ELSEVIER Theoretical Computer Science 290 (2003) 1057–1106

Theoretical Computer Science

www.elsevier.com/locate/tcs

Analysis of security protocols as *open* systems☆

Fabio Martinelli

Istituto per le Applicazioni Telematiche-C.N.R., via G. Moruzzi 1, I-56100 Pisa, Italy

Received 13 October 2000; received in revised form 17 April 2001; accepted 6 August 2002
Communicated by R. Gorrieri

Abstract

We propose a methodology for the formal analysis of security protocols. This originates from the observation that the verification of security protocols can be conveniently treated as the verification of *open* systems, i.e. systems which may have unspecified components. These might be used to represent a hostile environment wherein the protocol runs and whose behavior cannot be predicted a priori. We define a language for the description of security protocols, namely Crypto-CCS, and a logical language for expressing their properties. We provide an effective verification method for security protocols which is based on a suitable extension of partial model checking. Indeed, we obtain a decidability result for the secrecy analysis of protocols with a finite number of sessions, bounded message size and new nonce generation.

Keywords: Security protocol analysis; Process algebras; Partial model checking; Temporal logic

1. Overview

The increasingly growing amount of security-sensitive information flowing into computer networks has attracted a lot of interest in the investigation of formal methods for the definition and analysis of the security properties which these systems must ensure.

Computer networks may consist of thousands of geographically distributed computers, and communicating between two of them may involve the exploiting of the communication features of many others in the network. This results in the need of establishing secure communication channels, so that the exchanged information is kept confidential all through the communication steps (*secrecy*). Furthermore, the communicating parties should be sure about the origin of the messages they receive (*message*

☆ This paper is a revised and extended version of [35].

E-mail address: fabio.martinelli@iit.cnr.it (F. Martinelli).

0304-3975/02/$ - see front matter
PII: S0304-3975(02)00596-0

authentication) or the identity of the other parties involved in the communication (*entity authentication*). Several communication protocols have been developed to try to solve these problems through cryptography [44]. The messages sent in the network are usually encrypted, i.e. manipulated so as to make the actual message recoverable only by the users who know some given information, such as a decryption key; furthermore, messages are sometimes digitally signed, i.e. manipulated so as to make the message recoverable by anyone, while making sure it may have been originated only by the user who knows a certain piece of information, i.e. the signature key.

Cryptography is a fundamental tool for ensuring security properties, but it is not sufficient by itself, as proved by many flaws found in cryptographic protocols (e.g., see [3,13,29,40]). This holds even by assuming cryptography as completely reliable, i.e. without considering crypto-analysis attacks but only how messages are exchanged among the parties involved in the protocol. Thus, cryptographic protocols are recognized to be prone to errors, even though, conceptually, they involve only a few communication steps between the parties.

Over the last few years, several techniques for finding flaws in such protocols have been developed (e.g., see [13,26,27,29,31,37,40,43]). Some of them are based on (finite) state-exploration, and can typically ensure error-freeness for bounded systems only. Other approaches are based on proof techniques for authentication logic (e.g., see [3,21,42]) or process algebra (e.g., see [2,8]). Type systems and other static analysis techniques have also been successfully exploited (e.g., see [16]).

Our approach is novel in this area. It was spurred by the observation that security protocols can be conveniently described by open systems. Generally speaking, a system is open if it has some unspecified components. This un-specification may depend on several factors: for instance, we are at an early development phase and hence not all the implementation details are fixed or perhaps we are simply unable to predict a component's behavior within a system. In both cases, however, we want to make sure that, whenever the unspecified component is definitely fixed, the resulting system works properly, e.g. fulfills a certain property. Thus, the intuitive idea underlying the verification of an open system is the following: an open system satisfies a property if and only if, whatever component is substituted to the unspecified one, the whole system satisfies this property. In the context of formal languages for the description of system behavior, an open system may be simply regarded as a term of this language which may contain "holes" (or placeholders). These are the unspecified components. Consider, for instance, a language for describing concurrent systems such as CCS [38] or CSP [20] and let $|$ be the operator for representing parallel-running systems; then $A|(\_)$ and $A|B|(\_)$ may be considered as open systems.

Several situations which commonly arise in computer security analysis may be regarded as instances of open systems verification, e.g.:

- Security protocols involve several parties sending and receiving information over a possibly insecure network. We can then imagine a hostile intruder being "present into" the network and being able to listen, tap into and fake messages to attack the protocol. Suppose, for instance, having a user A willing to send a message m to another user B. This situation is usually described through the term $A|B$. It would be better to consider instead the open system $A|B|(\_)$, where the unknown component

(the hole) may be used to take into account the presence of an intruder whose behavior we are not able to predict. Note that A and B are not necessarily aware of the presence of the intruder.

- Sometimes, we also need to consider situations where some of the parties to a protocol behave maliciously, trying to achieve an advantage for themselves. Indeed, such parties could not behave as prescribed by the protocol. In the above example, if we choose not to trust B, then the situation should be properly modeled by analyzing the context $A|(\_)$. In contrast with the previous example, where we add information about the presence of an external intruder, here we remove the information about the "intended" behavior of a certain user, e.g. B; however, A still assumes the presence of B with such "intended" behavior. This amounts to require A has a strategy to protect itself against whatever malicious behavior B may have during the participation to the protocol.

Thus, when analyzing security-sensitive systems, neither the enemy's behavior nor the malicious users' behavior should be fixed beforehand. This will prevent us from making unjustified assumptions which could lead to erroneous (and dangerous) verification results. To sum up, a system should be secure regardless of the behavior the malicious users or intruders may have, which is exactly a verification problem of open systems. [1] Indeed, our proposal for defining security properties as "open systems" properties is the following (e.g. see [34–36]):

$$\text{For every component } X \quad S \,|\, X \models p, \tag{1}$$

where X stands for the possible enemies, S is the system under examination, consisting of honest participants, $|$ is the parallel-composition operator, p is a logical formula expressing the security property and $\models$ is the truth relation. It roughly states that the property p holds for the system S, regardless of the component (i.e., intruder, malicious user, hostile environment, etc.) which may possibly interact with it. We can instantiate the previous idea by assuming the process X is an intruder trying to discover information which should remain enclosed in the system S. The confidentiality of this information may be expressed by the logical formula p. If (1) holds, then the piece of information would remain unknown to any attacker X of the system S.

However, we need a method to mechanically check the properties expressed as (1).

1.1. The analysis method

We give an outline of our analysis approach, which will be discussed in further detail in the remainder of this paper.

Note that such properties as (1) look like validity statements of mathematical logic, i.e.

$$\text{For every component } X \quad X \models p, \tag{2}$$

where the formula p must be checked for every structure X. The main difference is that in (1) we check the components X in combination with a system S.

[1] Note also the analogy between open system verification and game theory issues as remarked in [24].

Our aim is to reduce such a verification problem as (1) to such a validity checking problem as (2). To obtain this, we apply and extend the partial model checking techniques used for the compositional verification of concurrent systems (see [5,25]).

Consider a system S in combination with a process X and try to figure out if the whole system $S \mid X$ enjoys a property expressed by a formula p or not. Then, partial model checking techniques can be used to find the sufficient and necessary condition on X, expressed by a logical formula p_S, so the whole system $S \mid X$ satisfies p. Briefly, we have

$$S \mid X \models p \quad \text{iff} \quad X \models p_S. \tag{3}$$

The key to obtaining such results is that the semantics of several formal description languages is given in the *Structured Operational Semantics* (SOS) style: there is a set of premises/conclusion rules which are used to infer the activities of complex systems starting from the activities of their components. Assume, for example, to have a single rule, like the following one:

$$\frac{S_1 \xrightarrow{a} S_1' \quad S_2 \xrightarrow{b} S_2'}{S_1 \mid S_2 \xrightarrow{a} S_1' \mid S_2'},$$

where $S \xrightarrow{a} S'$ means the system S is able to perform the activity a and evolve in the system S'. Then, the semantics of the composition of two systems $(S_1 \mid S_2)$ is uniquely obtained from the semantics of the two components, say S_1 and S_2. This form of compositional reasoning simplifies our task. Suppose to have a set of formulas denoting system activities and that the formula p means the activity a may be performed. Assume also that $S_1 \xrightarrow{a} S_1'$ holds for some S_1'. Then, a sufficient condition on S_2 to have $S_1 \mid S_2$ satisfying p is simply S_2 performing the activity b. This condition may be described through a formula, say p_{S_1}. However, the behavior of $S_1 \mid S_2$ is completely determined by the previous SOS rule. Thus, the unique possibility to have the firing of the transition a by $S_1|S_2$ is through the application of the rule itself. Indeed, the condition S_2 enjoying p_{S_1} turns out to be also necessary. (The actual formal framework is rather more complex and depends on the logic used and the particular format of SOS rules, e.g. see [5,25,34].)

Using property (3), such verification problems as in (1) can be easily reduced to such problems as in (2).

Now, we only need to equip our logic with a suitable validity (satisfiability) decision procedure. The wide research on temporal logic provides us with several useful techniques and results. The computational complexity of validity decision procedures is usually unfeasible. It is however worth noticing that, at least for the properties analyzed in this paper, the validity (satisfiability) problem can be efficiently solved for formulas obtained after partial model checking, i.e. in polynomial time in their size. (However, note that the size of the formula could be exponential in the size of the system analyzed.) A software tool implementing our methodology has actually been developed [30].

Outline of the paper: The remainder of this paper is organized as follows. Section 2 defines the language (actually a process calculus) we use for the description of

cryptographic protocols. Section 3 defines a logical language for the definition of the secrecy properties of protocols. Section 4 explains the proposed approach. Section 5 shows examples of our analysis. Finally, Section 6 provides some concluding remarks. The proofs have been attached in the appendix for readability purposes.

2. Crypto-CCS: an operational calculus for the description of protocols

This section presents the calculus we use for the description of security protocols, which is a slight modification of *CCS* process algebra [38] using cryptography-modeling constructs and dealing with secret (confidential) values (hence the name Crypto-CCS). The model consists in a set of sequential agents able to communicate each other by exchanging messages.

To analyze cryptographic protocols, we need to formally model a wide range of functions, such as encryption, decryption, hashing, etc., and each of them enjoys specific algebraic properties which may significantly affect the correctness of such protocols. To manage such variety, we decided to parameterize our calculus through a set of message-manipulating rules. For the application of these rules, we added a new construct to the calculus. We thus build a framework which is parametric w.r.t. the specific crypto-system used and the way messages are manipulated.

2.1. Types, typed messages and inference systems

Here, we define the data handling part of calculus. First of all, we introduce the notions of type, message and typed message. Types are used to record the structure and kind of data. Since certain operations are meaningful only over data with a certain structure, types permits us to define managing rules that precisely corresponds to these operations.[2] Messages are the data manipulated by agents and typed messages are messages whose structure is explicitly represented. We introduce also the notion of inference system which models the possible operations on typed messages.

Types: Consider a set of symbols $\mathscr{B}T = \{T^1, \ldots, T^n\}$ that represent the *basic types* and a set of symbols $\mathscr{F} = \{F^1, \ldots, F^l\}$ that represent the *constructors* for structured types. Assume to have a countable set TV of *type variables*. Then, the set *Types* is defined by the following grammar:

$$t ::= x \mid T \mid F^1(t_1, \ldots, t_{\mathrm{ar}(F^1)}) \mid \ldots \mid F^l(t_1, \ldots, t_{\mathrm{ar}(F^l)}),$$

where $x \in TV$, $T \in \mathscr{B}T$ and $ar(F)$ is the number of arguments of the constructor F.

Messages: Consider a collection $\{\mathsf{T}^1, \ldots, \mathsf{T}^n\}$ of infinite sets which are pairwise disjoint, i.e. $\mathsf{T}^i \cap \mathsf{T}^j = \emptyset$ when $i \neq j$. Each set T^i contains the messages of the basic type T^i. Assume to have a countable set V of message variables. Then, the set *Msgs* of messages is defined by the following grammar:

$$m ::= x \mid M \mid F^1(m_1, \ldots, m_{\mathrm{ar}(F^1)}) \mid \ldots \mid F^l(m_1, \ldots, m_{\mathrm{ar}(F^l)}),$$

[2] Types will be also useful for defining some conditions for our analysis.

where $x \in V$, $M \in \bigcup_{j\in\{1,\ldots,n\}} \mathsf{T}^j$ and $ar(F)$ is as above. (Note that we use the set $\mathscr{F}$ of constructors both for types and messages.)

Typed messages: We can label each message with a type that denotes its structure. Consider an assignment δ that maps message variables to types and let $type : Msgs \mapsto Types$ be:

$$type(m) = \begin{cases} \delta(x) & \text{if } m = x, \\ T^i & \text{if } m \in \mathsf{T}^i, \text{ where } i \in \{1,\ldots,n\}, \\ F(type(m_1),\ldots,type(m_l)) & \text{if } m = F(m_1,\ldots,m_l). \end{cases}$$

Let *Tmsgs* be the set of typed messages defined as $\{m{:}\ type(m) \mid m \in Msgs\}$. Types, messages and typed messages without variables are said to be *closed*. The set of closed typed messages of a closed-type T is $Tmsgs(T)$. For the sake of readability, we sometimes leave out a type from a typed message since this can be inferred from the message itself. Equality between typed messages means syntactic equality. Furthermore, we write $\langle\langle m_i : T_i\rangle\rangle_{i\in I}$, with $I = \{1,\ldots,n\}$, for a sequence $m_1 : T_1 \ldots m_n : T_n$.

Inference system: Agents are able to obtain new messages from the set of messages produced or received through an inference system. This system consists in a set of inference schemata. An inference schema can be written as

$$IS = \frac{m_1 : t_1 \cdots m_n : t_n}{m_0 : t_0},$$

where $m_1 : t_1, \ldots, m_n : t_n$ is a set of premises (possibly empty) and $m_0 : t_0$ is the conclusion. Consider a pair of assignments (ρ_1, ρ_2), with $\rho_1 : V \mapsto Msgs$, $\rho_2 : TV \mapsto Types$, then let $m[\rho_1]$ be the message m where each variable x is replaced with $\rho_1(x)$ (a similar definition applies to the type $t[\rho_2]$). Given a sequence of closed typed messages $\langle\langle m'_1 : t'_1, \ldots, m'_n : t'_n\rangle\rangle$, we say that a closed typed message $m : t$ can be inferred from $\langle\langle m'_1 : t'_1, \ldots, m'_n : t'_n\rangle\rangle$ through the application of the schema IS (written as $\langle\langle m'_1 : t'_1, \ldots, m'_n : t'_n\rangle\rangle \vdash_{IS} m : t$) if there exists a pair of assignments (ρ_1, ρ_2), with $\rho_1 : V \mapsto Msgs$, $\rho_2 : TV \mapsto Types$, s.t. $m_0[\rho_1] : t_0[\rho_2] = m : t$ and $m_i[\rho_1] : t_i[\rho_2] = m'_i : t'_i$, for $i \in \{1,\ldots,n\}$. (Note that, given $\langle\langle m'_1 : t'_1, \ldots, m'_n : t'_n\rangle\rangle$, it is decidable whether or not there exists $m : t$ s.t. $\langle\langle m'_1 : t'_1, \ldots, m'_n : t'_n\rangle\rangle \vdash_{IS} m : t$.) A *deduction* (or proof) for a closed typed message $m : t$ is a finite tree, rooted in $m : t$, whose nodes are messages built from their descendants through the application of an instance of an inference schema. Given an inference system, we can define an inference function $\mathscr{D}$ s.t. if ϕ is a finite set of closed messages, then $\mathscr{D}(\phi)$ is the set of closed messages that can be deduced starting from ϕ. We assume that $\mathscr{D}(\phi)$ is decidable.

Example 2.1. Table 1 shows an inference system for the modeling of cryptographic functions similar to the one used in [27,31,43]. Consider a set of basic types $\mathscr{B}T = \{Key, Agents, Nonces\}$ which stand for encryption keys, agent names and random numbers. Consider also a set of constructors $\mathscr{F} = \{E, (-1), \times\}$ which are, respectively, the constructors for encryptions, decryption keys and pairs. For instance, the followings are types: $Nonces \times Agents$, $Agents \times E(Key, Nonces)$ (we use the infix notation for the pairing construct), $E(x, Agents \times Agents)$ (with $x \in TV$) and Key^{-1} (we use the

Table 1
A simple inference system

$$\frac{x:t_1 \quad y:t_2}{(x,y):t_1\times t_2}\ (1) \qquad \frac{(x,y):t_1\times t_2}{x:t_1}\ (2) \qquad \frac{(x,y):t_1\times t_2}{y:t_2}\ (3)$$

$$\frac{x:t \quad y:Key}{E(y,x):E(Key,t)}\ (4) \qquad \frac{E(y,x):E(Key,t) \quad y^{-1}:Key^{-1}}{x:t}\ (5)$$

Table 2
Inference of $m:T$ from $(E(k,m),k^{-1}):(E(Key,T)\times Key^{-1})$

$$\frac{\dfrac{m':T'}{E(k,m):E(Key\times T)}\ (2) \quad \dfrac{m':T'}{k^{-1}:Key^{-1}}\ (3)}{m:T}\ (5)$$

superscript notation for the inverse constructor). Given a set of messages ϕ, then $m:T\in\mathscr{D}(\phi)$ iff $m:T$ can be inferred by the rules (1–5). Rule 1 builds the pairs of two messages; Rules 2 and 3 are used to obtain the components of a pair; Rule 4 allows messages to be encrypted using a key, while Rule 5 allows messages to be decrypted using the corresponding inverse key. Consider the closed typed message $m':T'=(E(k,m),k^{-1}):(E(Key,T)\times Key^{-1})$ (note we use commas to separate messages in pairs). Then, $m:T\in\mathscr{D}(\{m':T'\})$ since there exists a deduction of $m:T$ from $m':T'$ (see Table 2).

2.2. Agents and systems

We define the control part of our calculus for the description of cryptographic protocols. Basically, we consider (compound) systems which consist of sequential agents running in parallel. A sequential agent may be used to represent one (or more) user's sessions of a security protocol. These sessions typically consist of ordered sequences of actions for each user. A protocol which consists of the concurrent execution of several sessions of protocol participants may be described by a compound system.

The terms of our calculus are generated by the following grammar:

$$\begin{aligned}
&(\text{Compound systems:})\ S ::= \ S\backslash L \mid S_1\|S_2 \mid A_\phi,\\
&(\text{Sequential agents:})\ A ::= \ \mathbf{0} \mid \pi.A \mid A_1+A_2 \mid [m=m']A_1;A_2 \mid\\
&\qquad\qquad [\langle\langle m_i\rangle\rangle_{i\in I}\vdash_{IS} x:T]A_1;A_2,\\
&(\text{Prefix constructs:})\ \ \pi ::= c!m \mid c?x:T \mid \chi^x_{c,T} \mid gen^{x,I}_{T^k},
\end{aligned}$$

where $m,m',m_1,\ldots,m_n$ are *closed* messages or variables, x is a message variable, T,T^k are closed types and T^k is also basic, C is a finite set of channels with $c\in C$, ϕ is a finite set of *closed* typed messages, L is a subset of C and $i\in I\subseteq\mathbb{N}$ (the set of natural numbers).

We briefly give the informal semantics of sequential agents and compound system as well as some static constraints on the terms of the calculus.

Sequential agents:

- **0** is the process that does nothing.
- $\pi.A$ is the process that can perform an action according to the particular prefix construct π and then behaves as A:
  - $c!m$ allows the message m to be sent on channel c.
  - $c?x:T$ allows messages $m:T$ to be received on channel c. The message received substitutes the variable x.
  - $\chi^x_{c,T}$ is used to eavesdrop a communication on channel c which occurs in other sub-components of the system. The eavesdropped message substitutes the variable x.
  - $gen^{x,i}_T$ is used to generate new random messages of a basic type T. The message generated substitutes the variable x.
- $A_1 + A_2$ is the process that non-deterministically decides to behave as A_1 or A_2.
- $[m = m']A_1; A_2$ is the matching construct. If the two messages are equal to each other, then the process behaves as A_1, otherwise as A_2.
- $[\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} x:T]A_1; A_2$ is the inference construct. If, applying a case of inference schema IS with the premises $\langle\langle m_i:T_i\rangle\rangle_{i\in I}$, a message $m:T$ can be inferred, then the process behaves as A_1 (where x is replaced with m); otherwise the process behaves as A_2. This is the message-manipulating construct of the calculus: we can build a new message by using the messages in $\langle\langle m_i\rangle\rangle_{i\in I}$ and the inference rule IS.

