Modeling and Verifying Ad Hoc Routing Protocols *

Mathilde Arnaud[1,2], Véronique Cortier[2], and Stéphanie Delaune[1]

[1] LSV, ENS Cachan & CNRS & INRIA, France
[2] LORIA, CNRS & INRIA Nancy Grand Est, France

Abstract. Mobile ad hoc networks consist of mobile wireless devices which autonomously organize their infrastructure. In such a network, a central issue, ensured by routing protocols, is to find a route from one device to another. Those protocols use cryptographic mechanisms in order to prevent a malicious node from compromising the discovered route.

We present a calculus for modeling and reasoning about security protocols, including in particular secured routing protocols. Our calculus extends standard symbolic models to take into account the characteristics of routing protocols and to model wireless communication in a more accurate way. Then, by using constraint solving techniques, we propose a decision procedure for analyzing routing protocols for a bounded number of sessions and for a fixed network topology. We demonstrate the usage and usefulness of our approach by analyzing the protocol SRP applied to DSR.

1 Introduction

Mobile ad hoc networks consist of mobile wireless devices which autonomously organize their communication infrastructure: each node assumes the function of a router and relays packets on paths to other nodes. Finding these paths in an a priori unknown and constantly changing network topology is a crucial functionality of any ad hoc network. Specific protocols, called *routing protocols*, are designed to ensure this functionality known as *route discovery*.

Prior research in ad hoc networking has generally studied the routing problem in a non-adversarial setting, assuming a trusted environment. Thus, many of the currently proposed routing protocols for mobile ad hoc networks are assumed to be employed in a friendly environment (e.g. [13, 7]). Recent research has recognized that this assumption is unrealistic [6, 10, 4]. It is important to prevent malicious nodes from compromising the discovered routes. Since then, secure versions of routing protocols have been developed to ensure that mobile ad hoc networks can work even in an adversarial setting [18, 6, 11]. Those protocols use cryptographic mechanisms such as encryption, signature, MAC, to prevent a malicious node to insert and delete nodes inside a path.

* This work has been partially supported by the ANR project SeSur AVOTÉ.

Formal modeling and analysis techniques are well-adapted for checking correctness of security protocols. Formal methods have for example been proved successful for authentication or key establishment security protocols and a multitude of effective frameworks have been proposed (e.g. [16, 12, 2] to cite only a few). However, there are very few attempts to use formal methods in the context of mobile ad hoc networks. They indeed involve several subtleties (like a different intruder model or a particular network topology) that cannot be reflected in existing work.

Our contributions. The first main contribution of this paper is the proposition of a calculus, inspired from CBS# [10], which allows mobile wireless networks and their security properties to be formally described and analyzed. We model cryptography as a black box (the perfect cryptography assumption), thus the attacker cannot break cryptography, e.g. decrypt a message without having the appropriate decryption key. To model routing protocols in an accurate way, some features need to be taken into account. Among them:

– *Network topology*: nodes can only communicate (in a direct way) with their neighbor.
– *Broadcast communication*: the main mode of communication is broadcasting and only adjacent nodes receive messages
– *Internal states*: nodes are not memory-less but store some information in routing tables with impact on future actions.

There are also some implications for the attacker model. Indeed, in most existing formal approaches, the attacker controls the entire network. This abstraction is reasonable for reasoning about classical protocols. However, in the context of routing protocols, this attacker model is too strong and leads to a number of false attacks. The constraints on communication also apply to the attacker. Our attacker can only intercept messages at his location.

Our second main contribution is to provide a decision procedure for analyzing routing protocols for a bounded number of sessions and for a fixed topology. We first show how the analysis of routing protocols can be reduced to (generalized) constraint systems solving. We then adapt and generalize existing techniques [5] for solving our more general constraint systems. We demonstrate the usage and usefulness of our model and techniques by analyzing SRP (Secure Routing Protocol) [11] applied on the protocol DSR (Dynamic Routing Protocol) [8]. This allows us to retrieve an attack presented first in [4].

Related work. Recently, several frameworks have been proposed to model wireless communication and/or routing protocols in a more accurate way (e.g. [15, 17, 10]). However, these models do not take into account several features that seem crucial to model routing protocols (e.g. neighborhood of two nodes). Moreover, in [17], they only provide a semi-decision procedure (no decidability result). In [10], it seems that they only perform their analysis for a particular scenario of attacker, given as a fixed process. In this work, we consider a bounded number of sessions but we do not need to specify what are the different steps performed by the attacker to mount an attack.

2 Models for protocols

2.1 Messages

Cryptographic primitives are represented by function symbols. More specifically, we consider a *signature* $(\mathcal{S}, \mathcal{F})$ made of a set of *sorts* $\mathcal{S}$ and *function symbols* $\mathcal{F}$ together with arities of the form $ar(f) = s_1 \times \ldots \times s_k \to s$. We consider an infinite set of *variables* $\mathcal{X}$ and infinite set of *names* $\mathcal{N}$ that typically represent nonces or agent names. In particular, we consider a special sort loc for the nodes of the network. We assume that names and variables are given with sorts. We also assume an infinite subset $\mathcal{N}_{\mathsf{loc}}$ of names of sort loc. The set of *terms of sort s* is defined inductively by:

$$
\begin{array}{lll}
t ::= & & \text{term of sort } s \\
\quad \mid & x & \text{variable } x \text{ of sort } s \\
\quad \mid & a & \text{name } a \text{ of sort } s \\
\quad \mid & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F}
\end{array}
$$

where t_i is a term of some sort s_i and $ar(f) = s_1 \times \ldots \times s_k \to s$. We assume a special sort terms that subsumes all the other sorts and such that any term is of sort terms. We write $var(t)$ for the set of variables occurring in a term t. The term t is said to be a *ground* term if $var(t) = \emptyset$.

