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 arbitrary cryptographic protocols and security properties. We show that it is always sufficient to consider a bounded number of agents b (actually $b = 2$ in most of the cases): 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 [13], CASRUL/Datac [19], casper/FDR [26].

Though cryptographic protocols are often described in a concise way (see e.g. [7]), the verification problem is difficult for two reasons:

1. The number of agents potentially using the protocol is unbounded, as well as the number of protocol sessions.
2. 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. [15, 14, 9, 2]).

The verification tools have either to assume stronger properties on the protocols (e.g. [20, 10, 27, 2]) or to consider a bounded number of sessions (hence a bounded number of agents) only [3, 25, 16, 22], in which case the security problem becomes co-NP-complete [25]. 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 [6, 5].

In this paper, we consider a simple reduction, which works for any protocols and security properties typically considered for automated verification. We show that it is always sufficient to consider a bounded number of agents b (actually $b = 2$; 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 agents ids and

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

consider finitely many (4 most of the time) instances of the protocol roles, without loosing information. This proves actually that the instanciation techniques in [19] are complete. This also provides completeness results for abstraction used in [6, 23]. 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 [5, 6] 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.

Our result extends and clarifies a side result of [18]. Indeed, J. Heather and S. Schneider chow that one may consider only four agents (three honest, one dishonest) using implicitly that an agent may talk to herself. We prove actually that in J. Heather and S. Schneider case, 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 security property, we simply project every honest identity on one single honest identity and every dishonest identity on one single dishonest identity. 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. It is simply more convenient for our proof and 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 [11]) with a reduction of the Millen-Rueß model [21] to Horn clauses. 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. A more detailed definition can be found in [11]. 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 sets. This result assumes however that 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 [24, 20] explicitly disallow this possibility. That is why we consider in section 3.2 models in which an agent cannot 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 roles in the protocol.

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 [5] 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 member 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. It is a bit lengthy to describe the two parts in details: we will only show here examples and the less standard constructions. The reader is referred to [11] for more details.

2.1 Messages and traces

The set of messages is the 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:

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

Terms of sort Agent are called *agents*. 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.

- srv_i are intended to be server names.
- 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. 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.

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 [11] that this is a reasonable definition since other models can be reduced to this one.

$\mathsf{Agent}(x) \Rightarrow I(x, t)$	The intruder knows all agents ids.
$\mathsf{Da}(x) \Rightarrow I(\mathsf{prv}(x), t)$	The intruder knows all keys of compromised agents.
$I(x, t) \Rightarrow I(\mathsf{pub}(x), t)$	The intruder knows all public keys.
$I(x, t), I(y, t) \Rightarrow I(\{x\}_y, t)$	The intruder can encrypt a known message with a known key.
$I(\{x\}_{\mathsf{shr}(y)}, t), I(\mathsf{shr}(y), t) \Rightarrow I(x, t)$	The intruder can retrieve the clear text of a message encrypted with a known symmetric key.
$T([x, s] \cdot t) \Rightarrow I(x, [x, s] \cdot t)$	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.

Fig. 1. Some of the clauses defining I

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. Some typical clauses defining I are displayed on figure 1. There are other clauses for e.g. (un)pairing.

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

2.3 Protocol dependent clauses

We sketch here how to define the set of valid traces T on the Yahalom protocol. In this section, a, b will stand for variables of sort Agent, x, y, z for variables of sort $\mathsf{Message}$, s, t and e for variables of sort respectively Num, Trace and Event.

$$\begin{array}{l}
A \to B : A, N_a \\
B \to S : B, \{A, N_a, N_b\}_{\mathsf{shr}(B)} \\
S \to A : \{B, K_{ab}, N_a, N_b\}_{\mathsf{shr}(A)}, \{A, K_{ab}\}_{\mathsf{shr}(B)} \\
A \to B : \{A, K_{ab}\}_{\mathsf{shr}(B)}, \{N_b\}_{K_{ab}}
\end{array}$$

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:

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

Fresh is an auxiliary predicate (defined in figure 2), 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 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:

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

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 2 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

