Security properties: two agents are sufficient

Hubert Comon-Lundh and Véronique Cortier**

Laboratoire Spécification et Vérification, CNRS, INRIA
Ecole Normale Supérieure de Cachan,
{comon,cortier}@lsv.ens-cachan.fr

Abstract. We consider an important family of cryptographic protocols and a class of security properties which encompasses secrecy and authentication. We show that it is always sufficient to consider a bounded number of agents b ($b = 2$ for secrecy properties for example): if there is an attack involving n agents, then there is an attack involving at most b agents.

1 Introduction

The task of automatically verifying cryptographic protocols has now been undertaken by several research groups, because of its relevance to secure internet transactions. Let us cite for instance (this is far from being exhaustive): CAPSL [15], CASRUL/Datac [21], casper/FDR [29].

Though cryptographic protocols are often described in a concise way (see e.g. [8]), the verification problem is difficult for two reasons:

1. The protocol descriptions contain free variables called *roles*, which may be instanciated by any identities thereafter called *agents*. A protocol may be executed several times: we get several protocol *sessions*. Both the number of agents and the number of sessions are unbounded.
2. We assume the presence of an attacker which may record and analyze the messages sent through the net and which may forge and send new messages. The size of messages which can be forged by an intruder is also unbounded.

And, in fact, even for simple properties such as secrecy and for subclasses of protocols, the verification problem is undecidable (see e.g. [17, 16, 10, 3]).

The verification tools have either to assume stronger properties on the protocols (e.g. [23, 11, 30, 3]) or to consider a bounded number of sessions (hence a bounded number of agents) only [4, 28, 18, 25], in which case the security problem becomes co-NP-complete [28]. Yet another solution is to consider an upper approximation of the set of execution sequences, in such a way that, when no attack is found on this upper approximation, then there is no attack on the protocol. This is typically the approach of [7, 6].

In this paper, we consider a simple reduction, which works for any protocols and security properties typically considered for automated verification. We show

** Partially supported by INRIA project SECSI and RNTL project EVA.

that it is always sufficient to consider a bounded number of agents b (actually $b = 2$ for secrecy properties; we will discuss this point later): if there is an attack involving n agents, then there is an attack involving at most b agents. Such a result is useful for automatic tools: we may forget the universal quantifications on agent ids and consider finitely many instances of the protocol roles, without loosing information. This actually proves that the instanciation techniques in [21] are complete. This also provides completeness results for abstraction used in [7, 26]. Of course, the verification problem will remain undecidable, because we cannot faithfully give a bound on the number of sessions. Still, approximation techniques such as [6, 7] can be simplified and when considering a bounded number of sessions we may assume w.l.o.g. that only these b agents are involved. This reduction result also provides with a decision result for cryptographic protocols against a passive intruder, provided the messages are labeled by a nonce session and a rule number (which is often the case of implemented protocols).

Our result extends and clarifies a side result of [20]. Indeed, J. Heather and S. Schneider show that one may consider only four agents (three honest, one dishonest) using implicitly that an agent may talk to herself. We actually prove that in J. Heather and S. Schneider setting, only two agents are sufficient. In addition, our reduction result holds for more general security properties and also holds when an agent is disallowed to speak with herself.

The proof of our result is not difficult, once the protocol and its properties are expressed as Horn clauses: given an attack against a secrecy property, we simply project every honest identity on one single honest identity and every dishonest identity on one single dishonest identity. For more general security properties like authentication, we need $k + 1$ agents where k is the number of agents variables used in the security property. Actually, the result can be stated for a class of Horn clauses, which encompasses protocols descriptions. Everybody has her (his) favorite model. We do not argue that the Horn clause model is better than others. However, this model has two advantages for the setting of our result:

1. We do not need to define precisely the intruder power: our result holds for any intruder provided it satisfies some slight hypotheses, defined in our model. In particular, our reduction also holds considering the usual equational theories associated to the exclusive or, modular exponentiation...
2. For conditional security properties of the form “if ϕ then ψ”, we show that the reduction still holds when the negation of the property ψ can be expressed in Horn clauses, which is the case for authentication. Horn clauses are therefore an appropriate model to express the class of properties for which the reduction holds.

We claim that most other models can be reduced to this one, hence our reduction also applies to other models of cryptographic protocols. In order to support this claim, we provide (in [13]) with a reduction of the Millen-Rueß model [24] to Horn clauses, while keeping the number of agents involved in an attack. We hope that this will provide with enough evidence that the reduction result works for other models as well. (It is not possible to show in detail all reductions from other models to Horn clauses).

Our paper is organized as follows. We introduce our model in section 2. In section 3.1 we prove that, if there is an attack involving n agents, then there is an attack involving at most 2 agents, besides the constant agents which might be used in the protocol description. In other words, we show that we have to consider only instances of the roles in a two-element set. This is quite intuitive since the protocols never refer to the representation of agents identities: they are indeed irrelevant when verifying properties of cryptographic protocols. However, the formal proof of this result reveals two assumptions. First, the same agent may play different roles in a given protocol session: "an agent may talk to herself". Most of the models do not discard this ability. However, it may be considered as more realistic that an agent cannot play several roles in the same session. Some models [27, 23] explicitly disallow this possibility. In addition, this reduction result holds only for properties expressed as purely negative clauses, which is not the case of authentication properties. That is why we consider in section 3.2 more general security properties and models in which an agent may be forbidden to talk with herself. We prove in this case that, if there is an attack involving n agents, then there is an attack involving at most $k+1$ agents where k is the number of agent variables occurring in the security property.

2 The model

We define a trace model by means of Horn clauses, in which terms are messages. A similar representation can also be found in [6] for instance. The important feature is that we only use Horn clauses, which contain at least one positive literal. Hence there is a least Herbrand model $\mathcal{H}$, which is the intended trace semantics: the possible traces are the members of $T_{\mathcal{H}}$, the interpretation of the unary predicate T in $\mathcal{H}$. Clauses come in two parts: the first part is protocol independent and the second part is protocol specific.

2.1 Messages and traces

We consider a set of (ground) terms built over a set of function symbols $\mathcal{F}$ and basic sorts: $\mathsf{Num}, \mathsf{Agent}, \mathsf{Ha}, \mathsf{Da}, \mathsf{Message}, \mathsf{Event}, \mathsf{Trace}$. $\mathcal{F}$ contains the following function symbols:

$$\begin{array}{rrl@{\qquad}rrl}
0 : & & \to \mathsf{Num} & n_i : & \mathsf{Agent}^{k_i}, \mathsf{Num} & \to \mathsf{Message} \\
s : & \mathsf{Num} & \to \mathsf{Num} & st : & \mathsf{Agent}, \mathsf{Num}, \mathsf{Num}, \mathsf{Message} & \to \mathsf{Event} \\
h : & & \to \mathsf{Ha} & \bot : & & \to \mathsf{Trace} \\
d : & & \to \mathsf{Da} & [_,_]\cdot_ : & \mathsf{Event}, \mathsf{Num}, \mathsf{Trace} & \to \mathsf{Trace} \\
s_h : & \mathsf{Ha} & \to \mathsf{Ha} & srv_i & & \to \mathsf{Agent} \\
s_d : & \mathsf{Da} & \to \mathsf{Da} & & &
\end{array}$$

Terms of sort Agent are called *agents*, terms of sort $\mathsf{Message}$ are called *messages*. All other symbols, including the classical cryptographic primitives for building keys, encryption and pairs take messages as arguments and return a message. This set of cryptographic primitives is denoted by:

$$\mathcal{F}_{\mathsf{msg}} = \{<\_,\_>, \{\_\}_\_, \mathsf{pub}(\_), \mathsf{prv}(\_), \mathsf{shr}(\_)\}.$$

In addition, we may have e.g. hash function symbols.