Example 1. For example, we will consider the specific signature $(\mathcal{S}_1, \mathcal{F}_1)$ defined by $\mathcal{S}_1 = \{\mathsf{loc}, \mathsf{lists}, \mathsf{terms}\}$ and $\mathcal{F}_1 = \{hmac, \langle\rangle, ::, \perp\}$, with the following arities: $hmac : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$, $\langle\rangle : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$, $:: : \mathsf{loc} \times \mathsf{lists} \to \mathsf{lists}$, $\perp :\to \mathsf{lists}$. The sort lists represents lists of terms of sort loc. The symbol $::$ is the list constructor. $\perp$ is a constant representing an empty list. The term $hmac(m, k)$ represents the keyed hash message authentication code computed over message m with key k while $\langle\rangle$ is a pairing operator. We write $\langle t_1, t_2, t_3\rangle$ for the term $\langle t_1, \langle t_2, t_3\rangle\rangle$, and $[t_1; t_2; t_3]$ for $t_1 :: (t_2 :: (t_3 :: \perp))$.

Substitutions are written $\sigma = \{^{t_1}/_{x_1}, \ldots, ^{t_n}/_{x_n}\}$ with $\mathsf{dom}(\sigma) = \{x_1, \ldots, x_n\}$. We only consider *well-sorted* substitutions, that is substitutions for which x_i and t_i have the same sort. σ is *ground* if and only if all of the t_i are ground. The application of a substitution σ to a term t is written $\sigma(t)$ or $t\sigma$. A most general unifier of two terms t and u is a substitution denoted by $\mathsf{mgu}(t, u)$. We write $\mathsf{mgu}(t, u) = \perp$ when t and u are not unifiable.

The ability of the intruder is modeled by a deduction relation $\vdash \subseteq 2^{\mathsf{terms}} \times \mathsf{terms}$. The relation $S \vdash t$ represents the fact that the term t is computable from the set of terms S. The deduction relation can be arbitrary in our model thus is left unspecified. It is typically defined through a deduction system. For example, for the term algebra $(\mathcal{S}_1, \mathcal{F}_1)$ defined in Example 1, the deduction system presented in Figure 1 reflects the ability for the intruder to concatenate terms, to compute MAC when he knows the key, and to build lists. Moreover, he is able to retrieve components of a pair or of a list.

$$\frac{u \in T}{T \vdash u}\ (\mathrm{A}) \qquad \frac{T \vdash a \quad T \vdash l}{T \vdash a :: l}\ (\mathrm{LC}) \qquad \frac{T \vdash a :: l}{T \vdash a}\ (\mathrm{LH}) \qquad \frac{T \vdash a :: l}{T \vdash l}\ (\mathrm{LT})$$

$$\frac{T \vdash u \quad T \vdash v}{T \vdash hmac(u, v)}\ (\mathrm{C}) \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v\rangle}\ (\mathrm{P}) \qquad \frac{T \vdash \langle u, v\rangle}{T \vdash u}\ (\mathrm{UL}) \qquad \frac{T \vdash \langle u, v\rangle}{T \vdash v}\ (\mathrm{UR})$$

Fig. 1. Deduction system for hmac and lists.

2.2 Process calculus

Several calculi already exist to model security protocols (e.g. [2, 1]). However, for our purpose, a node, i.e. a process, has to perform some specific actions that can not be easily modeled in such calculi. For instance, a node stores some information, e.g. the content of its routing table. We also need to take into account the network topology and to model broadcast communication. Such features can not be easily modeled in these calculi. Actually, our calculus is inspired from CBS# [10], which allows mobile wireless networks and their security properties to be formally described and analyzed. However, we extend this calculus to be able to check neighborhood properties.

The intended behavior of each node of the network can be modeled by a *process* defined by the grammar given below. Our calculus is parametrized by a set $\mathcal{L}$ of formulas.

$P, Q ::=$	Processes
$\quad 0$	null process
$\quad \mathsf{out}(u).P$	emission
$\quad \mathsf{in}\ u[\Phi].P$	reception, $\Phi \in \mathcal{L}$
$\quad \mathsf{store}(u).P$	storage
$\quad \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q$	reading
$\quad \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q$	conditional, $\Phi \in \mathcal{L}$
$\quad P \mid Q$	parallel composition
$\quad !P$	replication
$\quad \mathsf{new}\ m.P$	fresh name generation

The process $\mathsf{out}(u).P$ emits u and then behaves like P. The process $\mathsf{in}\ u[\Phi].P$ expects a message m of the form u such that Φ is true and then behaves like $P\sigma$ where $\sigma = \mathsf{mgu}(m, u)$. If Φ is the true formula, we simply write $\mathsf{in}\ u.P$. The process $\mathsf{store}(u).P$ stores u in its storage list and then behaves like P. The process $\mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q$ looks for a message of the form u in its storage list and then, if such an element m is found, it behaves like $P\sigma$ where $\sigma = \mathsf{mgu}(m, u)$. If no element of the form u is found, then it behaves like Q.

