can be seen as instances of the following general1 scheme:

> "P guarantees a security property S if, whatever hostile environment is considered with a certain initial knowledge ϕ_I, then P *satisfies* the specification $\alpha_S(P)$",

where the relation *satisfies* and the function $\alpha_S(P)$ are property dependent parameters.

[4] Indeed $GNDC$ depends on the set ϕ_I, but we will omit it for the sake of readability.

It would be useful to find the most restrictive $\alpha_S(P)$ as it would induce the strongest property, up to the chosen notion of behaviour (i.e., the chosen relation *satisfies*). The idea is to use an $\alpha_S(P)$ which returns an *encapsulation* of protocol P, i.e., a version of P which is completely isolated from the environment. Intuitively, this secure *encapsulation* of P should correspond to the execution of P in a perfectly secure network where only the honest parties are present. In our process algebra setting, this corresponds to the *restriction* of all public channels where protocol messages are sent. This makes it impossible for an intruder to *interfere* on the protocol execution. Let us briefly explain this important point. Consider a preorder between processes $\leq$. Next, consider the induced equivalence $\approx$ as $\leq \cap \leq^{-1}$. Also suppose that for every process P we have

$$(P \,\|\, \underline{0}) \setminus C \approx P \setminus C$$

where $\underline{0}$ is the process that does nothing. This means that the process restricted on C is equivalent to the protocol in composition with the intruder that does nothing. The previous property holds for every security property we have studied. Please also note that, by definition, $\underline{0} \in \mathcal{E}_C^{\phi_I}$ for every ϕ_I. So it is very natural to consider α functions and processes P such that:

$$P \setminus C \leq \alpha(P)$$

This simply means that the protocol P is correct (as it satisfies its specification $\alpha(P)$) at least when it is not under attack. This condition can be somehow seen as a reasonable criterion for any *good* protocol: it must be correct at least when it is not under attack! Under this observation, it is clear that $P \in NDC$ implies $P \in GNDC^{\alpha}_{\leq}$.

If NDC holds then we can say that the hostile environment has no effect at all on P, since it still behaves as if it were isolated. It is important to note that what we observe of P behaviour are exactly the events we have discussed in the previous section. It is easy to see that, if we specify all the events corresponding to a certain set of properties, then NDC will imply all of them. [5]

When P does not guarantee NDC we say that an *interference* is possible, i.e., a behaviour that is caused by the hostile environment. It comes out that an attack to a security properties is revealed by NDC as an interference of the enemy on the protocol. This allows us to use NDC in order to detect different attacks all at once. In the next section we illustrate this issue with an example.

4 An Example

In this section we consider a larger protocol and we show how the analysis of (some of) the security properties presented above, can be carried out in a uniform way. The protocol is the Woo-Lam public key one, which has been proposed for

[5] Note that, the notion of *satisfies* must be chosen as the stronger one used by the security properties considered.

mutual entity authentication and key-exchange. The protocol, as reported in [27], is the following sequence of 7 messages:

$$
\begin{array}{lrcl}
\text{Message 1} & A \to KDC & : & A, B \\
\text{Message 2} & KDC \to A & : & \{PK_B\}_{SK_{KDC}} \\
\text{Message 3} & A \to B & : & \{A, N_A\}_{PK_B} \\
\text{Message 4} & B \to KDC & : & A, B, \{N_A\}_{PK_{KDC}} \\
\text{Message 5} & KDC \to B & : & \{PK_A\}_{SK_{KDC}}, \{\{N_A, K, A, B\}_{SK_{KDC}}\}_{PK_B} \\
\text{Message 6} & B \to A & : & \{\{N_A, K, A, B\}_{SK_{KDC}}, N_B\}_{PK_A} \\
\text{Message 7} & A \to B & : & \{N_B\}_K
\end{array}
$$

Alice sends to KDC a request of connection with Bob. Then KDC replies with a certified copy of B's public key. This copy is indeed signed with KDC's secret key, i.e., only KDC may have generated it. Alice checks the signature and sends to Bob a challenge N_A encrypted with Bob's public key. Bob forwards N_A to KDC encrypting it with KDC's public key and adding both his own identifier and the one of Alice. KDC is now ready to generate a certificate containing the challenge N_A, the fresh session key K and the two identifiers A and B. KDC sends this to Bob (encrypted with Bob's public key) together with a signed copy of Alice's public key. Bob checks the signature and forwards to Alice the certificate received from KDC together with a challenge N_B, all encrypted with Alice's public key. Finally, Alice sends back to Bob the challenge N_B encrypted with the new session key K.

