Automatic Verification of Cryptographic Protocols through Compositional Analysis Techniques^{*}

Davide Marchignoli¹ and Fabio Martinelli²

¹ Dipartimento di Informatica, Università di Pisa

² Dipartimento di Matematica, Università di Siena

Abstract. We present a tool for checking automatically the correctness of cryptographic protocols with finite behaviour. The underlying theory has been proposed in [13] and borrows some compositional analysis concepts for process algebras (see [3,8]). Here we extend the theory by showing an interesting relation among security properties.

1 Introduction

The great amount of security-sensitive information which flows in computer networks has stimulated a lot of research in formal methods for the definition and the analysis of security properties of communicating systems. A typical example of security property is that only legitimate users can access some kind of information, or a particular service, or also that parties in a communication get assurance about the identity of their correspondents.

Authentication protocols are the instruments used to design systems ensuring such properties; in turn they are based on cryptographic systems. The aim of cryptographic systems is that of permitting the exchange of messages through insecure media guaranteeing that only users who know a certain piece of information (*key*), can retrieve the actual content of the messages.

Unfortunately, cryptography constitutes only a building block in design of secure protocols and it is not sufficient by itself, as proved by many flaws found in authentication protocols (see [2,6,11,18]); moreover authentication protocols, even those involving only few communications between parties, are recognized to be prone to errors.

In the last years, several techniques to analyze communication protocols w.r.t. security properties have been developed (see [4,6,10,11,12,18]). Some of them are essentially based on the analysis of finite state systems, and typically can ensure error freedom only for a finite part of the behaviour of systems.

Another approach for the analysis of cryptographic protocols is based on proof techniques for authentication logics (see [2,7,17]) or for process algebras

^{*} Work partially supported by CNR, Progetto Strategico “Modelli e Metodi per la Matematica e l’Ingegneria and by MURST Progetto “Tecniche Formali per la Specifica, l’Analisi, la Verifica e la Trasformazione dei Sistemi Software”.

(see [1]). In general, these methods are not fully automated and need non trivial human efforts to analyze systems. An interesting exception is the work of Kindred and Wing ([7]) in which, an original fully automated approach for checking that a protocol enjoys some properties expressed in a logical language $\mathcal{L}$ is introduced.

The methodology proposed in this paper is based on a transposition of compositional analysis techniques for process algebras (also known as partial model checking or partial evaluation in [3,8]). This approach is novel in area of security and has been proposed by the second author as a unifying theoretical framework for the analysis of several security properties (see [13,14]). The tool proposed here shows as these ideas can be fruitfully employed in practice.

Many researchers specify security properties of a system in terms of their behaviour w.r.t. any environment in which they can operate (see [1,5,9,14]). We believe that this is a very natural approach and that partial evaluation techniques can be a valid tool for the analysis of this kind of properties. The intuitive idea of partial evaluation is the following: verifying that a system S , sharing the execution environment with a generic process X , enjoys a property expressed by a logical formula F , is equivalent to verify that X itself satisfies a particular formula $F//S$, computed according to the evaluation of S , in such a way to guarantee:

$$S \parallel X \models F \text{ iff } X \models F//S.$$

In our framework the process X can be regarded as an intruder that tries to discover some information; the formula F is used to state the requirement that in no way X can obtain such information. Our strategy will be:

- design suitable languages for protocol description and property specification;
- develop partial evaluation techniques (almost automatically);
- develop satisfiability procedure for the logic.

The latter is a key point for understanding our proposal. There is a related verification problem for open reactive systems recently defined by Kupferman and Vardi, namely *module checking*. The problem is to verify that every behaviour of a system, induced by the interaction with an arbitrary environment, satisfies a temporal logic formula. It is interesting to note that compositional analysis techniques can be used to tackle this problem too (see [15] for a deeper discussion).

2 An operational language for the description of protocols

2.1 Types and typed messages

We assume given a set of *basic type symbols* $\mathsf{T}^1, \dots, \mathsf{T}^n$, a set of *type constructor symbols* $F_1, \dots, F_m$ with arity function $ar : \{1, \dots, m\} \rightarrow \mathbb{N}$ and inductively define the set of *types* as:

$$T ::= \mathsf{T}^i \mid F_j(T_1, \dots, T_{ar(j)}) \quad i \in \{1, \dots, n\}, j \in \{1, \dots, m\}$$

$$\begin{array}{c}
\frac{m : T_1 \quad m' : T_2}{(m, m') : T_1 \times T_2} \quad (1) \qquad \frac{(m, m') : T_1 \times T_2}{m : T_1} \quad (2) \qquad \frac{(m, m') : T_1 \times T_2}{m' : T_2} \quad (3) \\
\frac{m : T_1 \quad k : Key}{E(k, m) : E(Key, T_1)} \quad (4) \qquad \frac{E(k, m) : E(Key, T) \quad k^{-1} : Key^{-1}}{m : T} \quad (5)
\end{array}$$

Fig. 1. Example of Inference System

We assume that any *basic type* T^i is “populated” by a finite set of *basic messages* BT_i and a countable set of *random messages* RT_i s.t. $BT_i \cap RT_i = \emptyset$. We will write $m : T$ for m is a message of type T . Messages of type $F_i(T_1, \dots, T_{ar(i)})$ are defined inductively as the minimum set satisfying:

$$m_1 : T_1, \dots, m_{ar(j)} : T_{ar(j)} \implies F_j(m_1, \dots, m_{ar(j)}) : F_j(T_1, \dots, T_{ar(j)})$$

in general we will allow messages to contain variables for other messages. Here we fix some notation:

- $SubM(m : T)$ denote the set of submessages of $m : T$;
- a message $m : T$ is *pure* if no variable occurs in $SubM(m : T)$;
- $Msgs(T)$ is the set of pure messages of type T ;
- a pure message $m : T$ is *initial* if $\forall T^i \text{ } SubM(m : T) \cap Msgs(T^i) \subseteq BT_i$.