Consider, for instance, the inference system in Table 1 and the sequential agent:

$$
\begin{array}{ll}
c?x:E(Key,T). & \{\text{receives } x\}\\
[x\ k^{-1} \vdash_5 y:T]\ \{\text{and tries to decrypt it}\} & \\
\quad (out!y.\mathbf{0}) & \{\text{with success}\}\\
\quad ; out!err.\mathbf{0} & \{\text{with failure}\}.
\end{array}
$$

The agent may receive a message and try to decrypt it using the inverse key k^{-1}: if it succeeds, then the calculated value is sent on channel *out*, otherwise the system outputs an error message *err*.

Given a message m in $\mathscr{D}(\phi)$, we can find a sequential agent X_ϕ which may build m by using several times the inference construct (since proofs always consist of a finite number of applications of the inference rules).

Compound systems:

- The system $S\backslash L$ is prevented from performing actions whose channel belongs to the set L, except for synchronizations.
- A compound system $S\|S_1$ performs an action a if either of its sub-components performs a, and a synchronization action ($\tau_{c,m}$), if the sub-components perform complementary actions, i.e. send-receive actions. It is worth noticing that, unlike *CCS*, our synchronization actions carry information about the message exchanged and the channel used. In this way, we can model eavesdropping. Indeed, the agents of one component, e.g. S, might know the message exchanged during the synchronization of the other component, i.e. S_1, by simultaneously performing an eavesdropping action χ.

Table 3
Operational semantics, where the symmetric rules for $\|_1, \|_2$ and $\|_\chi$ are left omitted

$$(!)\ \frac{}{(c!m.A)_\phi \xrightarrow{c!m} (A)_\phi} \qquad (?)\ \frac{m : T \in Tmsgs(T)}{(c?x : T.A)_\phi \xrightarrow{c?m} (A[m/x])_{\phi \cup \{m:T\}}}$$

$$(\chi)\ \frac{m : T \in Tmsgs(T)}{(\chi^x_{c,T}.A)_\phi \xrightarrow{\chi_{c,m}} (A[m/x])_{\phi \cup \{m:T\}}} \qquad (gen)\ \frac{g = \mathscr{R}^T(i)}{(gen^{x,i}_T.A)_\phi \xrightarrow{\tau_g} (A[g/x])_{\phi \cup \{g:T\}}}$$

$$([]_1)\ \frac{m = m' \quad (A_1)_\phi \xrightarrow{a} (A'_1)_{\phi'}}{([m = m']A_1; A_2)_\phi \xrightarrow{a} (A'_1)_{\phi'}} \qquad ([]_2)\ \frac{m \neq m' \quad (A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}}{([m = m']A_1; A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}}$$

$$(+_1)\ \frac{(A_1)_\phi \xrightarrow{a} (A'_1)_{\phi'}}{(A_1 + A_2)_\phi \xrightarrow{a} (A'_1)_{\phi'}} \qquad (+_2)\ \frac{(A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}}{(A_1 + A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}}$$

$$(\mathscr{D}_1)\ \frac{\langle\langle m_i : T_i\rangle\rangle_{i\in I} \vdash_{IS} m : T \quad (A_1[m/x])_{\phi \cup \{m:T\}} \xrightarrow{a} (A'_1)_{\phi'}}{([\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} x : T]A_1; A_2)_\phi \xrightarrow{a} (A'_1)_{\phi'}}$$

$$(\mathscr{D}_2)\ \frac{\nexists (m : T)\langle\langle m_i : T_i\rangle\rangle_{i\in I} \vdash_{IS} m : T \quad (A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}}{([\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} x : T]A_1; A_2)_\phi \xrightarrow{a} (A'_2)_{\phi'}}$$

$$(\|_1)\ \frac{S \xrightarrow{a} S'}{S\|S_1 \xrightarrow{a} S'\|S_1} \qquad (\|_2)\ \frac{S \xrightarrow{c?m} S' \quad S_1 \xrightarrow{c!m} S'_1}{S\|S_1 \xrightarrow{\tau_{c,m}} S'\|S'_1} \qquad (\|_\chi)\ \frac{S \xrightarrow{\tau_{c,m}} S' \quad S_1 \xrightarrow{\chi_{c,m}} S'_1}{S\|S_1 \xrightarrow{\tau_{c,m}} S'\|S'_1}$$

$$(\backslash L_1)\ \frac{S \xrightarrow{a} S' \quad channel(a) \notin L}{S\backslash L \xrightarrow{a} S'\backslash L}$$

- Finally, the term A_ϕ represents a system which consists of a single sequential agent whose knowledge, i.e. the set of messages it has, is described by ϕ. The agent's knowledge increases as it receives (or eavesdrops) messages (see rules $(?, \chi)$ in Table 3), infers new messages from the messages it knows (see rules $\mathscr{D}_1$ and $\mathscr{D}_2$) and generates new random messages (see rule (gen)). Sometimes we omit to represent agent's knowledge when this can be easily inferred from the context.

Remark 2.1. Note that we do not allow constructs for modeling recursion, thus our systems may have only finite computations. Adding recursion is possible, but it would make undecidable the verification problem of the properties we are going to consider in this paper. It is also worth noticing that, using typed channels, we implicitly assume that the receiver of a message can recognize the structure of the received messages, even if it may not be able to retrieve the actual meaning of such messages.

Static constraints: We assume our terms to respect some well-formedness conditions that can be statically checked. In particular:

Bindings: The inference construct $[\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} x : T]A_1; A_2$ binds the variable x in A_1, whereas the variable x must not appear in A_2. The prefix constructs $c?x : T.A$, $\chi^x_{c,T}.A$, $gen^{x,i}_T.A$ bind the variable x in A. A sequential agent is said to be closed if every

variable is bound. We assume that each variable may be bound at most once. In the sequel, we only consider compound systems made of closed agents.

Agent's knowledge: For every sequential agent A_ϕ, we require that all the closed messages that appear in A belong to its knowledge ϕ. More formally, let *is_closed* be a function that given a message m returns $\{m\}$ if this is closed, $\emptyset$ otherwise. Then, let $M(A)$ be

$$M(c!e.A) = is\_closed(e) \cup M(A),$$

$$M(c?x:T.A) = M(\chi^x_{c,T}.A) = M(gen^{x,i}_T.A) = M(A),$$

$$M(A + A') = M(A) \cup M(A'),$$

$$M([m = m']A; A') = is\_closed(m) \cup is\_closed(m') \cup M(A) \cup M(A'),$$

$$M([\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} x:T]A; A') = \left(\bigcup_{i\in I} is\_closed(m_i)\right) \cup M(A) \cup M(A').$$

So, for every sequential agent A_ϕ, we require $M(A) \subseteq \phi$.

Random message generation: We partition each set of basic values T^k, $k \in \{1,\ldots,n\}$, into two disjoint sets, the initial values IT^k (finite) and the random values RT^k (infinite and countable). Thus, we distinguish between the values of type T^k which may initially be known by the agents, i.e. IT^k, and the ones which are guessed later on, i.e. RT^k. The construct $gen^{x,i}_{T^k}$ allows to guess a random value of a basic-type T^k, i.e. a message in RT^k. (Random messages of structured types can be built using random values of basic types as sub-components.) Since these values must be randomly guessed, two different agents should not be able to guess the same value and each guessed value should be different from the previous ones. To formally model this, we consider a set of *injective* functions $\mathscr{R}^{T^k} : \mathbb{N} \mapsto \mathsf{RT}^k$, with T^k basic type, supplying the values guessed by the agents. We only consider compound systems s.t. the prefix constructs $gen^{x,i}_T$ appear only once, given i and T. Moreover, if $gen^{x,i}_T$ occurs in an agent, and $\mathscr{R}^{T^k}(i) = g$, then g must not occur in any knowledge of the agents of the system. The set of random values of a basic type which belong to a system S, namely $rand(S)$, consists of the RT^i values contained in the knowledge of the sequential agents in S plus the messages g, so the prefix $gen^{x,i}_T$ belongs to an agent in S, with $g = \mathscr{R}^T(i)$.

Remark 2.2. Our assumptions on the random messages clearly limit the intruder capabilities of attacking a protocol in our model. For example, let the intruder know an encrypted message. Then, in principle, it could try to decrypt it by iteratively guessing every possible decryption key. This is called a *brute force* attack. In our model, each guessed message is always different from those previously guessed and also from the initial ones. Hence, the guessed decryption keys will always be different from the key used to encrypt the message and so the intruder will not be able to decrypt the message. In a real programming system, since the set of possible keys is usually bounded, the *brute force* attack would lead the intruder to guess the correct decryption key and therefore the secret message. However, the set of possible keys is usually large enough

to make similar attacks computationally infeasible. Our model tries to reflect this situation. (This form of abstraction is often assumed when applying formal methods to security protocol analysis.)

2.2.1. Operational semantics and auxiliary notions

Here, we give the operational semantics of Crypto-CCS and some auxiliary definitions.

The activities of the agents of the systems are described by the actions they can perform. The set Act of actions which may be performed by a compound system is defined as: $Act = \{c?m, c!m, \chi_{c,m}, \tau_{c,m} \mid c \in C,\ m \in Msgs,\ m \text{ closed}\} \cup \{\tau_g \mid g \in \mathsf{RT}^i\}$. Below, we give a definition of the function *channel* that, given an action, returns a channel (*void* if the channel is not specified), and *message* that, given an action, returns its message.

	$c!m$	$c?m$	$\tau_{c,m}$	$\chi_{c,m}$	τ_g
Channel	c	c	*void*	c	*void*
Message	$\{m\}$	$\{m\}$	$\{m\}$	$\{m\}$	$\{g\}$

We denote a sequence of actions with Greek letters (the empty sequence is ε). The function $msgs$, defined as $msgs(a\gamma) = message(a) \cup msgs(\gamma)$ and $msgs(\varepsilon) = \emptyset$, returns the set of communicated messages in a sequence of actions. As usual, we use a Labeled Transition System (LTS) to assign semantics to our calculus. The semantics of *closed* terms is given by the least set of action relations induced by the rules shown in Table 3.

Let $Sort(S)$ be the set of channels that occur in S. This set represents the channels where agents of S may communicate. We consider $S \doteq Nil$ when $\forall a \in Act : S \overset{a}{\not\to}$ (i.e. when there is no system S' s.t. $S \overset{a}{\to} S'$). Let $SubM(m)$ be the set of subterms of m and $SubM(\phi)$ be $\bigcup_{m\in\phi} SubM(m)$. We say that a message m is initial, i.e. composed only of initial values, if $SubM(m) \cap \mathsf{RT}^k = \emptyset$ for each $k \in \{1, \ldots, n\}$.

For notational convenience, we sometimes use $S\|_L X$ instead of $(S\|X)\backslash L$. As a notation we also use $S \overset{\gamma}{\mapsto} S'$ if γ is a finite sequence of actions a_i, $1 \leqslant i \leqslant n$ s.t. $S = S_0 \overset{a_1}{\to} \cdots \overset{a_n}{\to} S_n = S'$.

Note 1. *If a sequential agent sends out a message during a computation, then this message can be deduced by the agent's knowledge, i.e. if* $X_\phi \overset{\gamma}{\mapsto} X'_{\phi'} \overset{c!m}{\to} X''_{\phi''}$ *then* $m \in \mathscr{D}(\phi')$. *Moreover, the static constraints on systems are preserved after performing activities, i.e. if* S *satisfies our constraints and* $S \overset{a}{\to} S'$ *then also* S' *satisfies our constraints. In the following, we also assume that systems in their initial configuration know only initial messages.*

Example 2.2. Consider a simple case of public key encryption, and apply as inference system the one in Table 1. For each user U, there is a pair of keys, i.e. $(PK(U), PK(U)^{-1})$, so that $PK(U)$ (the public key of U) is known by every one in the system and $PK(U)^{-1}$ (the private key of U) is only known by the user U. The public key

is used to encrypt messages. Since this is public, everyone can do it. The private key is used to decrypt messages. Since only U is assumed to know this key, even if an agent could eavesdrop the communication, it could not retrieve the message m. We thus model a situation where an agent A tries to send B a message m encrypted with the B public key.

The sender A_{ϕ_A}, with $\phi_A = \{m_A : T, PK(B) : Key\}$, is the following:

$$\begin{array}{ll} [m_A \ PK(B) \vdash_4 x : E(Key, T)] & \{\text{encrypts } m_A \text{ with } PK(B)\} \\ (c_{AB}!x.A_1) & \{\text{sends it}\} \\ ; \mathbf{0}. & \end{array}$$

The receiver B_{ϕ_B}, with $\phi_B = \{PK(B)^{-1} : Key^{-1}\}$, is the following:

$$\begin{array}{ll} c_{AB}?y : E(Key, T). & \{\text{receives } y\} \\ [y \quad PK(B)^{-1} \vdash_5 z : T] & \{\text{tries to decrypt it}\} \\ B_1 & \{\text{successful}\} \\ ; \mathbf{0} & \{\text{failed}\}, \end{array}$$

where A_1 and B_1 stand for the possible continuations after the protocol of A and B, respectively. Finally, the description of the compound system is $Sys = A_{\phi_A} \| B_{\phi_B}$. Briefly, the agent A builds the encrypted message, then sends it on channel c_{AB}. The agent B receives an encrypted message from channel c_{AB} and then tries to decrypt it using the inverse (private) key $PK(B)^{-1}$. If the protocol has been run properly, the variable z of B should contain m_A.

3. A logical language for the description of protocol properties

We illustrate a logical language ($\mathscr{L}_K$) for the specification of the functional and security properties of a compound system. We have extended a normal multimodal logic (e.g., see [46]) with operators which make it possible to specify whether a message belongs to an agent's knowledge after a computation γ performed by the whole system, starting from a fixed initial knowledge. The syntax of the logical language $\mathscr{L}_K$ is defined by the following grammar:

$$F ::= \mathbf{T} \mid \mathbf{F} \mid \langle a \rangle F \mid [a]F \mid \bigwedge_{i \in I} F_i \mid \bigvee_{i \in I} F_i \mid m \in K^{\phi}_{X,\gamma} \mid \exists \gamma : m \in K^{\phi}_{X,\gamma},$$

where $a \in Act$, m is a *closed* message, X is an agent identifier, I is an index set and ϕ a finite set of *closed* typed messages. The language without $m \in K^{\phi}_{X,\gamma}$ and $\exists \gamma : m \in K^{\phi}_{X,\gamma}$ ("knowledge" operators) is called $\mathscr{L}$.

Informally, $\mathbf{T}$ and $\mathbf{F}$ are the true and false logical constants; 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; $\bigvee_{i \in I}$ ($\bigwedge_{i \in I}$) represents the logical disjunction (conjunction). As usual, we consider $\bigvee_{i \in \emptyset}$ ($\bigwedge_{i \in \emptyset}$) as $\mathbf{F}(\mathbf{T})$. A system S satisfies a formula $m \in K^{\phi}_{X,\gamma}$ if S can perform a

Table 4
Semantics of the logical language

$S \models \mathbf{T}$,	for every process S	
$S \models \mathbf{F}$,	for no process S	
$S \models \bigwedge_{i\in I} F_i$	iff	$\forall i \in I : S \models F_i$
$S \models \bigvee_{i\in I} F_i$	iff	$\exists i \in I : S \models F_i$
$S \models \langle a\rangle F$	iff	$\exists S' : S \xrightarrow{a} S'$ and $S' \models F$
$S \models [a]F$	iff	$\forall S' : S \xrightarrow{a} S'$ implies $S' \models F$
$S \models m \in K^{\phi}_{X,\gamma}$	iff	$\exists S' : (S \overset{\gamma}{\mapsto} S') \downarrow_X = \gamma'$ and
		$m : T \in \mathscr{D}(\phi \cup msgs(\gamma'))$
$S \models \exists\gamma : m \in K^{\phi}_{X,\gamma}$	iff	$\exists\gamma : S \models m \in K^{\phi}_{X,\gamma}$

computation γ of actions and an agent of S, identified by X, can infer the message m starting from the set of messages ϕ plus the messages it has come to know during the computation γ. The formula $\exists\gamma : m \in K^{\phi}_{X,\gamma}$ is satisfied by a system S if there exists a computation γ and an agent X of S s.t. X_ϕ can infer m during the computation γ.

We assume that a unique identifier can be assigned to every sequential agent in a compound system (e.g., the path from the root to the sequential agent term in the parsing tree of the compound system term). Then, given a sequence of transitions $S \overset{\gamma}{\mapsto} S'$ of a compound term S, let $(S \overset{\gamma}{\mapsto} S') \downarrow_X$ be the sequence of actions of the agent identified by X in S, that have contributed to the transitions of the whole system.[3] Finally, the formal semantics of a formula $F \in \mathscr{L}_K$ w.r.t. a compound system S is inductively defined in Table 4

We can establish a decidability result for the sub-logic which consists only of the logical constants, disjunctions and the possibility modality.

Lemma 3.1. *Consider a formula $F \in \mathscr{L}$, which consists only of the logical constants, disjunctions and the possibility modality, and a finite set of closed messages ϕ. It is decidable if there exists a sequential agent X s.t. $X_\phi \models F$.*

[3] For simplification, here we leave out the technical details. We can however achieve this result by suitably adding information on the transitions, e.g., see [10].

3.1. Verification problems for security protocols

The formula $\exists\gamma : m \in K^{\phi}_{X,\gamma}$ plays a central role in the analysis of security protocols, because it makes it possible to express secrecy properties. Indeed, it may be used to check whether the agent X can discover a certain message or not. We are mainly interested in the study of properties like:

No agent (intruder), communicating with the agents of the system S, can retrieve a secret that should only be shared by some agents of S,

or dually

There exists an agent (intruder) that, communicating with the agents of the system S, can retrieve a secret that should only be shared by some agents of S.

In our model, the latter property can be formally restated as follows:

$$\exists X_{\phi} \quad \text{s.t.} \quad S \| X_{\phi} \models \exists\gamma : m \in K^{\phi}_{X,\gamma}, \tag{4}$$

where m is the secret message. However, we are not interested in every computation γ of $S\|X$. Indeed, consider the following example where the system S consists of the single agent $\mathbf{0}_{\{m:T\}}$, whose unique secret is m. Then, due to our semantic modeling of receiving actions (see rule (?) in Table 3), it is possible for an intruder to obtain any secret messages. For instance, let Y be $c?x:T.\mathbf{0}$, then:

$$\mathbf{0}_{\{m:T\}} \| Y_{\emptyset} \overset{c?m:T}{\rightarrow} \mathbf{0}_{\{m:T\}} \| \mathbf{0}_{\{m:T\}}.$$

Thus, Y discovers m after performing the receiving action $c?m:T$. This means that S satisfies property (4), i.e. it is not secure, which contradicts our intuition. However, note that the receiving action $c?m:T$ simply models the potential communication capability of Y (and thus of the system $S\|Y$) with an external (omniscient) environment. But, when we fix X_{ϕ}, we want to consider $S\|X_{\phi}$ as a *closed* system and study only its internal communications, i.e. the actual communications between the system S and its "hostile" environment represented by X. We thus consider the following formulation:

$$\exists X_{\phi} \quad \text{s.t.} \quad (S\|X_{\phi})\backslash L \models \exists\gamma : m \in K^{\phi}_{X,\gamma}, \tag{5}$$

where L is the set of channels on which S and X can communicate, i.e. $Sort(S\|X)$. Note that all the possible computations of $(S\|X_{\phi})\backslash L$ are actually the internal computations of $S\|X$. In the context $(S\|(\_))\backslash L$ the intruder X is forced to communicate only with the system S. In our working example, i.e. $S\|Y_{\emptyset}$, no internal action is possible because S does not perform any output which may synchronize with the receiving actions of Y. Thus, the system $(S\|Y_{\emptyset})\backslash L$ does nothing.[4] So, as the intuition suggests, S does not satisfy property (5). Note that if we consider the system $S' = c!m.\mathbf{0}_{\{m:T\}}$,

[4] Actually, when considering a generic agent X, we also need to take into account the guessing actions. However, by construction, guessed values will always be different from m.

whose unique secret is m, then

$$S' \| Y_{\emptyset} \stackrel{\tau_{c,m}}{\rightarrow} \mathbf{0}_{\{m:T\}} \| \mathbf{0}_{\{m:T\}},$$

which clearly shows that S' does satisfy (5) as the intuition suggests (because S' is making public its secret). For the previous considerations we make the following assumption.