We also assume that every agent is a message and every message is an event; we have the subsort relations $\mathsf{Agent} \leq \mathsf{Message} \leq \mathsf{Event}$, $\mathsf{Ha} \leq \mathsf{Agent}$ and $\mathsf{Da} \leq \mathsf{Agent}$. Let us comment a little bit:

– Num is only used for internal representations of session numbers, nonces... It is important to provide with one representation since we will consider Herbrand models. However, such a representation is irrelevant in what follows. In particular, neither the intruder nor the agents have access to this representation.
– There are two non-standard sorts Da, Ha. The terms of these sorts are respectively $s_d^k(d)$ and $s_h^k(h)$ and are intended to represent compromised and honest agents respectively. Again, this is for internal representation only. Of course, this distinction is never used in the protocol description. It is however necessary in the protocol property definition: typically, we want to state that a secret *shared by honest agents* remains unknown to the intruder, hence we need a way to express that an agent is honest. The use of s_d, s_h as a mean to represent agents identities is arbitrary. It could be replaced with any other representation.
– srv_i are intended to be server names. They are constant names in the protocols, thus they are not counted in the number of agents needed for an attack.
– n_i is a collection of function symbols, which are used to represent nonces (randomly generated data): n_i is intended to take as arguments some agent ids (who generates the nonce and who are supposed to receive the nonce) and a session number. i is intended to be the protocol step. Note that we may also consider a single symbol n with an additionnal argument i. Then $n_i(\_)$ is simply a notation for $n(i,\_)$.
– st is intended to represent the local state of an agent: the term $st(a,i,j,m)$ represents the agent a being at the j^{th} step of the role i and having the message m in his memory. Events will consist of either sending a message or increasing a local memory. Traces are sequences of pairs of an event and a session number.
– We do not assume any a priori typing of messages (there is no a priori way to distinguish between a nonce and a pair for instance), though any such policy could be specified at the protocol description level.

By abuse of notation, we will sometimes write e.g. 2 instead of $s(s(0))$, $< x, y, z >$ instead of $< x, < y, z >>$, or $\{x, y\}_z$ instead of $\{< x, y >\}_z$.

We will sometimes use unary predicate symbols instead of sort names in order to explicitly state the sort of a variable. For instance, we may write $\mathsf{Agent}(x)$, expressing that x is of sort Agent (other authors use the notation $x : \mathsf{Agent}$). Such unary predicate symbols can only be used with variable arguments. Though this is not used in the present paper, note that the definition of such unary symbols is a tree automaton [12], hence their least Herbrand interpretation is a recognizable set of trees.

2.2 Protocol independent clauses

We sketch here and in the following section how to design a set of Horn clauses defining valid traces. We also show in [13] that this is a reasonable definition since other models can be reduced to this one.

We use a binary predicate symbol I to describe the intruder knowledge. I takes two arguments: a message m and a trace t; $I(m,t)$ means that message m is known to the intruder after executing t. The first clauses describe the initial knowledge of the intruder, the second ones describe his ability to analyze and synthesize messages, the third ones his ability to learn new messages.

Initial knowledge

$\mathsf{Agent}(x) \Rightarrow I(x,t)$	The intruder knows all agent ids.
$\mathsf{Agent}(x) \Rightarrow I(\mathsf{pub}(x),t)$	The intruder knows all public keys.
$\mathsf{Da}(x) \Rightarrow I(\mathsf{prv}(x),t)$ $\mathsf{Da}(x) \Rightarrow I(\mathsf{shr}(x),t)$	The intruder knows all keys of compromised agents..

Analysis and Synthesis

$I(x,t), I(y,t) \Rightarrow I(< x,y >,t)$	The intruder can compose messages.
$I(x,t), I(y,t) \Rightarrow I(\{x\}_y,t)$	The intruder can encrypt a known message with a known key.
$I(\{x\}_{\mathsf{pub}(y)},t), I(\mathsf{prv}(y),t) \Rightarrow I(x,t)$ $I(\{x\}_{\mathsf{prv}(y)},t), I(\mathsf{pub}(y),t) \Rightarrow I(x,t)$ $I(\{x\}_y,t), I(y,t), \mathsf{Sym}(y) \Rightarrow I(x,t)$	The intruder can retrieve the clear text of a encrypted message when he knows the inverse of the key.

The predicate Sym is defined in figure 1.

Interception and Memorization

$T([x,s]\cdot t) \Rightarrow I(x,[x,s]\cdot t)$	All messages sent through the network are available to the intruder.
$I(x,t) \Rightarrow I(x,y\cdot t)$	The intruder remembers a message whatever is added to the trace.

Protocol independent clauses also contain the definition of some auxiliary predicates, which are described on figure 1 as well as the clause $T(\bot)$, which states that the empty trace is a trace. How to continue a trace is protocol-dependent.

The variables a, b stand for variables of sort Agent, x, y, z, m for variables of sort $\mathsf{Message}$, s, t and e for variables of sort respectively Num, Trace and Event.

Definition of Sym:

$$\mathsf{Agent}(x) \Rightarrow \mathsf{Sym}(x)$$

$$\mathsf{Agent}(x_1), \ldots, \mathsf{Agent}(x_{k_i}), \mathsf{Num}(s) \Rightarrow \mathsf{Sym}(n_i(x_1, \ldots, x_k, s))$$

$$\mathsf{Message}(x_1), \ldots, \mathsf{Message}(x_k) \Rightarrow \mathsf{Sym}(f(m_1, \ldots, m_k)) \quad f \in \{< \_, \_ >, \{\_\}_\_, \mathsf{shr}(\_)\}$$

Definition of Sup:

$$\mathsf{Num}(x) \Rightarrow \mathsf{Sup}(s(x), 0)$$

$$\mathsf{Sup}(x, y) \Rightarrow \mathsf{Sup}(s(x), s(y))$$

Definition of Neq:

$$\mathsf{Sup}(s, s') \Rightarrow \mathsf{Neq}(s, s')$$

$$\mathsf{Sup}(s', s) \Rightarrow \mathsf{Neq}(s', s)$$

Definition of Fresh:

$$\Rightarrow \mathsf{Fresh}(\perp, s)$$

$$\mathsf{Fresh}(t, s), \mathsf{Sup}(s, s') \Rightarrow \mathsf{Fresh}([e, s'] \cdot t, s)$$

Definition of In:

$$\Rightarrow \mathsf{In}([e, s], [e, s] \cdot t)$$

$$\mathsf{In}(x, t) \Rightarrow \mathsf{In}(x, [e, s] \cdot t)$$

Definition of NotPlayed:

$$\Rightarrow \mathsf{NotPlayed}(a, i, k, s, \perp)$$

$$\mathsf{NotPlayed}(a, k, i, s, t) \Rightarrow \mathsf{NotPlayed}(a, k, i, s, [m, s'] \cdot t)$$

$$\mathsf{NotPlayed}(a, k, i, s, t), \mathsf{Neq}(s, s') \Rightarrow \mathsf{NotPlayed}(a, k, i, s, [e, s'] \cdot t)$$

$$\mathsf{NotPlayed}(a, k, i, s, t), \mathsf{Neq}(i, j) \Rightarrow \mathsf{NotPlayed}(a, k, i, s, [st(a, k, j, m), s] \cdot t)$$

$$\mathsf{NotPlayed}(a, k, i, s, t), \mathsf{Neq}(k, k') \Rightarrow \mathsf{NotPlayed}(a, i, s, [st(a, k', j, m), s] \cdot t)$$

Fig. 1. Definitions of the auxiliary predicates

2.3 Protocol dependent clauses

We describe here how to define the set of valid traces T on the Yahalom protocol (see e.g. [8] for a description and further references). This protocol aims at establishing a session key between two agents, using a trusted server. In this section, a, b will stand for variables of sort Agent, x, y, z, m for variables of sort $\mathsf{Message}$, s, t and e for variables of sort respectively Num, Trace and Event.

$$
\begin{aligned}
&A \rightarrow B : A, N_a \\
&B \rightarrow S : B, \{A, N_a, N_b\}_{\mathsf{shr}(B)} \\
&S \rightarrow A : \{B, K_{ab}, N_a, N_b\}_{\mathsf{shr}(A)}, \{A, K_{ab}\}_{\mathsf{shr}(B)} \\
&A \rightarrow B : \{A, K_{ab}\}_{\mathsf{shr}(B)}, \{N_b\}_{K_{ab}}
\end{aligned}
$$

We first state that, at any point, we may start a new session of the protocol assigning roles to any of the agents. This is expressed by:

$$
\begin{aligned}
\mathsf{Fresh}(t, s), T(t) \Rightarrow T(\ &[st(a, 1, 1, < a, b, srv >), s] \\
&\cdot[st(b, 2, 1, < b, srv >), s] \\
&\cdot[st(srv, 3, 1, srv), s] \cdot t)
\end{aligned}
$$

Fresh is an auxiliary predicate (defined in figure 1), which holds when the number s is larger than any number occurring in t. Then the trace t can be extended accordingly.