An *inference system* IS is a set of *inference schemata*

$$\frac{m_1 : T_1 \quad \dots \quad m_n : T_n}{m : T}$$

in which $m_1 : T_1, \dots, m_n : T_n$ is a (possibly empty) set of *premises* and $m : T$ is the *conclusion*. A *proof* for a typed message $m : T$ is a finite tree, rooted in $m : T$, whose nodes are built from their descendants by applying a rule schema. We say that $m : T$ is *deducible* from a set of messages ϕ (and write $\phi \vdash m : t$) if there exists a proof of $m : T$ whose leaves have premises contained in ϕ . Each inference system induces a *deduction function* $\mathcal{D}(\phi) = \{m : T \mid \phi \vdash m : T\}$.

Here we present a formalization of a deduction system similar to those used by many authors (see [10,12]). Among the basic types we have *Key* for encryption keys; the type constructors are *Pair* for pair formation, *E* for encryption and $\cdot^{-1}$ for key inverse; messages $Pair(m_1, m_2) : Pair(T_1, T_2)$ will be more succinctly written as $(m_1, m_2) : T_1 \times T_2$. The inference system is presented in figure 1. The interesting rules are 4 and 5; the former permits the encryption of messages by using a key, and the latter permits to deduce the clear message from the encrypted message and the decryption key.

2.2 Syntax and semantics

We briefly introduce the syntax of systems. A system (term) is generated by the grammar in figure 2, where $m : T, m' : T'$ are typed messages, $\langle \langle m_i : T_i \rangle \rangle_{i \in I}$ is a sequence of typed messages, C is a finite set of channels with $c \in C$, x is a message variable, ϕ is a finite set of pure typed messages, L is a subset of C and

$S ::= S \setminus L \mid S_1 \parallel S_2 \mid (A)_\phi$	<i>Composed Systems</i>
$A ::= Nil \mid a.A \mid A_1 + A_2 \mid [m : T = m' : T']A_1; A_2$	<i>Sequential Agents</i>
$\mid [\langle m_i : T_i \rangle]_{i \in I} \vdash_{IS} x : T]A_1; A_2$	
$a ::= c!m : T \mid c?(x) : T \mid \tau \mid \tau_{c,m:T} \mid \chi_{c,x:T} \mid gen_T^{x,(i,j)}$	<i>Actions</i>

Fig. 2. Syntax of systems

$i, j \in \mathbb{N}$ (the set of natural numbers). The set of channels occurring in a term A will be written as $Sort(A)$.

The inference construct $[\langle m_i : T_i \rangle]_{i \in I} \vdash_{IS} x : T]A_1; A_2$ acts as a binder for the variable x in A_1 , and the prefix constructs $c?(x) : T.A$, $\chi_{c,x:T}.A$, $gen_T^{x,(i,j)}.A$ act as binders for the variable x in A .

Free and bound variables are defined in the usual way and consequently also open and closed agents (systems). Hereafter we will consider only closed agents in which every bound variable has a different name. The set Act of actions, which can be performed by a compound system, is defined as:

$$Act = \{\tau, \tau_{g:T^i}, c?m : T, c!m : T, \tau_{c,m:T}, \chi_{c,m:T} \mid c \in C, m : T, g \in RT^i\}$$

The projections *channel* that given an action returns a channel and *msgs* that given a sequence of actions returns its messages can be straightforwardly defined.

The language described here is essentially CCS with values (see [16]), a different syntax is used for simplicity and some new construct is added to deal with message formation and deduction.

As in CCS the operators Nil , $a.A$, $A_1 + A_2$, $S \setminus L$, $S_1 \parallel S_2$ denote respectively: the process that can do nothing, the sequential composition of action a to process A , the choice among agent A_1 and agent A_2 , the restriction of process S to actions not in L and the parallel composition of systems S_1 and S_2 . Again as in CCS we have the internal action (denoted by τ) and the communication actions send and receive of message $m : T$ over channel c (denoted respectively by $c!m : T$ and $c?m : T$).

In order to model insecure channels, we also consider the action $\tau_{c,m:T}$ arising from the communication of a message $m : T$ on an insecure channel c . The message can be listened by a process performing the *eavesdrop* action $\chi_{c,x:T}$ over channel c . In order to keep the message handling separated from the communication aspects of the language, some new operator is introduced. The *matching* operator $[m : T = m' : T']A_1; A_2$ permits to check the equality between typed messages $m : T$ and $m' : T'$ executing the residuals A_1 or A_2 accordingly. The *deduction* operator $[\langle m_i : T_i \rangle]_{i \in I} \vdash_{IS} x : T]A_1; A_2$ permits to deduce a new message x applying an inference schema IS to the set of messages $\langle m_i : T_i \rangle_{i \in I}$; the residual A_1 is chosen if IS is applicable else the residual A_2 is chosen. By using this construct a finite number of times, an agent can build the proof of every message in $\mathcal{D}(\phi)$. Typically it may be used to decrypt messages by applying a rule such as 5 in figure 1. Also we need to record the *knowledge* of an agent, i.e.