Secured routing protocols typically perform some checks on the route they received before accepting a message. Thus we will typically consider the logic $\mathcal{L}_{\mathsf{route}}$ defined by the following grammar:

$\Phi ::=$	Formula
$\mathsf{check}(a,b)$	neighborhood of two nodes
$\mathsf{checkl}(c,l)$	local neighborhood of a node in a list
$\mathsf{route}(l)$	validity of a route
$\mathsf{loop}(l)$	existence of a loop in a list
$\Phi_1 \wedge \Phi_2$	conjunction
$\Phi_1 \vee \Phi_2$	disjunction
$\neg\Phi$	negation

Given a graph $G = (V, E)$, the semantics $[\![\Phi]\!]$ of a formula $\Phi \in \mathcal{L}_{\mathsf{route}}$ is recursively defined by:

- $[\![\mathsf{check}(a,b)]\!] = 1$ iff $(a,b) \in E$, with a, b of sort loc,
- $[\![\mathsf{checkl}(c,l)]\!] = 1$ iff c is of sort loc, l is of sort lists, c appears exactly once in l, and for any l' sub-list of l,
  - if $l' = a :: c :: l_1$, then $(a,c) \in E$.
  - if $l' = c :: b :: l_1$, then $(b,c) \in E$.
- $[\![\mathsf{route}(l)]\!] = 1$ iff l is of sort lists and for every l' sub-list of l, if $l' = a :: b :: l_1$, then $(a,b) \in E$.
- $[\![\mathsf{loop}(l)]\!]$ iff l is of sort lists and there exists an element appearing at least twice in l,
- $[\![\Phi_1 \wedge \Phi_2]\!] = [\![\Phi_1]\!] \wedge [\![\Phi_2]\!]$, $[\![\Phi_1 \vee \Phi_2]\!] = [\![\Phi_1]\!] \vee [\![\Phi_2]\!]$, and $[\![\neg\Phi]\!] = \neg[\![\Phi]\!]$.

Example 2. Consider the secure routing protocol SRP introduced in [11], with the assumption that each node knows who his neighbors are. SRP is not a routing protocol by itself, it describes a generic way for securing source-routing protocols. We study its application to the DSR protocol [8]. DSR is a protocol which is used when an agent S (the source) wants to communicate with another agent D (the destination), which is not his immediate neighbor. In an ad hoc network, messages can not be sent directly to the destination, but have to travel along a path of nodes.

To discover a route to the destination, the source constructs a request packet and broadcasts it to its neighbors. The request packet contains its name S, the name of the destination D, an identifier of the request id, a list containing the beginning of a route to D, and a hmac computed over the content of the request with a key K_{SD} shared by S and D. It then waits for an answer containing a route to D with a hmac matching this route, and checks that it is a plausible route, *i.e.* checks that the route does not contain a loop and that his neighbor in the route is indeed a neighbor of S in the network.

Consider the signature given in Example 1 and let $S, D, \mathsf{req}, \mathsf{rep}, id, K_{SD}$ be names ($S, D \in \mathcal{N}_{\mathsf{loc}}$) and x_L be a variable of sort lists. The process executed by a node S initiating the search of a route towards a node D is:

$$P_{\mathsf{init}}(S,D) = \mathsf{new}\ id.\mathsf{out}(u_1).\mathsf{in}\ u_2[\Phi_S].0$$

$$\text{where} \begin{cases} u_1 = \langle \mathsf{req}, S, D, id, S :: \bot, hmac(\langle \mathsf{req}, S, D, id\rangle, K_{SD})\rangle \\ u_2 = \langle \mathsf{rep}, D, S, id, x_L, hmac(\langle \mathsf{rep}, D, S, id, x_L\rangle, K_{SD})\rangle \\ \Phi_S = \mathsf{checkl}(S, x_L) \wedge \neg \mathsf{loop}(x_L) \end{cases}$$

The names of the intermediate nodes are accumulated in the route request packet. Intermediate nodes relay the request over the network, except if they have already seen it. An intermediate node also checks that the received request is locally correct by verifying whether the head of the list in the request is one of its neighbors. Below, $V \in \mathcal{N}_{\mathsf{loc}}$, x_S, x_D and x_a are variables of sort loc whereas x_r is a variable of sort lists and x_{id}, x_m are variables of sort terms. The process executed by an intermediary node V when forwarding a request is as follows:

$$P_{\mathsf{req}}(V) = \mathsf{in}\ w_1[\Phi_V].\mathsf{read}\ t\ \mathsf{then}\ 0\ \mathsf{else}\ (\mathsf{store}(t).\mathsf{out}(w_2).0)$$

$$\text{where} \begin{cases} w_1 = \langle \mathsf{req}, x_S, x_D, x_{id}, x_a :: x_r, x_m\rangle \\ \Phi_V = \mathsf{check}(V, x_a) \\ t = \langle x_S, x_D, x_{id}\rangle \\ w_2 = \langle \mathsf{req}, x_S, x_D, x_{id}, V :: (x_a :: x_r), x_m\rangle \end{cases}$$

When the request reaches the destination D, it checks that the request has a correct hmac and that the first node in the route is one of his neighbors. Then, D constructs a route reply, in particular it computes a new hmac over the route accumulated in the request packet with K_{SD}, and sends the answer back over the network..