Remark 3.1. We analyze the context $(S\|(\_))\backslash L$ ($S\|_L(\_)$ by using the short notation) when we want to study the secrecy properties for a system S against an intruder (or a malicious) agent X, with $Sort(X) \subseteq L$.

We usually assume the intruders have some initial knowledge. This will depend on the system under investigation.

Example 3.1. Consider the protocol in the Example 2.2 and look for an intruder X_{ϕ_X}, if any, who can actively interact along the communication channel c_{AB} and discover the secret message m_A, i.e.

$$(A_{\phi_A} \| B_{\phi_B} \| X_{\phi_X}) \backslash \{c_{AB}\} \models \exists \gamma : m_A \in K_{X,\gamma}^{\phi_X}.$$

If we assume that $m_A \notin \mathscr{D}(\phi_X)$ and $PK(B)^{-1} \notin \mathscr{D}(\phi_X)$, then, for every X which can only communicate via c_{AB}, the previous statement does not hold. So, no intruder will be able to know the exchanged message m_A under the above hypothesis. In the sequel, we show how to prove this fact.

Formulation (5) models a scenario where a system is attacked by an external agent. But, we need also to consider more subtle situations, e.g., where a legitimate user of the system is willing to act maliciously against the other legitimate participants. Consider a system that consists of n sequential agents $A_{i_{\phi_i}}$, with $i \in \{1,\ldots,n\}$. To check whether the agent A_i may apply a successful strategy in order to obtain a secret m of some other participants, say A_j with $i \neq j$, we can study if

$$\exists X_{\phi_i} \quad \text{s.t.} \quad (\|_{l \in \{1,\ldots,n\} \backslash \{i\}} A_{l_{\phi_l}} \| X_{\phi_i}) \backslash L \models \exists \gamma : m \in K_{X,\gamma}^{\phi_i}. \tag{6}$$

We can also imagine more complex situations where a system is under attack by an external intruder and by a group of internal agents, i.e. we can study if $\exists X_{\phi_i} \exists Y_{\phi_{i_1}}^{i_1} \ldots \exists Y_{\phi_{i_k}}^{i_k}$ s.t.

$$(\|_{l \in \{1,\ldots,n\} \backslash IA \backslash \{i\}} A_{l_{\phi_l}} \|_{l' \in IA} Y_{\phi_{l'}}^{l'} \| X_{\phi_i}) \backslash L \models \begin{array}{l} \exists \gamma : m \in K_{X,\gamma}^{\phi_i} \\ \bigvee_{l' \in IA} \exists \gamma : m \in K_{Y^{l'},\gamma}^{\phi_{l'}} \end{array}, \tag{7}$$

where $IA = \{i_1,\ldots,i_k\}$ and $i \notin IA$.

Note that we only deal with secrecy properties; however, several other security properties can be encoded as secrecy ones, e.g., for the authentication properties see [30,33].

4. Analysis method

4.1. Intuition

In the previous sections, we set a framework for defining security properties (so far secrecy only) for cryptographic protocols as properties of open systems. Now, we want to define suitable techniques which may help us to solve the verification problem. In particular, we want to follow and extend the analysis approach for open systems developed in [34], which is based on partial model checking techniques (e.g., see [5,25]).

The central idea is to turn the problem of verifying an open system into a validity (satisfiability) checking problem of a certain logic. To achieve this, we use the partial model checking techniques proposed within the concurrency theory for compositional analysis of systems: suppose we have a compound system $S_1 \| S_2$ that has to satisfy a certain property F. Using partial model checking techniques, we can reduce this verification problem to checking whether either component, say S_2, satisfies a new property $F/\!/S_1$ (see Section 4.3 for its formal definition), which is obtained by "evaluating" the behavior of the other component S_1 (hence the name "partial evaluation").

Observing the compositional analysis proposed in [5,25], it can be noted that it is semantic driven. Likewise, our partial evaluation functions derive from the inspection of the operational rules of calculus. Below we try to give an insight in the way partial model checking (or partial evaluation) works and the main difficulties we encounter in this specific field for the analysis of cryptographic protocols.

Possibility formula: Consider the system $S \doteq c?x : T.\mathbf{0}$ and suppose we want to determinate the necessary and sufficient condition on a sequential agent X to obtain

$$S \| X \models \langle \tau_{c,m'} \rangle \mathbf{T}.$$

This statement means that the system $S \| X$ performs an internal communication of the message m' on the channel c. Recall that $S \| X$ may perform the $\tau_{c,m'}$ action (according to the operational semantics in Table 3) *if and only if* either situation holds (the rule to infer the transition is in brackets):

(1) (Rule $\|_1$) S performs $\tau_{c,m'}$ in isolation.
(2) (Rule $\|_1$ symm.) X performs $\tau_{c,m'}$ in isolation.
(3) $S \| X$ performs a form of synchronization, i.e.
- (a) (Rule $\|_2$) S performs a reading action and X the complementary sending action.
- (b) (Rule $\|_2$ symm.) S performs a sending action and X the complementary reading action.
- (c) (Rule $\|_\chi$) S performs an internal communication action and X an eavesdropping action.
- (d) (Rule $\|_\chi$ symm.) S performs an eavesdropping action and X an internal communication action.

Cases 1, 3b, 3c and 3d are not possible since S only performs reading actions. Furthermore, case 2 is not possible either since X is a sequential agent and does not perform

Table 5
Partial evaluation function for $S\|(\_)$ and $\mathscr{L}$

$$
\begin{aligned}
\mathbf{T}/\!/S &\doteq \mathbf{T}\\
\mathbf{F}/\!/S &\doteq \mathbf{F}\\
(\langle a\rangle F)/\!/S &\doteq \langle a\rangle(F/\!/S) \vee \bigvee_{S \xrightarrow{a} S'} F/\!/S' \qquad (a \neq \tau_{c,m}, \tau_g)\\
(\langle \tau_{c,m}\rangle F)/\!/S &\doteq \bigvee_{S \xrightarrow{c!m} S'} \langle c?m\rangle(F/\!/S') \vee \bigvee_{S \xrightarrow{c?m} S'} \langle c!m\rangle(F/\!/S') \vee\\
&\quad \bigvee_{S \xrightarrow{\tau_{c,m}} S'} \langle \chi_{c,m}\rangle F/\!/S' \vee \bigvee_{S \xrightarrow{\tau_{c,m}} S'} F/\!/S'\\
(\langle \tau_g\rangle F)/\!/S &\doteq \begin{cases} \langle \tau_g\rangle(F/\!/S) & g \notin rand(S)\\ \bigvee_{S \xrightarrow{\tau_g} S'} F/\!/S' & \text{otherwise} \end{cases}\\
([a]F)/\!/S &\doteq [a](F/\!/S) \wedge \bigwedge_{S \xrightarrow{a} S'} F/\!/S' \qquad (a \neq \tau_{c,m}, \tau_g)\\
([\tau_{c,m}]F)/\!/S &\doteq \bigwedge_{S \xrightarrow{c!m} S'} [c?m](F/\!/S') \wedge \bigwedge_{S \xrightarrow{c?m} S'} [c!m](F/\!/S') \wedge\\
&\quad \bigwedge_{S \xrightarrow{\tau_{c,m}} S'} [\chi_{c,m}]F/\!/S' \wedge \bigwedge_{S \xrightarrow{\tau_{c,m}} S'} F/\!/S'\\
([\tau_g]F)/\!/S &\doteq \begin{cases} [\tau_g](F/\!/S) & g \notin rand(S)\\ \bigwedge_{S \xrightarrow{\tau_g} S'} F/\!/S' & \text{otherwise} \end{cases}\\
\bigvee_{i\in I} F_i/\!/S &\doteq \bigvee_{i\in I}(F_i/\!/S)\\
\bigwedge_{i\in I} F_i/\!/S &\doteq \bigwedge_{i\in I}(F_i/\!/S)
\end{aligned}
$$

internal actions. To summarize, X can only perform the corresponding sending action, i.e. $X \xrightarrow{c!m'}$, which is expressed by the formula

$$X \models \langle c!m'\rangle \mathbf{T}.$$

Thus $\langle \tau_{c,m'}\rangle \mathbf{T}/\!/S$ is $\langle c!m'\rangle \mathbf{T}$. The *necessity* modality is accordingly treated.

Disjunction: The reduction of a formula which is either a conjunction or a disjunction can be separately achieved for each operand, and the resulting formulas connected by the corresponding logical operator, i.e. conjunction or disjunction. E.g., for the disjunction operator, we have:

$$
\begin{aligned}
&S\|X \models F_1 \vee F_2 && \text{iff}\\
&S\|X \models F_1 \text{ or } S\|X \models F_2 && \text{iff}\\
&X \models F_1/\!/S \text{ or } X \models F_2/\!/S && \text{iff}\\
&X \models F_1/\!/S \vee F_2/\!/S.
\end{aligned}
$$

Note that, at least for the sub-language $\mathscr{L}$, the construction of the formula eventually comes to an end (see Table 5): as a matter of fact, each time a reduction is performed, either the system S performs a computation step or the formula to be analyzed has a smaller size than the previous one. It must be borne in mind that we consider systems which can only perform finite sequences of actions.

Knowledge formula: The formula which deals with secrecy, i.e. $\exists\gamma : m \in K^{\phi}_{X,\gamma}$, is more complex and needs more work. As noted in the Remark 3.1, the context $(S\|(\_))\backslash L$ ($S\|_L(\_)$ for short) is more suitable than $S\|(\_)$ for secrecy analysis. Thus, we only perform the partial model checking of the formula $\exists\gamma : m \in K^{\phi}_{X,\gamma}$ in contexts like $S\|_L(\_)$.

To understand how we can perform the partial model checking, note that

$$S\|_L X \models \exists\gamma : m \in K^{\phi}_{X,\gamma} \tag{8}$$

means $S\|_L X$ performs a computation γ s.t. X can infer $m : T$ from the messages discovered during this computation and its initial knowledge ϕ. We have two different situations, depending on the length of γ:

(1) γ is empty;
(2) γ is equal to $a\gamma'$ for some action a and computation γ' and so $S\|_L X$ performs an action a by reaching a configuration $S'\|_L X'$. In turn, there is a computation γ' from $S'\|_L X'$ s.t. X may infer m.

More formally, statement (8) is equivalent to

$$S\|_L X \models m \in K^{\phi}_{X,\varepsilon}$$

$$\bigvee_{(a,\phi')\in AK} \langle a\rangle(\exists\gamma' : m \in K^{\phi'}_{X,\gamma'}) \tag{9}$$

where AK is $\{(a,\phi') \mid \exists S', X', \phi' \text{ s.t. } S\|_L X_\phi \xrightarrow{a} S'\|_L X'_{\phi'}\}$. Each pair in AK represents an action that $S\|_L X_\phi$ can perform and the resulting knowledge of X. The logical formula in (9) may be considered as an unwinding step of the semantic definition of $\exists\gamma : m \in K^{\phi}_{X,\gamma}$ w.r.t. the system S.

Now, note that $m \in K^{\phi}_{X,\varepsilon}$ is equivalent to **T** or **F**, depending on $m : T \in \mathscr{D}(\phi)$ or not. Thus, the first disjunct in the formula in (9) is **T** or **F**. Next, we are going to prove that, under some assumptions, it is possible to finitely iterate the unwinding step for the formula $\exists\gamma' : m \in K^{\phi'}_{X,\gamma'}$ until we obtain a formula which only consists of disjunctions, possibility formulas and the logical constants **T** and **F**. We can eventually apply partial model checking techniques to these simple logical operators and checking the formula $\exists\gamma : m \in K^{\phi}_{X,\gamma}$ would be possible. So, it is worthy using formulation (9) instead of (8).

To illustrate the problems, we encounter during the unwinding phase, consider again the set AK. This can be represented by the combination of the following three sets:

(1) $AK_I = \{(a,\phi') \mid \exists S', X', \phi' \text{ s.t. } S\|_L X_\phi \xrightarrow{a} S'\|_L X'_{\phi'} \text{ by rules } \|_2, \|_\chi \text{ or symm.}\}$. This set represents the possible interactions between S and X.
(2) $AK_S = \{(a,\phi) \mid \exists S' \text{ s.t. } S\|_L X_\phi \xrightarrow{a} S'\|_L X_\phi \text{ by rule } \|_1\}$. This set represents the actions performed by S only.
(3) $AK_X = \{(a,\phi') \mid \exists X', \phi' \text{ s.t. } S\|_L X_\phi \xrightarrow{a} S\|_L X'_{\phi'} \text{ by rule } \|_1 \text{ symm.}\}$. This set represents the actions performed by X only.

Consider a pair (a,ϕ') in the first set. Then, if

$$S\|_L X \models \langle a\rangle(\exists\gamma' : m \in K^{\phi'}_{X,\gamma'} n)$$

it must be

$$\exists S', X', \phi' : S\|_L X_\phi \xrightarrow{a} S'\|_L X'_{\phi'} \quad \text{and} \quad S'\|_L X'_{\phi'} \models \exists \gamma' : m \in K^{\phi'}_{X,\gamma'}.$$

We can again unwind the formula $\exists \gamma' : m \in K^{\phi'}_{X,\gamma'}$, now with respect to S'. Note that the system S' may perform less steps than S. Thus, if we could limit ourselves only to the pairs in AK_I, then our unwinding phase would eventually terminate. The same reasoning applies to pairs in AK_S. However, if we also consider the pairs in AK_X, which correspond to the situation where only X acts, then this reduction strategy is not guaranteed to finish. Indeed, the size of the system S will not decrease. Nevertheless, in the context $S\|_L(\_)$, the agent X can only guess random messages without interacting with S. Thus, since we consider only systems with finite behavior, we have the following observation.

Observation 4.1. *If we could fix an upper bound for the number of generation actions that are performed by the agent X before any interaction with S, then the unwinding process would eventually terminate.*

In the next subsection, we will identify a class of sequential agents, namely the *well-behaved* ones, fulfilling the requirement stated in the previous observation. The partial model checking of the knowledge formula $\exists \gamma : m \in K^{\phi}_{X,\gamma}$ can be solved when we consider only well behaved agents. Moreover, we will make some general assumptions on the inference system. These allow us to safely stick to the class of well-behaved agents as attackers when analyzing the security of a system. Indeed, we will prove that, if a generic sequential agent is able to successfully attack the system, then a well-behaved agent can do that too.

4.2. Well-behaved agents

Roughly speaking, an agent is *well behaved* if it only performs *well-behaved* computations, where a computation is such if and only if, whenever its first action is the guessing of a random value, then this value occurs in a message successively sent and between these two events there are only guessing actions. We thus require that the intruder actually uses its random generated messages in a sending action. First, we formalize the concept of *well-behaved* agent.

Definition 4.1. A sequential agent X_ϕ is *well behaved* if, during its execution, it only performs well-behaved computations, where a computation γ is *well behaved iff* whenever $\gamma = \tau_{g_1}\gamma'$ then $\gamma' = \tau_{g_2} \ldots \tau_{g_n} c!m'\gamma''$, with $n \geqslant 1$, and $\{g_i\}_{i\in\{1,\ldots,n\}} \subseteq SubM(m')$.

The set of basic types that can appear in an agent is finite. Moreover, for each type, the set of different random messages that can appear as submessages in a message is limited. Thus, we can find an upper bound for the number of random generation actions that may appear in any computation of a well-behaved agent. By observation (4.1), the partial model checking phase eventually terminates when the unknown component is a well behaved agent.

We make two assumptions on the deduction function. Using these assumptions, we can prove that it is sufficient to consider only *well-behaved* agents as intruders in the system. Indeed, if an intruder attacks the system, then also a well behaved one can perform a similar attack. The two assumptions are the following.

Assumption 4.1. Given a message m, suppose $g:T^i$ is a random value not occurring either in m or in any of the messages in ϕ. Then, we have $m \in \mathscr{D}(\phi \cup \{g:T^i\})$ iff $m \in \mathscr{D}(\phi)$.

Consider the deduction function $\mathscr{D}$ which has only the following rule schema:

$$\frac{y:T \quad x:T'}{F(y):F(T)},$$

where F is a type constructor. Assume also to have a basic message $m:T$ and a random value $g':T'$. We can see that $\mathscr{D}$ does not enjoy the previous assumption. Indeed, we have that $F(m):F(T) \in \mathscr{D}(\{m:T, g':T'\})$ but $F(m):F(T) \notin \mathscr{D}(\{m:T\})$, whereas $g':T'$ does not occur both in $m:T$ and in $F(m):F(T)$.

Note, however, that there is no relationship between the message $F(y):F(T)$ and $x:T'$, indeed any type T' message is just sufficient to deduce $F(y):F(T)$. Perhaps, this rule can only be used to test the presence of any type T' message in the intruder's knowledge. However, since an intruder can always infer a random-type T' message, this rule seems useless. Inference systems commonly used in literature do not have any similar rule. According to the next assumption, basic-type messages cannot be forged.

Assumption 4.2. If m is a message and $m \in \mathscr{D}(\phi)$, then every basic-type value of m must be a submessage of some message in ϕ.

For example, this assumption prevents the deduction function by producing random values which are not within the set ϕ, i.e. which are not guessed using the appropriate construct *gen*. For example, the following rule does not fulfill this assumption:

$$\frac{m:T}{E(g,m):E(Key,T)},$$

where g is a value of a basic type. Note that this assumption implies that new basic-type messages can only be obtained by performing a guessing action.

Finally, under the previous assumptions, we can prove the following lemma which tells that we can limit ourselves to consider well-behaved sequential agent only in the analysis of formulas on the form $\exists\gamma : m \in K^{\phi}_{X,\gamma}$.

Lemma 4.1. *Given a system S and an initial message m, suppose that there exists a sequential agent X_ϕ s.t. $Sort(S\|X) \subseteq L$ and $S\|_L X_\phi \models \exists\gamma : m \in K^{\phi}_{X,\gamma}$. Then, a well-behaved agent X'_ϕ exists s.t. $S\|_L X'_\phi \models \exists\gamma : m \in K^{\phi}_{X',\gamma}$.*

Table 6
Partial evaluation function for $(\_)\backslash L$ and $\mathscr{L}$

$$
\begin{aligned}
\mathbf{T}/\!/_L &\doteq \mathbf{T}\\
\mathbf{F}/\!/_L &\doteq \mathbf{F}\\
(\langle a\rangle F)/\!/_L &\doteq \begin{cases} \langle a\rangle(F/\!/_L) & channel(a)\notin L,\\ \mathbf{F} & \text{otherwise.}\end{cases}\\
([a]F)/\!/_L &\doteq \begin{cases} [a](F/\!/_L) & channel(a)\notin L,\\ \mathbf{T} & \text{otherwise.}\end{cases}\\
\bigvee_{i\in I} F_i/\!/_L &\doteq \bigvee_{i\in I}(F_i/\!/_L)\\
\bigwedge_{i\in I} F_i/\!/_L &\doteq \bigwedge_{i\in I}(F_i/\!/_L)
\end{aligned}
$$

4.3. Partial evaluation functions

Finally, we present the partial evaluation functions. For the logical language $\mathscr{L}$, that has no knowledge operators, the partial evaluation function for the context $S\|X$ is given in Table 5. The main difference w.r.t. the work in [5] is about the constraints on the generation actions. Indeed, when considering the case about the generation of an action τ_g, we cannot let the intruder guess a random message which belongs to S, i.e. it is in $rand(S)$. The next statement is about the correctness of the partial evaluation function.