$$\begin{array}{c}
(?) \frac{m : T \in Msgs(T)}{(c?(x) : T.A)_\phi \xrightarrow{c?m:T} (A[m/x])_{\phi \cup \{m:T\}}} \\
(gen) \frac{g = \mathcal{R}^\top(i, j)}{(gen_T^{x,(i,j)}.A)_\phi \xrightarrow{\tau g:T} (A[g/x])_{\phi \cup \{g:T\}}} \\
(\chi) \frac{m : T \in Msgs(T)}{(\chi_{c,x:T}.A)_\phi \xrightarrow{\chi_{c,m:T}} (A[m/x])_{\phi \cup \{m:T\}}} \\
(\parallel_1) \frac{m : T \neq m' : T' \quad (A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}}{([m : T = m' : T']A_1; A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}} \quad (!) \frac{m : T \in D(\phi)}{(c!m : T.A)_\phi \xrightarrow{c!m:T} (A)_\phi} \\
(\parallel_2) \frac{m : T = m' : T' \quad (A_1)_\phi \xrightarrow{a} (A'_1)_{\phi'}}{([m : T = m' : T']A_1; A_2)_\phi \xrightarrow{a} (A'_1)_{\phi'}} \quad (+_1) \frac{(A_1)_\phi \xrightarrow{a} (A'_1)_{\phi'}}{(A_1 + A_2)_\phi \xrightarrow{a} (A'_1)_{\phi'}} \\
(\mathcal{D}_1) \frac{\langle \langle m_i : T_i \rangle \rangle_{i \in I} \vdash_{IS} m : T \quad (A_1[m/x])_{\phi \cup \{m:T\}} \xrightarrow{a} (A'_1)_{\phi'}}{([\langle \langle m_i : T_i \rangle \rangle_{i \in I} \vdash_{IS} x : T]A_1; A_2)_\phi \xrightarrow{a} (A'_1)_{\phi'}} \\
(\mathcal{D}_2) \frac{\bar{A}(m : T) \langle \langle m_i : T_i \rangle \rangle_{i \in I} \vdash_{IS} m : T \quad (A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}}{([\langle \langle m_i : T_i \rangle \rangle_{i \in I} \vdash_{IS} x : T]A_1; A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}} \\
(\setminus L) \frac{S \xrightarrow{\mu} S' \quad \mu \in Act - Msgs_L}{S \setminus L \xrightarrow{\mu} S' \setminus L} \quad (\chi_1) \frac{S \xrightarrow{\tau_{c,m:T}} S' \quad S_1 \xrightarrow{\chi_{c,m:T}} S'_1}{S \parallel S_1 \xrightarrow{\tau_{c,m:T}} S' \parallel S'_1} \\
(\parallel_1) \frac{S \xrightarrow{a} S'}{S \parallel S_1 \xrightarrow{a} S' \parallel S_1} \quad (\parallel_2) \frac{S \xrightarrow{c?m:T} S' \quad S_1 \xrightarrow{c!m:T} S'_1}{S \parallel S_1 \xrightarrow{\tau_{c,m:T}} S' \parallel S'_1}
\end{array}$$

Fig. 3. Operational semantics.

the set of messages that an agent can use to deduce new messages; this is accomplished using the notation $(A)_\phi$ for the agent A with set of messages ϕ . Let us make some assumptions on the capability of sequential agents to guess random values. Random generated messages (nonces) are used to witness the freshness of messages during executions of the protocol (runs). To model the characteristics of these messages, we assume that for every basic type T^i there is a subset of messages of this kind, i.e. RT^i . A particular operation $gen_T^{x,(i,j)}$ permits to *guess* a random value of a basic type T . Random messages of composed types can be built by using basic random values as subcomponents. Since it should be quite unlikely to generate twice the same random message, we assume that any $gen_T^{x,(i,j)}$ always instantiate a different value. This is achieved using, for each basic type T^k , an injective function $\mathcal{R}^{\mathsf{T}^k} : \mathbb{N} \times \mathbb{N} \mapsto \mathsf{RT}^k$ associating to the pair (i, j) the i^{th} value *guessed* by the j^{th} process of the system. Moreover we assume that systems in their initial configuration contain only *initial* messages.

The formal behaviour of a compound term is described by means of a *Labelled Transition Systems* $\langle S, Act, \{\xrightarrow{a}_{a \in A}\} \rangle$, where:

- S is the set of states (compound terms);
- Act is the set of actions defined above;
- $\xrightarrow{a}_{a \in A}$ is a set of *transition relations* on S , defined as the minimum set closed under the rules in figure 3.

For sake of conciseness we have omitted the symmetrical rules $+_2, \parallel_2, \chi_2$ and we have used $M\text{sgs}_L$ for $\{c!m:T, c?m:T, \chi_{c,m:T} \mid c \in L, m:T \in M\text{sgs}(T)\}$.

Given a finite sequence of actions $\gamma = \gamma_1, \dots, \gamma_n$ we will write $S \xrightarrow{\gamma} S'$ instead of $S = S_0 \xrightarrow{\gamma_1} \dots \xrightarrow{\gamma_n} S_n = S'$. Given a sequence of transitions $S \xrightarrow{\gamma} S'$ and an agent X of S , we use $(S \xrightarrow{\gamma} S') \downarrow_X$ to denote the subsequence of actions performed by X .

3 A logical language for the description of protocol properties

We present a logical language ($\mathcal{L}_K$) for the specification of the functional and security properties of a compound system. We need to extend a normal multi-modal logic with operators dealing with the knowledge of agents. More precisely we need operators to describe if a given agent can deduce a particular message in a given execution γ .

The syntax of the logical language $\mathcal{L}_K$ is defined by the following grammar:

$$F ::= \mathbf{T} \mid \mathbf{F} \mid \langle a \rangle F \mid [a]F \mid \wedge_{i \in I} F_i \mid \vee_{i \in I} F_i \mid m : T \in K_{X,\gamma}^\phi \mid \exists \gamma : (m : T) \in K_{X,\gamma}^\phi$$

where $m : T$ is a pure typed message, X is an agent identifier, I is an index set, ϕ is a finite set of pure typed messages and γ is a sequence of actions (ϵ is the empty sequence). Informally, the $\langle a \rangle F$ modality expresses the *possibility* to perform an action a and then satisfy F . The $[a]F$ modality expresses the *necessity* that after performing an action a the system satisfies F . A system S satisfies a formula $m : T \in K_{X,\gamma}^\phi$ if S can perform a sequence γ of actions and an agent X of S , with knowledge ϕ , can deduce $m : T$ using ϕ plus the messages he has acquired in performing the sequence γ . This formula plays a central role in the analysis of authentication protocols, since these are often based on the sharing, between two parties, of a secret (a particular message that is assumed no one else knows). Hence, pieces of information are used to witness the identity of agents and the eventual disclosure of particular information can have dangerous consequences. For instance, the existence of a sequence γ such that an agent X_ϕ can deduce $m : T$ (i.e. the secret) can be expressed formally using the formula $\exists \gamma : (m : T) \in K_{X,\gamma}^\phi$. The language without $m : T \in K_{X,\gamma}^\phi$ and $\exists \gamma : (m : T) \in K_{X,\gamma}^\phi$ is called $\mathcal{L}$.