This protocol looks quite complex. As a matter of fact it mixes the requests for public keys with entity authentication and key-exchange. In particular the first two messages and the first part of message 5 are for public keys distribution. The challenges N_A and N_B are used to guarantee mutual entity authentication. Finally, the certificate provides (authenticated) key-distribution. Indeed, this protocol contains two errors with respect to the correct version [29] that cause a number of attacks. We now analyze the protocol by applying the ideas developed in the previous section. First of all we point out in detail which are the various security properties that the protocol should guarantee:

- public key PK_A and PK_B should be authentic from KDC (this is why they are signed);
- it is also important that the session key K is authentic, i.e., no enemy should be able to force A and B using a faked session key;
- of course, the session key K should also remain secret;
- also the challenges should remain secret (since they are always sent encrypted) but this is not crucial since they are used only for guaranteeing entity authentication; in protocols where the nonces are also used for generating a new session key secrecy requirement becomes crucial;
- finally, the protocol should guarantee mutual entity authentication between A and B.

We now show all the events that are used to model such properties as suitable annotation to the protocol, where the secrecy event $learnt(K)$ is not reported,

1	$A \to KDC$	:	A, B	$run(A,B)$
2	$KDC \to A$	:	$\{PK_B\}_{SK_{KDC}}$	$received(PK_B)$
3	$A \to B$	:	$\{A, N_A\}_{PK_B}$	$run(B,A)$
4	$B \to KDC$	:	$A, B, \{N_A\}_{PK_{KDC}}$	
5	$KDC \to \mathbf{E}(B)$	:	$\{PK_A\}_{SK_{KDC}}$, $\{\{N_A, K, A, B\}_{SK_{KDC}}\}_{PK_B}$	
$5'$	$\mathbf{E}(KDC) \to B$	:	$\{\mathbf{PK_E}\}_{\mathbf{SK}_{KDC}}$, $\{\{N_A, K, A, B\}_{SK_{KDC}}\}_{PK_B}$	$received(\mathbf{PK_E})$
6	$B \to \mathbf{E}(A)$	:	$\{\{N_A, K, A, B\}_{SK_{KDC}}, N_B\}_{\mathbf{PK_E}}$	$\mathbf{learnt(K)}$
$\mathbf{6'}$	$\mathbf{E}(B) \to A$	:	$\{\{N_A, K, A, B\}_{SK_{KDC}}, N_B\}_{\mathbf{PK_A}}$	$received(K)$ $commit(A,B)$
7	$A \to B$	:	$\{N_B\}_K$	$commit(B,A)$

Table 1. An attack to secrecy and message authenticity

as it is performed by the omitted enemy. This allows us check all the properties in just one step following our NDC approach.

1	$A \to KDC$	:	A, B	$run(A,B)$
2	$KDC \to A$	:	$\{PK_B\}_{SK_{KDC}}$	$received(PK_B)$
3	$A \to B$	:	$\{A, N_A\}_{PK_B}$	$run(B,A)$
4	$B \to KDC$	:	$A, B, \{N_A\}_{PK_{KDC}}$	
5	$KDC \to B$	:	$\{PK_A\}_{SK_{KDC}}$, $\{\{N_A, K, A, B\}_{SK_{KDC}}\}_{PK_B}$	$received(PK_A)$
6	$B \to A$	:	$\{\{N_A, K, A, B\}_{SK_{KDC}}, N_B\}_{PK_A}$	$received(K)$ $commit(A,B)$
7	$A \to B$	:	$\{N_B\}_K$	$commit(B,A)$