The process executed by the destination node D is the following:

$$P_{\mathsf{dest}}(D, S) = \mathsf{in}\ v_1[\Phi_D].\mathsf{out}(v_2).0$$

$$\text{where} \begin{cases} v_1 = \langle \mathsf{req}, S, D, x_{id}, x_a :: x_l, hmac(\langle \mathsf{req}, S, D, x_{id}\rangle, K_{SD})\rangle \\ \Phi_D = \mathsf{check}(D, x_a) \\ v_2 = \langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l, hmac(\langle D, S, x_{id}, x_a :: x_r\rangle, K_{SD})\rangle \end{cases}$$

Then, the reply travels along the route back to S. The intermediary nodes check that the route in the reply packet is locally correct (that is that they appear once in the list and that the nodes before and after them are their neighbors) before forwarding it. The process executed by an intermediary node V when forwarding a reply is the following:

$$P_{\mathsf{rep}}(V) = \mathsf{in}\ w'[\Phi'_V].\mathsf{out}(w')$$

$$\text{where} \begin{cases} w' = \langle \mathsf{rep}, x_D, x_S, x_{id}, x_r, x_m\rangle \\ \Phi'_V = \mathsf{checkl}(V, x_r) \end{cases}$$

2.3 Execution model

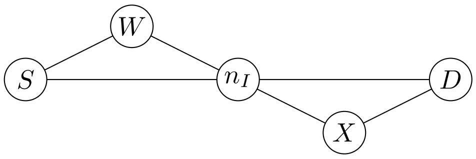
Each process is located at a specified node of the network. Unlike classical Dolev-Yao model, the intruder does not control the entire network but can only interact with its neighbors. More specifically, we assume given a graph $G = (V, E)$ with $V \subseteq \mathcal{N}_{\mathsf{loc}}$, that represents the topology of the network. We assume that one special node $n_I \in V$ is controlled by the intruder. This node is then called *malicious*. A *concrete configuration* of the network is represented by a triplet $(\mathcal{P}; \mathcal{S}; \mathcal{I})$ where:

– $\mathcal{P}$ is a multiset of expressions of the form $\lfloor P \rfloor_n$ where null processes, i.e. expressions of the form $\lfloor 0 \rfloor_n$ are removed. $\lfloor P \rfloor_n$ represents the (ground) process P located at the node $n \in V$. We will write $\lfloor P \rfloor_n \cup \mathcal{P}$ instead of $\{\lfloor P \rfloor_n\} \cup \mathcal{P}$.
– $\mathcal{S}$ is a set of expressions of the form $\lfloor t \rfloor_n$ with $n \in V$ and t a ground term. $\lfloor t \rfloor_n$ represents the fact that the node n has stored the term t.
– $\mathcal{I}$ is a set of terms representing the messages seen by the intruder.

Example 3. Consider the topology described below. A typical initial configuration for the SRP protocol is

$$K_0 = \lfloor P_{\mathsf{init}}(S, D) \rfloor_S \mid \lfloor P_{\mathsf{dest}}(D, S) \rfloor_D; \emptyset; \mathcal{I}_0$$

The nodes S and D want to communicate, n_i is a malicious node whereas the others, i.e. W and X are honest. We assume that each node has an empty storage list and the initial knowledge of the intruder is $\mathcal{I}_0$.



Each honest node broadcasts its message to all its neighbors. To capture more malicious behaviors, we allow the node controlled by the intruder to send messages only to some specific neighbor. The communication system is formally defined by the rules of Figure 2. They are parametrized by the underlying graph G.

The relation $\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$. We may write $\rightarrow_G$ instead of $\rightarrow$ when the underlying network topology G is not clear from the context.

Example 4. Continuing Example 2, the following sequence of transitions is enabled from the initial configuration K_0.

$$K_0 \rightarrow^* \lfloor \mathsf{in}\ u_2[\Phi_S].0 \rfloor_S \cup \lfloor P_{\mathsf{dest}}(D, S) \rfloor_D; \emptyset; \mathcal{I}_0 \cup \{u_1\}$$

$$\text{where} \quad \begin{cases} u_1 = \langle \mathsf{req}, S, D, id, S :: \bot, hmac(\langle \mathsf{req}, S, D, id \rangle, K_{SD}) \rangle \\ u_2 = \langle \mathsf{rep}, D, S, id, x_L, hmac(\langle \mathsf{rep}, D, S, id, x_L \rangle, K_{SD}) \rangle \\ \Phi_S = \mathsf{checkl}(S, x_L) \wedge \neg \mathsf{loop}(x_L) \end{cases}$$

During this transition, S broadcasts a request to find a route to D to its neighbors. The intruder n_I is a neighbor of S, so he learns the request message. Assuming that the intruder knows the names of its neighbors, i.e. $W, X \in \mathcal{I}_0$, he can then build a fake message request:

$$m = \langle \mathsf{req}, S, D, id, [X; W; S], hmac(\langle \mathsf{req}, S, D, id, \rangle, K_{SD}) \rangle$$

and send it to D. Since $(X, D) \in E$, D accepts this message and the resulting configuration of the transition is

$$\lfloor \mathsf{in}\ u_2[\Phi_S].0 \rfloor_S \cup \lfloor \mathsf{out}(v_2\sigma).0 \rfloor_D; \emptyset; \mathcal{I}_0 \cup \{u_1\}$$

$$\text{where} \quad \begin{cases} v_2 = \langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l, hmac(\langle D, S, x_{id}, x_a :: x_r \rangle, K_{SD}) \rangle \\ \sigma = \{{}^{id}/_{x_{id}}, {}^{X}/_{x_a}, {}^{[W;S]}/_{x_L}\} \end{cases}$$