Now, if a has started a session s, and if she has not already sent the first message of this session, she can do it, hence extending the trace, and moving to stage 2 for this session:

$$
\left.\begin{array}{r}
T(t), \\
\mathsf{In}([st(a, 1, 1, < a, b, srv >), s], t), \\
\mathsf{NotPlayed}(a, 2, s, t)
\end{array}\right\} \Rightarrow
\begin{array}{l}
T(\ [< a, n_1(a, s) >, s] \\
\quad \cdot[st(a, 1, 2, < a, b, srv, n_1(a, b, s) >), s] \\
\quad \cdot t)
\end{array}
$$

This uses the auxiliary predicates In and $\mathsf{NotPlayed}$ which are intended to be respectively the membership test on traces and a test that this step has not already been completed for the same session (see figure 1 for complete definitions).

Finally, let us describe how the last step of the protocol is translated: we require a to have completed the first step and assume that she receives a message of the expected form. This message may be forged by the intruder: we do not include receive events in the trace since messages that are possibly received are identical to messages that can be forged by the intruder.

$$
\left.\begin{array}{r}
T(t), \\
\mathsf{In}([u_1, s], t), \\
\mathsf{NotPlayed}(a, 3, s, t), \\
I(< \{b, x, n_1(a, b, s), y\}_{\mathsf{shr}(a)}, z >, t)
\end{array}\right\} \Rightarrow T([< z, \{y\}_x >, s] \cdot [u_2, s] \cdot t)
$$

where $u_1 = st(a,1,2,<a,b,srv,n_1(a,b,s)>)$ and $u_2 = st(a,1,3,<a,b,srv,n_1(a,b,s), x>)$.

2.4 The model

Now, we assume defined the sets of Horn clauses $\mathcal{C}_I, \mathcal{C}_D$ for the protocol independent clauses and the protocol dependent clauses. For a protocol P, we let $\mathcal{C}_P$ be $\mathcal{C}_I \cup \mathcal{C}_D$. We assume that $\mathcal{C}_P$ does not contain negative clauses (i.e. we only specify what is possible). Then $\mathcal{C}_P$ has a least Herbrand model $\mathcal{H}_P$.

Definition 1. *A* valid trace *for the protocol P is a member of the interpretation of T in $\mathcal{H}_P$.*

2.5 Attacks

Let ϕ be the security property that we want to check. We assume that ϕ can be expressed as a clause using the primitives described in previous sections. This is not a strong restriction since, at least the trace properties can be expressed in this way (and possibly other properties which relate different traces), as shown by the following examples.

Example 1. We can express that $u(x, y, s)$ (or $u(x, y)$ if we want to express the secrecy of a constant data) is a (long term) secret shared by x and y by:

$$(\forall t, x, y, s).\neg T(t) \vee \neg \mathsf{Ha}(x) \vee \neg \mathsf{Ha}(y) \vee \neg I(u(x, y, s), t)$$

which means that, in any trace t, if x and y are honest agents, then $u(x, y, s)$ is unknown to the intruder.

Example 2. We can express that $u(x, y, s)$ is a short term secret. I does not know $u(x, y, s)$ as long as session s is not completed for the agent x playing the first role:

$$(\forall t, x, y, s).\neg T(t) \vee \neg \mathsf{Ha}(x) \vee \neg \mathsf{Ha}(y) \vee \neg I(u(x, y, s), t) \vee \neg \mathsf{NotPlayed}(x, 1, 3, s, t).$$

If we assume that the last message of the protocol is sent by x then we express here that, in any trace t, if x and y are honest agents, then $u(x, y, s)$ is unknown to the intruder unless the session is already completed.

Example 3. We can express an authentication property: if x receives the message $m(x, y, s)$, then it has been sent previously by y: $(\forall t, x, y, s)$

$$\neg T(t) \vee \neg \mathsf{Ha}(x) \vee \neg \mathsf{Ha}(y) \vee \neg I(m(x, y, s), t) \vee \mathsf{In}([st(y, 2, 2, m(x, y, s)), s], t).$$

Definition 2. *A protocol P* satisfies a property ϕ *iff $\mathcal{H}_P \models \phi$.*

Dually, there is an attack when $\mathcal{H}_P \not\models \phi$. In such a case (by compactness), there is a finite subset $\mathcal{H}_0$ of $\mathcal{H}_P$ such that $\mathcal{H}_0 \not\models \phi$:

Definition 3. *An* attack *on P for ϕ is a finite subset $\mathcal{H}_0$ of $\mathcal{H}_P$ such that $\mathcal{H}_0 \not\models \phi$. $\mathcal{H}_0$ is an* attack with n agents *if there are at most n distinct terms of sorts* Agent *in $\mathcal{H}_0$.*

For instance, if the property ϕ is a "trace property", $\mathcal{H}_0$ may contain a single predicate $T(t)$ where t is a finite trace which violates the property.

2.6 Relevance of the model

The model we present here is actually an extension of the Millen-Rueß model [24, 14] (hereafter referred to as the MR model), expressed in Horn clauses. The MR model is itself inspired from Paulson's model [27] and from the strand spaces [31]. Formally, we proved in [13] that for each protocol of the MR model, we can associate a finite set of Horn clauses $\mathcal{C}_P$ and a finite set of purely negative clauses Φ_P such that P is insecure if and only if there is an attack on $\mathcal{C}_P$ for some $\phi \in \Phi_P$. Such a reduction preserves the number of agents involved in the attack [13].

3 Reduction to a fixed number of agents

We are now ready to state and prove our reduction results. In section 3.1, we prove that for secrecy properties, it is sufficient to consider 2 agents. In section 3.2, we extend this result when an agent is disallowed to speak to herself and/or for more general security properties like authentication.

3.1 From n agents to 2 agents

In this section, we also consider purely negative properties, which easily encompasses secrecy, but does not encompass authentication in a natural way. We will discuss this in section 3.2. We show that if there is an attack with n agents, then we can construct an attack with 2 agents: given an attack using n agents, we project every honest identity on a single honest identity and every dishonest identity on a single dishonest identity. Since prtotols and properties do not rely on internal agents representations, we obtain a valid attack using only two agents. This projection uses the fact that our model allows an agent to speak to herself, which is the case of most of the models for cryptographic protocols [24, 31, 19, 6, 16, 4]. However, a similar result holds even if an agent is disallowed to speak to herself (see subsection 3.2). We also consider here purely negative properties, which easily encompasses secrecy, but does not encompass authentication in a natural way. We will discuss this in section 3.2.

We emphasize that our result holds for any model of protocols which do not make use of our internal representation of agent ids (which is the case of any model we know). More precisely:

Definition 4. *A set of clauses $\mathcal{C}$ is* admissible *if it does not use the symbols s_h, s_d. A clause is said* purely negative *if it only contains negative literals.*

The clauses which were proposed in the previous sections are admissible. Furthermore, any protocol specification can not use our particular representation of names, hence it is always represented as an admissible set of clauses.

Theorem 1. *Let $\mathcal{C}_P$ be an admissible set of clauses. Let ϕ be a purely negative admissible clause. If there is an attack of P for ϕ, using n agents, then there is an attack using (at most) two agents.*

Proof. We first introduce some notations. Let $\mathcal{M}$ be the set of messages, $\mathcal{T}$ be the set of all positive ground literals, and Σ_g be the set of mappings from variables to ground terms, which are compatible with the sort constraints.
Given a Horn clause $c = B_1(\boldsymbol{x}), ..., B_n(\boldsymbol{x}) \Rightarrow A(\boldsymbol{x})$ where $B_1(\boldsymbol{x}), \ldots, B_n(\boldsymbol{x}), A(\boldsymbol{x})$ are positive literals whose free variables are contained in $\boldsymbol{x}$, and a subset $\mathcal{S}$ of $\mathcal{T}$, we define $c(\mathcal{S})$ as follows:

$$c(\mathcal{S}) \stackrel{\text{def}}{=} \{A(\boldsymbol{x})\sigma \mid \sigma \in \Sigma_g, \forall i, B_i(\boldsymbol{x})\sigma \in \mathcal{S}\}.$$

Then, the immediate consequence relation $F_{\mathcal{C}}$ is the mapping from $2^{\mathcal{T}}$ to $2^{\mathcal{T}}$ defined by:

$$F_{\mathcal{C}}(\mathcal{S}) \stackrel{\text{def}}{=} \mathcal{S} \cup \bigcup_{c \in \mathcal{C}} c(\mathcal{S}).$$

For simplicity, we will write F_P for the mapping $F_{\mathcal{C}_P}$.