3.1 Semantics

We assume given a deduction function $\mathcal{D}$ which enjoys the following assumptions (e.g. the one defined in figure 1):

- 1 For every type T the set of messages in $\mathcal{D}(\phi) \cap M\text{sgs}(T)$ is finite and constructible¹, when ϕ is a finite set. We need this assumption since we want to be able to perform an automatic analysis.

¹ Here, it means that we have an effective procedure that returns an explicit enumeration of $\mathcal{D}(\phi) \cap M\text{sgs}(T)$, i.e. its canonical index.

- 2 If IS is an inference schema and δ a bijection between random values then:
 $m_1 : T_1 \dots m_n : T_n \vdash_{IS} m : T$ iff $\delta(m_1 : T_1) \dots \delta(m_n : T_n) \vdash_{IS} \delta(m : T)$. The idea under this assumption is to avoid deduction systems that are not general and depend on particular random values.
- 3 If $m : T$ is a typed message and $m : T \in \mathcal{D}(\phi)$, then every submessage of $m : T$ of a basic type must be a submessage of some message in ϕ . We want that messages of basic type cannot be forged.

We define the semantics of a formula $F \in \mathcal{L}_K$ w.r.t. an LTS associated with a composed system S as follows:

$$\begin{array}{ll}
\text{For every } S \text{ we have } S \models \mathbf{T}, & \text{for no } S \text{ we have } S \models \mathbf{F} \\
S \models \wedge_{i \in I} F_i & \text{iff } \forall i \in I : S \models F_i \\
S \models \vee_{i \in I} F_i & \text{iff } \exists i \in I : S \models F_i \\
S \models \langle a \rangle F & \text{iff } \exists S' : S \xrightarrow{a} S' \text{ and } S' \models F \\
S \models [a] F & \text{iff } \forall S' : S \xrightarrow{a} S' . S' \models F \\
S \models m : T \in K_{x,\gamma}^\phi & \text{iff } \exists S' : (S \xrightarrow{\gamma} S') \downarrow_X = \tilde{\gamma} \text{ and} \\
& m : T \in \mathcal{D}(\phi \cup \text{msgs}(\tilde{\gamma})) \\
S \models \exists \gamma : (m : T) \in K_{x,\gamma}^\phi & \text{iff } \exists \gamma : S \models m : T \in K_{x,\gamma}^\phi
\end{array}$$

4 Partial evaluation techniques

To compute the partial evaluation it is convenient to assume a particular behaviour of agents with regard to the generation of random values. In particular, we want that, if an agent performs a sequence of actions whose first action is the guessing a random value $g : T$, then this value will be eventually sent as submessage of some message $m' : T'$ during the sequence; moreover we require that between these two events only guessing actions are performed. These agents are called *well behaved* (see [13] for a formal definition). As notation we use $S \parallel_L X$ for $(S \parallel X) \setminus L$ and we consider $S = \text{Nil}$ when $\forall a \in \text{Act } S \not\xrightarrow{a}$. Since we are interested in the analysis of formulas like $\exists \gamma : m \in K_{X,\gamma}^\phi$, we can restrict ourselves to consider only this particular kind of sequential agents. In fact we can prove that if there exists a sequential agent X_ϕ s.t. $S \parallel_L X_\phi \models \exists \gamma : (m : T) \in K_{X,\gamma}^\phi$ then a well behaved agent X'_ϕ exists s.t. $S \parallel_L X'_\phi \models \exists \gamma : (m : T) \in K_{X',\gamma}^\phi$. In figure 4 we give the partial evaluation function for $\parallel_L$ and $\exists \gamma : (m : T) \in K_{X,\gamma}^\phi$, where² and:

$$\begin{aligned}
\text{succ}(S) &= \{(c, m' : T, S') \mid S \xrightarrow{c?m':T} S' \text{ and } m' : T \in \mathcal{D}(\phi)\}, \\
\text{Rsucc}(S) &= \{(c, m' : T, \langle g_1 : T_1, \dots, g_n : T_n \rangle, S') \mid X_\phi \xrightarrow{\tau_{g_1:T_1}, \dots, \tau_{g_n:T_n}} X'_\phi \wedge \\
&\quad S \xrightarrow{c?m':T} S' \wedge m' : T \in \mathcal{D}(\phi') \wedge \{g_i : T_i\}_{i \in I} \subseteq \text{SubM}(m' : T)\}.
\end{aligned}$$

The set $\text{succ}(S)$ represents the sending actions (and relative successors of S) that can be performed by the intruder. The set $\text{Rsucc}(S)$ represents the sequences

² For sake of simplicity, we avoid the problem of considering always a different index of the *gen* actions in the translated formulas (this problem can be easily solved by global counters).

of guessing of random values followed by a sending of a message that can be performed by the intruder. By observing the compositional analysis proposed in [3,8] it can be noted that it is somewhat semantic driven. Analogously, our partial evaluation can be derived by inspection of the operational semantics of the language. Between brackets we have put in evidence the corresponding intruder behaviour. It is worthwhile noticing that $\text{succ}(S)$ is a finite set (by assumption 1).