The security properties that this protocol should guarantee have in common that trace inclusion is their suitable $\approx$ relation. Hence, an interference is a trace that is composed only by the security events and that is possible when the enemy is active, but not when the protocol runs in isolation. Consider now the attack sequence in Table 1, where the interference trace is reported in the right column. In this attack E exploits (one of) the mistakes in the specification of the protocol as reported in [27]. In particular, it is necessary to include in the certificate of the public key also the identifier of the corresponding owner. Hence, in messages 2 and 5 the certificate should be $\{PK_B, B\}_{SK_{KDC}}$ and $\{PK_A, A\}_{SK_{KDC}}$, respectively. The enemy can thus substitute in message 5 the signed public key of A with its own signed key. Note that the enemy can obtain a signed copy of its own key by just running honestly the protocol with another user. After that, the sixth message will be encrypted by Bob with the public key of the enemy thus allowing the interception of the certificate and, consequently, of the session key K. By observing the events we can identify two attacks:

1	$\mathbf{E}(A) \to KDC$	: A, B	
2	$KDC \to \mathbf{E}(A)$	: $\{PK_B\}_{SK_{KDC}}$	
3	$\mathbf{E}(A) \to B$	: $\{A, N_A\}_{PK_B}$	$run(B, A)$
4	$B \to KDC$	: $A, B, \{N_A\}_{PK_{KDC}}$	
5	$KDC \to \mathbf{E}(B)$	: $\{PK_A\}_{SK_{KDC}}$, $\{\{N_A, K, A, B\}_{SK_{KDC}}\}_{PK_B}$	
$5'$	$\mathbf{E}(KDC) \to B$	: $\{\mathbf{PK_E}\}_{\mathbf{SK}_{KDC}}$, $\{\{N_A, K, A, B\}_{SK_{KDC}}\}_{PK_B}$	$received(\mathbf{PK_E})$
6	$B \to \mathbf{E}(A)$	: $\{\{N_A, K, A, B\}_{SK_{KDC}}, N_B\}_{\mathbf{PK_E}}$	$\mathbf{learnt(K)}$
7	$\mathbf{E}(A) \to B$	: $\{N_B\}_K$	$commit(B, A)$

Table 2. An attack to secrecy, authenticity and entity authentication

1. the protocol does not guarantee the authenticity of Alice public key in message 5; we observe this through event $received(PK_E)$; as we stated above, this is caused by the absence of Alice identifier inside the signature;
2. the protocol does not guarantee the secrecy of K; this is revealed by event $learnt(K)$.

Note also that we do not observe any entity authentication attack. Alice and Bob are convinced to communicate one another at the end of the protocol. Thus, the secrecy attack over K becomes even more dangerous, as the enemy can easily eavesdrop every future communication encrypted with K between Alice and Bob and they have no way of discovering this.

Indeed it is easy to show an attack similar to the previous one in order to obtain also an entity authentication failure, e.g. see Table 2.

The idea is that the role plaied by A in the previous attack could be fully simulated by the enemy as done here. Since we have $commit(B, A)$ with no $run(A, B)$ the entity authentication attack is indeed revealed. Hence, the protocol does not guarantee the authentication of A with respect to B, i.e., the enemy is indeed able to impersonate A. Note that the attack on message authenticity and secrecy are still valid.

Other complex attacks are possible, involving two or three parallel sessions of the protocol. All of them are based on the fact that the certificate does not include the corresponding identifier. As we already stated, this is an error in the version reported in [27]. In the original protocol [28] the certificates are correct. However in such a version there is another error, causing different failures, that has been corrected two months later by the same authors in [29].

5 Conclusion

Our non interference-based approach to the analysis of protocols has been mechanized [8, 9], resulting in a tool that checks NDC on finite state representations of the honest participants and the (most general) enemy. With the help of this tool, we have been able to show failures upon two unflawed (to the best of our knowledge) protocols: Woo & Lam public key one-way authentication protocol and ISO public key two-pass parallel authentication protocol; and new failures upon three flawed protocols: Encrypted Key Exchange, Station to Station, Woo & Lam symmetric key one-way authentication protocol (the last one reported in [8, 9]). Many other protocols have been analyzed, most of those reported in the cryptographic protocol library [6], and we have been able to capture the attacks reported there. We are now improving the efficiency of our tool in order to be able to analyze larger, commercial protocols for e-commerce, such as SET [22].