Definition of Sup:

$$\begin{aligned} \mathsf{Num}(x) &\Rightarrow \mathsf{Sup}(s(x), 0) \\ \mathsf{Sup}(x, y) &\Rightarrow \mathsf{Sup}(s(x), s(y)) \end{aligned}$$

Definition of Fresh:

$$\begin{aligned} &\Rightarrow \mathsf{Fresh}(\perp, s) \\ \mathsf{Fresh}(t, s), \mathsf{Sup}(s, s') &\Rightarrow \mathsf{Fresh}([e, s'] \cdot t, s) \end{aligned}$$

Definition of In:

$$\begin{aligned} \mathsf{Trace}([e, s] \cdot t) &\Rightarrow \mathsf{In}([e, s], [e, s] \cdot t) \\ \mathsf{In}(x, t) &\Rightarrow \mathsf{In}(x, [e, s] \cdot t) \end{aligned}$$

Definition of $\mathsf{NotPlayed}$:

$$\begin{aligned} &\Rightarrow \mathsf{NotPlayed}(a, i, s, \perp) \\ \mathsf{NotPlayed}(a, i, s, t), \mathsf{Sup}(s, s') &\Rightarrow \mathsf{NotPlayed}(a, i, s, [e, s'] \cdot t) \\ \mathsf{NotPlayed}(a, i, s, t), \mathsf{Sup}(s', s) &\Rightarrow \mathsf{NotPlayed}(a, i, s, [e, s'] \cdot t) \\ \mathsf{NotPlayed}(a, i, s, t), \mathsf{Sup}(i, j) &\Rightarrow \mathsf{NotPlayed}(a, i, s, [st(a, j, m), s] \cdot t) \end{aligned}$$

Fig. 2. Definitions of the auxiliary predicates

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

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:

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

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: $(\forall t,x,y,s)$

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

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 [21, 12] (hereafter referred to as the MR model), expressed in Horn clauses. The MR model is itself inspired from Paulson's model [24] and from the strand spaces [28]. Formally, we proved in [11] 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 $\phi \in \Phi_P$.

3 Reduction to a fixed number of agents

3.1 From n agents to 2 agents

We show that if there is an attack with n agents, then an attack with 2 agents can be constructed: given an attack using n agents, we project every honest identity on one single honest identity and every dishonest identity on one dishonest identity. Then 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 [21, 28, 17, 5, 14, 3]. 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 3.3.

We emphasize that our result holds for all models of protocols which do not make use of our internal representation of agents ids. 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. *Let L be a positive literal of $\mathcal{H}_P$, then $\overline{L}$ is in $\mathcal{H}_P$.*

This is prove 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 $B_1(\boldsymbol{x}),\ldots,B_k(\boldsymbol{x}) \Rightarrow A(\boldsymbol{x}) \mid C$ 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:
$$\begin{cases}\mathsf{Da}(x) \Rightarrow A(x,y)\\ \mathsf{Da}(y) \Rightarrow A(x,y)\\ \qquad\;\; \Rightarrow A(x,x)\end{cases}$$

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 3.3 an extension of theorem 1 for formulas containing positive literals.

3.2 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). For, 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 and we introduce the definition of protocol clauses:

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

- *ϕ does not contain the symbols s_h, s_d,*
- $\mathsf{Distinct}$ *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 $\mathsf{Distinct}$ whose semantic is exactly the pairs of distinct agents. In this case, an *admissible* clause should also verify that $\mathsf{Distinct}$ 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 correspondence between two compromised agents does not increase the intruder knowledge, which is the case of all "real" protocols ([7]). This leads to a specification which can be reduced to the above one.

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 is a purely negative admissible clause. If there is an attack on P for ϕ using n agents, then there is an attack on P for ϕ using at most $k+1$ agents, where k is the number of variables of sort* Ha *occurring in 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, while the specification of the intruder power ($\mathcal{C}_I$) should not give specific knowledge depending on honest agent: either data (depending on agents) are known for all agents (like agent ids) or data are known only for compromised agents (like private keys).

Proof. We keep the notations of the proof of theorem 1. Again, we consider a subset $\mathcal{H}_0$ of $\mathcal{H}_P$ which falsifies ϕ. As before, since ϕ is purely negative, we may assume that $\mathcal{H}_0$ does not contain any negative literal.