$$\begin{aligned}
& \exists \gamma : (m : T) \in K_{X,\gamma}^\phi // S \doteq \\
& \bigvee_{(c, m' : T', S') \in \text{succ}(S)} \langle c!m' : T' \rangle (\exists \gamma : (m : T) \in K_{X,\gamma}^\phi // S') \quad (\text{sending}) \vee \\
& \bigvee_{(c, m' : T', \langle \langle g_i : T_i \rangle \rangle_{i \in I}, S') \in \text{Rsucc}(S)} \langle \langle \tau_{g_i : T_i} \rangle \rangle_{i \in I} \langle c!m' : T' \rangle \\
& \quad (\exists \gamma : (m : T) \in K_{X,\gamma}^{\phi \cup \langle \langle g_i : T_i \rangle \rangle_{i \in I}} // S') \quad (\text{guessing}) \vee \\
& \bigvee_{S \xrightarrow{c!m' : T'} S'} \langle c?m' : T' \rangle (\exists \gamma : (m : T) \in K_{X,\gamma}^{\phi \cup \{m' : T'\}} // S') \quad (\text{receiving}) \vee \\
& \bigvee_{S \xrightarrow{\tau_{c, m' : T'}} S'} \langle \chi_{c, m' : T'} \rangle (\exists \gamma : (m : T) \in K_{X,\gamma}^{\phi \cup \{m' : T'\}} // S') \quad (\text{eaves - drop.}) \vee \\
& \bigvee_{S \xrightarrow{a} S'} \exists \gamma : (m : T) \in K_\gamma^\phi // S' \quad (\text{idling}) \vee \\
& m : T \in K_{X,\epsilon}^\phi // S \quad (\text{trivial}) \\
& m : T \in K_{X,\epsilon}^\phi // S = \exists \gamma : (m : T) \in K_{X,\gamma}^\phi // \text{Nil} = \begin{cases} \mathbf{T} & m : T \in \mathcal{D}(\phi) \\ \mathbf{F} & m : T \notin \mathcal{D}(\phi) \end{cases}
\end{aligned}$$

Fig. 4. Partial evaluation function for $\|_L$ and $\exists \gamma : (m : T) \in K_{X,\gamma}^\phi$.

The next proposition states the correctness of the partial evaluation, where we assume that X_ϕ is a well behaved agent.

Proposition 1. *Given a system S , with $\text{Sort}(S) \cup \text{Sort}(X) \subseteq L$, a finite set of typed messages ϕ and an initial message $m : T$ then:*

$$S\|_L X_\phi \models \exists \gamma : (m : T) \in K_{X,\gamma}^\phi \text{ iff } X_\phi \models \exists \gamma : (m : T) \in K_{X,\gamma}^\phi // S.$$

Unfortunately, the formula $F = (\exists \gamma : (m : T) \in K_{X,\gamma}^\phi) // S$ presents various infinitary disjunctions, which are due to the analysis of the generation of random values by the agent X . We can prove, by our assumptions on the deduction function, that it is not fundamental which sequence of generation actions is performed, the essential thing is the correct kind of types that are generated. So we can give a translation from this formula to one without infinitary disjunctions, s.t. the satisfiability is preserved, i.e. F is satisfiable iff $\tilde{F}$ is satisfiable (see [13]). This translated formula $\tilde{F}$ presents only finitary disjunctions. This translation can be performed during the generation of F and leave unchanged the finitary part of the formula. We have reduced the verification of the existence of an agent X_ϕ s.t. $S\|_L X_\phi \models \exists \gamma : (m : T) \in K_{X,\gamma}^\phi$ to a satisfiability problem in a sublogic of $\mathcal{L}$. Moreover, the decidability problem for this sublogic is simple and we can build an agent (i.e. intruder) for a satisfiable formula $\tilde{F}$. Hence, we can state:

Theorem 1. *Given a system S , with $\text{Sort}(S) \subseteq L$, a finite set of typed messages ϕ and an initial message $m : T$ then is decidable if $\exists X_\phi$ with $\text{Sort}(X) \subseteq L$ s.t. $S\|_L X_\phi \models \exists \gamma : (m : T) \in K_{X,\gamma}^\phi$.*

5 Authentication properties

The theory we have presented in the previous sections deals in particular with so called *secrecy* properties, namely that certain pieces of information remain enclosed in a particular context. Among other interesting properties are the authentication ones. The definition of authentication used in [10] can be restated as follows:

Whenever a Sender A completes a run of the protocol, apparently with Receiver B , then B has recently been running a protocol, apparently with A .

We define two distinct actions *start*, *finish* to model the starting of Sender and the termination of Receiver respectively: when Sender starts it issues the action *start* and when Receiver terminates it issues the action *finish*. It is assumed that such actions cannot be performed by others than Sender and Receiver. In this setting it is possible to formalize the authentication property as follows (see [10]):

$$\Phi = \text{for any run } \gamma \quad (finish \in \gamma \downarrow_B \implies start \in \gamma \downarrow_A)$$

Please note that, since the set of runs is prefix closed, this property also implies that *start* precedes *finish* in any γ .

It should appear clear that by adapting the compositional analysis techniques of previous section, this property could be easily checked; here we prefer to show a reduction of the verification of this property to a particular secrecy property, that can be directly handled by our theory (and so by our tool). We believe that this is an interesting result of its own, since to our knowledge this is the first attempt to perform a similar reduction.

We define an encoding $\mathcal{S}$ over systems as:

$$\begin{aligned} \mathcal{S}(A \parallel B) &= (A' \parallel B') \\ A' &= A[start := c!start_v] \parallel c?(y) : special.Nil \\ B' &= B[finish := c'!finish_v] \parallel c?(x) : special.Nil \end{aligned}$$

where c, c' are channels not occurring in $A \parallel B$ and $start_v, finish_v$ are distinguished values. Moreover we assume:

- The intruder cannot interact over channel c, c' . This seems reasonable since these actions appear only for checking purpose; please note that this hypothesis matches the hypothesis that *start* and *finish* actions cannot be executed by the intruder.
- Values $start_v$ and $finish_v$ are basic values such that $start_v, finish_v \notin \phi_X$ and $start_v \in \phi_A \setminus \phi_B, finish_v \in \phi_B \setminus \phi_A$.