It is well-known that the set of positive literals $\mathcal{H}_P^+$ of the least Herbrand model $\mathcal{H}_P$ is the least fixed point of F_P:

$$\mathcal{H}_P^+ = \bigcup_{k=1}^{+\infty} F_P^k(\emptyset)$$

For every $L \in \mathcal{H}_0$ there is a minimal index n_L such that $L \in F_P^{n_L}(\emptyset)$.

We define now the projection function: we map every honest agent to h and every dishonest agent to d : for every literal L, let $\overline{L}$ be the literal L in which every maximal subterm of sort Ha is replaced with h and every maximal subterm of sort Da is replaced with d:

$$\begin{aligned}
\overline{f(t_1, \ldots, t_n)} &\stackrel{\text{def}}{=} f(\overline{t_1}, \ldots, \overline{t_n}) \;\text{ If } f \notin \{s_h, s_d\} \\
\overline{s_h(t)} &\stackrel{\text{def}}{=} h \\
\overline{s_d(t)} &\stackrel{\text{def}}{=} d
\end{aligned}$$

Our proof relies on the following lemma which ensures that if a positive literal is in $\mathcal{H}_P$ then its projection is also in $\mathcal{H}_P$.

Lemma 1. *If L be a positive literal of $\mathcal{H}_P$, then $\overline{L}$ is in $\mathcal{H}_P$.*

This is proved by induction on n_L. If $n_L = 0$, there is no literal such that $n_L = 0$ thus there is nothing to prove.

Suppose the property true for $n_l \leq n$ and consider a positive literal L of $\mathcal{H}_P$ such that $n_L = n + 1$. There exists a clause c_L and positive literals $L_1, \ldots, L_k \in \mathcal{H}_P^+$ such that $L \in c_L(\{L_1, \ldots, L_k\})$ with $n_{L_i} \leq n$ for all $1 \leq i \leq k$. By induction hypothesis, $\overline{L_1}, \ldots, \overline{L_k} \in \mathcal{H}_P^+$. In addition, c_L is on the form $B_1(\boldsymbol{x}), \ldots, B_k(\boldsymbol{x}) \Rightarrow A(\boldsymbol{x})$ with $L = A(\boldsymbol{x})\sigma$, $L_i = B_1(\boldsymbol{x})\sigma$ for some $\sigma \in \Sigma_g$. Since c_L is an admissible clause, it does not contains the symbols s_h and s_d thus $\overline{L} = A(\boldsymbol{x})\overline{\sigma}$ and $\overline{L_i} = B_1(\boldsymbol{x})\overline{\sigma}$. Hence $\overline{L} \in c_L(\{\overline{L_1}, \ldots \overline{L_k})$ and $\overline{L} \in \mathcal{H}_P^+$.

We are now ready to complete the proof. Assume that $\mathcal{H}_0$ is a finite subset of $\mathcal{H}_P$ such that $\mathcal{H}_0 \not\models \phi$. Since ϕ is assumed to be purely negative, we may assume w.l.o.g. that $\mathcal{H}_0$ only contains positive literals.

Let $\mathcal{H}_1 = \{\overline{L} \mid L \in \mathcal{H}_0\}$. The set $\mathcal{H}_1$ is still finite and, by lemma 1, $\mathcal{H}_1 \subset \mathcal{H}_P$. Let us show that $\mathcal{H}_1 \not\models \phi$. Let $\phi\sigma$ an instance of ϕ falsified by $\mathcal{H}_0$. Then $\overline{\phi\sigma}$ is falsified by $\mathcal{H}_1$. Since ϕ is an admissible clause $\overline{\phi\sigma} = \phi\overline{\sigma}$, thus $\mathcal{H}_1 \not\models \phi$. $\square$

Actually, this theorem does not hold when ϕ may contain positive literals.

Example 4. Let $\mathcal{C}_P$ be:

$$\begin{cases} \mathsf{Da}(x) \Rightarrow A(x,y) \\ \mathsf{Da}(y) \Rightarrow A(x,y) \\ \qquad\quad \Rightarrow A(x,x) \end{cases}$$

and ϕ be $A(x,y)$. $\neg A(h, s_h(h))$ is an attack and there is no attack with a single honest agent.

We will consider in section 3.2 an extension of theorem 1 for formulas containing positive literals.

3.2 Extensions of our reduction result

Disallowing an agent to speak with herself
In the last section we used the ability for an agent to speak with herself, which was not explicitly ruled out by the specification. There are however examples in which the existence of an attack relies on this ability:

Example 5. Consider the following "toy" example where an agent A sends a secret to an agent B:

$$A \to B : \{A, B, N_a\}_{\mathsf{pub}(B)}, \{secret\}_{\{A,A,N_a\}_{\mathsf{pub}(B)}}.$$

B is able to build the compound key $\{A, A, N_a\}_{\mathsf{pub}(B)}$ and gets the secret. One can show that N_a will remain unknown to the intruder, thus $\{A, A, N_a\}_{\mathsf{pub}(B)}$ is unknown to the intruder unless $A = B$. Thus this protocol is flawed only if an honest agent sends a secret to herself.

We are now considering explicitly disallowing such self-conversations between honest agents. Still, a dishonest agent is enabled to speak with himself, which actually does not bring any new information to the intruder (see remark 1 below). Following this, we add a predicate symbol $\mathsf{Distinct}$ defined by the set of clauses:

$$\mathcal{C}_{\neq} \stackrel{\text{def}}{=} \begin{cases} \mathsf{Distinct}(x,y), \mathsf{Ha}(x), \mathsf{Ha}(y) \Rightarrow \mathsf{Distinct}(s_h(x), s_h(y)) \\ \mathsf{Ha}(x) \Rightarrow \mathsf{Distinct}(h, s_h(x)) \\ \mathsf{Ha}(x), \mathsf{Da}(y) \Rightarrow \mathsf{Distinct}(x,y) \\ \mathsf{Distinct}(x,y) \Rightarrow \mathsf{Distinct}(y,x) \\ \mathsf{Da}(x), \mathsf{Da}(y) \Rightarrow \mathsf{Distinct}(x,y) \end{cases}$$

The least Herbrand model of $\mathsf{Distinct}$ consists of pairs $(s_h^k(h), s_d^m(d))$, $(s_d^m(d), s_h^k(h))$, $(s_d^m(d), s_d^k(d))$ and $(s_h^i(h), s_h^j(h))$ with $i \neq j$.

We redefine the notion of an admissible clause:

Definition 5. *A clause ϕ is* admissible *if*

– *ϕ does not contain the symbols s_h, s_d,*
– Distinct *occurs only negatively in ϕ.*

We can specify that the sender a is distinct from the (expected) receiver b with admissible clauses: it suffices to add negative literals $\mathsf{Distinct}(a, b)$. Note however that such a property is not expressible in e.g. the Millen-Rueß model. The protocol model $\mathcal{H}_P$ is now the least Herbrand model of $\mathcal{C}_{\neq} \cup \mathcal{C}_P$. All other definitions are unchanged.

Remark 1. If we want to specify that an agent is not allowed to speak with herself, even for dishonest agents, we can introduce a predicate Distinct whose semantic is exactly the pairs of distinct agents. In this case, an *admissible* clause should also verify that Distinct occurs at most once, which is sufficient to express that an agent is not allowed to speak to herself. In addition, the protocol has to verify that the message exchanges between two compromised agents does not increase the intruder knowledge, which is the case of all "real" protocols ([8]). This leads to a specification which can be reduced to the above one.

More general security properties

Theorems 1 assumes that ϕ is purely negative, which is necessary according to Example 4.

We have seen in section 2.5 that such a restriction to negative properties is not a problem for secrecy. On the other hand, an authentication property (non-injective agreement) is naturally expressed as

$$\neg T(t) \vee \neg \mathsf{Ha}(x) \vee \neg \mathsf{Ha}(y) \vee \neg I(m(x, y, s), t) \vee \mathsf{In}([st(x, m(x, y, s)), s], t)$$

which involves a positive literal. However, it is still possible to handle such properties. Let us extend the definition of admissible security properties to a class which encompasses authentication and secrecy properties.