$$\begin{array}{rcl}
\{\lfloor \mathsf{in}\ u_j[\Phi_j].P_j\rfloor_{n_j} \mid (n,n_j)\in E \wedge [\![\Phi_j\sigma_j]\!]=1\} & \rightarrow & \{\lfloor P_j\sigma_j\rfloor_{n_j}\} \cup \lfloor P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}' \\
\cup\, \lfloor \mathsf{out}(t).P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} & & \begin{cases} \text{if } (n,n_I)\in E \text{ then } \mathcal{I}'=\mathcal{I}\cup\{t\} \\ \text{if } (n,n_I)\notin E \text{ then } \mathcal{I}'=\mathcal{I} \end{cases} \\
\text{where } \sigma_j = \mathsf{mgu}(t,u_j) & & \\
\lfloor \mathsf{in}\ u[\Phi].P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} & \rightarrow & \lfloor P\sigma\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} \\
& & \text{if } (n_I,n)\in E,\ \mathcal{I}\vdash t,\ \sigma=\mathsf{mgu}(t,u) \text{ and } [\![\Phi\sigma]\!]=1 \\
\lfloor \mathsf{store}(t).P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} & \rightarrow & \lfloor P\rfloor_n \cup \mathcal{P};\lfloor t\rfloor_n\cup\mathcal{S};\mathcal{I} \\
\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\lfloor t\rfloor_n\cup\mathcal{S} & \rightarrow & \lfloor P\sigma\rfloor_n \cup \mathcal{P};\lfloor t\rfloor_n\cup\mathcal{S};\mathcal{I} \\
& & \text{where } \sigma=\mathsf{mgu}(t,u) \\
\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} & \rightarrow & \lfloor Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} \\
& & \text{if for all } t \text{ such that } \lfloor t\rfloor_n\in\mathcal{S},\ \mathsf{mgu}(t,u)=\perp \\
\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} & \rightarrow & \lfloor P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} \qquad \text{if } [\![\Phi]\!]=1 \\
\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} & \rightarrow & \lfloor Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} \qquad \text{if } [\![\Phi]\!]=0 \\
\lfloor P_1 \mid P_2\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} & \rightarrow & \lfloor P_1\rfloor_n \cup \lfloor P_2\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} \\
\lfloor !P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} & \rightarrow & \lfloor P\alpha\rfloor_n \cup \lfloor !P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} \\
& & \text{where } \alpha \text{ is a renaming of the bound variables of } P \\
\lfloor \mathsf{new}\ m.P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} & \rightarrow & \lfloor P\{^{m'}/_m\}\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I} \\
& & \text{where } m' \text{ is a fresh name}
\end{array}$$

Fig. 2. Concrete transition system.

As usual, an attack is defined as a reachability property.

Definition 1. *Let G be a graph. There is an attack on a configuration with a hole $(\mathcal{P}[\_];\mathcal{S};\mathcal{I})$ for the network topology G and the formula Φ if there exist $n, \mathcal{P}', \mathcal{S}', \mathcal{I}'$ such that $(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0];\mathcal{S};\mathcal{I}) \rightarrow_G^* (\lfloor \mathsf{out}(\mathsf{error})\rfloor_n \cup \mathcal{P}', \mathcal{S}', \mathcal{I}')$ where error is a special symbol.*

The usual secrecy property can be typically encoded by adding a witness process in parallel. For example, the process $W = \mathsf{in}\ s.\_$ can only evolve if it receives the secret s. Thus the secrecy preservation of s on a configuration $(\mathcal{P};\mathcal{S};\mathcal{I})$ can be defined by the (non) existence of an attack on the configuration $(\mathcal{P} \cup \lfloor W\rfloor_{n_I};\mathcal{S};\mathcal{I})$ and the formula true.

Example 5. For SRP, the property we want to check is that the list of nodes obtained by the source through the protocol represents a path in the graph. We can easily encode this property by replacing the null process in $P_{\mathsf{init}}(S,D)$ by a hole, and testing the formula $\neg\mathsf{route}(x_L)$. Let $P'_{\mathsf{init}}(S,D)$ be the resulting process. Then, we recover the attack mentioned in [4] with the topology given in Example 3, and from the initial configuration:

$$K'_0 = \lfloor P'_{\mathsf{init}}(S,D)\rfloor_S \mid \lfloor P_{\mathsf{dest}}(D,S)\rfloor_D;\emptyset;\mathcal{I}_0.$$

Indeed, we have that:

$$\begin{array}{rl} K'_0 \to^* & \lfloor \mathsf{in}\ u_2[\Phi_S].P \rfloor_S \cup \lfloor \mathsf{out}(m').0 \rfloor_D; \emptyset; \mathcal{I} \\ \to & \lfloor \mathsf{in}\ u_2[\Phi_S].P \rfloor_S \cup \lfloor 0 \rfloor_D; \emptyset; \mathcal{I}' \\ \to & \lfloor \mathsf{if}\ \neg\mathsf{route}([X;W;S])\ \mathsf{then}\ \mathsf{out}(\mathsf{error}).0\ \mathsf{else}\ 0 \rfloor_S; \emptyset; \mathcal{I}' \\ \to & \lfloor \mathsf{out}(\mathsf{error}).0 \rfloor_S; \emptyset; \mathcal{I}' \end{array}$$

$$\text{where} \quad \begin{cases} m' = \langle \mathsf{rep}, D, S, id, [X;W;S], hmac(\langle D, S, id, [X;W;S]\rangle, K_{SD})\rangle \\ \mathcal{I} = \mathcal{I}_0 \cup \{u_1\}, \text{and} \\ \mathcal{I}' = \mathcal{I}_0 \cup \{u_1\} \cup \{m'\}. \end{cases}$$

3 Symbolic semantics

It is difficult to directly reason with the transition system defined in Figure 2 since it is infinitely branching. Indeed, a potentially infinite number of distinct messages can be sent at each step by the intruder node. That is why it is often interesting to introduce a *symbolic* transition system where each intruder step is captured by a single rule (e.g. [3]).