Over the system³ $\mathcal{S}(A \parallel B)$ we will consider the following property:

$$\Psi = \text{for any run } \gamma \quad (finish_v \in K_{A,\gamma} \implies start_v \in K_{B,\gamma}).$$

³ Actually, our tool is able to deal also with agents that are built by using parallel composition.

where $K_{A,\gamma}$ ($K_{B,\gamma}$) represents the knowledge of the agent A (B) after the system has performed the sequence γ . Under the above assumptions we can state:

Proposition 2. *If L is such that $\text{Sort}((A \parallel B) \parallel X) \setminus \{\text{start}, \text{finish}\} \subseteq L$ then:*

$$((A \parallel B) \parallel X) \setminus L \models \Phi \text{ iff } (\mathcal{S}(A \parallel B) \parallel X) \setminus L \models \Psi.$$

We can use our tool to check if a system satisfies Ψ by checking if it does not satisfy:

$$\exists \gamma : \text{finish}_v \in K_{A,\gamma} \wedge \text{start}_v \notin K_{B,\gamma}.$$

Moreover if we force the intruder to eavesdrop any message sent over channels c and c' , we can check the property above simply by inspecting the knowledge of the intruder i.e. checking the following property:

$$\exists \gamma : \text{finish}_v \in K_{X,\gamma} \wedge \text{start}_v \notin K_{X,\gamma}.$$

6 Technical framework for the implementation

In this section we show the more interesting theoretical aspects of our tool. In order to implement the partial evaluation function we have to specify how succ , $R\text{succ}$ and the membership of messages in $\mathcal{D}(\phi)$ can be computed. We define the *size* of a message $|m : T|$ as 1 if T is a basic type, and as $1 + \max\{|T_i|\}_{i \in \{1, \dots, \text{ar}(j)\}}$ if $T = F_j(T_1, \dots, T_{\text{ar}(j)})$. Let $\text{Msgs}(|T|)$ be the set of messages whose type has a *size* equal or smaller than T .

We consider as deduction system the one presented in figure 1, that can be proved to enjoy our assumptions (see [13]). In the following we define a canonical representation of the knowledge of agents with the aim to compute easily $\mathcal{D}(\phi) \cap \text{Msgs}(T)$ and $m : T \in \mathcal{D}(\phi)$ (a similar representation, but for a different problem has been presented in [7]).

Definition 1. ϕ is downward closed (*DC*) iff $\forall m : T \in \mathcal{D}(\phi) \setminus \phi$ we have $m : T \in \mathcal{D}(\phi \cap \text{Msgs}(|T| - 1))$.

It is not difficult to prove that with ϕ *DC* we have $m : T \in \mathcal{D}(\phi)$, iff there is a proof of $m : T$ that uses only growing rules (namely rules in which the size of the conclusion is bigger than the sizes of the premises). Hence to decide if $m : T \in \mathcal{D}(\phi)$ it is enough to follow recursively the structure of the message, checking if submessages of $m : T$ belong to ϕ . Also to compute $\mathcal{D}(\phi) \cap \text{Msgs}(T)$ we simply follow the structure of the messages, getting for basic types T_b the list $\phi \cap \text{Msgs}(T_b)$, and then correctly reconstruct an appropriate list of messages.

Definition 2. ϕ is minimal iff $\forall m : T \in \phi$ we have $m : T \notin \mathcal{D}(\phi \cap \text{Msgs}(|T| - 1))$.

Definition 3. ϕ is a base for Γ if $\mathcal{D}(\phi) = \Gamma$ and ϕ is minimal and downward closed.

The property of *minimality* ensures that no unnecessary message belongs to the base, indeed if $m : T$ can be deduced by ϕ which is *DC* then it can be deduced by other messages in ϕ , say $m_1 : T_1, \dots, m_n : T_n$, with $|T_i| < |T|$ for $i \in \{1, \dots, n\}$ and so $m : T \in \mathcal{D}(\phi \setminus \{m : T\})$, with $\phi \setminus \{m : T\}$ *DC* and $\mathcal{D}(\phi) = \mathcal{D}(\phi \setminus \{m : T\})$. Moreover this representation enjoys the following strong property:

Proposition 3. *Given $\mathcal{D}(\phi)$ with ϕ base, if ψ is a base for $\mathcal{D}(\phi)$ then we have $\psi = \phi$.*

The last thing we have to specify is how a base ϕ can be updated to a base ϕ' in such a way to have $\mathcal{D}(\phi') = \mathcal{D}(\phi \cup \{m : T\})$. Given ϕ base for $\mathcal{D}(\phi)$ then we define $Add(m : T, \phi)$ such that $\mathcal{D}(Add(m : T, \phi)) = \mathcal{D}(\phi \cup \{m : T\})$:

```

Add(m : T,  $\phi$ ) =
  { $m_1 : T_1, \dots, m_n : T_n$ } = Decompose( $m : T, \phi$ )
   $\phi_0 = \phi \cup \{m : T\}$ 
  for i=1 to n do  $\phi_i = Add(m_i : T_i, \phi_{i-1})$ 
  Include( $m : T, \phi_n - \{m : T\}$ )

```

where:

- $Decompose(m : T, \phi)$ is the set of messages that can be derived from $m : T$; more precisely we compute $Decompose(m : T, \phi)$ as the set of messages that can be deduced starting from message $m : T$ and applying exactly one “destructor” rule (projection rule for pairs and decryption rule for encryptions). In this way we inductively consider smaller and smaller messages to be inserted in ϕ until undecomposable messages are reached;
- $Include(m : T, \phi)$ is the *minimal* $\phi' \subseteq (\phi \cup \{m : T\})$ such that $\mathcal{D}(\phi') = \mathcal{D}(\phi \cup \{m : T\})$. To obtain $Include(m : T, \phi)$ we take advantage of the fact that all the relevant submessages of $m : T$ have already been included in ϕ , so we simply need to remove from ϕ messages directly derivable from $m : T$. What we do is to remove from ϕ all the messages that can be deduced starting from $m : T$ and applying “constructor” rules (pair formation and encryption).

Proposition 4. *We have that $Add(m : T, \phi)$ is a base for $\mathcal{D}(\phi \cup \{m : T\})$.*

From a practical point of view, our work permits the so called *on the fly* analysis technique, i.e. if there are some errors, these can be found even without the explicit analysis of the whole system.