Proposition 4.1. *Given a system S, a sequential agent X and a logical formula $F\in\mathscr{L}$ then*

$$S\|X\models F \quad \textit{iff} \quad X\models F/\!/S.$$

Sometimes, it is also useful to consider the context $(\_)\backslash L$, which consists only of the restriction operator. The partial evaluation function $/\!/_L$ for such context w.r.t. the formulas in $\mathscr{L}$ is given in Table 6. The next proposition states its correctness.

Proposition 4.2. *Given a system S', a set of channels L and a logical formula $F\in\mathscr{L}$, we have*

$$(S')\backslash L\models F \quad \textit{iff} \quad S'\models F/\!/_L.$$

Example 4.1. Consider the context $(S\|(\_))\backslash\{c\}$, where $S=c?x:T.\mathbf{0}$. Then, by applying Propositions 4.1 and 4.2, we have

$$
\begin{aligned}
&(S\|(X))\backslash\{c\}\models\langle c?m\rangle\mathbf{T}\vee\langle\tau_{c,m}\rangle\mathbf{T} && \text{iff Proposition 4.2,}\\
&S\|X\models(\langle c?m\rangle\mathbf{T}\vee\langle\tau_{c,m}\rangle\mathbf{T})/\!/_{\{c\}} && \text{iff Proposition 4.1,}\\
&X\models((\langle c?m\rangle\mathbf{T}\vee\langle\tau_{c,m}\rangle\mathbf{T})/\!/_{\{c\}})/\!/S.
\end{aligned}
$$

Note that $(\langle c?m\rangle\mathbf{T}\vee\langle\tau_{c,m}\rangle\mathbf{T})/\!/_{\{c\}}=\langle\tau_{c,m}\rangle\mathbf{T}$ and $\langle\tau_{c,m}\rangle\mathbf{T}/\!/S=\langle c!m\rangle\mathbf{T}$.

Table 7
Partial evaluation function for $S\|_L X$, with $Sort(S\|X) \subseteq L$, and $\exists\gamma : m \in K^{\phi}_{X,\gamma}$

$$
\begin{aligned}
&\exists\gamma : m \in K^{\phi}_{X,\gamma}/\!/S \doteq \\
&\bigvee_{(c,m',S')\in Send(S)} \langle c!m'\rangle(\exists\gamma : m \in K^{\phi}_{X,\gamma}/\!/S') && \text{(sending)} \vee \\
&\bigvee_{(c,m',\langle\langle g_i\rangle\rangle_{i\in I},S')\in RSend(S)} \langle\langle \tau_{g_i}\rangle\rangle_{i\in I}\langle c!m'\rangle \\
&\qquad (\exists\gamma : m \in K^{\phi\cup\langle\langle g_i\rangle\rangle_{i\in I}}_{X,\gamma}/\!/S') && \text{(guessing and sending)} \vee \\
&\bigvee_{S\xrightarrow{c!m'} S'} \langle c?m'\rangle(\exists\gamma : m \in K^{\phi\cup\{m'\}}_{X,\gamma}/\!/S') && \text{(receiving)} \vee \\
&\bigvee_{S\xrightarrow{\tau_{c,m'}} S'} \langle \chi_{c,m'}\rangle(\exists\gamma : m \in K^{\phi\cup\{m'\}}_{X,\gamma}/\!/S') && \text{(eaves-dropping)} \vee \\
&\bigvee_{S\xrightarrow{a} S'(a=\tau_{c,m},\tau_g)} \exists\gamma : m \in K^{\phi}_{\gamma}/\!/S' && \text{(idling)} \vee \\
&m \in K^{\phi}_{X,\varepsilon}/\!/S && \text{(nothing to do)}
\end{aligned}
$$

$$
m \in K^{\phi}_{X,\varepsilon}/\!/S \doteq \exists\gamma : m \in K^{\phi}_{X,\gamma}/\!/Nil \doteq \begin{cases} \mathbf{T} & m \in \mathscr{D}(\phi) \\ \mathbf{F} & m \notin \mathscr{D}(\phi) \end{cases}
$$

where :

$$
\begin{aligned}
Send(S) &= \{(c,m',S') \mid S \xrightarrow{c?m'} S' \text{ and } m' \in \mathscr{D}(\phi)\} \\
RSend(S) &= \{(c,m',\langle g_1,\ldots,g_n\rangle,S') \mid \exists X_\phi \text{ s.t. } X_\phi \xrightarrow{\tau_{g_1},\ldots,\tau_{g_n}} X'_{\phi'}, S \xrightarrow{c?m'} S', \\
&\qquad m' \in \mathscr{D}(\phi'), \{g_i\}_{i\in\{1,\ldots,n\}} \subseteq SubM(m')\backslash(SubM(\phi)\cup rand(S))\}
\end{aligned}
$$

Now, we will focus on the treatment for $\exists\gamma : m \in K^{\phi}_{X,\gamma}$ since this is the one mostly involved in the analysis of security properties. We have already illustrated the idea: first, we unwind this formula in a disjunction of possibility formulas; next, we apply the partial model checking function to these simpler formulas. We give the partial evaluation function for $\exists\gamma : m \in K^{\phi'}_{X,\gamma}$ and $S\|_L X$, with X well behaved, in Table 7. We grouped together the formulas corresponding to the same behavior (in brackets) of the intruder X:

- *Sending*: This disjunction takes into account the sending actions of the intruder on which the system S is willing to synchronize. The intruder performs these actions only starting from its initial knowledge ϕ, without guessing any new random message. Indeed, $Send(S)$ (see Table 7) is a set of triples which represent the channel used, the message sent and the relative derivative of the system S. Each message must be inferable from the knowledge ϕ.
- *Guessing and sending*: Here the intruder performs a preliminary sequence of guessing actions before sending. Since we consider well-behaved agents, then these random values must occur in the message sent. Indeed, $RSend(S)$ (see Table 7) is a set of quadruples which represent the channel used, the message sent, the guessing sequence of the random values and the relative derivative of the system S. (Note that $Send(S)$ and $RSend(S)$ depend on the knowledge ϕ, although not explicitly mentioned.) In practice, the number of guessings of random messages is related to the receiving actions of the system.

- *Receiving*: This disjunction takes into account the messages that the intruder may receive from S. The messages received increase the intruder's knowledge ϕ.
- *Eavesdropping*: This disjunction takes into account the messages that the intruder may eavesdrop from S. The eavesdropped messages increase the intruder's knowledge ϕ.
- *Idling*: This disjunction takes into account the possible configurations that are reached by an action of the system only.
- *Nothing to do*: This takes into account the possibility that the intruder has already gathered enough information to deduce m.

It is worth noticing that these partial evaluation functions are only determined by the logic chosen to describe the security properties and by the operational semantics of the calculus. Thus, in our approach, we make no assumptions about the intruder's capabilities; the intruders are simply sequential agents of the system, with a certain initial knowledge.

The next statement is about the correctness of the partial evaluation, where we assume that X is a well-behaved sequential agent.

Proposition 4.3. *Given a system S and a well-behaved sequential agent X_ϕ where ϕ is finite and $Sort(S\|X) \subseteq L$, then if m is an initial message, we have*

$$S\|_L X_\phi \models \exists\gamma : m \in K^{\phi}_{X,\gamma} \quad \textit{iff} \quad X_\phi \models \exists\gamma : m \in K^{\phi}_{X,\gamma}//S.$$

Eventually, we reduced our verification problem to a satisfiability problem. In particular, the partial evaluation of the formula $\exists\gamma : m \in K^{\phi}_{X,\gamma}//S$ actually produces a formula in $\mathscr{L}$, i.e. which only consists of the logical constants, disjunctions and *possibility* formulas. Our next and last step is to provide a satisfiability procedure for the sublogic which only consists of the previous operators.

4.4. Main result

Note that the formula $F = \exists\gamma : m_1 \in K^{\phi}_{X,\gamma}//S$ shows several infinitary disjunctions, i.e. disjunctions whose index set is infinite. This makes it difficult to establish satisfiability results about such formulas. Indeed, the two index sets, i.e. $Send(S)$ and $RSend(S)$, might be infinite. In particular, $RSend(S)$ is always infinite since, by construction, we assume an infinite number of basic-type random values. Moreover, $Send(S)$ is also infinite whenever the set of messages of a certain-type T that can be inferred from ϕ is infinite and the system S is willing to receive on a channel of type T. Since we want to obtain a theory to be effectively mechanized, we add two further assumptions to our deduction function. These enable us to find a formula which has only disjunctions with finite indexes and which can be satisfied if and only if the formula obtained after the partial evaluation phase can be satisfied. Our assumptions seem not too restrictive: as a matter of fact, the inference systems commonly used in literature enjoy them (and also the one given in Table 2).

According to the following assumption, for each type T, a finite number of messages can be inferred from a finite set of messages.

Assumption 4.3. When ϕ is finite, $\mathscr{D}(\phi) \cap Tmsgs(T)$ is finite.

This ensures that $Send(S)$ is a finite set, when ϕ is finite. As a matter of fact, $Send(S)$ consists of triples of the form $\langle c, m, S' \rangle$, where c is a channel of S, m is a message that can be inferred from ϕ and S' is a derivative of S s.t. $S \stackrel{c?m}{\rightarrow} S'$. The set of channels of S is finite. According to the previous assumption, the set of messages of a certain-type T that can be inferred by the intruder is finite. Thus, only a finite number of different messages m may appear as a second element of the triples. Finally, this also ensures that only a finite number of different derivatives S' of S may appear as a third element.

The reason for the next assumption is to avoid an inference system that depends on particular random values. First, we need to introduce the concept of *type-preserving* bijection. We say that a bijection σ between random values of basic type is type preserving if whenever $\sigma(g : T) = g' : T'$ then $T' = T$. Type-preserving bijections naturally extend to bijections between actions, sequence of actions, sequential agents and compound systems in a straightforward way.

Assumption 4.4. If IS is an inference schema and σ a type-preserving bijection between random values, then: $m_1 : T_1 \ldots m_n : T_n \vdash_{IS} m : T$ *iff* $\sigma(m_1 : T_1) \ldots \sigma(m_n : T_n) \vdash_{IS} \sigma(m : T)$.

A schema that does not satisfy this assumption is the following:

$$\frac{g : T}{m : T'},$$

where $g : T, m : T'$ are closed messages and $g \notin SubM(m)$. Consider the inference system which only consists of the previous rule, and additionally, that $\sigma(g : T) = g' : T$ for some $g' \neq g$. Then, $m : T' \in \mathscr{D}(\{g : T\})$, but $\sigma(m : T') = m : T' \notin \mathscr{D}(\{\sigma(g : T)\}) = \mathscr{D}(\{g' : T\}) = \emptyset$.

In our context, anytime a random message is generated, it is brand-new; there are no relationships between new random messages and the others existing in the system. The latter two assumptions on the inference system ensure that, when analyzing the secrecy of an initial message, it is not necessary to look for a certain sequence of random messages, it is sufficient to generate a sequence of messages with the appropriate types. The following example may help.

Example 4.2. Consider an inference system having only one inference schema:

$$\frac{x : T}{Hash(x) : Hash(T)} (\vdash_{\text{hash}}),$$

which might model one-way functions, i.e. functions that are easily computed but whose inverses are computationally hard. The inference system leaves out a rule to infer x by knowing $Hash(x)$. Hence, the Hash function cannot be formally inverted.

Consider the system S which receives a value y on the channel c. Then, it communicates a secret message $m : T$ depending on $Hash(y) = Hash(m)$ or not. Thus,

we will have:

$$S = c?y:T.\\ \quad [y \vdash_{\text{hash}} z : Hash(T)]([z = Hash(m)]c!m.\mathbf{0};\mathbf{0})\\ \quad ;\mathbf{0},$$

where m is a confidential message of S. We analyze the context $S\|_L X$ where $L = \{c\}$. Let us assume the initial knowledge of the intruders X is empty. Thus, intruders can only infer a new message and send it to the system. So, consider the set:

$$RSend(S) = \{(c, g_1, \langle g_1\rangle, S') \mid \exists X_\phi \text{ s.t. } X_\phi \overset{\tau_{g_1}}{\mapsto} X', S \overset{c?g_1}{\to} S', g_1 \in \mathsf{RT}^T \backslash rand(S)\}.$$

This set is infinite, because there is an infinite number of messages in RT^T. However, each derivative S' in the quadruples of $RSend(S)$ has the following form:

$$[g \vdash_{\text{hash}} z : Hash(T)][z = Hash(m)]c!m.\mathbf{0};\mathbf{0},$$

where g is a random value which differs from any initial value of S. Thus, $Hash(g)$ will always be different from $Hash(m)$ and so m will be never sent on the channel c. The results of the analysis do not depend on a specific random value. We can simply choose one of them, representative of all the others.

Note that the computations of a system S do not significantly differ when S receives a random message instead of another one, provided these are both newly created. The only difference is the "renaming" of the two random values in the messages exchanged during the computation. So, if we are interested in studying the secrecy of an initial value m, which does not contain any random message, then we can safely limit ourselves to consider only a subset of the quadruples existing in $RSend(S)$, without considering every possible random message generation sequence.

To make formal this idea, we define a relationship among the random messages sequences. Then we illustrate it through an example.

Definition 4.2. Consider two sequences $\langle\langle g_i : T_i\rangle\rangle_{i\in I}$ and $\langle\langle g'_i : T'_i\rangle\rangle_{i\in I}$, each consisting of distinct random messages. If there exists a type-preserving bijection σ s.t. $\sigma(\{g_i\}_{i\in I}) = \{g'_i\}_{i\in I}$ and $\sigma(g) = g$ if $g \notin \{g_i\}_{i\in I} \cup \{g'_i\}_{i\in I}$, then we say the two sequences are related by σ, i.e. $\langle\langle g_i : T_i\rangle\rangle_{i\in I} \equiv_\sigma \langle\langle g'_i : T'_i\rangle\rangle_{i\in I}$.

Example 4.3. Consider the sequences $\langle\langle g_1 : T, g_2 : T\rangle\rangle$ and $\langle\langle g_2 : T, g_3 : T\rangle\rangle$. Then, we have that $\langle\langle g_1 : T, g_2 : T\rangle\rangle \equiv_\sigma \langle\langle g_2 : T, g_3 : T\rangle\rangle$ where $\sigma(g_1) = g_3$, $\sigma(g_2) = g_2$, $\sigma(g_3) = g_1$ and σ is the identity elsewhere. Note that two sequences may be related even if they have some values in common.

We can prove that, if $S \overset{\gamma}{\mapsto}$ then $\sigma(S) \overset{\sigma(\gamma)}{\mapsto}$, i.e. two systems related by a bijection σ perform computations which are related by this bijection σ. Assumption 4.4 is required to prove this result.

Lemma 4.2. *Consider a system S and assume that $\langle\langle g_i : T_i\rangle\rangle_{i\in I} \equiv_\sigma \langle\langle g'_i : T'_i\rangle\rangle_{i\in I}$. Assume also that in S there is no subterm $gen_T^{x,k}\, y$ s.t. $\mathsf{RT}^T(k)$ is equal to $g_i : T_i$ or $g'_i : T'_i$ for some $i \in I$. Then, we have*

$$S \overset{\gamma}{\mapsto} S_1 \quad \Rightarrow \quad \sigma(S) \overset{\sigma(\gamma)}{\mapsto} \sigma(S_1).$$

The next lemma shows that if an intruder performs a successful secrecy attack on the system after a sequence of guessing actions γ, then another attack exists with a sequence γ' related to γ by a bijection σ. This implies that we need to consider only one of these sequences in our secrecy analysis.

Lemma 4.3. *Let $\langle\langle g_i : T_i\rangle\rangle_{i\in I} \equiv_\sigma \langle\langle g'_i : T'_i\rangle\rangle_{i\in I}$; consider a system S and a sequential agent X, with $Sort(S\|X) \subseteq L$, and let no subterm $gen_T^{x,k}$ in $S\|X$ be s.t. $\mathsf{RT}^T(k)$ is equal to $g_i : T_i$ or $g'_i : T'_i$ for some $i \in I$; furthermore, assume $SubM(\phi) \cap (\{g_i : T_i\}_{i\in I} \cup \{g'_i : T'_i\}_{i\in I}) = \emptyset$. If $m : T$ is an initial message, then*

$$S\|_L X_{\phi\cup\{g_i : T_i\}_{i\in I}} \models \exists\gamma : m \in K_{X,\gamma}^{\phi\cup\{g_i : T_i\}_{i\in I}}$$

iff

$$\sigma(S)\|_L \sigma(X)_{\phi\cup\{g'_i : T'_i\}_{i\in I}} \models \exists\gamma : m \in K_{\sigma(X),\gamma}^{\phi\cup\{g'_i : T'_i\}_{i\in I}}.$$

This result provides us with the technical guidance required to perform a translation from the formula F, obtained through the partial evaluation functions in Table 7, to another one, denoted by $\tilde{F}$, in such a way that F is satisfiable iff $\tilde{F}$ is satisfiable. This translated formula $\tilde{F}$ shows only finitary disjunctions. The idea is to quotient $RSend(S)$ and take only a representative for each class of tuples whose sequences of random message generation can be related to each other through a bijection.

Definition 4.3. The relation $\equiv^e$ over tuples of $RSend(S)$ is defined as:

$$(c, m : T, \langle\langle g_i : T_i\rangle\rangle_{i\in I}, S') \equiv^e (c', m' : T', \langle\langle g'_i : T'_i\rangle\rangle_{i\in I}, S'_1)$$

iff

$$c = c', T = T', \exists\sigma : \langle\langle g_i : T_i\rangle\rangle_{i\in I} \equiv_\sigma \langle\langle g'_i : T'_i\rangle\rangle_{i\in I}, S'_1 = \sigma(S'), m' = \sigma(m)$$

Intuitively, two tuples of $RSend(S)$ are related by $\equiv^e$ if, whenever they have the same typed channel, then the two sequences of random messages are related by a bijection σ and also the messages and the derivatives of S are related by this bijection. Note $\equiv^e$ is an equivalence relation. The quotient of $RSend(S)$ w.r.t. the relation $\equiv^e$, namely $RSend(S)/_{\equiv^e}$, turns out to be finite.

Lemma 4.4. *$RSend(S)/_{\equiv^e}$ is a finite set.*

The translation $\tilde{F}$ is simply the partial evaluation function in Table 7 where $RSend(S)$ is replaced with $RSend(S)/_{\equiv^e}$.

The next lemma states that we can just study the satisfiability problem of $\tilde{F}$ instead of that of F.

Lemma 4.5. *Under the hypothesis of Proposition* 4.3, *let* $F = \exists\gamma : m_1 \in K^{\phi}_{X,\gamma}//S$. *Then, we have*

$$\exists X_\phi \text{ s.t. } X_\phi \models F \quad \text{iff} \quad \exists Y_\phi \text{ s.t. } Y_\phi \models \tilde{F}.$$

The two agents X and Y are related by a bijection. Thus, the main difference is that they guess different sequences of random values, yet related by type-preserving bijections.

Finally, we have reduced the checking of the existence of an agent X_ϕ s.t. $S\|_L X_\phi \models \exists\gamma : m \in K^{\phi}_{X,\gamma}$ to a satisfiability problem for a finitary formula in a sublogic of $\mathscr{L}$. The main result of this paper is the following.

Theorem 4.1 (Martinelli [35]). *Consider a system* S, *with* $Sort(S) \subseteq L$, *a finite set of typed messages* ϕ *and an initial message* $m : T$. *It is decidable if there exists* X_ϕ, *with* $Sort(X) \subseteq L$, *s.t.*

$$S\|_L X_\phi \models \exists\gamma : m \in K^{\phi}_{X,\gamma}.$$

Moreover, if $\exists\gamma : m \in K^{\phi}_{X,\gamma}//S$ is satisfiable, then we can build an agent (the attacker) which is a model of such a formula.

So far, in our analysis, we only considered sequential agents as possible intruders. However, this is not a significant restriction. As a matter of fact, if there is an attack performed by a compound system, then there is an attack by a sequential agent whose knowledge is the union of the knowledge of the agents of the compound system. There is only a technical restriction which prescribes that the system S under investigation must not use the eavesdropping action (as the intuition suggests since S should consist only of honest agents).