Now, we let θ be an instance of ϕ which is falsified by $\mathcal{H}_0$. If $x_1, \ldots x_k$ are the variables of sort Ha in ϕ, we let $s_h^{m_1}(h), \ldots, s_h^{m_p}(h)$ be the set $\{x_1\theta, \ldots, x_k\theta\}$ with $m_1 < \ldots < m_p$ ($p \leq k$). Next, we define the projection function as follows:

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

Again, we let $\mathcal{H}_1 = \{\overline{L} \mid L \in \mathcal{H}_0\}$ and we are going to prove that $\mathcal{H}_1 \subseteq \mathcal{H}_P$ and $\mathcal{H}_1$ falsifies $\phi\overline{\theta}$. This will conclude the proof since $\mathcal{H}_1$ will be an attack with $p+1$ agents: $d, h, s_h(h), \ldots, s_h^{p-1}(h)$, $p \leq k$.

Actually, with the following lemma, the proof that $\mathcal{H}_1 \subseteq \mathcal{H}_P$ is very much the same as in theorem 1:

Lemma 2. *If* $\mathsf{Distinct}(u_1, u_2) \in F_P^n(\emptyset)$*, then* $\overline{\mathsf{Distinct}(u_1, u_2)} \in F_P^n(\emptyset)$.

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 $u_1, u_2 \notin \{t_1, \ldots, t_k\}$, 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 $\overline{\mathsf{Distinct}(u_1, u_2)} = \mathsf{Distinct}(d, d) \in F_P(\emptyset)$;
- if $u_1 \in \{t_1, \ldots, t_k\}$ and $u_2 \notin \{t_1, \ldots, t_k\}$ (or the converse), then $\overline{\mathsf{Distinct}(u_1, u_2)} = \mathsf{Distinct}(s_h^j(h), d)$ (or $\mathsf{Distinct}(d, s_h^j(h))$), which also belongs to $F_P(\emptyset)$;
- if $u_1, u_2 \in \{t_1, \ldots, t_k\}$: $u_1 = s_h^{m_i}(h)$, $u_2 = s_h^{m_j}(h)$ with $i \neq j$, then $\overline{\mathsf{Distinct}(u_1, u_2)} = \mathsf{Distinct}(s_h^i(h), s_h^j(h)) \in F_P^{|j-i|}(\emptyset)$. In this last case, $|j-i| \leq |m_j - m_i|$ by construction, hence the result.

End of the proof of lemma 2.
As in theorem 1, we can now prove by induction on n_L that for any literal $L \in \mathcal{H}_P$, $\overline{L} \in \mathcal{H}_P$: $\mathsf{Distinct}$ literals are handled by lemma 2. We also need here that there is no variable of sort Ha in the clauses, in order to ensure the well-sortedness of $\overline{\sigma}$ (since, now, for some terms $t : \mathsf{Ha}$, $\overline{t}$ is no longer of sort Ha). $\square$

Note that the bound $k+1$ can be reached for some protocols P and some properties ϕ.

Example 6. 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 Lowe attack, 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.

3.3 Extensions

Theorems 1 and 2 assume 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, authentication 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 properties to a class which encompasses authentication and secrecy properties.

Definition 6. *A security property ϕ is* admissible *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, the t_i's are variables of sort Trace *and the u_i's are terms with variables of sort* Num *or* Ha. *In addition ϕ must still verify that:*

- *it does not contain the symbols s_h, s_d,*
- *if a ground subterm of some u_i is of sort* Agent *then it is of sort* Ha.