3.1 Constraint systems

As in [9, 5, 14], groups of executions can be represented using constraint systems. However, compared to previous work, we have to enrich constraint systems in order to cope with the formula that are checked upon the reception of a message and also in order to cope with generalized disequality tests for reflecting cases where agents reject messages of the wrong form.

Definition 2 (constraint system). *A* constraint system *is a finite sequence $\mathcal{C}$ of* constraints *of the form $t = u$ (unification constraint), $\mathcal{I} \Vdash u$ (deduction constraint), $\forall X. t \neq u$ (disequality constraint), and Φ (formula of $\mathcal{L}_{\mathsf{route}}$), where t, u are terms, $\mathcal{I}$ is a non empty set of terms, and X is a set of variables. Moreover, the sequence gives an order $<$ on left hand sides of deduction and unification constraints such that:*

- *If $(\mathcal{I} \Vdash u) \in \mathcal{C}$ and $(\mathcal{I}' \Vdash u') \in \mathcal{C}$, then either $\mathcal{I} \subseteq \mathcal{I}'$ and $\mathcal{I} \leq \mathcal{I}'$, or $\mathcal{I}' \subseteq \mathcal{I}$ and $\mathcal{I}' \leq \mathcal{I}$.*
- *If $(T \star u) \in \mathcal{C}$ where $\star \in \{=, \Vdash\}$ and $x \in Var(T)$, then*
$$T_x = min_<\{T' \mid (T' \star v) \in \mathcal{C}, x \in Var(v), \star \in \{=, \Vdash\}\}$$
exists and $T_x < T$.

The ordering condition ensures that variables are always introduced by an unification constraint or a deduction constraint, which is always the case when modeling protocols. We denote by $rvar(\mathcal{C})$ the set of variables introduced in $\mathcal{C}$ in the right-hand-side of a unification constraint or a deduction constraint: $rvar(\mathcal{C})$ represents in fact all the free variables appearing in $\mathcal{C}$.

A *solution* to a constraint system $\mathcal{C}$ is a ground substitution θ such that $dom(\theta) = rvar(\mathcal{C})$ and for all $T = U \in \mathcal{C}$, $T\theta = U\theta$; for all $S \Vdash U \in \mathcal{C}$, $S\theta \vdash U\theta$; for all $(\forall X. T \neq U) \in \mathcal{C}$, then $T\theta$ and $U\theta$ are not unifiable (even renaming the variables of X with fresh variables); and for all formula Φ, $[\![\Phi\sigma]\!] = 1$.

Example 6. Let $\mathcal{C} = \{\mathcal{I}_0 \cup \{u_1\} \Vdash v_1;\ \mathcal{I}_0 \cup \{u_1, v_2\} \Vdash u_2;\ \Phi_S;\ \neg\mathsf{route}(x_L)\}$

$$\text{with} \begin{cases} u_1 = \langle \mathsf{req}, S, D, id, S :: \bot, hmac(\langle \mathsf{req}, S, D, id\rangle, K_{SD})\rangle \\ u_2 = \langle \mathsf{rep}, D, S, id, x_L, hmac(\langle \mathsf{rep}, D, S, id, x_L\rangle, K_{SD})\rangle \\ \Phi_S = \mathsf{checkl}(S, x_L) \wedge \neg\mathsf{loop}(x_L) \\ v_1 = \langle \mathsf{req}, S, D, x_{id}, x_a :: x_l, hmac(\langle \mathsf{req}, S, D, x_{id}\rangle, K_{SD})\rangle \\ v_2 = \langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l, hmac(\langle D, S, x_{id}, x_a :: x_r\rangle, K_{SD})\rangle \end{cases}$$

We have that $\mathcal{C}$ is a constraint system, and $\sigma = \{{}^{id}/_{x_{id}}, {}^{X}/_{x_a}, {}^{[W;S]}/_{x_L}\}$ is a solution of the constraint system $\mathcal{C}$.

3.2 Transition system

Concrete executions can be finitely represented by executing the transitions *symbolically.* A *symbolic configuration* is a quadruplet $\mathcal{P}; \mathcal{S}; \mathcal{I}, \mathcal{C}$ where

- $\mathcal{P}$ is a multiset of expressions of the form $\lfloor P \rfloor_n$ where null processes are removed. $\lfloor P \rfloor_n$ represents the process P located at the node $n \in V$;
- $\mathcal{S}$ is a set of expressions of the form $\lfloor t \rfloor_n$ with $n \in V$ and t a term (not necessarily ground).
- $\mathcal{I}$ is a set of terms representing the messages seen by the intruder.
- $\mathcal{C}$ is a constraint system.