Definition 6. *A clause ϕ is* admissible security property *if ϕ is of the form*

$$C \vee \mathsf{In}(u_1, t_1) \vee \cdots \vee \mathsf{In}(u_n, t_n),$$

where C is a purely negative clause, $t_1, \ldots, t_n, u_1, \ldots, u_n$ are terms and

– *the symbols s_h, s_d do not occur in ϕ*
– Distinct *occurs only negatively in ϕ.*

Reduction result

Our reduction result will now depend on the security property under consideration: if the property ϕ uses k distinct agents variables then if there is an attack, there is an attack with (at most) $k + 1$ agents.

Theorem 2. *Assume $\mathcal{C}_P$ is an admissible set of clauses, which does not contain any variable of sort* Ha*, and ϕ an admissible security property. If there is an attack on P for ϕ using n agents, then there is an attack on P for ϕ using at most $k_1 + k_2 + 1$ agents, where k_1 is the number of variables of sort* Ha *occurring in ϕ and k_2 is the number of variables of sort* Agent *which are not of subsort* Ha*, occurring in the positive literals of ϕ.*

Note that disallowing variables of sort Ha in $\mathcal{C}_P$ is not a real restriction. Indeed, the specification of the protocol itself ($\mathcal{C}_D$) should not distinguish between honest and dishonest agents. Moreover, the intruder capabilities ($\mathcal{C}_I$) should not depend on specific honest agents: an intruder knows *all* the public information of honest agents and the private information of dishonest agents *only*.

Proof. We keep the notations of the proof of theorem 1. This new reduction result is proved in two steps: first, we reduce ϕ to a purely negative clause and then we use a projection function to get our result.

First step: ϕ is of the form $C \vee \mathsf{In}(u_1, t_1) \vee \cdots \vee \mathsf{In}(u_n, t_n)$. We define a predicate $\widetilde{\mathsf{In}}$ whose interpretation will coincide with the complement of the interpretation of In in $\mathcal{H}_P$. This could be performed using general techniques such as in [5, 9]. We give however here a direct definition:

$$\begin{aligned}
\widetilde{\mathsf{Trace}}(x) &\Rightarrow \widetilde{\mathsf{In}}(y, x)\\
\widetilde{\mathsf{Event}}(x) &\Rightarrow \widetilde{\mathsf{In}}([x, s], y)\\
\widetilde{\mathsf{Num}}(x) &\Rightarrow \widetilde{\mathsf{In}}([x, s], y)\\
&\Rightarrow \widetilde{\mathsf{In}}(x, \perp)\\
\widetilde{\mathsf{In}}(x, y), \mathsf{Diff}(x, z) &\Rightarrow \widetilde{\mathsf{In}}(x, y \cdot z)
\end{aligned}$$

$\widetilde{\mathsf{Trace}}, \widetilde{\mathsf{Event}}, \widetilde{\mathsf{Num}}$ are defined by complementation of tree automata: their interpretation coincides with the complement of $\mathsf{Trace}, \mathsf{Event}$ and Num respectively in $\mathcal{H}_P$. Diff is defined by the following clauses:

$$\begin{aligned}
&\Rightarrow \mathsf{Diff}(f(x_1, \ldots, x_n), g(y_1, \ldots, y_k)) \;\; \forall f \neq g\\
\mathsf{Diff}(x_i, y_i) &\Rightarrow \mathsf{Diff}(f(x_1, \ldots, x_n), f(x_1, \ldots, x_n)) \; 1 \leqslant i \leqslant n
\end{aligned}$$

We let $C_\neg$ be these additional clauses. Note that $C_\neg$ is not an admissible set of clauses but we show in the second step of the proof that we can still handle such clauses since they are not used elsewhere in the protocol clauses.

Let $\mathcal{H}_{P,\phi}$ be the least Herbrand model of $\mathcal{C}_P \cup \mathcal{C}_{\neq} \cup C_\neg$. It contains $\mathcal{H}_P$ (since all clauses in $C_\neg$ are headed with predicate symbols which do not occur in any clause of $\mathcal{C}_P \cup \mathcal{C}_{\neq}$). Moreover, for every ground terms u, t, $\mathcal{H}_{P,\phi} \models \widetilde{\mathsf{In}}(u, t)$ iff $\mathcal{H}_P \not\models \mathsf{In}(u, t)$. Indeed, the interpretation of Diff in $\mathcal{H}_{P,\phi}$ is the set of pairs of distinct ground terms. Next, the interpretations of $\widetilde{\mathsf{In}}$ and In in $\mathcal{H}_{P,\phi}$ have an empty intersection: we prove this result by contradiction and by induction on the minimal index n such that both $\mathsf{In}(s, t)$ and $\widetilde{\mathsf{In}}(s, t)$ belong to $F^n_{P,\phi}(\emptyset)$. Finally, every pair of terms (s, t) is either in the interpretion of In or in the interpretation of $\widetilde{\mathsf{In}}$, by induction on the size $|s| + |t|$.

Second step: projection. Let $\psi \stackrel{\mathrm{def}}{=} C \vee \neg\widetilde{In}(u_1, t_1) \vee \cdots \vee \neg\widetilde{In}(u_n, t_n)$ and assume there exists a ground substitution θ such that $\mathcal{H}_P \not\models \phi\theta$. Then $\mathcal{H}_{P,\phi} \not\models \psi\theta$. Now, since $\psi\theta$ is purely negative, there is a finite subset $\mathcal{C}_0$ of ground instances of $\mathcal{C}_P \cup \mathcal{C}_{\neq} \cup \mathcal{C}_{\neg}$ such that $\mathcal{C}_0 \models L$ for every literal $\neg L$ occurring in $\psi\theta$.

We let $\mathcal{H}_0$ be the set of (positive) literals defined by:

$$\begin{aligned}\mathcal{H}_0 = \quad & \{P(t_1, \dots, t_m) \in \mathcal{H}_{P,\phi} \mid P \neq \widetilde{\mathsf{In}}, P \neq \mathsf{Diff}, \mathcal{C}_0 \models P(t_1, \dots, t_m)\} \\ & \cup \{\widetilde{\mathsf{In}}(u_i\theta, t) \in \mathcal{H}_{P,\phi} \mid i \in \{1, \dots, n\}, \mathcal{C}_0 \models \widetilde{\mathsf{In}}(u_i\theta, t)\} \\ & \cup \{\mathsf{Diff}(u_i\theta, t) \in \mathcal{H}_{P,\phi} \mid i \in \{1, \dots, n\}, \mathcal{C}_0 \models \mathsf{Diff}(u_i\theta, t)\}\end{aligned}$$

Since $\mathcal{C}_0$ is a set of Horn clauses, for $L \in \mathcal{H}_0$, there is an integer n_L and a clause $C_L \in \mathcal{C}_0$ such that $L \in F^{n_L}_{\mathcal{C}_0}(\emptyset)$, $L \notin F^{n_L-1}_{\mathcal{C}_0}(\emptyset)$, $C_L = \neg L_1 \vee \ldots \vee \neg L_k \vee L$, $L_1, \dots, L_k \in F^{n_L-1}_{\mathcal{C}_0}(\emptyset)$. With such definitions, $\mathcal{H}_0$ has the following properties:

- $\mathcal{H}_0$ is a finite subset of $\mathcal{H}_{P,\phi}$
- $\mathcal{H}_0$ falsifies $\psi\theta$
- because of the form of the clauses defining $\widetilde{\mathsf{In}}$, for every $L \in \mathcal{H}_0$, if $C_L = \neg L_1 \vee \ldots \vee \neg L_k \vee L$, then either $L_1, \dots, L_k \in \mathcal{H}_0$, or else $L = \mathsf{Diff}(u_i\theta, t)$.