Future extensions of the approach include the modeling of cryptographic protocols with more concrete information, e.g., time and probability, that can be helpful in order to discover time/probability dependent attacks that cannot be revealed in a purely nondeterministic setting. Some initial work in this direction is [12], where the NDC idea has been extended in the context of a discrete time process algebra, and applied to prevent timing covert channels in multilevel computer systems.

Our NDC-based approach has been developed for a CCS-like calculus that is powerful enough to model most cryptographic protocols. However, recursive protocols as well as protocols for mobile systems (where channel names are passed as values in a communication) cannot be easily modeled. Hence, we are planning to study the extension of our approach to richer calculi, such as the spi-calculus [2].

References

[1] M. Abadi. Security protocols and specifications. In *Proc. Foundations of Software Science and Computation Structures*, volume 1578 of *LNCS*, pages 1–13, 1999.

[2] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. *Information and Computation*, 148(1):1–70, 1999.

[3] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, January 1996.

[4] D. E. Bell and L. J. La Padula. Secure computer systems: Unified exposition and multics interpretation. *ESD-TR-75-306, MITRE MTR-2997*, March 1976.

[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] J. Clark and J. Jacob. "A Survay of Authentication Protocols Literature: version 1.0". November 1997.
http://www.cs.york.ac.uk/~jac/papers/drareview.ps.gz.

[7] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–133, 1984.

[8] A. Durante, R. Focardi, and R. Gorrieri. A compiler for analysing cryptographic protocols using non-interference. Submitted for publication.

[9] 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.

[10] R. Focardi and R. Gorrieri. A classification of security properties for process algebras. *Journal of Computer Security*, 3(1):5–33, 1994/1995.

[11] 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), September 1997.

[12] R. Focardi, R. Gorrieri, and F. Martinelli. Information flow analysis in a discrete-time process algebra. In *Proceedings of CSFW'00*, 2000. IEEE Press. To appear.

[13] R. Focardi, R. Gorrieri, and F. Martinelli. Message authentication through non-interference. In *Proc. of 8th International Conference in Algebraic Methodology and Software Technology* (AMAST), 2000. To appear.

[14] R. Focardi, R. Gorrieri, and F. Martinelli. Secrecy in security protocols as non-interference. In *Workshop on secure architectures and information flow*, volume 32 of *ENTCS*, 2000.

[15] 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, LNCS 1708, 1999.

[16] J. A. Goguen and J. Meseguer. Security policy and security models. In *Proc. of the 1982 Symposium on Security and Privacy*, pages 11–20. IEEE Press, 1982.

[17] 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.

[18] 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.

[19] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[20] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proceedings of TACAS'96*, pages 146–166. LNCS 1055, 1996.

[21] G. Lowe. A hierarchy of authentication specification. In *Proceedings of the 10th Computer Security Foundation Workshop*, pages 31–43. IEEE press, 1997.

[22] MasterCard. Secure electronic payment protocol, November 1995. Draft Version 1.2.

[23] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[24] S. Schneider. Verifying authentication protocols with csp. In *Proceedings of the 10th Computer Security Foundation Workshop*, pages 3–17. IEEE press, 1997.

[25] S. Schneider. Formal analysis of a non-repudiation protocol. In *Proceedings of CSFW'98*, pages 54–65. IEEE Press, 1998.

[26] S. Schneider. Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9), September 1998.

[27] F.B. Schneier. *Applied Cryptography*. Wiley, 1996.

[28] T. Y. C. Woo and S. S. Lam. Authentication for distributed systems. *IEEE Computer*, 25(1):39–51, 1992.

[29] T. Y. C. Woo and S. S. Lam. Authentication revised. *IEEE Computer*, 25(3):10, 1992.

[30] 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.

[31] J. Zhou and D. Gollman. A fair non-repudiation protocol. In *Proc. of Symposium in Research in Security and Privacy*, pages 55–61. IEEE Press, 1996.