Symbolic transitions are defined in Figure 3, they mimic concrete ones. In particular, for the second rule, the set I of processes ready to input a message is split into three sets: the set J of processes that accept the message T, the set K of processes that reject the message T because T does not unify with the expected pattern U_j, and the set L that reject the message T because the condition ϕ is not fulfilled.

We can easily check that whenever $\mathcal{P}; \mathcal{S}; \mathcal{I}, \mathcal{C} \rightarrow_s \mathcal{P}'; \mathcal{S}'; \mathcal{I}', \mathcal{C}'$ where $\mathcal{P}; \mathcal{S}; \mathcal{I}, \mathcal{C}$ is a symbolic configuration then $\mathcal{P}'; \mathcal{S}'; \mathcal{I}', \mathcal{C}'$ is still a symbolic configuration, that is $\mathcal{C}'$ is a constraint system.

Example 7. For example, executing the same transitions as in Example 5 symbolically, we reach the following configuration :

$$K_s = \lfloor \mathsf{out}(\mathsf{error}).0 \rfloor_S;\ \emptyset;\ \mathcal{I}_0 \cup \{u_1, v_2\};\ \mathcal{C}$$

where $\mathcal{C}$ is the constraint system defined in Example 6.

3.3 Soundness and completeness

We show that our symbolic transition system reflects exactly the concrete transition system, i.e. each concrete execution of a process is captured by one of the

$$\begin{array}{rcl}
\{\lfloor \mathsf{in}\ u_i[\Phi_i].P'_i\rfloor_{n_i} \mid i \in I\} & \rightarrow_s & \{\lfloor P'_j\rfloor_{n_j} \mid j \in J\} \cup \lfloor P\rfloor_n \\
\lfloor \mathsf{out}(t).P\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & & \cup \{\lfloor \mathsf{in}\ u_k[\Phi_k].P'_k\rfloor_{n_k} \mid k \in K \cup L\} \cup \mathcal{P}; \mathcal{S}; \mathcal{I}'; \\
& & \mathcal{C} \cup \{t = u_j;\ \Phi_j \mid j \in J\} \\
& & \quad \cup \{\forall (\mathit{Var}(u_k) \smallsetminus \mathit{rvar}(\mathcal{C})).\, t \neq u_k \mid k \in K\} \\
& & \quad \cup \{t = u_l\alpha_l;\ \neg\Phi_l\alpha_l \mid l \in L\}
\end{array}$$

where $\lfloor P'\rfloor_{n'} \in \mathcal{P}$ implies that $(n, n') \notin E$ or P' is not of the form $\mathsf{in}\ U'[\Phi'].Q'$, $I = J \uplus K \uplus L$, and α_l is a renaming of $\mathit{Var}(u_k) \smallsetminus \mathit{rvar}(\mathcal{C})$ by fresh variables if $(n, n_I) \in E$ then $\mathcal{I}' = \mathcal{I} \cup \{t\}$ else $\mathcal{I}' = \mathcal{I}$.

$$\begin{array}{rcl}
\lfloor \mathsf{in}\ u[\Phi].P\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & \rightarrow_s & \lfloor P\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \cup \{\mathcal{I} \Vdash u;\ \Phi\} \text{ if } (n_I, n) \in E \\
\\
\lfloor \mathsf{store}(t).P\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & \rightarrow_s & \lfloor P\rfloor_n \cup \mathcal{P}; \lfloor t\rfloor_n \cup \mathcal{S}; \mathcal{I}; \mathcal{C} \\
\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & \rightarrow_s & \lfloor P\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \cup \{t = u\} \quad \text{where } \lfloor t\rfloor_n \in \mathcal{S} \\
\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & \rightarrow_s & \lfloor Q\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \cup \{\forall X.\, t \neq u \mid \lfloor t\rfloor_n \in \mathcal{S}\} \\
& & \hfill \text{where } X = \mathit{Var}(u) \smallsetminus \mathit{rvar}(\mathcal{C}) \\
\\
\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & \rightarrow_s & \lfloor P\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \cup \{\Phi\} \\
\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & \rightarrow_s & \lfloor Q\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \cup \{\neg\Phi\} \\
\\
\lfloor P_1 \mid P_2\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & \rightarrow_s & \lfloor P_1\rfloor_n \cup \lfloor P_2\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \\
\lfloor !P\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & \rightarrow_s & \lfloor P\alpha\rfloor_n \cup \lfloor !P\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \\
& & \hfill \text{where } \alpha \text{ is a renaming of the bound variables of } P \\
\lfloor \mathsf{new}\ m.P\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} & \rightarrow_s & \lfloor P\{^{m'}/_{m}\}\rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C} \\
& & \hfill \text{where } m' \text{ is a fresh name}
\end{array}$$

Fig. 3. Symbolic transition system.

symbolic execution. To do that, we explain how a concrete and a symbolic execution match each other, and by a case study, we show that the link is preserved when taking a transition (see detailed proofs in Appendix **??**).

Proposition 1. *Let $K = \mathcal{P}[\_]; \mathcal{S}; \mathcal{I}$ be a concrete configuration with a hole, and Φ a formula. Then there is an attack on K and Φ if and only if $\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})\ \mathsf{else}\ 0]; \mathcal{S}; \mathcal{I}, \emptyset \rightarrow_s^* \lfloor \mathsf{out}(u).P_s\rfloor_n \cup \mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C}$ and the constraint system $\mathcal{C} \cup \{\mathsf{error} = u\}$ has a solution.*

We deduce that checking for an attack can be reduced to checking the existence of a solution for reachable constraint systems.