Theorem 4.2. *Consider a system* S, *with* $Sort(S) \subseteq L$ *and no construct* $\chi^x_{c,T}$ *occurring in it, a system* $Y = \|_{k\in\{1,\ldots,l\}} Y^k_{\phi_k}$ *consisting of a set of sequential agents* Y_k *with* $Sort(Y^k_{\phi_k}) \subseteq L$ *for* $k \in \{1,\ldots,l\}$, *and an initial message* $m : T$. *Suppose that, for some* $i \in \{1,\ldots,l\}$, *we have*

$$S\|_L Y \models \exists\gamma : m \in K^{\phi_i}_{Y^i,\gamma}$$

then there exists a sequential agent X_ϕ, *with* $\phi = \bigcup_{k\in\{1,\ldots,l\}} \phi_k$, *s.t.*

$$S\|_L X_\phi \models \exists\gamma : m \in K^{\phi}_{X,\gamma}.$$

4.4.1. About the complexity of the procedure

Our decision procedure consists of building the formula through partial model checking and studying its satisfiability. This check can be done in linear time in the size of

this formula. (Furthermore, this can be done *on-the-fly*, i.e. during the construction of the formula itself.[5]) However, the size of the formula depends on the specific inference system used. Thus, the complexity of the procedure cannot easily be estimated.

Nevertheless, we will try to roughly estimate the size of the formula in the simple case where S is a sequential agent and the inference system only consists of a basic type and the pairing function. This should provide the reader with an idea of the efficiency of our procedure (see also Section 5.2 for a significant example).

Let n be the max length of a computation of S. Let l be the max height of the types used in S. Let k be the maximum number of different prefix constructs that can be enabled in each state of S. Then, the size of the formula after partial evaluation is bound by

$$k^n(|\phi_S| + |\phi_X| + 2^l n)^{2^l n}.$$

If we do not consider the intruder able to guess new random messages, then the size of the formula is bound by

$$k^n(|\phi_S| + |\phi_X|)^{2^l n}.$$

Usually, l is very small (e.g., 3 or 4). However, if we consider compound systems, then n grows linearly with the number of sequential agents. Thus, as expected, our procedure suffers the state explosion problem.

5. Examples

We show two examples of analysis using our method. The first is simple and will be explained in the details. The second one is more complex and has been checked by using our verification tool PaMoChSA, which implements the approach described in this paper (e.g., see [30]).

5.1. Example 3.1

First, consider the simple system of the Example 3.1.

An agent A is going to send a confidential message m_A to the agent B. The sender A_{ϕ_A}, with $\phi_A = \{m_A, PK(B)\}$, is the following:

$$[m_A \quad PK(B) \vdash_4 x : E(Key, T)](c_{AB}!x.\mathbf{0}); \mathbf{0}.$$

The receiver B_{ϕ_B}, with $\phi_B = \{PK(B)^{-1}\}$, is the following:

$$c_{AB}?y : E(Key, T).([y \quad PK(B)^{-1} \vdash_5 z : T]\mathbf{0}; \mathbf{0}).$$

We study whether or not there is an intruder which can interact with our system and retrieve the message m_A, i.e.

$$\exists X_{\phi_X} \text{ s.t. } (A_{\phi_A} \| B_{\phi_B} \| X_{\phi_X}) \backslash \{c_{AB}\} \models \exists \gamma : m_A \in K_{X,\gamma}^{\phi_X}.$$

[5] Thus, we are able to find an attack, if any, even without building the whole formula.

Table 8
Unfolding of $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X} // S$

$$
\begin{array}{l}
\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X} // S \doteq \\
\quad \text{(guessing)} \\
\langle\langle \tau_{g_1} \rangle\rangle \langle c_{AB}!E(PK(B), g_1)\rangle (\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{g_1\}} // S_1) \\
\vee \quad \text{(guessing)} \\
\langle\langle \tau_{g_1}, \tau_{g_2} \rangle\rangle \langle c_{AB}!E(g_2, g_1)\rangle (\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{g_1, g_2\}} // S_2) \\
\vee \quad \text{(receiving)} \\
\langle c_{AB}?E(PK(B), m_A)\rangle (\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B), m_A)\}} // S_3) \\
\vee \quad \text{(eaves-dropping)} \\
\langle \chi_{c_{AB}, E(PK(B), m_A)}\rangle (\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B), m_A)\}} // S_4) \\
\vee \quad \text{(idling)} \\
\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X} // S_4 \\
\vee \quad \text{(nothing to do)} \\
\mathbf{F}
\end{array}
$$

where

$$
\begin{array}{l}
S_1 = A_{\phi_A} \| [E(PK(B), g_1) \quad PK(B)^{-1} \vdash_5 z : T] \mathbf{0}; \mathbf{0} \\
S_2 = A_{\phi_A} \| [E(g_2, g_1) \quad PK(B)^{-1} \vdash_5 z : T] \mathbf{0}; \mathbf{0}. \\
S_3 = \mathbf{0} \| c_{AB}?y : E(Key, T).[y \quad PK(B)^{-1} \vdash_5 z : T] \mathbf{0}; \mathbf{0} \\
S_4 = \mathbf{0} \| [E(PK(B), m_A) \quad PK(B)^{-1} \vdash_5 z : T] \mathbf{0}; \mathbf{0}
\end{array}
$$

The initial messages of $S = A \| B$ are $m_A, PK(B)^{-1}$ and $PK(B)$. It is reasonable to assume that the initial knowledge ϕ_X of X is simply the public key $PK(B)$. (Other messages can be guessed by the intruder during the attack.) We apply the partial evaluation rules in Table 7 (along with the reduction for infinitary disjunctions).

Note 2. *In the remainder of this section, we state that a formula F* reduces to *a formula F' whenever $F = \bigvee_{i \in I} F_i$, $F' = \bigvee_{i \in I'} F_i$ and $I' \subset I$; moreover, we require that F is satisfiable iff F' is satisfiable.*

- First, we consider $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X} // S$ where $S = A_{\phi_A} \| B_{\phi_B}$. To calculate this formula, we need to calculate $Send(S)$ and $RSend(S)_{/\equiv^e}$ (see Table 7). In particular, we have:
  - $Send(S) = \{\}$. Indeed, note that $Send(S)$ represents the possible messages that the intruder may generate from its knowledge and the system is willing to synchronize with. However, the system S may only receive messages of type $E(Key, T)$ whereas the intruder can derive no messages of that type from ϕ_X.
  - $RSend(S)_{/\equiv^e} = \{(c_{AB}, E(PK(B), g_1) : E(Key, T), \langle\langle g_1 : T \rangle\rangle, S_1), (c_{AB}, E(g_2, g_1) : E(Key, T), \langle\langle g_1 : T, g_2 : Key \rangle\rangle, S_2)\}$ where S_1 and S_2 are given in Table 8. The first tuple of $RSend(S)_{/\equiv^e}$ represents the intruder, which guesses a message g_1 of type T and uses it in a communication with S. The second tuple of $RSend(S)_{/\equiv^e}$ represents the intruder which guesses a message g_1 of type T and a message g_2 of type Key and builds the encryption of g_1 with g_2.

  So we obtain the formula in Table 8. We can make a first reduction by noticing that the *eavesdropping* behavior of the intruder is more dangerous than the *idling* one; i.e. if the formula corresponding to idling is satisfiable then also the formula relative to eavesdropping is satisfiable. Thus, the disjunction of the eavesdropping and the idling formula reduces to the eavesdropping formula. Moreover, note that

Table 9
Unfolding of $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{g_1\}} // S_1$

$\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{g_1\}} // S_1 \doteq$

(receiving)

$\langle c_{AB}?E(PK(B), m_A)\rangle(\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A),g_1\}} // S_5)$

$\vee$ (nothing to do)

$\mathbf{F}$

where

$S_5 = \mathbf{0} \| [E(PK(B), g_1) \quad PK(B)^{-1} \vdash_5 z : T]\mathbf{0}; \mathbf{0}$

the formula $G \vee \mathbf{F}$ is equivalent to G. Thus, the last three subformulas reduce to

$$\langle \chi_{c_{AB},E(PK(B),m_A)} \rangle (\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A)\}} // S_4).$$

In addition, note that $S_4 \overset{a}{\not\mapsto}$ for every action a (so $S = Nil$); thus, by definition in Table 7, $(\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A)\}} // S_4)$ is $\mathbf{F}$ because $m_A \notin \mathscr{D}(\phi_X \cup \{E(PK(B), m_A)\})$.

- Next, we analyze: $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{g_1\}} // S_1$. We have that $Send(S_1) = \{\}$ and $RSend(S_1)_{/\equiv^e} = \{\}$ because $S_1 \overset{c?m}{\not\mapsto}$ for every action $c?m$. So, we obtain the formula in Table 9. This formula reduces to its first argument, i.e.

$$\langle c_{AB}?E(PK(B), m_A)\rangle(\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A),g_1\}} // S_5).$$

However, note that $S_5 \overset{a}{\not\mapsto}$ for every $a \in Act$; thus, by definition of the partial evaluation function, $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A),g_1\}} // S_5$ is $\mathbf{F}$ because $m_A \notin \mathscr{D}(\phi_X \cup \{E(PK(B), m_A), g_1\})$. Thus, the whole formula in Table 9 reduces to $\mathbf{F}$.
- Next, we analyze: $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{g_1,g_2\}} // S_2$. This formula reduces to $\mathbf{F}$. The reasoning proceeds as above.
- Next, we analyze: $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A)\}} // S_3$. We first compute $Send(S_3)$ and $RSend(S_3)$. In particular, we have:
  - $Send(S_3) = \{(c_{AB}, E(PK(B), m_A), S_4)\}$, because $E(PK(B), m_A) \in \mathscr{D}(\phi_X \cup \{E(PK(B), m_A)\})$.
  - $RSend(S_3)_{/\equiv^e} = \{(c_{AB}, E(PK(B), g_1) : E(Key, T), \langle\langle g_1 : T\rangle\rangle, S_5), (c_{AB}, E(g_2, g_1) : E(Key, T), \langle\langle g_1 : T, g_2 : Key\rangle\rangle, S_6)\}$ where S_6 is given in Table 10.

  Note that S_5 and S_6 are equal to Nil, i.e. they can perform no actions. Thus, by definition of the partial evaluation function, $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A),g_1\}} // S_5$ and $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A),g_1,g_2\}} // S_6$ are $\mathbf{F}$, because $m_A \notin \mathscr{D}(\phi_X \cup \{E(PK(B), m_A), g_1, g_2\})$ (and so $m_A \notin \mathscr{D}(\phi_X \cup \{E(PK(B), m_A), g_1\})$). Thus, the whole formula in Table 10 reduces to $\mathbf{F}$.

Table 10
Unfolding of $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A)\}} // S_3$

$\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A)\}} // S_3 \doteq$

(guessing)

$\langle\langle \tau_{g_1} \rangle\rangle \langle c_{AB}!E(PK(B), g_1)\rangle (\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A),g_1\}} // S_5)$

$\vee$ (guessing)

$\langle\langle \tau_{g_1}, \tau_{g_2} \rangle\rangle \langle c_{AB}!E(g_2, g_1)\rangle (\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A),g_1,g_2\}} // S_6)$

$\vee$ (nothing to do)

$\mathbf{F}$

where

$S_6 = \mathbf{0} \| [E(g_2, g_1) \quad PK(B)^{-1} \vdash_5 z : T]\mathbf{0}; \mathbf{0}$

Table 11
Final unfolding of $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X} // S$

$\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X} // S \doteq$

(guessing)

$\langle\langle \tau_{g_1} \rangle\rangle \langle c_{AB}!E(PK(B), g_1)\rangle \mathbf{F}$

$\vee$ (guessing)

$\langle\langle \tau_{g_1}, \tau_{g_2} \rangle\rangle \langle c_{AB}!E(g_2, g_1)\rangle \mathbf{F}$

$\vee$ (receiving)

$\langle c_{AB}?E(PK(B), m_A)\rangle \mathbf{F}$

$\vee$ (eaves-dropping, idling, nothing to do)

$\mathbf{F}$

- Next, we analyze: $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{E(PK(B),m_A)\}} // S_4$. This formula reduces to $\mathbf{F}$. The reasoning proceeds as for $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X \cup \{g_1,g_2\}} // S_2$.

To summarize, we obtain that $\exists\gamma : m_A \in K_{X,\gamma}^{\phi_X} // S$ reduces to the formula in Table 11 and finally to $\mathbf{F}$. Thus, there is no intruder with initial knowledge ϕ_X which can discover the exchanged message m_A.

5.2. Needham Schroeder public key protocol

This protocol has became paradigmatic for testing analysis tools for cryptographic protocols. It has a subtle flaw discovered by Lowe [27] which arises in the presence of a malicious agent.

Table 12
Needham Schroeder public key protocol

$A \mapsto B: \{N_a, A\}_{PK(B)}$	$A \mapsto B: \{N_a, A\}_{PK(B)}$
$B \mapsto A: \{N_a, N_b\}_{PK(A)}$	$B \mapsto A: \{N_a, N_b, B\}_{PK(A)}$
$A \mapsto B: \{N_b\}_{PK(B)}$	$A \mapsto B: \{N_b\}_{PK(B)}$
Flawed version	Lowe's corrected version

Table 13
Description in Crypto-CCS of the Needham Schroeder public key protocol

Let NSPK be $A\|B\|_L X_{\phi_X}$, where $\phi_X = \{A, B, PK(X)^{-1}, PK(X), PK(A), PK(B)\}$, $L = \{c_{AB}, c_{AX}, c_{BX}\}$, $A = A_B + A_X$, $B = B_A + B_X$ with:

$$
\begin{aligned}
A_B \;=\; & [N_a \quad A \vdash_1 y][y \quad PK(B) \vdash_4 y_1] \\
& c_{AB}!y_1. \\
& c_{AB}?r_1 : E(Key, Nonce \times Nonce). \\
& [r_1 \quad PK(A)^{-1} \vdash_5 y_2][y_2 \vdash_2 y_3] \\
& [y_3 = N_a] \\
& [y_2 \vdash_3 y_4][y_4 \quad PK(B) \vdash_4 y_5] \\
& c_{AB}!y_5
\end{aligned}
\qquad
\begin{aligned}
B_A \;=\; & c_{AB}?r_2 : E(Key, Nonce \times Id). \\
& [r_2 \quad PK(B)^{-1} \vdash_5 z_1][z_1 \vdash_3 z_2] \\
& [z_2 = A] \\
& [z_1 \vdash_2 z_3][z_3 \quad N_b \vdash_1 z_4] \\
& [z_4 \quad PK(A) \vdash_4 z_5] \\
& c_{AB}!z_5.c_{AB}?r_3 : E(Key, Nonce) \\
& [r_3 \quad PK(B)^{-1} \vdash_5 z_6][z_6 = N_b]
\end{aligned}
$$

and

$$
\begin{aligned}
A_X &= A_B[N_a^X/N_a, PK(X)/PK(B), c_{AX}/c_{AB}] \\
B_X &= B_A[N_b^X/N_b, PK(X)/PK(A), c_{BX}/c_{AB}]
\end{aligned}
$$

Note: The tailing **0** is omitted, i.e. $[m = m']A_1$ stands for $[m = m']A_1; \mathbf{0}$ (similarly for the deduction constructs).

In Table 12, we show the *intended* execution of the protocol by using the notation which is commonly found in literature. In the flawed version, the sender A communicates to B a fresh nonce N_a (i.e. a randomly guessed value) and its name encrypted with the public key of B (thus only B can decrypt this message). Next, the receiver B communicates to A the just received nonce N_a and a fresh nonce N_b, both encrypted with the public key of A. Finally, the sender A communicates to the receiver the nonce N_b encrypted with the public key of B. At the end of a successful run between a sender A and a receiver B, only these two processes should know N_a and N_b. [6]

We analyzed the Crypto-CCS specification in Table 13 which consists of three agents: A, B and X. The agent A may act as initiator both with B and X; the agent B acts as responder, while the agent X may act as initiator or responder of the protocol. The specification for X is not given, thus we check whether the system is secure

[6] These nonces could be used to establish a new communication channel with a new shared key that is a function of these values.

Table 14
The Lowe's attack described in common notation

(1) $A \mapsto X : E(PK(X),(N_a,A))$
(2) $X(A) \mapsto B : E(PK(B),(N_a,A))$
(2) $B \mapsto X(A) : E(PK(A),(N_a,N_b))$
(1) $X \mapsto A : E(PK(A),(N_a,N_b))$
(1) $A \mapsto X : E(PK(X),N_b)$
(2) $X(A) \mapsto B : E(PK(A),N_b)$

against whatever behavior the agent X could have. We only specified the intruder's initial knowledge, i.e. the public keys of A and B, the names of A and B and its private and public key. We need not give the nonces to the intruder because it can guess them by itself. We performed our analysis and, as expected, we found the flaw in [27]: an intruder X masks as A to B and discovers both nonces N_a and N_b.

The attack is given in Table 14 (we use $X(A)$ as the intruder involved in the communication as agent A): the attack consists of two concurrent sessions: in the first one, the agent A initiates the protocol with X; in the second one, the agent X communicates with B pretending to be A. The steps of the attack 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 $E(PK(A),(N_a,N_b))$, which contains the fresh nonce N_b, encrypted with the public key of A. Now, the intruder cannot directly decrypt the message, but can send the message to the agent A. The agent A correctly decrypts $E(PK(A),(N_a,N_b))$ and resends the nonce N_b to X, encrypted with the public key of X, since it thinks it is the second message of its run with X. Eventually, X discovers N_b and sends it to B.

We corrected the protocol as done [27] (see Table 12) and checked that there are no flaws. Indeed, the second message encodes also the name of the sender, i.e. B. Thus, the presented attack is no more possible, because A would receive from X a message whose sender is B. At that point, A should quit the session with X.

Note that, in our analysis, we can guarantee there are no attacks only in the system configuration we analyzed, i.e. only a session between a single sender and a single receiver. In principle, we cannot extend the results to systems consisting of more users and sessions. Moreover, it is interesting to remark that the Lowe's attack is possible only in the presence of a legitimate user which acts maliciously against the other users of the system. Indeed, if we consider the agent X as an external one, i.e. a legitimate agent does not start a session with it, then the attack is not possible.

5.2.1. *Experimental results with* PaMoChSA

The current implementation of our verification tool PaMoChSA is in the *ocaml v3.0* language. The tool receives as inputs: the description of the system under investigation, the intruder's initial knowledge, and a predicate on the knowledge of the intruder,

Table 15
Experimental results about Needham–Schroeder public key protocol with AMOCHSA

Spec name	Significant stored nodes	User time (s)	Attack?
NSPK-flawed	18623	3.58	YES
NSPK-correct	267 659	58	NO
NSPK-flawed-no-gen	319	0.06	YES
NSPK-correct-no-gen	793	0.19	NO

e.g., if a message may be deduced from it. The tool looks for possible attacks on the specification, i.e. if an intruder, by interacting with the system, is able to reach a configuration where its knowledge satisfies the predicate. If the tool discovers a possible attack, then it outputs its description.

In Table 15, we report the experimental results about the verification of the Needham–Schroeder Public Key protocol (NSPK, for short) on a Pentium III (750MHz, 772 Mb) with Red-Hat 6.2 operating system. The first column lists the name of the file with the Crypto-CCS protocol specification, the intruder initial knowledge and the predicate to be checked; the second column lists the number of the stored nodes of the formula obtained during the partial model checking phase; the third column reports the user time for the whole analysis and finally the fourth column reports the results of the analysis (if the system is not correct, then a successful attack sequence is given). The last two lines list the cases where intruders cannot generate any new names.

6. Concluding remarks and related work

We proposed a novel approach for modeling and checking the security properties of cryptographic protocols. This relies on the observation that security protocols may be naturally described as open systems, i.e. systems which may have some unspecified components. These may be used to represent a hostile intruder whose behavior cannot be predicted or else a malicious agent which may not follow its program to obtain an advantage for itself. The verification phase consists of checking that, for any instance of the unknown component, the resulting system satisfies a property expressed in a formula of a suitable temporal logic. We argue this paradigm is fit for specifying and analyzing both network and system security. As a matter of fact, we have currently applied partial evaluation techniques to the analysis of such properties as *non-interference* [36], *timed non-interference* [15], *secrecy* [35] and *authentication* [30]. All the results rely on the following schema:

(1) design suitable languages for system description and property specification;
(2) develop corresponding partial evaluation techniques;
(3) develop a satisfiability procedure for the logic used.