Let us now define the projection. If $x_1, \dots x_{k_1}$ are the variables of sort Ha in ψ, we let $s_h^{m_1}(h), \dots, s_h^{m_p}(h)$ be the set $\{x_1\theta, \dots, x_{k_1}\theta\}$ with $m_1 < \ldots < m_p$ ($p \leq k_1$). If $y_1, \dots y_{k_2}$ are the variables of u_i, t_i, of sort Agent but not of subsort Ha, we let $s_d^{n_1}(d), \dots, s_d^{m_q}(d)$ be the set $\{y_1\theta, \dots, y_{k_2}\theta\}$ with $n_1 < \ldots < n_q$ ($q \leq k_2$). Next, we define the projection function as follows:

$$\begin{cases} \overline{f(t_1, \dots, t_n)} \stackrel{\mathrm{def}}{=} f(\overline{t_1}, \dots, \overline{t_n}) & \text{If } f(t_1, \dots, t_n) \text{ is not of sort } \mathsf{Ha} \text{ or } \mathsf{Da} \\ \overline{s_h^{m_i}(h)} \stackrel{\mathrm{def}}{=} s_h^{i-1}(h) & \text{For } i = 1, ..., p \\ \overline{s_d^{n_i}(d)} \stackrel{\mathrm{def}}{=} s_d^{i-1}(d) & \text{For } i = 1, ..., q \\ \overline{t} \stackrel{\mathrm{def}}{=} s_d^{q}(d) & \text{Otherwise} \end{cases}$$

Again, we let $\mathcal{H}_1 = \{\overline{L} \mid L \in \mathcal{H}_0\}$ and we are going to prove that $\mathcal{H}_1 \subseteq \mathcal{H}_{P,\phi}$ and $\mathcal{H}_1$ falsifies $\psi\overline{\theta}$. This will conclude the proof since considering $\mathcal{H}_1'$ the set $\mathcal{H}_1$ where the literals $\widetilde{\mathsf{In}}(u_i, t_i)\theta$ are replaced with $\neg\mathsf{In}(u_i, t_i)\theta$, we have $\mathcal{H}_1' \subseteq \mathcal{H}_P$ since $\mathcal{H}_P \models \neg\mathsf{In}(u_i, t_i)\theta$ iff $\mathcal{H}_{P,\phi} \models \widetilde{\mathsf{In}}(u_i, t_i)\theta$ from the first part of the proof. Moreover $\mathcal{H}_1'$ falsifies $\phi\overline{\theta}$, thus $\mathcal{H}_1'$ will be an attack with $p + q + 1$ agents: $d, s_d(d), \dots, s_d^q(d), h, s_h(h), \dots, s_h^{p-1}(h)$, $p + q \leq k_1 + k_2$.

Actually, with the three following lemmas, the proof that $\mathcal{H}_1 \subseteq \mathcal{H}_{P,\phi}$ is similar to the proof of theorem 1:

Lemma 2. *For every ground terms g_1, g_2, if $\mathsf{Distinct}(g_1, g_2) \in F^n_{P,\phi}(\emptyset)$, then $\overline{\mathsf{Distinct}(g_1, g_2)} \in F^n_{P,\phi}(\emptyset)$.*

Proof of lemma 2:
We may assume $n > 0$. Let $t_i = x_i\theta$. Then there are three possible situations (let us recall that $\mathsf{Distinct}$ only occurs positively in $\mathcal{C}_{\neq}$):

- if $g_1, g_2 \notin \{t_1, \ldots, t_k\}$, then using that the least Herbrand model of $\mathsf{Distinct}$ consists of pairs $(s^k(h), s^m(d)), (s^m(d), s^k(h)), (s^m(d), s^k(d))$ and $(s^i(h), s^j(h))$ with $i \neq j$, we have that $\overline{\mathsf{Distinct}(g_1, g_2)} = \mathsf{Distinct}(d, d) \in F_{P,\phi}(\emptyset)$;
- if $g_1 \in \{t_1, \ldots, t_k\}$ and $g_2 \notin \{t_1, \ldots, t_k\}$ (or the converse), then $\overline{\mathsf{Distinct}(g_1, g_2)} = \mathsf{Distinct}(s_h^j(h), d)$ (or $\mathsf{Distinct}(d, s_h^j(h))$), which also belongs to $F_{P,\phi}(\emptyset)$;
- if $g_1, g_2 \in \{t_1, \ldots, t_k\}$: $g_1 = s_h^{m_i}(h), g_2 = s_h^{m_j}(h)$ with $i \neq j$, then $\overline{\mathsf{Distinct}(g_1, g_2)} = \mathsf{Distinct}(s_h^i(h), s_h^j(h)) \in F_{P,\phi}^{|j-i|}(\emptyset)$. In this last case, $|j-i| \leq |m_j - m_i|$ by construction, hence the result.

End of the proof of lemma 2.

Lemma 3. *For every $i = 1, ..., n$, for every term t, if $\mathsf{Diff}(u_i\theta, t) \in \mathcal{H}_{P,\phi}$ then $\overline{\mathsf{Diff}(u_i\theta, t)} \in \mathcal{H}_{P,\phi}$.*

Proof of lemma 3:
By construction of the predicate Diff, we have $\mathsf{Diff}(u, v) \in \mathcal{H}_{P,\phi}$ if and only if $u \neq v$. Thus it is sufficient to show that $u_i\overline{\theta} = \overline{t}$ implies $u_i\theta = t$. Assume $u_i\overline{\theta} = \overline{t}$. Since the variables of u_i, which are of sort Agent, are contained in $\{x_1, \ldots x_{k_1}\} \cup \{y_1, \ldots y_{k_2}\}$, the maximal subterms of $u_i\theta$ of sort Agent belong to:

$$\mathcal{A} \stackrel{\text{def}}{=} \{s_h^{m_1}(h), \ldots, s_h^{m_p}(h), s_d^{n_1}(d), \ldots, s_d^{m_q}(d)\},$$

thus the maximal subterms of $u_i\overline{\theta}$ of sort Agent belong to:

$$\{d, s_d(d), \ldots, s_d^{q-1}(d), h, s_h(h), \ldots, s_h^{p-1}(h)\}.$$

Assume t contains a maximal subterm of sort Agent, which does not belongs to $\mathcal{A}$, then one of the maximal subterms of sort Agent of $\overline{t}$ is $s_d^q(d)$. Thus $u_i\overline{\theta}$ and $\overline{t}$ can not be equal. Consequently, the maximal subterms of t of sort Agent belong to $\mathcal{A}$. Since the function $\bar{\cdot}$ is injective on $\mathcal{A}$, we get $u_i\theta = t$.
End of the proof of lemma 3.

Lemma 4. *If $\widetilde{\mathsf{Trace}}(t) \in \mathcal{H}_{P,\phi}$ (resp. $\widetilde{\mathsf{Event}}(t)$, resp. $\widetilde{\mathsf{Num}}(t)$) then $\widetilde{\mathsf{Trace}}(\overline{t}) \in \mathcal{H}_{P,\phi}$ (resp. $\widetilde{\mathsf{Event}}(\overline{t})$, resp. $\widetilde{\mathsf{Num}}(\overline{t})$).*

Proof sketch of lemma 4: In every state of a minimal deterministic tree automaton accepting $\widetilde{\mathsf{Trace}}$, either all terms of sort Agent are accepted or no term of sort Agent is accepted. *End of proof of lemma 4.*

As in theorem 1, we prove now that, if $L \in \mathcal{H}_0$, then $\overline{L} \in \mathcal{H}_{P,\phi}$. This is proved by induction on n_L. If $n_L = 0$, there is nothing to prove. Otherwise, let $C_L = \neg L_1 \vee \ldots \vee \neg L_k \vee L$. By property of $\mathcal{H}_0$, either $L = \mathsf{Diff}(u_i\theta, t)$, in which case we conclude using lemma 3, or else $L_1, \ldots, L_k \in \mathcal{H}_0$. Then, by induction hypothesis, $\overline{L_1}, \ldots, \overline{L_k} \in \mathcal{H}_{P,\phi}$. Moreover, $C_L = C\sigma$ for some clause $C \in \mathcal{C}_P \cup \mathcal{C}_{\neq} \cup \mathcal{C}_{\neg}$. If $C \in \mathcal{C}_P$, as in the proof of theorem 1, $\overline{C_L} = C\overline{\sigma}$ and we conclude that $\overline{L} \in \mathcal{H}_{P,\phi}$. If $C \in \mathcal{C}_{\neq}$, L is of the form $\mathsf{Distinct}(g_1, g_2)$ and we conclude using lemma 2.

Finally, if $C \in \mathcal{C}_\neg$, either $L = \widetilde{\mathsf{Trace}}(t)$ (resp. $\widetilde{\mathsf{Event}}(t)$, resp. $\widetilde{\mathsf{Num}}(t)$) and we conclude using lemma 4, or else C is one of the clauses defining $\widetilde{\mathsf{In}}$. However, since s_h, s_d do not occur in these clauses, $\overline{C\sigma} = C\overline{\sigma}$ and we can simply apply the induction hypothesis.

This shows that $\mathcal{H}_1 \subseteq \mathcal{H}_{P,\phi}$, hence concludes the proof as noticed above. $\square$

Remark: Note that our reduction result strongly depends on the fact that the negation of the predicate In can also be expressed by a set of Horn clauses.