7 Optimizations

In this section we try to highlight some further optimizations for our analysis of protocols. We have already seen that the formula produced by the partial evaluation function can be reduced to a finitary one, still preserving satisfiability. Here we present other reductions on the formulas that can improve the efficiency

of the verification method. From the definition it follows that the deduction functions are *monotonic*, i.e. if $\phi \subseteq \phi'$ then $\mathcal{D}(\phi) \subseteq \mathcal{D}(\phi')$. This leads to the following fact, if $\phi \subseteq \phi'$ then:

$$S \parallel_L X_\phi \models \exists \gamma : (m : T) \in K_{X,\gamma}^\phi \implies S \parallel_L X_{\phi'} \models \exists \gamma : (m : T) \in K_{X,\gamma}^{\phi'}.$$

The above implication follows from the fact that if $X_\phi \xrightarrow{\gamma}$ then $X_{\phi'} \xrightarrow{\gamma}$ too. So an intuitive approach could be to consider the behaviours of an intruder where his knowledge grows as much as possible. This idea has been exploited by many researchers (see [18]), in particular Shmatikov and Stern claim that they first proved the soundness of this approach. Their model differs from ours, since it is based on asynchronous communications. Here we transpose their idea in our formal context and we show also the soundness of our reduction, that can be stated more easily due to our logical characterization. If we look at the partial evaluation function we note that there are two possible behaviours for an intruder w.r.t. a system that can perform a communication of a message $m : T$ on a channel c (i.e. a $\tau_{c,m:T}$ action), he can wait otherwise he can eavesdrop the communication. It is clear that by idling he loses the possibility to increase his knowledge, and if an intruder can derive $m' : T'$ by starting from ϕ then he could derive it by starting from $\phi \cup \{m : T\}$. So we have that if:

$$(\exists \gamma : (m' : T') \in K_{X,\gamma}^\phi // S') \text{ is satisfiable}$$

then

$$\langle \tau_{c,m:T} \rangle (\exists \gamma : (m' : T') \in K_{X,\gamma}^{\phi \cup \{m:T\}} // S') \text{ is satisfiable.}$$

Since we consider disjunctive formulas, we can safely cut off the part of the partially evaluated formula, which is obtained through the analysis of idling behaviour of the intruder with respect to a communication action, since if this formula is satisfiable then the formula corresponding to the eaves-dropping of this communication is satisfiable.

Another suggestion is to avoid the intruder sending a message if an honest participant in the protocol can do that. We can formally state this reduction in our formalism as an equivalence of the satisfiability problem between:

$$\langle c?m:T \rangle \langle c!m:T \rangle (\exists \gamma : (m' : T') \in K_{X,\gamma}^\phi // S') \text{ and } (\exists \gamma : (m' : T') \in K_{X,\gamma}^\phi // S')$$

where $S \xrightarrow{\tau_{c,m:T}} S'$ and $m : T \in \phi$.

It is clear that if $(\exists \gamma : (m' : T') \in K_{X,\gamma}^\phi // S')$ is not satisfiable then also $\langle c?m : T \rangle \langle c!m : T \rangle (\exists \gamma : (m' : T') \in K_{X,\gamma}^\phi // S')$ is not satisfiable, otherwise it is like to say that $\langle a \rangle \langle b \rangle \mathbf{F}$ is satisfiable. The other side of the equivalence is similar.

The partial order reduction techniques can be applied too. In this way it is possible to exploit the independence between actions (i.e. performing of one of the two actions do not prevent the performing of the other), for example in the case that the system has two separate agents that both can perform a sending action. Hence, we can prove that:

$$\begin{aligned} & \langle c?m : T \rangle \langle c'?m' : T' \rangle (\exists \gamma : (m'' : T'') \in K_{X,\gamma}^\phi // S') \vee \\ & \langle c'?m' : T' \rangle \langle c?m : T \rangle (\exists \gamma : (m'' : T'') \in K_{X,\gamma}^\phi // S') \end{aligned}$$

is equivalent (from the satisfiability point of view) to

$$\langle c?m : T \rangle \langle c'?m' : T' \rangle (\exists \gamma : (m'' : T'') \in K_{X,\gamma}^\phi // S').$$

8 Needham Schroeder Public Key protocol

In this section we show an example protocol that has become paradigmatic for testing tools for cryptographic protocol analysis. For a long time it has been considered correct, and also proved within a logical framework. It has a simple flaw, that arises when the system is considered in presence of another agent. Below we present the *intended* execution between a sender and a receiver, by using the notation used in literature.

$$\begin{array}{ll}
 A \mapsto B : \{N_a, A\}_{PK(B)} & A \mapsto B : \{N_a, A\}_{PK(B)} \\
 B \mapsto A : \{N_a, N_b\}_{PK(A)} & B \mapsto A : \{N_a, N_b, A\}_{PK(A)} \\
 A \mapsto B : \{N_b\}_{PK(B)} & A \mapsto B : \{N_b\}_{PK(B)} \\
 \text{flawed version} & \text{corrected version}
 \end{array}$$

In the flawed version the sender A communicates to B a fresh nonce N_a and its name encrypted with the public key of B (so only B , who knows the private key, can decrypt this message). Then the receiver B communicates to A the nonce N_a that he has received before and a fresh nonce N_b encrypted with the public key of A . Finally the sender communicates to the receiver the nonce N_b . In the intention of the designer of the protocol, at the end of a run between a sender A and a receiver B , it must be that only A and B know N_a and N_b (these nonces can be used to establish a new communication with a new shared key that is function of these values).

Our specification is based on the description of the behaviour of the two components separately. We have tested our specification and as expected we have found a flaw, even if a slight different w.r.t. the one presented in [10]. An intruder is able to know the nonce N_b . To perform the verification we have only specified the initial knowledge of a possible intruder, i.e. the public keys of A and B , the names of A and B , and his private and public keys. We do not need to give the nonces to the agents, since contrary to other approaches, our framework allows the intruder to guess them autonomously.