The application of the previous steps to a specific setting requires some efforts (just like in the secrecy analysis of this paper). Nevertheless, this method has a wide range of applications, because the techniques used are general and flexible. In particular,

partial model checking has been defined for expressive temporal logics, e.g. μ-calculus [22], and for a class of system description languages based on certain formats of SOS [5,25,34].

Partial evaluation functions ideally describe the behavior and abilities of the unknown component, i.e. the enemy. This is a main feature of our method: when you assume that the enemy may be a process definable in the calculus, then enemy's abilities are directly inferred. It is interesting to note that this also provides a sort of formal justification of the enemy's abilities in the so-called most powerful intruder method by Dolev and Yao (e.g., see [11]). The idea is to consider a fixed intruder which can eavesdrop, tap into and fake messages exchanged in a network. Table 7 shows that our intruders have precisely these abilities.

The main contributions of this paper can be summarized as follows:

- Security properties can be naturally described as properties of open systems. Our method provides a uniform approach for the definition of both network and system security. We used a single context, i.e. $S\|_L X$, since several security properties can be defined using a similar context (see [17]). However, the same ideas may apply to more complex contexts (open systems) as well.
- Partial model checking techniques are flexible and may be applied to several analysis frameworks. These techniques might also be used to formally infer potential intruders' abilities also for other computational models and analysis scenarios (e.g., different contexts).
- We provide a decidability result for the secrecy analysis of protocols with a finite number of sessions. In particular, we allow intruders to guess new random values but we consider messages of bounded size.
- A suitable process calculus for describing security protocols has been proposed. This calculus is equipped with a construct to model in which way messages can be inferred from other messages. The presence of this construct is justified by the observation that, in the description of security protocols, we need to model several different cryptographic functions. These might have specific algebraic features which can be conveniently encoded in an inference system. Our analysis theory is completely parametric with respect to the given inference system, provided this relies on some (mild) assumptions.

We can highlight (at least) two limitations of our technique:

- We consider only protocols with a limited number of parties and sessions. (Note that the secrecy analysis for general protocols is undecidable because it can be reduced to the halting problem.) However, in [28,47], preliminary results have been obtained about how, for a large class of protocols, the correctness with an unbounded number of participants and sessions can be inferred from the results of the analysis of a system with a finite number of parties and sessions. Hopefully, the combination of this method with ours may contribute to the development of fully automated analysis approaches for a large class of security protocols.
- We consider communications over *typed* channels, i.e. processes cannot receive messages unless they do comply with the type imposed on the channel. Hence, our approach does not take into account the so-called *type flaw attacks* which are based on a misunderstanding of the structure of the received messages. In [19], however,

under some mild assumptions on the protocols, it has been shown that, if a system is secure against attacks which do not exploit-type flaws, then that system is secure also against type flaw attacks. This result has been obtained by considering the *strand spaces* [48] as security protocol models. Whether this holds in our framework will require further study.

We plan also to extend our approach by admitting a restricted form of recursion and adopting a method to symbolically represent the knowledge that the intruders would obtain during the computation with an unlimited system. (Such symbolic techniques could be also useful to avoid bounds on the message size, e.g., see [4].) Moreover, we believe that the security analysis method explained in this paper is flexible and may be suitable for other languages and analysis contexts as well, e.g. for such mobile languages as ambient calculus [9] or π-calculus [39] (for some preliminary results see [32]).

6.1. Related work

The literature on security properties analysis and verification of cryptographic protocols is wide. For an impressive overview of cryptographic protocols and cryptographic systems, see [44]. Here, we briefly recall some approaches related to process algebra and logic theory.

The most powerful intruder approach has been recasted in the process algebra setting by many authors, e.g. Lowe [27], Roscoe [43] and Schneider [45]. The idea is to analyze the system under investigation only against a single process which describe the behavior of the most powerful intruder. Marrero et al. [31] use a model with sequential agents where the intruder's behavior is implicitly assumed. Although it is not necessary to give an explicit description of the intruder's behavior, yet this is fixed a priori through some axioms that represent its capabilities. It is worthy noticing that, at least for the trace properties used here (also called safety properties), our approach based on the universal quantification over possible intruders and the one based on the most powerful one are equivalent (see [17]). However, there are more complex security properties which mix several aspects as, e.g., deadlock detection, branching points, liveness and/or fairness, where the most powerful intruder approach is not directly applicable and the quantification over all possible intruders seems unavoidable. For instance, see the discussions in [12,36] about the BNDC property, i.e. a non-interference property based on the notion of *bisimulation*, and about *non-repudiation* [17,23]. Note that our verification method has been useful to check BNDC-like properties [36].

In [13,16,17], an attempt is proposed to analyze *information flow* and cryptographic protocols within the same conceptual framework. A general schema for defining security properties is given [17]. This schema is called *Generalized NDC* and it is a suitable extension of a non-interference property, i.e. NDC, defined in [12]. The main idea is that a system is secure whenever its normal execution behavior cannot be "significantly" altered even in the presence of an intruder (or a malicious agent). The schema relies on the quantification over any possible enemy as ours, but the specification of the "correct" behavior is given through a process. The analysis method is the most general intruder approach and a systematic way of constructing such intruder is proposed.

A sufficient condition (see [17]) is given that shows how considering only this process is enough for a large class of security properties. One of the main goals of such approach is to provide a uniform framework for the comparison of security properties (e.g., see [14]).

Some authors found the π-calculus [39] suitable for describing security protocols. This is mainly due to its management of names, some of which can be seen as secret messages. In [2], Abadi and Gordon proposed an approach based on proof theoretic tools for a variant of the π-calculus, namely the spi-calculus, which embodies some constructs for modeling cryptography. The innovative idea was to model the intruders using the testing equivalence theory for π-calculus (e.g., see [7]). In [1], Abadi proposed a type system for secrecy properties for the spi-calculus; Boreale et al. (e.g., see [8]) provided a compositional proof system for equivalence testing which may be used to deal with both secrecy and authenticity properties. Another approach relies on control flow analysis techniques of the π-calculus [6]. By controlling how the information is exchanged along channels, confinement properties can be studied, i.e. whether the information is sent via a particular channel or remains enclosed in a system. This approach has also been extended to mobile ambients [9] (possibly with an unbounded number of states) by Nielson and Nielson [41].

Other approaches are based on proof theoretic methods (e.g., see [2,3,18,21,42,45]). Some of them use temporal and modal logic concepts and make it possible to prove that a system, even without a finite behavior, enjoys security properties. In general, these methods are not fully automated and need non-trivial human efforts to analyze systems, while counter-examples are not directly feasible. An interesting exception is the work by Kindred and Wing [21], where the authors propose a fully automated approach for checking that a protocol enjoys some properties expressed in a logical language L (which has to satisfy some requirements). The protocol is represented through a set of formulas P. From this set P, by using the rules and axioms of the logic, it is possible to infer all formulas which the protocol enjoys (the "theory" of P). This set is finitely representable and the membership problem is decidable. Thus, the verification that the protocol satisfies a property expressed by a formula F corresponds to check whether the formula F belongs to the "theory" of P.

In [23], Kremer and Raskin analyze the non-repudiation properties of security protocols using ATL logic. This logic has been specifically developed to describe the properties of open systems, so this approach and ours are somehow related to each other. However, both the computational model and the solution methodology differ. (Nevertheless, it would be very interesting to extend our framework for the treatment of non-repudiation properties and then compare the efficiency of the two approaches.)

Acknowledgements

We would like to thank the anonymous referees for their helpful comments. Thanks are also due to Marinella Petrocchi for checking our specification of the NSPK protocol with PaMoChSA.

This work has been partially supported by Microsoft Research Europe (Cambridge); by MIUR project "MEFISTO"; by MURST project "Tecniche e strumenti software per l'analisi della sicurezza delle comunicazioni in applicazioni telematiche di interesse economico e sociale"; by CNR project "Strumenti, ambienti ed applicazioni innovative per la società dell'informazione" and finally by CSP with the project "SeTAPS".

Appendix A. Proofs

Lemma 3.1. *Consider a formula $F \in \mathscr{L}$, which consists only of the logical constants, disjunctions and the possibility modality, and a finite set of closed messages ϕ. It is decidable if there exists a sequential agent X_ϕ s.t. $X_\phi \models F$.*

Proof. We prove the thesis by structural induction on F; furthermore, if F is satisfiable, we construct a model X_F for such a formula.

- $F = \mathbf{F}$. Then, no sequential agent models F, and thus F is not satisfiable.
- $F = \mathbf{T}$. Then, every sequential agent models F. Let $(X_F)_\phi$ be $(\mathbf{0})_\phi$.
- $F = F_1 \vee F_2$. Then, F is satisfiable iff F_1 is satisfiable or F_2 is satisfiable. If both F_1 and F_2 are not satisfiable then also F is not satisfiable. Otherwise, suppose that F_1 (or F_2) is satisfiable. Then, by structural induction, there is $(X_{F_1})_\phi$ which models F_1. Let $(X_F)_\phi$ be $(X_{F_1})_\phi$.
- $F = \langle a \rangle F_1$. By inspection of the action a:
  - $c!m$. If $m \notin \mathscr{D}(\phi)$ then F is not satisfiable. (Recall that whenever an agent sends a message, then this message must deducible from its knowledge.) Otherwise, assume $m \in \mathscr{D}(\phi)$. If F_1 is not satisfiable then also F is not satisfiable. On the other hand, by structural induction, let $(X_{F_1})_\phi$ be a model for F_1. Then, let $(X_F)_\phi$ be $(c!m.X_{F_1})_\phi$, if $m \in \phi$, otherwise let $(X_F)_\phi$ be $(p.c!x.X'_{F_1})_\phi$, where: 1) p is a proof of m from ϕ whose root is an assignment to the variable x; 2) x is a variable that does not appear in X_{F_1}; 3) X'_{F_1} is the term X_{F_1} where m is replaced with x.
  - $c?m$. If F_1 is not satisfiable by a sequential agent whose knowledge is $\phi \cup \{m\}$ then also F is not satisfiable. Otherwise, by structural induction, let $(X_{F_1})_{\phi \cup \{m\}}$ be a model for F_1. Then, let $(X_F)_\phi$ be $(c?x:T.X'_{F_1})_\phi$, where: 1) T is the type of m, 2) x is a variable that does not occur in X_{F_1}; 3) X'_{F_1} is the term X_{F_1} where the message m is replaced with x.
  - τ_g. This case is similar to the previous one, with the appropriate prefix construct $gen_T^{x,i}$.
  - $\chi_{c,m}$. This case is similar to the previous one, with the appropriate prefix construct $\chi_{c,T}^x$. □

Proposition 4.1. *Given a system S, an agent X and a logical formula $F \in \mathscr{L}$, then*

$$S \| X \models F \quad \textit{iff} \quad X \models F//S.$$

Proof. The proof is performed by structural induction on the formula F. We describe only one case, the others are similar.

- $F = \langle a \rangle F'$, with $a \neq \tau_{c,m}, \tau_g$. We have $S \| X \models \langle a \rangle F'$ iff $S \| X \xrightarrow{a} S' \| X'$ and $S' \| X' \models F'$. Since, $a \neq \tau_{c,m}, \tau_g$, the unique rules to derive the transition $S \| X \xrightarrow{a} S' \| X'$ may be $\|_1$ and the symmetric of $\|_1$. So, $S \| X \xrightarrow{a} S' \| X'$ iff either $S \xrightarrow{a} S'$ for some S' and $X' = X$ or $X \xrightarrow{a} X'$ and $S' = S$. By, induction hypothesis, we have that $S' \| X \models F'$ iff $X \models F' // S'$, and $S \| X' \models F'$ iff $X' \models F' // S$. So, we eventually obtain that $S \| X \models \langle a \rangle F'$ iff $X \models \vee_{S' : S \xrightarrow{a} S'} F' // S'$ or $X \models \langle a \rangle (F' // S)$ and the thesis follows. □

Proposition 4.2. *Given a set of channels L and a logical formula $F \in \mathscr{L}$, we have*

$$(S) \backslash L \models F \quad \textit{iff} \quad S \models F //_L.$$

Proof. The proof is performed by structural induction on the formula F. We describe only one case, the others are similar.

- $F = \langle a \rangle F'$. We have that $S \backslash L \models \langle a \rangle F'$ iff $S \backslash L \xrightarrow{a} S' \backslash L$ and $S' \backslash L \models F'$. Note that $S \backslash L \xrightarrow{a} S' \backslash L$, for some S', only if $channel(a) \notin L$ and, in this case, by induction hypothesis, we have that $S' \backslash L \models F'$ iff $S' \models F' //_L$. So if $channel(a) \notin L$ then $S \backslash L \models \langle a \rangle F$ iff $S \models \langle a \rangle (F' //_L)$. When $channel(a) \in L$, for no S we may have $S \backslash L \models \langle a \rangle F$, or equivalently, $S \models \mathbf{F}$. □

Lemma 4.1. *Given a system S and an initial message m, suppose that there exists a sequential agent X_ϕ s.t. $Sort(S \| X) \subseteq L$ and $S \|_L X_\phi \models \exists \gamma : m \in K^\phi_{X,\gamma}$. Then, a well-behaved agent X'_ϕ exists s.t. $S \|_L X'_\phi \models \exists \gamma : m \in K^\phi_{X',\gamma}$.*

Proof. Assume that $S \|_L X \overset{\gamma'}{\mapsto}$ and $m \in K^\phi_{X,\gamma'}$, i.e. $m \in \mathscr{D}(\phi \cup msgs(\gamma))$ where $\gamma' \downarrow_X = a_1 \ldots a_j = \gamma$. We must have $X_\phi \overset{\gamma}{\mapsto}$. Thus, the formula $\langle a_1 \rangle \ldots \langle a_n \rangle \mathbf{T}$ is satisfiable by an agent whose knowledge is ϕ. Then, by the proof of Lemma 3.1, we can find a process Y_ϕ s.t. $Y_\phi \overset{\gamma}{\mapsto}$, and whose structure is as follows:

$$Y = pc_1.pc_2. \ldots \overbrace{[\ldots] \ldots [\ldots \vdash x]}^{p} pc_i \ldots . pc_{j-1}.pc_j.\mathbf{0},$$

where pc_i are prefix constructs and p represents a proof of x. Note that each prefix construct pc_i corresponds to action a_i in γ. Deduction constructs may only appear before a sending action. Moreover, note that the prefix construct which immediately follows the end of a proof must be the sending of the inferred value, e.g., $pc_i = c!x$.

We will build a well-behaved agent X' s.t $S \|_L X' \models \exists \gamma : m \in K^\phi_{X',\gamma}$. The agent X' is obtained by iteratively reordering the prefix constructs of X, until we obtain a well-behaved agent. In particular, we move each generation action just before the first place where the generated value is necessary, i.e. before the proof of the first sent message m' s.t. g occurs in m'.

Let X_0 be X. We construct a sequence of agents X_k s.t. $S \|_L X_k \overset{\gamma'_k}{\mapsto}$, $\gamma'_k \downarrow_X = \gamma_k$ and $msgs(\gamma_k) = msgs(\gamma)$. Moreover, the relative order between communication actions with S is never changed. Thus, also X_k is a successful attacker of S.

Assume that in X_k we can find a prefix construct $pc_i = gen_T^{x,i}$, whose corresponding action in $\gamma_k = a_1 \dots a_j$ is $a_i = \tau_g$ and:

- $pc_{i+1} = \chi_{c,T}^{y}, c?y:T$, then we simply let X_{k+1} be the agent X_k where pc_i and pc_{i+1} are swapped. Note that the generation action a_i has no influence on the action a_{i+1}, thus the agent X_{k+1} executes a computation γ_{k+1} where a_i and a_{i+1} are inverted with respect to the sequence γ_k.
- We have that $a_l = c!m'$ and g does not occur in m', where $l > i$ is the smallest index s.t. $a_l = c!m'$, for some m', and $a_h = \tau_{g_h}$, for all h s.t. $i < h < l$. Note that m' can be deduced by $\phi \cup msgs(a_1 \dots a_{l-1})$. Note also that g does not appear as submessage of $\phi \cup msgs(a_1, \dots, a_{i-1})$ since it is newly created and it cannot be received (or eavesdropped) as submessage in every action in a_h with $1 < h \leqslant i-1$ since it is never sent as submessage and the other processes cannot deduce it. (Indeed, g is a random value of basic type which can be only guessed by X_k and by Assumption 4.2 this message cannot be deduced by no other agent in S.) Note also that $g \notin msgs(a_{i+1} \dots a_{l-1})$ since these are guessing actions of new random messages. Thus, we have that $m' \in \mathscr{D}(\phi \cup msgs(a_1 \dots a_{l-1}))$, and g does not occur in $\phi \cup msgs(a_1 \dots a_{i-1} a_{i+1} \dots a_{l-1})$. We can thus apply Assumption 4.1, and we get $m' \in \mathscr{D}(\phi \cup msgs(a_1 \dots a_{i-1} a_{i+1} \dots a_{l-1}))$. Let p' a proof of m from $\phi \cup msgs(a_1 \dots a_{i-1} a_{i+1} \dots a_{l-1})$ whose root is an assignment to the variable x. Note that g does not appear in p'. Thus, its guessing it is not necessary in the proof p'. Note also that the generation action a_i has no influence on the successive generation action a_h, for $i < h < l-1$. So, let X_{k+1} be the process X_k where each construct pc_h, with $i < h \leqslant l$, is shifted left of one position, the proof p, that precedes pc_l in X_k, is replaced with p' and furthermore the pc_l construct of X_{k+1} is pc_i of X_k.

Finally, we obtain an agent whose guessing actions are grouped together and inserted just before the sending of a message which contains the guessed values as submessages. Actually, it is also possible that the tail of the agent consists of a sequence of guessing actions followed by the **0** term. We can safely remove them and we obtain a well-behaved agent. Call this agent X' and the result follows. □

Proposition 4.3. *Given a system S and a well-behaved sequential agent X_ϕ where ϕ is finite and $Sort(S \| X) \subseteq L$, then if m is an initial message we have*

$$S \|_L X_\phi \models \exists \gamma : m \in K_{X,\gamma}^{\phi} \quad \textit{iff} \quad X_\phi \models \exists \gamma : m \in K_{X,\gamma}^{\phi} // S.$$

Proof. By induction on the max length of the computations of S.

Base case: $S \doteq Nil$. We observe that X can only guess new random messages during its future behavior. But, by Assumption 4.1, this does not significantly improve its capabilities for the deduction of initial messages. Thus, X is a successful attacker iff $m \in \mathscr{D}(\phi)$.

Induction step:

$$S \|_L X_\phi \models \exists \gamma : m \in K_{X,\gamma}^{\phi} \quad \text{iff}$$
$$\exists \gamma : S \|_L X_\phi \models m \in K_{X,\gamma}^{\phi} \quad \text{iff}$$

$$
\begin{array}{ll}
S\|_L X_\phi \models m \in K^\phi_{X,\varepsilon} & \vee \\
\exists \gamma, \gamma = a\gamma' \wedge S\|_L X_\phi \models m \in K^\phi_{X,\gamma} & \text{iff} \\
X_\phi \models m \in K^\phi_{X,\varepsilon} & \vee
\end{array}
$$

(1) $\exists a \exists S', X' : S\|_L X_\phi \xrightarrow{a} S'\|_L X'_{\phi'} \wedge S'\|_L X'_{\phi'} \models \exists \gamma' : m \in K^{\phi'}_{X',\gamma'}$ $\vee$
(Interactions between X and S through rules $\|_2, \|_\chi$ and symm.)

(2) $\exists a \exists S' : S\|_L X_\phi \xrightarrow{a} S'\|_L X_\phi \wedge S'\|_L X_\phi \models \exists \gamma' : m \in K^\phi_{X,\gamma'}$ $\vee$
(S only acts through rule $\|_1$)

(3) $\exists a \exists X' : S\|_L X_\phi \xrightarrow{a} S\|_L X'_{\phi'} \wedge S\|_L X'_{\phi'} \models \exists \gamma' : m \in K^{\phi'}_{X',\gamma'}$
(X only acts through rule $\|_1$ symm.)