3.3 Discussion on the bound $k_1 + k_2 + 1$

We show here that the bound $k_1 + k_2 + 1$ can be reached for some protocols P and some properties ϕ when $k_1 = 1$ or $k_2 = 0$. We have not built examples in the general case (when $k_1 \neq 1, k_2 \neq 0$) because it seemed technical but we are convinced that it could be obtained by a combination of the two following examples.

Example 6. (bound $k_1 + 1$ when $k_2 = 0$) Let $k \geq 2$. Consider the following protocol, inspired from the Needham-Schroeder public key protocol. $a_1, \ldots, a_k$ are variables of sort Agent.
Let $u = < a_1, \ldots, a_k >$.
Initialization

$$\mathsf{Fresh}(t,s), T(t) \Rightarrow T([st(a_1,1,u),s] \cdot [st(a_2,1,a_2),s] \cdot \cdots \cdot [st(a_k,1,a_k),s] \cdot t)$$

First message: $A_1 \to A_2 : \{A_1, A_2, \ldots, A_k, N_{A_1}\}_{\mathsf{pub}(A_2)}$, $A_i \neq A_j$, for $i \neq j$.

$$\left.\begin{array}{r} T(t), \mathsf{Distinct}(a_i,a_j) \quad i \neq j \\ \mathsf{In}([st(a_1,1,u),s],t), \\ \mathsf{NotPlayed}(a_1,2,s,t) \end{array}\right\} \Rightarrow \begin{array}{l} T(\ [\{u, n_1(a_1,\ldots,a_k,s)\}_{\mathsf{pub}(a_2)}, s] \\ \quad \cdot [st(a_1, 2, < u, n_1(a_1,\ldots,a_k,s) >), s] \\ \quad \cdot t) \end{array}$$

Second message: $A_2 \to A_1 : \{N_{A_1}, N_{A_2}\}_{\mathsf{pub}(A_1)}$

$$\left.\begin{array}{r} T(t), \mathsf{Distinct}(a_i,a_j) \quad i \neq j \\ I(\{u,x\}_{\mathsf{pub}(a_2)}, t) \\ \mathsf{In}([st(a_2,1,a_2),s],t), \\ \mathsf{NotPlayed}(a_2,2,s,t) \end{array}\right\} \Rightarrow \begin{array}{l} T(\ [\{x, n_2(a_1,\ldots,a_k,s)\}_{\mathsf{pub}(a_1)}, s] \\ \quad \cdot [st(a_2, 2, < u, n_2(a_1,\ldots,a_k,s) >), s] \\ \quad \cdot t) \end{array}$$

Third message: $A_1 \to A_2 : \{N_{A_2}\}_{\mathsf{pub}(A_2)}$

$$\left.\begin{array}{r} T(t), I(\{n_1(a_1,\ldots,a_k,s), y\}_{\mathsf{pub}(a_1)}, t) \\ \mathsf{In}([st(a_1,2,< u,n >),s],t), \\ \mathsf{NotPlayed}(a_2,3,s,t) \end{array}\right\} \Rightarrow \begin{array}{l} T(\ [\{y\}_{\mathsf{pub}(a_1)}, s] \\ \quad \cdot [st(a_1, 3, < u, n, y >), s] \\ \quad \cdot t) \end{array}$$

where $n = n_1(a_1, \ldots, a_k, s)$.
We could also add some other rules to make the roles of $a_3, \ldots, a_k$ less fictitious.
We consider the property:

$$\phi = \neg\mathsf{Ha}(x_1) \vee \ldots \vee \neg\mathsf{Ha}(x_k) \vee \neg I(n_2(x_1,\ldots,x_k,s),t).$$

Then, following the attack describet by G. Lowe in [22], there is an attack on ϕ, using $k+1$ agent ids. Let us sketch why every attack on ϕ uses at least $k+1$ agent ids. Assume there is an attack, then there exist $t, s, a_1, \ldots, a_k$ such that $I(n_2(a_1, \ldots, a_k, s), t) \in \mathcal{H}_P$ where $\mathcal{H}_P$ is the least Herbrand model and $a_1, \ldots, a_k$ are honest agents. Since a_2 produces $n_2(a_1, \ldots, a_k, s)$ only if $\mathsf{Distinct}(a_i, a_j)$ for $i \neq j$ holds and since $a_1, \ldots, a_k$ are honest agents, we have that $a_1, \ldots, a_k$ are distinct. In addition, if no dishonest identity is used, then the intruder cannot decrypt any message thus he can not obtain $n_2(a_1, \ldots, a_k, s)$. Consequently, at least one compromised identity has been used, thus at least $k+1$ identities have been used for the attack.

Example 7. (bound k_2+1 when $k_1 = 1$) Let $k \geq 2$. Consider the following toy protocol:

$$A_1 \rightarrow A_2 : \{A_1, \ldots, A_{k+1}\}_{\mathsf{prv}(A_1)}, \bigcup_{\substack{\{i_1, \ldots, i_{k+1}\} \subset \{1, \ldots, k+1\} \\ \mid \#\{i_1, \ldots, i_{k+1}\} \leq 1}} < A_{i_1}, \ldots, < A_{i_k}, A_{i_{k+1}} >$$

The agent A sends an authentication message $\{A_1, \ldots, A_{k+1}\}_{\mathsf{prv}(A_1)}$ and all possible $k+1$-tuples of agents names where at least two agent's name are equal. This can be encoded by one single clause:

$$T(t), \mathsf{Fresh}(t, s) \Rightarrow T([\{a_1, \ldots, a_{k+1}\}_{\mathsf{prv}(a)}] \cdot \bigcup_{\substack{\{i_1, \ldots, i_{k+1}\} \subset \{1, \ldots, k+1\} \\ \mid \#\{i_1, \ldots, i_{k+1}\} \leq k}} [< a_{i_1}, \ldots, < a_{i_k}, a_{i_{k+1}} >, s] \cdot t)$$

where $a_1, \ldots, a_{k+1}$ are variables of sort Agent and $\#S$ denotes the cardinal of a set S.
To reach the bound k_2 of theorem 2, the important part is the choice of the security property. Consider:

$$\phi = \neg \mathsf{Ha}(x) \vee \neg \mathsf{Agent}(y_1) \vee \cdots \vee \neg \mathsf{Agent}(y_k) \vee \neg I(\{x, y_1, \ldots, y_k\}_{\mathsf{prv}(x)}, t) \\ \vee \mathsf{In}(< x, y_1, \ldots, y_k >, t)$$

Then, using the notation of theorem 2, $k_1 = 1$ and $k_2 = k$. To obtain an attack, there must be a trace t where the intruder is able to know a message of the form $\{a_1, \ldots, a_{k+1}\}_{\mathsf{prv}(a_1)}$, where a_1 is an honest agent identity. This means that the clause $\phi\theta$ must has been used for some ground substitution θ, since the other clauses do not allow the intruder to encrypt with an honest private key. Assume the $\{x\theta, y_1\theta, \ldots, y_k\theta\}$ contains less than $k+1$ elements. Then, by construction of the clauses, the tuple $< x\theta, y_1\theta, \ldots, y_k\theta >$ is in the trace t thus we do not have an attack. Consequently, to have an attack, we need at least $k+1 = k_2 + k_1$ agents.

4 Limitations

In this section, we discuss about limitations of our reduction results. First, it does not hold for security properties expressed as observational equivalence. Secondly, as stated in the introduction, it does not imply any reduction result on the number of sessions. We actually show that an arbitrary number of interleaved sessions can be necessary to obtain a secret data.

4.1 Observational equivalence

Secrecy properties may be expressed using reachability properties but may also be defined using observational equivalence. For example, in spi-calculus [1], a protocol $P(z)$ preserves the secrecy of z if for any terms M and M', the protocols $P(M)$ and $P(M')$ are barbed equivalent, i.e. if for any process O, the processes $P(M) \mid O$ and $P(M') \mid O$ are weakly bisimilar. Intuitively, $P(z)$ preserves the secrecy of z if an observer can not tell the difference between $P(M)$ and $P(M')$. We will not define here the semantics of the spi-calculus and the barbed equivalence, the reader is referred to [1] for precise definitions.

For such properties, our method does not work: the weak bisimilarity is not stable by projection on agent identities. Finding a subclass of processes in spi-calculus for which a similar reduction result holds is an open problem.