Then we can reduce such a case to the purely negative case and we get:

Theorem 3. *Assume that $\mathcal{C}_P$ is an admissible set of clauses, which does not contain any variable of sort* Ha*, ϕ is an admissible security property, then if there is an attack on P for ϕ using n agents, there is an attack on P for ϕ using at most $k+1$ agents, where k is the number of variables of sort* Ha *occurring in ϕ.*

For instance, 3 agents are sufficient if we consider the above-specified authentication property.
Proof sketch:
For every positive literal $L = \mathsf{In}(u_i,t_i)$ occurring in ϕ, we construct a set of Horn clauses C_L defining a predicate $\widetilde{L}$ and such that:

- the least Herbrand model $\mathcal{H}_{P,\phi}$ of $\mathcal{C}_P \cup \mathcal{C}_{\neq} \cup \bigcup_L C_L$ contains $\mathcal{H}_P$;
- for every (well sorted) ground substitution σ, $\mathcal{H}_P \not\models L\sigma$ iff $\mathcal{H}_{P,\phi} \models \widetilde{L}\sigma$;
- the new set of clauses $\mathcal{C}_P \cup \bigcup_L C_L$ is admissible.

We first construct C_L using the complementation techniques, which yields a definition of the predicates negations (see e.g. [4, 8]). Let $x_1^i, \ldots, x_n^i$ be the variables of u_i. The set of clauses C_L is defined by:

$$\begin{aligned} &\Rightarrow \widetilde{L}(x_1^i,\ldots,x_n^i,\perp) \\ \widetilde{L}(x_1^i,\ldots,x_n^i,t), \mathsf{Diff}(u_i,y) &\Rightarrow \widetilde{L}(x_1^i,\ldots,x_n^i,y\cdot t) \end{aligned}$$

These clauses satisfy the above two first conditions. However, they make use of a predicate symbol Diff, whose semantics is the set of pairs of distinct terms, and the definition of Diff is not admissible. Then, we remove clauses defining Diff which are not admissible, and replace negative literals $\neg\mathsf{Diff}(x,y)$ where x,y are of sort Agent with $\neg\mathsf{Distinct}(x,y)$. The resulting clauses satisfy the three above conditions since the semantics for the new definition of Diff, restricted to instanciations of pairs (u_i,y), is still a set of pairs of distinct terms. $\square$

4 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 $(k+1)^n$ 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 only involving 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 [1]). Since our reduction result ensures that only a finite number of agents have to be considered, we conclude that secrecy is decidable in $\mathsf{EXP}(n) \times \mathsf{PTIME}$ where n is the number of roles of the protocol.

Acknowledgments : We would to thank Michael Périn and anonymous referees for their helpful comments.

References

1. D. M. Allester. Automatic recognition of tractability in inference relations. In *Journal of the ACM 40(2)*, pages 284–303, April 1993.
2. R. Amadio and W. Charatonik. On name generation and set-based analysis in the dolev-yao model. In *Proc. CONCUR 02*. Springer-Verlag, 2002.
3. 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.
4. 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.
5. 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.
6. 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.
7. J. Clark and J. Jacob. A survey of authentication protocol literature: Version, 1997.
8. H. Comon. Disunification: a survey. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*. MIT Press, 1991.
9. H. Comon and V. Cortier. Tree automata with one memory, set constraints and cryptographic protocols. Technical Report LSV-01-13, LSV, 2001.

10. H. Comon, V. Cortier, and J. Mitchell. Tree automata with memory, set constraints and ping pong protocols. In *Proc. ICALP 2001*, 2001.
11. 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.
12. 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.
13. G. Denker, J. Millen, and H. Rueß. The capsl integrated protocol environment. Technical Report SRI-CSL-2000-02, SRI International, Oct. 2000.
14. 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.
15. S. Even and O. Goldreich. On the security of multi-party ping-pong protocols. In *Proc. IEEE Symp. on Foundations of Computer Science*, 1983.
16. 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.
17. 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.
18. 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.
19. 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/`.
20. G. Lowe. Towards a completeness result for model checking of security protocols. *Journal of Computer Security*, 7(2–3):89–146, 1999.
21. J. Millen and H. Rueß. Protocol-independent secrecy. In *RSP: 21th IEEE Computer Society Symposium on Research in Security and Privacy*, 2000.
22. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security*, 2001.
23. 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.
24. L. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1):85–128, 1998.
25. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *14th IEEE Computer Security Foundations Workshop*, 2001.
26. P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and A. Roscoe. *The modelling and analysis of security protocols: the CSP approach*. Addison-Wesley, 2000.
27. 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.
28. 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.