Condition (1) takes into account every possible synchronization between S and X; these are (from the point of view of X agent):

- Receiving a typed message on a channel c. Because only a finite set of messages can be sent from S on every channel and the number of channels is finite, we can consider only a finite number of actions a. So we have that $X_\phi \xrightarrow{c?m'} X'_{\phi \cup \{m':T'\}}$ and $S'\|_L X'_{\phi \cup \{m':T'\}} \models \exists \gamma' : m \in K^{\phi \cup \{m':T'\}}_{X',\gamma'}$. By induction hypothesis we have that $X'_{\phi \cup \{m':T'\}} \models \exists \gamma : m \in K^{\phi \cup \{m':T'\}}_{X',\gamma} // S'$. So considering all the possible a actions of this kind we obtain

$$\bigvee_{S \xrightarrow{c!m'} S'} \langle c?m' \rangle (\exists \gamma : m \in K^{\phi \cup \{m':T'\}}_{X',\gamma} // S').$$

- Sending of a typed message. By induction hypothesis, the following disjunction takes in account these cases:

$$\bigvee_{(c,m',S') \in Send(S)} \langle c!m' \rangle (\exists \gamma : m \in K^{\phi}_{X',\gamma} // S'),$$

where $Send(S) = \{(c, m', S') \mid S \xrightarrow{c?m'} S' \text{ and } m' \in \mathscr{D}(\phi)\}$.

- Eavesdropping of a communication internal to the system S. Such communications can be in a finite number, and hence by induction hypothesis these cases can be treated as

$$\bigvee_{S \xrightarrow{\tau_{c,m'}} S'} \langle \chi_{c,m'} \rangle (\exists \gamma : m \in K^{\phi \cup \{m':T'\}}_{X',\gamma} // S').$$

Condition (2) takes in account actions performed by the system S without interaction with the agent X, by induction hypothesis these actions can be taken into account using the following formula:

$$\bigvee_{S \xrightarrow{a} S' (a = \tau_{c,m}, \tau_g)} (\exists \gamma : m \in K^{\phi}_{X,\gamma} // S').$$

The last condition (3) is more difficult to translate than the previous ones. As a matter of fact, this formulation does not directly permit us to use the induction hypothesis on S.

However, our restriction to the analysis of well-behaved processes and our requirements on the *Sort* of S and X help us. In particular, we will prove that the unique interesting possibility, for this condition, is that the initial part of the successful computation γ between S and X consists of a sequence of guessing actions by X, followed by an internal action which corresponds to the sending of a message m' from X to S.

First, note that $a=\tau_{g_1}$ for some random message g_1. Next, let γ^* be $\gamma\downarrow_X$; then, since X is well behaved, $\gamma^*=a\gamma_1' c!m'\gamma_1''$, where $\gamma_1'=\tau_{g_2}\ldots\tau_{g_n}$; moreover, the random values g_i occur in m'. By definition of well-behaved process, we have that

$$X_\phi \overset{a\gamma_1'}{\mapsto} X^1_{\phi\cup msgs(a\gamma_1')} \overset{c!m'}{\to} X'_{\phi\cup msgs(a\gamma_1')} \overset{\gamma_1''}{\mapsto}.$$

Now, consider the computation γ of $S\|_L X_\phi$. We can re-write it as $\gamma_1\tau_{c,m'}\gamma_2$ where in $a\gamma_1$ there are no interactions between S and X. So, we have

$$S\|_L X_\phi \overset{a\gamma_1}{\mapsto} S_1\|_L X^1 \overset{\tau_{c,m'}}{\to} S_2\|_L X' \overset{\gamma_2}{\mapsto}.$$

The sequence $a\gamma_1$ must be the interleaving of two sequences of actions: the ones of X, i.e. $a\gamma_1'$, and the ones of S, say γ_S. We may have two cases depending on γ_S empty or not:

- If γ_S is not empty, then we have also the following computation:

  $$S\|_L X_\phi \overset{\gamma_S}{\mapsto} S_1\|_L X_\phi \overset{a\gamma_1'\tau_{c,m'}\gamma_2}{\mapsto}$$

  and $m\in K^\phi_{X,\gamma_S a\gamma_1'\tau_{c,m'}\gamma_2}$. Thus, we can now apply the induction hypothesis, and we get $X_\phi\models\exists\gamma: m\in K^\phi_{X,\gamma}//S_1$. But, note that $\exists\gamma: m\in K^\phi_{X,\gamma}//S_1$ is a disjunct of $\exists\gamma: m\in K^\phi_{X,\gamma}//S$, which is already taken into account by condition (2) because actions in γ_S may be only internal or guessing ones.
- If γ_S is empty, then

  $$S\|_L X_\phi \overset{a\gamma_1'\tau_{c,m'}}{\mapsto} S'\|_L X'_{\phi\cup msgs(a\gamma_1')}$$

  and

  $$S'\|_L X'_{\phi\cup msgs(a\gamma_1')}\models\exists\gamma: m\in K^{\phi\cup msgs(a\gamma_1')}_{X',\gamma}.$$

  We can now apply induction hypothesis and so we obtain

  $$X'_{\phi\cup msgs(a\gamma_1')}\models(\exists\gamma: m\in K^{\phi\cup msgs(a\gamma_1')}_{X',\gamma}//S')$$

  and finally

  $$X_\phi\models\langle\tau_{g_1}\rangle\ldots\langle\tau_{g_n}\rangle\langle c!m'\rangle(\exists\gamma: m\in K^{\phi\cup msgs(a\gamma_1')}_{X',\gamma}//S').$$

By the other hand, if

$$X_\phi\models\langle\tau_{g_1}\rangle\ldots\langle\tau_{g_n}\rangle\langle c!m'\rangle(\exists\gamma: m\in K^{\phi\cup msgs(a\gamma_1')}_{X',\gamma}//S')$$

then it can be proven that a computation γ exists s.t. $S\|_L X_\phi\models m\in K^\phi_{X,\gamma}$.

So, by considering all possible initial sequences of random generation actions we have

$$\bigvee_{(c,m',\langle\langle g_i\rangle\rangle_{i\in I},S')\in RSend(S)} \langle\langle \tau_{g_i}\rangle\rangle_{i\in I}\langle c!m'\rangle(\exists\gamma : m \in K^{\phi'}_{X',\gamma'}//S'),$$

where $\phi' = \phi \cup \langle\langle g_i\rangle\rangle_{i\in I}$ and

$$\begin{aligned} RSend(S) = \{&(c,m',\langle g_1,\ldots,g_n\rangle,S') \mid \exists X_\phi \text{ s.t. } X_\phi \overset{\tau_{g_1},\ldots,\tau_{g_n}}{\mapsto} X'_{\phi'}, S \overset{c?m'}{\to} S',\\ &m' \in \mathscr{D}(\phi'), \{g_i\}_{i\in\{1,\ldots,n\}} \subseteq SubM(m')\backslash(SubM(\phi)\\ &\cup rand(S))\}. \quad \square \end{aligned}$$

Lemma A.1. *Consider a system S and assume that $\langle\langle g_i : T_i\rangle\rangle_{i\in I} \equiv_\sigma \langle\langle g'_i : T'_i\rangle\rangle_{i\in I}$. Assume also that in S there is no subterm $gen^{x,k}_T$ s.t. $\mathsf{RT}^T(k)$ is equal to $g_i : T_i$ or $g'_i : T'_i$ for some $i \in I$. Then we have*

$$S \overset{a}{\to} S_1 \quad \Rightarrow \quad \sigma(S) \overset{\sigma(a)}{\to} \sigma(S_1).$$

Proof. By structural induction on S and by inspection on the rules used to infer the transitions of S.

First, we consider that S is a sequential agent. Then we may have the following cases:

- $S = \mathbf{0}$. Trivial.
- $S = pc.A$. Then we may have the following cases depending on pc:
  - $pc = c!m$. Thus $S \overset{c!m}{\to} A$ and similarly $\sigma(S) = c!\sigma(m).\sigma(A) \overset{c!\sigma(m)}{\to} \sigma(A)$.
  - $pc = c?x : T$. For whatever $m \in Tmsgs(T)$ we have $S \overset{c?m}{\to} A[m/x]$. Since $\sigma(S) = c?x : T.\sigma(A)$, we have that $\sigma(S) \overset{c?m}{\to} \sigma(A)[m/x]$ for whatever $m \in Tmsgs(T)$. Thus, also $\sigma(S) \overset{c?\sigma(m)}{\to} \sigma(A)[\sigma(m)/x] = \sigma(A[m/x])$.
  - $pc = \chi^x_{c,T}$. Similar to the case above.
  - $pc = gen^{x,i}_T$. We have that $S \overset{\tau_g}{\to} A[g/x]$, where $g = \mathsf{RT}^T(i)$, and $\sigma(S) = gen^{x,i}_T.\sigma(A)$. Due to our hypothesis, g is different from the random messages in $\{g_i : T_i\}_{i\in I} \cup \{g'_i : T'_i\}_{i\in I}$. Hence, $\sigma(g) = g$ (by our assumptions on σ) and $\sigma(S) \overset{\tau_g}{\to} \sigma(A)[g/x] = \sigma(A[g/x])$.
- $S = A_1 + A_2$. If $S \overset{a}{\to} A'$ it means that either $A_1 \overset{a}{\to} A'$ or $A_2 \overset{a}{\to} A'$. In both cases, by structural induction, the thesis follows.
- $S = [m = m']A_1; A_2$. We have that $\sigma(S) = [\sigma(m) = \sigma(m')]\sigma(A_1); \sigma(A_2)$. If $S \overset{a}{\to} A'$, it must be that either $m = m'$ and $A_1 \overset{a}{\to} A'$ or $m \neq m'$ and $A_2 \overset{a}{\to} A'$. Now, if $m = m'$ then also $\sigma(m) = \sigma(m')$ and by structural induction the thesis follows. If $m \neq m'$ then we have also $\sigma(m) \neq \sigma(m')$ and once again the thesis follows by structural induction.
- $S = [\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} x : T]A_1; A_2$. We have that $\sigma(S) = [\langle\langle \sigma(m_i)\rangle\rangle_{i\in I} \vdash_{IS} x : T]\sigma(A_1); \sigma(A_2)$. If $S \overset{a}{\to} A'$, it must be that either $\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} m : T$ and $A_1[m/x] \overset{a}{\to} A'$ or for no $m : T$ we have $\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} m : T$ and $A_2 \overset{a}{\to} A'$. Now, if $\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} m : T$ then, by Assumption 4.4, we have $\langle\langle \sigma(m_i)\rangle\rangle_{i\in I} \vdash_{IS} \sigma(m)$; by structural induction the thesis follows since $\sigma(A)[\sigma(m)/x] = \sigma(A[m/x])$. Analogously, if for no $m : T$ we have that

$\langle\langle m_i\rangle\rangle_{i\in I} \vdash_{IS} m : T$ then for no $\sigma(m)$ we have $\langle\langle\sigma(m_i)\rangle\rangle_{i\in I} \vdash_{IS} \sigma(m : T)$ and the result follows by structural induction.

The case where S is $S'\backslash L$ is trivial. Now consider that $S = S_1 \| S_2$.

- If $S \xrightarrow{a} S'$ by means of the operational rule $\|_1$ (or its symmetric) the result trivially follows.
- If $S \xrightarrow{\tau_{c,m}} S'$ by means of the rule $\|_2$ (communication) then it means that $S_1 \xrightarrow{c?m} S_1'$ and $S_2 \xrightarrow{c!m} S_2'$. By structural induction, we get $\sigma(S_1) \xrightarrow{c?\sigma(m)} \sigma(S_1')$ and $\sigma(S_2) \xrightarrow{c!\sigma(m)} \sigma(S_2')$. Thus the thesis follows since, by the operational rule $\|_2$, we have $\sigma(S) \xrightarrow{\tau_{c,\sigma(m)}} \sigma(S')$. The symmetric case is similar.
- If $S \xrightarrow{a} S'$ by means of the rule $\|_\chi$ (or its symmetric) the result similarly follows as above. □

Lemma 4.2. *Consider a system S and assume that $\langle\langle g_i : T_i\rangle\rangle_{i\in I} \equiv_\sigma \langle\langle g_i' : T_i'\rangle\rangle_{i\in I}$. Assume also that in S there is no subterm $gen_T^{x,k}$ s.t. $\mathsf{RT}^T(k)$ is equal to $g_i : T_i$ or $g_i' : T_i'$ for some $i \in I$. Then, we have*

$$S \overset{\gamma}{\mapsto} S_1 \quad \Rightarrow \quad \sigma(S) \overset{\sigma(\gamma)}{\mapsto} \sigma(S_1).$$

Proof. By induction on the length of the computation γ and by exploiting Lemma A.1. □

Lemma 4.3. *Let $\langle\langle g_i : T_i\rangle\rangle_{i\in I} \equiv_\sigma \langle\langle g_i' : T_i'\rangle\rangle_{i\in I}$; consider a system S and a sequential agent X, with $Sort(S\|X) \subseteq L$, and let no subterm $gen_T^{x,k}$ in $S\|X$ be s.t. $\mathsf{RT}^T(k)$ is equal to $g_i : T_i$ or $g_i' : T_i'$ for some $i \in I$; furthermore, assume $SubM(\phi) \cap (\{g_i : T_i\}_{i\in I} \cup \{g_i' : T_i'\}_{i\in I}) = \{\}$. If $m : T$ is an initial message, then:*

$$S\|_L X_{\phi\cup\{g_i:T_i\}_{i\in I}} \models \exists\gamma : m \in K_{X,\gamma}^{\phi\cup\{g_i:T_i\}_{i\in I}}$$
$$\text{iff}$$
$$\sigma(S)\|_L \sigma(X)_{\phi\cup\{g_i':T_i'\}_{i\in I}} \models \exists\gamma : m \in K_{\sigma(X),\gamma}^{\phi\cup\{g_i':T_i'\}_{i\in I}}.$$

Proof. If $S\|_L X_{\phi\cup\{g_i:T_i\}_{i\in I}} \models \exists\gamma : m \in K_{X,\gamma}^{\phi\cup\{g_i:T_i\}_{i\in I}}$ then there exists γ such that

$$(S\|_L X_{\phi\cup\{g_i:T_i\}_{i\in I}} \overset{\gamma}{\mapsto}) \downarrow_X = \bar{\gamma}$$

with $m : T \in \mathscr{D}(\phi \cup \{g_i : T_i\}_{i\in I} \cup msgs(\bar{\gamma}))$. We are in the hypothesis of Lemma 4.2 so we have $\sigma(S)\|_L \sigma(X)_{\phi\cup\{g_i':T_i'\}_{i\in I}} \overset{\gamma'}{\mapsto}$ with $\gamma' = \sigma(\gamma)$. Hence, we can see that $(\sigma(S)\|_L \sigma(X)_{\phi\cup\{g_i':T_i'\}_{i\in I}} \overset{\gamma'}{\mapsto}) \downarrow_{\sigma(X)} = \overline{\gamma'} = \sigma(\bar{\gamma})$. So we have

$$\begin{aligned}
&m : T \in \mathscr{D}(\phi \cup \{g_i : T_i\}_{i\in I} \cup msgs(\bar{\gamma})) && \text{iff (Assumption 4.4)}\\
&\sigma(m : T) \in \mathscr{D}(\sigma(\phi \cup \{g_i : T_i\}_{i\in I} \cup msgs(\bar{\gamma}))) && \text{iff}\\
&\sigma(m : T) \in \mathscr{D}(\sigma(\phi) \cup \sigma(\{g_i : T_i\}_{i\in I}) \cup \sigma(msgs(\bar{\gamma}))) && \text{iff}\\
&m : T \in \mathscr{D}(\phi \cup \{g_i' : T_i'\}_{i\in I} \cup msgs(\overline{\gamma'})).
\end{aligned}$$

Finally, we have $\sigma(S)\|_L\sigma(X)_{\phi\cup\{g_i':T_i'\}_{i\in I}} \models \exists\gamma : m \in K_{\sigma(X),\gamma}^{\phi\cup\{g_i':T_i'\}_{i\in I}}$. The other direction can be proved by using a symmetric reasoning. □

Lemma 4.4. *$RSend(S)/_{\equiv^e}$ is a finite set.*

Proof. Consider two tuples $t, t' \in RSend(S)$, i.e.

$$t = (c, m : T, \langle\langle g_i : T_i\rangle\rangle_{i\in I}, S'), \quad t' = (c', m' : T', \langle\langle g_i' : T_i'\rangle\rangle_{i\in I'}, S_1').$$

Assume that $c = c'$, $T = T'$, $I = I'$ and that the sequences of the types of the random values guessed, i.e. $\langle\langle T_i\rangle\rangle_{i\in I}$ and $\langle\langle T_i'\rangle\rangle_{i\in I}$, are equal up to permutations (they represent the same multi-set of basic types). Moreover, assume that $S \xrightarrow{c?m} S'$, because in S there is an agent $A = c?x : T.A'$, which performs the action $A \xrightarrow{c?m} A'[m/x]$, and S_1' is equal to S' where $A'[m/x]$ is replaced with $A'[m'/x]$. (Roughly, the two derivatives S' and S_1' are due to the same sequential agent which performs a receiving action.)

Note that, if two messages, say m and m', with the same type differ, it means that at least an occurrence of a basic value of m is replaced in m' with an occurrence of a different basic value (but with the same type).

Now, we decorate the term representing the type T (T') of m (m') as follows:

- We color $black_b$ each basic type in the term of T, whose corresponding message of basic value b in m (m') is not in $\{g_i\}_{i\in I}$ ($\{g_i'\}_{i\in I}$).
- We color $white_i$ each basic type in the term of T, whose corresponding message in m (m') is the basic value g_i (g_i') in $\{g_i\}_{i\in I}$ ($\{g_i'\}_{i\in I}$).

We say that two colored terms T, T' representing the same type are compatible iff

- *Black condition*: If t_1 is a term of a basic type occurring in T, which is colored $black_b$ for some b, then the corresponding term in T' is colored $black_b$, and vice-versa;
- *White condition*: If two occurrences t_1, t_2 of terms of basic types have the same color in T then the corresponding occurrences t_1', t_2' in T' have also the same color (even if it could be different from the one of t_1, t_2), and vice-versa;

The *black* condition ensures that the two messages m, m', which originate the coloring of the two terms T, T' of the same type, agree on basic values which are not in $\{g_i, g_i'\}_{i\in I}$. The *white* condition ensures that equal random values in m correspond to equal random values in m'.

Eventually, under the previous assumptions and by assuming also that the colored terms of T and T' are compatible, we can find a bijection σ, s.t.

(1) $\langle\langle g_i : T_i\rangle\rangle_{i\in I} \equiv_\sigma \langle\langle g_i' : T_i'\rangle\rangle_{i\in I}$;
(2) $\sigma(m) = m'$;
(3) $\sigma(S_1) = S_1'$.

We build σ as follows. For $g \in \{g_i\}_{i\in I}$ let $\sigma(g) = g'$ if g' is the corresponding value of g in m'. For $g' \in \{g_i'\}_{i\in I}$ let $\sigma(g') = g$ if $\sigma(g) = g'$. Let $\sigma(g)$ be g if $g \notin \{g_i, g_i'\}_{i\in I}$. Clearly, σ is a type preserving function. We can also prove that σ is injective. In particular, consider $g_k, g_h \in \{g_i\}_{i\in I}$, with $k \neq h$, then clearly, $white_k$ is different from $white_h$. (Recall that the sequence of guessing consists of distinct random values.) Thus,

since T and T' are compatible, the color of the term corresponding to $\sigma(g_k)$ and the one of $\sigma(g_h)$ are not equal, but this means that $\sigma(g_k) \neq \sigma(g_h)$. This proves the first of the above points. To show that (2) holds, note that m' is the term m where each occurrence of g is replaced with $\sigma(g)$, thus $\sigma(m) = m'$. To show that (3) holds, note that S_1 and S_1' differ because there is an agent is S_1 which is $A'[m/x]$ whereas the corresponding agent in S_1' is $A'[m'/x]$. The random values in $\{g_i, g_i'\}_{i\in I}$ do not occur in S, by construction. Thus, $\sigma(S')$ is S' where m is replaced with $\sigma(m) = m'$, and so $\sigma(S') = S_1'$. From the previous facts we get $t \equiv^e t'$.