4.2 An unbounded number of sessions

Our result does not give any bound on the number of sessions. On the contrary, it is easy to see that an arbitrary number of interleaved sessions can be needed to obtain a secret, using the Post Correspondence Problem (PCP). The encoding we present here is inspired by the encoding given by Michael Rusinowitch (private communication) to show the undecidability of secrecy for general cryptographic protocols.

Let Σ a finite alphabet and $(u_i, v_i)_{1 \le i \le n}, u_i, v_i \in \Sigma^*$ an instance of PCP. We build the following protocol:

$$
\begin{aligned}
A \rightarrow B &: \{< 0, 0 >, 0\}_{K_{ab}} \\
B \rightarrow A &: \{< 0, 0 >, N_b\}_{K_{ab}} \\
A : \{< 0, 0 >, z\}_{K_{ab}} \rightarrow B &: \{< 0, 0 >, \{z\}_{N_a}\}_{K_{ab}} \\
B \rightarrow A &: \{< 0, 0 >, \{z\}_{N_a}\}_{K_{ab}} \\
A : \{<x, y>, \{z\}_{N_a}\}_{K_{ab}} \rightarrow B &: \{<xu_i, yv_i>, z'\}_{K_{ab}}, \{s\}_{\{<xu_i, xu_i>, 0\}_{K_{ab}}} \; 1 \le i \le n
\end{aligned}
$$

K_{ab} is a private key between two honest agents A and B, unknown to the intruder. The protocol encodes PCP. In addition, we have added a "counter" which ensures that as many sessions as the number of couples of words needed for the solution, have to be opened at the same time. All messages exchanged between honest agents are of the form $\{<u, v>, t\}_{K_{ab}}$, an intruder is not able to decrypt or modify them and the term t represents a nonce successively encrypted by other nonces. Let n_t denotes the number of encryption symbols used in t.

The second rule of the agent A consists in incrementing the number of encryption symbols. Note that this rule can be played only if the terms u and v are both equal to zero, i.e., the intruder has not started to build a solution.

Conversely, each time the agent A plays his last rule, the number of encryption has to be decremented. To simulate a solution of PCP for this instance, we have to proceed in the following way:

- opening as many sessions between A and B as the number of couples of words needed for the solution,
- then successively in each session, letting A playing his second rule by forwarding in the session $i+1$ the message obtained from A in the session i,
- eventually, playing successively, but in the reverse order of the sessions, the last rule of A by forwarding only the part corresponding the couple of word needed to build the solution at this step.

One can show by induction that, on the first hand, the secret s is revealed if and only if the instance $(u_i, v_i)_{1\leq i\leq n}$ of PCP has a solution and, on the second hand, an attack needs at least k parallel sessions where k is the number of couples of words of $(u_i, v_i)_{1\leq i\leq n}$ used to construct the smallest solution w:

$$w = u_{\phi(1)} \cdots u_{\phi(n)} = v_{\phi(1)} \cdots v_{\phi(n)}.$$

Our proof is quite informal here but it could be easily formalized in most models of cryptographic protocols.

5 Conclusions

We have shown that it is possible to restrict the number of agents without loss of generality: security properties which fail in an unbounded network, also fail in a small limited network. This does not assume any property of the protocols.

To prove a security property for some protocol P, it is therefore sufficient to consider finitely many instances of the roles of P, typically 2^n where n is the number of roles in P (or $(k+1)^n$ for authentication properties or if we don't allow an agent to be both the sender and the receiver of a message). These numbers are small since $n = 2$ for most protocols (sometimes $n = 3$). They can be further lowered since sessions involving only dishonest agents are not relevant.

This reduction result also provides with a decision result if we assume a passive attacker, i.e. an attacker who may only analyze the messages sent on the net but who cannot forge and send new messages. Indeed, in the presence of such an attacker (or eavesdropper), we can also assume that an agent cannot confuse messages from different sessions: it suffices to label the messages by a session nonce and the rule number (which is often the case for implemented protocols). Thus there is no need to consider interleaving of sessions. In addition, given a set of messages S and a message m, deciding whether the intruder may deduce m from S is in PTIME (side result of [2]). Since our reduction result ensures that only a finite number of agents have to be considered, we conclude that secrecy is decidable in $\mathsf{EXP}(n) \times \mathsf{PTIME}(s)$ where n is the number of roles of the protocol and s is the size of the protocol.

References

1. M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.
2. D. M. Allester. Automatic recognition of tractability in inference relations. In *Journal of the ACM 40(2)*, pages 284–303, April 1993.
3. R. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In *Proc. CONCUR 02*. Springer-Verlag, 2002.
4. R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. In *Proc. CONCUR, vol. 1877 of Lecture Notes in Computer Science, pages 380-394*, 2000.
5. R. Barbuti, P. Mancarella, D. Pedreshi, and F. Turini. A transformation approach to negation in logic programming. *Journal of Logic Programming*, 8:201–228, 1990.
6. B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW: Proceedings of 14th Computer Security Foundations Workshop*. IEEE Computer Society Press, 2001.
7. L. Bozga, Y. Lakhnech, and M. Périn. Pattern-based abstraction for verifying secrecy in protocols. In *Tools and Agorithms for the Construction and Analysis of Sytems (TACAS'03), to appear*, 2003.
8. J. Clark and J. Jacob. A survey of authentication protocol literature: Version, 1997.
9. H. Comon. Disunification: a survey. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*. MIT Press, 1991.
10. H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. Technical Report LSV-01-13, LSV, 2001.
11. H. Comon, V. Cortier, and J. Mitchell. Tree automata with memory, set constraints and ping pong protocols. In *Proc. ICALP 2001*, 2001.
12. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997. release October, 1rst 2002.
13. H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. In *Research Report LSV-02-10, Lab. Specification and Verification, ENS de Cachan, Cachan, France*, August 2002.
14. V. Cortier, J. Millen, and H. Rueß. Proving secrecy is easy enough. In *14th IEEE Computer Security Foundations Workshop*, pages 97–108. IEEE Computer Society, 2001.
15. G. Denker, J. Millen, and H. Rueß. The capsl integrated protocol environment. Technical Report SRI-CSL-2000-02, SRI International, Oct. 2000.
16. N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. of Workshop on Formal Methods and Security Protocols, Trento, 1999.*, 1999.
17. S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1983.
18. M. Fiore and M. Abadi. Computing symbolic models for verifying cryptographic protocols. In *Proc.14th IEEE Computer Security Foundations Workshop*, Cape Breton, Nova Scotia, June 2001.
19. J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *CSFW: Proc. 13th IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2000.

20. J. Heather and S. Schneider. Towards automatic verification of authentication protocols on an unbounded network. In *Proceedings of the 13th Computer Security Foundations Workshop (CSFW'00)*, pages 132–143, Cambridge, England, 2000. IEEE Computer Society Press.
21. F. Jacquemard, M. Rusinowitch, and L. Vigneron. Compiling and verifying cryptographic protocols. In *Proc. Logic Programming and Automated Reasoning*, volume 1955 of *Lecture Notes in Computer Science*, 2000. See also the CASRUL page `http://www.loria.fr/equipes/cassis/softwares/casrul/`.
22. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In T. Margaria and B. Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'96)*, volume 1055, pages 147–166, Passau, Germany, march 1996. Springer-Verlag, Berlin Germany. Also in Software Concepts and Tools, 17:93-102, 1996.
23. G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(2–3):89–146, 1999.
24. J. Millen and H. Rueß. Protocol-independent secrecy. In *RSP: 21th IEEE Computer Society Symposium on Research in Security and Privacy*, 2000.
25. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security*, 2001.
26. L. Paulson. Mechanized proofs for a recursive authentication protocol. In *Proceedings of the 10th Computer Security Foundations Workshop*, pages 84–95. IEEE Computer Society Press, 1997.
27. L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85–128, 1998.
28. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *14th IEEE Computer Security Foundations Workshop*, 2001.
29. P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and A. Roscoe. *The modelling and analysis of security protocols: the CSP approach*. Addison-Wesley, 2000.
30. S. D. Stoller. A bound on attacks on payment protocols. In *Proc. 16th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 61–70. IEEE Computer Society Press, June 2001.
31. J. Thayer, J. Herzog, and J. Guttman. Strand spaces: proving security protocols correct. In *Journal of Computer Security, Vol. 7*, pages 191–230. IEEE Computer Society, 1999.