The following is a behaviour of an intruder that causes N_b to be leaked (we use $X(A)$ to represent the intruder that takes part to a communication as the agent A):

$$\begin{array}{ll}
 A \mapsto X & : E(Xkey, (N_a, A)) \\
 X(A) \mapsto B & : E(Bkey, (N_a, A)) \\
 B \mapsto X(A) & : E(Akey, (N_a, N_b)) \\
 X \mapsto A & : E(Akey, (N_a, N_b)) \\
 X(A) \mapsto B & : E(Akey, N_a) \\
 A \mapsto X & : E(Xkey, N_b)
 \end{array}$$

The attack performed can be summarized as follows: the agent A starts a run of the protocol with the agent X ; then the agent X can simulate A in a run of the protocol with the agent B . The agent B sends to $X(A)$ the message $E(Akey, (N_a, N_b))$, which contains the fresh nonce N_b , encrypted with A public key. Now the intruder is not able to decrypt directly the message, but he can send the message to the agent A . The agent A will correctly decrypt

$E(Akey, (N_a, N_b))$ and then he reply the nonce N_b to X , encrypted with X public key, since he thinks that is the second message of his run with X . Now X knows N_b ! It is interesting to note that the above intruder is not very clever, since he sends to B , as last message, a non correct message (encrypted with a wrong key), by permitting B to understand that there are some problems. A clever intruder can wait to receive the correct message from A , and then send N_b back to B , correctly encrypted. Also this intruder can be found by our *secrecy* analysis. By performing *authentication* analysis only the latter intruder can be found. Indeed, we have corrected the protocol similarly to [10] and we have verified that there are no flaws. The presented attack is found in few seconds by our tool, and the verification of the corrected version takes less than a minute on a Pentium PC, with Linux operating system. It is interesting to note that we do not need to introduce a specification for an intruder.

9 Conclusions and related work

We have proposed a tool for checking security properties of cryptographic protocols. The underlying methodology is a transposition of the ideas proposed for the analysis of information flow security properties (i.e. *non interference*, see [13]). Hence, compositional analysis techniques seem to provide a unique conceptual framework for the study of security properties.

To our knowledge, the only previous attempt to analyze *non interference* and authentication protocols within the same conceptual framework, has been proposed by Focardi and Gorrieri ([4,6]), but in their work an explicit description for a particular (the most general) intruder is requested.

The same limitation is present in the seminal work of Lowe [10], who applies generic tools for verification of process algebra terms for the analysis of authentication protocols. In the aforementioned paper, Lowe shows how by starting from the results of the analysis on a finite number of runs, one can deduce the correctness of the whole behaviour of the protocol.

Perhaps, a work more similar to ours is the one of Marrero *et al.* (see [12]), where a model with sequential agents is used and the explicit description of an intruder is not needed since an axiomatic behaviour of the intruder is supposed. But the work is more limited in its scopes, since they do not permit the intruder to guess nonces and uses two different methodology for secrecy and authentication, and seems not to be directly generalizable (they make reasoning with a fixed theory), while generality and flexibility are major topics of our work. Actually, our approach is from the opposite direction, the behaviour of an intruder is automatically considered when one applies the point of view of compositional analysis.

Another interesting approach to model intruders, is the one of Abadi and Gordon (see [1]). They use the testing equivalence theory for a variant of the π -calculus to take in account the presence of unspecified intruders that try to leak some information. The idea is very appealing, but their approach differs from

ours since it relies mainly on proof techniques, while our approach is tailored for automatic analysis.

Acknowledgments We would like to thank the anonymous referees for their helpful comments.

References

1. M. Abadi and A. D. Gordon. Reasoning about cryptographic protocols in the spi calculus. In *Proc. of CONCUR'97*, volume 1243 of *LNCS*, pages 59–73, 1997.
2. M. Abadi and M. R. Tuttle. A semantics for a logic of authentication. In *Proc. of PODC'91*, pages 201–216. ACM Press, 1991.
3. H. R. Andersen. Partial model checking (extended abstract). In *Proc. of LICS'95*, pages 398–407. IEEE Computer Society Press, 1995.
4. R. Focardi, A. Ghelli, and R. Gorrieri. Using non interference for the analysis of security protocols. In *Proc. of DIMACS Workshop on Design and Formal Verification of Security Protocols*. DIMACS Center, Rutgers University, Sep. 1997.
5. R. Focardi and R. Gorrieri. A classification of security properties. *Journal of Computer Security*, 3(1):5–33, 1995.
6. R. Focardi and R. Gorrieri. The compositional security checker: A tool for the verification of information flow security properties. *IEEE Trans. Sof. Eng.*, 27:550–571, 1997.
7. D. Kindred and J. M. Wing. Fast, automatic checking of security protocols. In *Second USENIX Workshop on Electronic Commerce*, pages 41–52, Oakland, California, 1996.
8. K. G. Larsen and L. Xinxin. Compositionality through an operational semantics of contexts. *Journal of Logic and Computation*, 1(6):761–795, Dec. 1991.
9. P. Lincoln, J. Mitchell, M. Mitchell, and A. Scedrov. A probabilistic poly-time framework for protocol analysis. In *ACM Symposium in Computer and Communication Security*, 1998.
10. G. Lowe. Breaking and fixing the Needham Schroeder public-key protocol using FDR. In *Proc. of TACAS'96*, volume 1055 of *LNCS*, pages 147–166, 1996.
11. G. Lowe and B. Roscoe. Using CSP to detect errors in the TMN protocol. *IEEE Trans. Sof. Eng.*, 23(10):659–669, 1997.
12. W. Marrero, E. Clarke, and S. Jha. A model checker for authentication protocols. In *Proc. of DIMACS Workshop on Design and Formal Verification of Security Protocols*. Rutgers University, Sep. 1997.
13. F. Martinelli. Languages for description and analysis of authentication protocols. In *Proceedings of 6th Italian Conference on Theoretical Computer Science*. World Scientific, 1998.
14. F. Martinelli. Partial model checking and theorem proving for ensuring security properties. In *Proc. of CSFW'98*, pages 44–52. IEEE Computer Society Press, 1998.
15. F. Martinelli. An uniform approach for the analysis of open systems. Draft, 1998.
16. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
17. L. Paulson. Proving properties of security protocols by induction. In *Proc. of CSFW'97*. IEEE Computer Society Press, 1997.
18. V. Shmatikov and U. Stern. Efficient finite-state analysis for large security protocols. In *Proc. of CSFW'98*, pages 105–116. IEEE Computer Society Press, 1998.