On the other hand, if $t \equiv^e t'$ then it is possible to prove that the colorings of the type of the exchanged messages, say m and m', are compatible.

From this follows that, in order to establish if two tuples in $RSend(S)$ are equivalent, we simply need to establish that, the channels, the type of the messages, the multi-sets of basic types guessed are the same in the two tuples. Moreover, the derivatives of S are obtained through an action of the same sequential agent; finally, the colored terms of the type of the exchanged messages m and m' are compatible.

Note that the channels and types in S are finite. Fix a channel c and the corresponding type T. Now, we may have only a finite number of possible multi-sets of basic types which represents the occurrences of newly guessed random values in the message m with type T. Fix a sequence, and so the index set I is fixed. Then, we may have only a finite number of ways of coloring the term of the type T, with colors in $black_b$, with b basic value and $b \in \mathscr{D}(\phi)$, and $white_i$, with $i \in I$ and I finite, because the term of T is finite and the possible colors are finite. (Recall that, even though not explicitly represented, the set $RSend(S)$ depends on ϕ. This set is finite by Assumption 4.3; moreover, by Assumption 4.2, it is not possible to deduce basic values which are not in ϕ.) Thus, the compatibility relation among colorings of types, which is an equivalence relation, may have only a finite set of equivalence classes. Thus, it follows that we may have only a finite number of equivalence classes in $RSend(S)_{\equiv^e}$. □

Lemma 4.5. *Under the hypothesis of Proposition* 4.3, *let* $F = \exists\gamma : m_1 \in K^{\phi}_{X,\gamma}//S$. *Then, we have*

$$\exists X_\phi \text{ s.t. } X_\phi \models F \quad \textit{iff} \quad \exists Y_\phi \text{ s.t. } Y_\phi \models \tilde{F}.$$

Proof. By induction on the depth of the nesting of infinitary disjunctions in F.

($\Rightarrow$) If $\exists X_\phi \models F$ then either X models a formula in the finite part of F and, in this case, also of $\tilde{F}$, or there exists $t = (c, m : T, \langle\langle g_i : T_i\rangle\rangle_{i\in I}, S_1) \in RSend(S) : X_\phi \models F_t$, where F_t is

$$\langle\langle \tau_{g_i}\rangle\rangle_{i\in I}\langle c!m\rangle(\exists\gamma : m_1 \in K^{\phi\cup\langle\langle g_i:T_i\rangle\rangle_{i\in I}}_{X,\gamma}//S_1).$$

This fact implies that there exists X_ϕ s.t.

$$X_\phi \overset{\tau_{g_1},\ldots,\tau_{g_n}}{\mapsto} X'_{\phi'} \overset{c!m}{\to} X''_{\phi'} \wedge X''_{\phi'} \models \exists\gamma : m_1 \in K^{\phi'}_{X,\gamma}//S_1$$

with $S \overset{c?m}{\to} S_1$, $\phi' = \phi \cup \{g_1 : T_1, \ldots, g_n : T_n\}$, $m : T \in \mathscr{D}(\phi')$ and $\{g_i : T_i\}_{i\in I} \subseteq SubM(m : T)$.

There must be the case for some $t'=(c',m':T,\langle\langle g'_i:T'_i\rangle\rangle_{i\in I},S'_1)\in[t]_{\equiv^e}$ that $F_{t'}$ appears as a disjunction in $\tilde{F}$. Thus, we will show that there exists a process X^1_ϕ s.t. $X^1_\phi\models F_{t'}$ and consequently $X^1_\phi\models\tilde{F}$.

Let σ be a type preserving bijection between s.t. $\langle\langle g_i:T_i\rangle\rangle_{i\in I}\equiv_\sigma\langle\langle g'_i:T'_i\rangle\rangle_{i\in I}$, with $I=\{1,\ldots,n\}$ (such bijection must exist by construction, since $t'\equiv^e t$). Hence it is possible to construct a process X^1_ϕ such that

$$X^1_\phi \overset{\tau_{g'_1},\ldots,\tau_{g'_n}}{\mapsto} X'^1_{\phi'_1}$$

and $X'^1_{\phi'_1}=\sigma(X'_{\phi'})$. By Lemma 4.1 we can freely assume that X is in the form

$$gen^{v_1,i_1}_{T_1}.\ldots.gen^{v_n,i_n}_{T_n}.Z.$$

We simply consider X^1_ϕ as

$$gen^{v'_1,i'_1}_{T'_1}.\ldots.gen^{v'_n,i'_n}_{T'_n}.Z,$$

where $\mathscr{R}^{T'_k}(i'_k)=g'_k$ and $v'_k=v_j$ when $\sigma(g_j:T_j)=g'_k:T'_k$.

Since, $X'_{\phi'}\overset{c!m}{\to}X'_{\phi'}$ and $X'^1_{\phi'_1}=\sigma(X'_{\phi'})$ it must be that for some $X''^1_{\phi'_1}$ we have $X'^1_{\phi'_1}\overset{c!m'}{\to}X''^1_{\phi'_1}$, with $X''^1_{\phi'_1}=\sigma(X''_{\phi''})$. We also have $\sigma(S_1)=S'_1$, $\sigma(m)=m'$ and $\sigma(\langle\langle g_i\rangle\rangle_{i\in I})=\langle\langle g'_i\rangle\rangle_{i\in I}$. The hypothesis of Lemma 4.3 are fulfilled and so $S_1\|X''_{\phi'}\models\exists\gamma:m_1\in K^{\phi'}_{X,\gamma}$ implies $S'_1\|_L X''^1_{\phi'_1}\models\exists\gamma:m_1\in K^{\phi'_1}_{X,\gamma}$ and by Proposition 4.3 we get $X''^1_{\phi'_1}\models\exists\gamma:m_1\in K^{\phi'_1}_{X,\gamma}//S'_1$. The depth of the nesting of infinitary disjunctions in the latter formula is smaller than in the formula $\exists\gamma:m_1\in K^{\phi}_{X,\gamma}//S$, hence by applying the induction hypothesis, we get $\exists Y'_{\phi'_1}\models\exists\gamma:m_1\in\tilde{K}^{\phi'_1}_{Y',\gamma}//S'_1$. Finally, the thesis follows by considering Y_ϕ as

$$gen^{v'_1,i'_1}_{T'_1}.\ldots.gen^{v'_n,i'_n}_{T'_n}.p.c!y.Y'',$$

where p is a proof of m' from $\phi\cup\{g'_i:T'_i\}_{i\in I}$ whose root is an assignment to the variable y, and each value g'_i is replaced with v'_i; similarly, Y'' is the process Y', where each value g'_i is replaced with v'_i and m' is replaced with y. Note that $Y_\phi \overset{\tau_{g'_1}\ldots\tau_{g'_1}c!m'}{\mapsto} Y'_{\phi'_1}$.

($\Leftarrow$) The other direction is trivial. □

Theorem 4.1. *Consider a system S, with $Sort(S)\subseteq L$, a finite set of typed messages ϕ and an initial message $m:T$. It is decidable if there exists X_ϕ, with $Sort(X)\subseteq L$, s.t.*

$$S\|_L X_\phi\models\exists\gamma:m\in K^{\phi}_{X,\gamma}.$$

Proof. By using Proposition 4.3 and Lemma 4.5 we can reduce the decidability of the existence of such X_ϕ to a satisfiability problem of the reduced formula $\tilde{F}=\exists\gamma:m\in\tilde{K}^{\phi}_{X,\gamma}//S$. Note that such formula consists only of the logical constants, disjunctions and the possibility modality. Thus, we can apply Lemma 3.1 and the result follows. □

Theorem 4.2. *Consider a system S, with $Sort(S) \subseteq L$ and no construct $\chi^x_{c,T}$ occurring in it, a system $Y = \|_{k\in\{1,\ldots,l\}} Y^k_{\phi_k}$ consisting of a set of sequential agents Y_k with $Sort(Y^k_{\phi_k}) \subseteq L$ for $k \in \{1,\ldots,l\}$, and an initial message $m : T$. Suppose that, for some $i \in \{1,\ldots,l\}$, we have*

$$S\|_L Y \models \exists\gamma : m \in K^{\phi_i}_{Y^i,\gamma}$$

then there exists a sequential agent X_ϕ, with $\phi = \bigcup_{k\in\{1,\ldots,l\}} \phi_k$, s.t.:

$$S\|_L X_\phi \models \exists\gamma : m \in K^{\phi}_{X,\gamma}.$$

Proof. By induction on the minimal length n of sequences γ, s.t. $S\|_L\|_{k\in\{1,\ldots,l\}} Y^k_{\phi_k} \overset{\gamma}{\mapsto}$ and $\gamma\downarrow_{Y^i} = \gamma'$, with $m \in \mathscr{D}(msgs(\gamma') \cup \phi_i)$.

- $n = 0$. Then, let X be Y^i.
- *Induction step*. Then, consider $\gamma = a\gamma_1$ s.t.

$$S\|_L\|_{k\in\{1,\ldots,l\}} Y^k_{\phi_k} \xrightarrow{a} S'\|_L\|_{k\in\{1,\ldots,l\}} Y'^k_{\phi'_k} \overset{\gamma_1}{\mapsto} S''\|_L\|_{k\in\{1,\ldots,l\}} Y''^k_{\phi''_k}$$

  and $\gamma\downarrow_{Y^i} = \gamma'$, with $m \in \mathscr{D}(msgs(\gamma') \cup \phi_i)$. By induction hypothesis, we have that there exists $X'_{\phi'}$ with $\phi' = \bigcup_{k\in\{1,\ldots,l\}} \phi'_k$, s.t. $S'\|_L X' \models \exists\gamma : m \in K^{\phi'}_{X',\gamma}$. We may have different cases depending on the agents involved in the execution of a.
  - Only agents in S participated on the execution of a. Then, note that $\phi' = \phi$ and let X be X'.
  - Only agents in $\|_{k\in\{1,\ldots,l\}} Y^k_{\phi_k}$ participated on the execution of a. We may have two cases:
    $a = \tau_g$. Then, for some $j \in \mathbb{N}$ we have $g = \mathscr{R}^T(j)$. Let X be $(gen^{x,j}_T.X_1)_\phi$, where X_1 is the agent X' where each occurrence of the random value g replaced with x. (We assume that x does not appear in X'.)
    $a = \tau_{c,m'}$. Then, note that $\phi' = \phi$. So, let X be X'.
  - One agent of S and one of $\|_{k\in\{1,\ldots,l\}} Y^k_{\phi_k}$, say $Y^{l'}$, synchronize. We may have several cases depending on the action performed by $Y^{l'}$:
    $c!m'$. Then, note that $\phi' = \phi$. So, let X be $c!m'.X'$.
    $c?m' : T'$. Then, let X be $(c?x : T'.X_1)_\phi$, where X_1 is the agent X' where each occurrence of m' replaced with x. (We assume that x does not appear in X'.)
    $\chi_{c,m':T'}$. Then, let X be $(\chi^x_{c,T'}.X_1)_\phi$, where X_1 is the agent X' where each occurrence of m' replaced with x. (We assume that x does not appear in X'.) □

References

[1] M. Abadi, Secrecy by typing in security protocols, J. ACM 46 (5) (1999) 749–786.
[2] M. Abadi, A.D. Gordon, A calculus for cryptographic protocols: the spi calculus, Inform. Comput. 148 (1) (1999) 1–70.
[3] M. Abadi, M.R. Tuttle, A semantics for a logic of authentication, in: Proc. 10th Ann. ACM Symp. on Principles of Distributed Computing, ACM Press, New York, 1991, pp. 201–216.

[4] R.M. Amadio, D. Lugiez, On the reachability problem in cryptographic protocols, in: CONCUR '00, Lecture Notes in Computer Science, Vol. 1877, Springer, Berlin, 2000.
[5] H.R. Andersen, Partial model checking (extended abstract), in: Proc. 10th Ann. IEEE Symp. Logic in Computer Science, IEEE Computer Society Press, Silver Spring, MD, 1995, pp. 398–407.
[6] C. Bodei, P. Degano, F. Nielson, H.R. Nielson, Static analysis for the pi-calculus with applications to security, Inform. Comput. 168 (2001) 68–92.
[7] M. Boreale, R. De Nicola, Testing equivalence for mobile processes, Inform. Comput. 120 (2) (1995) 279–303.
[8] M. Boreale, R. De Nicola, R. Pugliese, Proof techniques for cryptographic processes, in: Proc. 14th Ann. Symp. on Logic in Computer Science (LICS) (Trento, Italy), IEEE Computer Society Press, Silver Spring, MD, 1999, pp. 157–166.
[9] L. Cardelli, A. Gordon, Mobile ambients, in: Proc. Foundations of Software Science and Computation Structures, Lectures Notes in Computer Science, Vol. 1378, Springer, Berlin, 1998, pp. 140–155.
[10] P. Degano, C. Priami, Enhanced operational semantics: a tool for describing and analysing concurrent systems, ACM Comput. Surveys 133 (2) 135–176.
[11] D. Dolev, A. Yao, On the security of public key protocols, IEEE Trans. Inform. Theory 29 (12) (1983) 198–208.
[12] R. Focardi, R. Gorrieri, A classification of security properties, J. Comput. Security 3 (1) (1995) 5–33.
[13] R. Focardi, R. Gorrieri, The compositional security checker: a tool for the verification of information flow security properties, IEEE Trans. Software Eng. 27 (1997) 550–571.
[14] R. Focardi, R. Gorrieri, F. Martinelli, A comparison of three authentication properties, Theoret. Comput. Sci., accepted for publication.
[15] R. Focardi, R. Gorrieri, F. Martinelli, Information flow analysis in a discrete-time process algebra, in: Proc. 13th IEEE Computer Security Foundations Workshop, IEEE Press, New York, 2000, pp. 170–184.
[16] R. Focardi, R. Gorrieri, F. Martinelli, Non interference for the analysis of cryptographic protocols, in: Proc. 27th Internat. Colloq. in Automata, Languages and Programming, Lectures Notes in Computer Science, Vol. 1853, Springer, Berlin, 2000, pp. 354–372.
[17] R. Focardi, F. Martinelli, A uniform approach for the definition of security properties, in: Proc. World Congress on Formal Methods (FM'99), Lecture Notes in Computer Science, Vol. 1708, Springer, Berlin, 1999, pp. 794–813.
[18] J.W. Gray III, J. McLean, Using temporal logic to specify and verify cryptographic protocols, in: Proc. 8th Computer Security Foundations Workshop, IEEE Computer Society Press, Silver Spring, MD, 1995.
[19] J. Heather, G. Lowe, S. Schneider, How to prevent type flow attacks on security protocols, in: Proc. 13th IEEE Computer Security Foundations Workshop, IEEE Press, New York, 2000, pp. 255–268.
[20] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, Englewood Cliffs, NJ, 1985.
[21] D. Kindred, J.M. Wing, Fast, automatic checking of security protocols, in: Proc. 2nd USENIX Workshop on Electronic Commerce, Oakland, CA, 1996, pp. 41–52.
[22] D. Kozen, Results on the propositional μ-calculus, Theoret. Comput. Sci. 27 (3) (1983) 333–354.
[23] S. Kremer, J.-F. Raskin, Formal verification of non-repudiation protocols: a game approach, in: Proc. Workshop in Formal Methods for Computer Security, June 2000.
[24] O. Kupferman, M.Y. Vardi, Module checking, in: R. Alur, T.A. Henzinger (Eds.), Proc. 8th Internat. Conf. on Computer Aided Verification, Lecture Notes in Computer Science, Vol. 1102, Springer, Berlin, 1996, pp. 75–86.
[25] K.G. Larsen, L. Xinxin, Compositionality through an operational semantics of contexts, J. Logic Comput. 1 (6) (1991) 761–795.
[26] G. Leduc, F. Germeau, Verification of security protocols using LOTOS—method and application, Comput. Comm. 23 (12) (2000) 1089–1103.
[27] G. Lowe, Breaking and fixing the Needham Schroeder public-key protocol using FDR, in: Proc. Tools and Algorithms for the Construction and the Analysis of Systems, Lecture Notes in Computer Science, Vol. 1055, Springer, Berlin, 1996, pp. 147–166.
[28] G. Lowe, Towards a completeness result for model checking of security protocols, in: Proc. 11th IEEE Computer Security Foundations Workshop, IEEE, New York, 1998, pp. 96–105.

[29] G. Lowe, B. Roscoe, Using CSP to detect errors in the TMN protocol, IEEE Trans. Software Eng. 23 (10) (1997) 659–669.
[30] D. Marchignoli, F. Martinelli, Automatic verification of cryptographic protocols through compositional analysis techniques, in: Proc. Internat. Conf. on Tools and Algorithms for the Construction and the Analysis of Systems (TACAS'99), Lecture Notes in Computer Science, Vol. 1579, Springer, Berlin, 1999.
[31] W. Marrero, E. Clarke, S. Jha, A model checker for authentication protocols, in: H. Orman, C. Meadows (Eds.), Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols, DIMACS Center, Rutgers University, September 1997.
[32] F. Martinelli, About compositional analysis for (finite) π-calculus processes, Proceedings of Foundations of Information Technology in the Era of Networking and Mobile Computing. IFIP 17th World Computer Congress, TC1 Stream, Kluwer, 2002, pp. 524–536.
[33] F. Martinelli, Encoding several security properties as secrecy ones, Technical Report IAT-B4-2001-20.
[34] F. Martinelli, Formal methods for the analysis of open systems with applications to security, Ph.D. Thesis, University of Siena, December 1998.
[35] F. Martinelli, Languages for description and analysis of authentication protocols, in: Proc. 6th Italian Conf. on Theoretical Computer Science, World Scientific, Singapore, 1998, pp. 304–315.
[36] F. Martinelli, Partial model checking and theorem proving for ensuring security properties, in: Proc. 11th Computer Security Foundations Workshop, IEEE Computer Society Press, Silver Spring, MD, 1998, pp. 44–52.
[37] C. Meadows, The NRL protocol analyzer: an overview, J. Logic Programming 26 (2) (1996) 113–131.
[38] R. Milner, Communication and Concurrency, International Series in Computer Science, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[39] R. Milner, J. Parrow, D. Walker, A calculus of mobile processes, Inform. Comput. 100 (1) (1992) 1–77.
[40] J.C. Mitchell, M. Mitchell, U. Stern, Automated analysis of cryptographic protocols using murphi, in: Proc. Symp. on Security and Privacy, IEEE Computer Society Press, Silver Spring, MD, 1997, pp. 141–153.
[41] H.R. Nielson, F. Nielson, Shape analysis for mobile ambients, in: Proc. POPL'00, ACM Press, New York, 2000, pp. 142–154.
[42] L.C. Paulson, Proving properties of security protocols by induction, in: Proc. 10th Computer Security Foundations Workshop, IEEE Computer Society Press, Silver Spring, MD, 1997.
[43] A.W. Roscoe, M.H. Goldsmith, The perfect spy for model-checking crypto-protocols, in: H. Orman, C. Meadows (Eds.), Proc. DIMACS Workshop on Design and Formal Verification of Security Protocols, DIMACS Center, Rutgers University, September 1997.
[44] F.B. Schneider, Applied Cryptography, Wiley, New York, 1996.
[45] S. Schneider, Verifying authentication protocols with CSP, in: Proc. 10th Computer Security Foundations Workshop, IEEE Computer Society Press, Silver Spring, MD, 1997.
[46] C. Stirling, Modal and temporal logics for processes, in: Logics for Concurrency: Structures versus Automata, Lecture Notes in Computer Science, Vol. 1043, Springer, Berlin, 1996, pp. 149–237.
[47] S. Stoller, A reduction for automated verification of authentication protocols, in: Workshop on Formal Methods and Security Protocols (FMSP'99), Trento, Italy, 1999.
[48] J. Thayer, J. Herzog, J. Guttman, Honest ideals on strand spaces, in: Proc. 11th IEEE Computer Security Foundations Workshop, IEEE Press, New York, 1998, pp. 66–78.