Example 8. Consider our former example of an attack on SRP , with initial configuration K_0. We can reach the configuration K_s, and the constraint system $\mathcal{C} \cup \{\mathsf{error} = \mathsf{error}\}$ has a solution σ (cf. Example 6), so there is an attack on K_0.

Note that our result holds for any signature, for any choice of predicates, and for processes possibly with replication. Of course, it then remains to decide the existence of a constraint system that has a solution.

4 Decidability result

In this section, we restrict ourselves to processes:

- without replication,
- where nodes can only perform neighborhood tests on themselves, i.e. whenever $\mathsf{check}(a, b)$ or $\mathsf{checkl}(a, l)$ appears in a process P, then P is encapsulated in node a,
- defined over the signature considered in Example 1 and formula of the logic $\mathcal{L}_{\mathsf{route}}$ restricted to formulas where the predicate route does not appear (Intuitively, nodes do not have the possibility to check the validity of an entire route).

We also restrict ourselves to properties $\Phi \in \mathcal{L}_{\mathsf{route}}$ such that, if Φ' is the disjunctive normal form of Φ, whenever the predicate route appears in Φ', it is always negated.
We also assume that the intruder is initially given an infinite number of names that he can use as its will, in particular for possibly passing some disequality constraints.
A concrete configuration $(K = \mathcal{P}[]; \mathcal{S}; \mathcal{I})$ is said *initial* if K is ground and if $\mathcal{N}_{\mathsf{loc}} \subseteq \mathcal{I}$.

Our second main contribution is to show that accessibility properties are decidable for a class of processes that model secure routing protocols, for a bounded number of sessions.

Theorem 1. *Let G be a finite graph, K be an initial concrete configuration and Φ a property. Deciding whether there is an attack on K and Φ for the topology G is NP-complete.*

Theorem 1 ensures in particular that we can decide whether a routing protocol like SRP can guaranty that any route accepted by the source is indeed a route (a path) in the network. The proof of Theorem 1 involves several steps.

1. First, since processes contain no replication, Proposition 1 ensures that it is sufficient to decide the existence of a solution for our class of constraint systems.
2. It has been shown in [5] that the existence of a solution of a constraint system (with only deduction constraints) can be reduced to the existence of a solution of a *solved* constraint system, where right-hand-sides of the constraints are variables only. We have extended this result to our generalized notion of constraint systems, *i.e.* with disequality tests and formula of $\mathcal{L}_1$, and for an intruder knowledge with an infinite number of names.
3. We then show how to decide the existence of a solution for a solved constraint system. It is not straightforward like in [5] since we are left with (non solved) disequality constraints and formula. The key step consists in showing that we can bound (polynomially) the size of the lists in a minimal attack.

5 Conclusion

We have shown that, for general processes that can reflect a given network topology, existence of attacks can be reduced to existence of constraint systems with a solution. As an illustration, for a large class of processes without replication that captures routing protocol like SRP applied on DSR, we have proved that the existence of an attack is NP-complete. In particular, we generalize existing works on solving constraint systems to properties like the validity of a route and to protocols with broadcasting.

Our results hold for an arbitrary (but fixed and finite) graph. We believe that we could adapt our techniques for deciding the existence of a network topology that would lead to an attack but this is left as future work. We also plan to consider how to model changes in the network topology in order to analyze the security of route updates.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proc. 28th Symposium on Principles of Programming Languages (POPL'01)*, pages 104–115. ACM Press, 2001.
2. M. Abadi and A. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Proc. 4th Conference on Computer and Communications Security (CCS'97)*, pages 36–47. ACM Press, 1997.
3. R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.
4. L. Buttyán and I. Vajda. Towards Provable Security for Ad Hoc Routing Protocols. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 94–105, New York, NY, USA, 2004. ACM.
5. V. Cortier, H. Comon-Lundh, and E. Zalinescu. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Transactions on Computational Logic (TOCL)*, 2009. To appear.
6. Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11:21–38, January 2005.
7. D. Johnson, D. Maltz, and J. Broch. DSR: The Dynamic Source Routing Protocol for multi-hop wireless ad hoc networks. In *Ad Hoc Networking*, pages 139–172, 2001.
8. D. B. Johnson, D. A. Maltz, and J. Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139–172. Addison-Wesley, 2001.
9. J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. of the 8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.
10. S. Nanz and C. Hankin. A Framework for Security Analysis of Mobile Wireless Networks. *Theor. Comput. Sci.*, 367(1):203–227, 2006.
11. P. Papadimitratos and Z. Haas. Secure routing for mobile ad hoc networks. In *Proc. SCS Communication Networks and Distributed Systems Modelling Simulation Conference (CNDS)*, 2002.

12. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.
13. C. E. Perkins and E. M. Belding-Royer. Ad-hoc on-demand distance vector routing. In *Proc. 2nd Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, pages 90–100, New Orleans, LA, USA, 1999.
14. M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*, pages 174–190. IEEE Comp. Soc. Press, 2001.
15. P. Schaller, B. Schmidt, D. Basin, and S. Capkun. Modeling and verifying physical properties of security protocols for wireless networks. In *Proc. 22nd Computer Security Foundations Symposium (CSF'09)*. IEEE Comp. Soc. Press, 2009.
16. F. J. Thayer, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.
17. S. Yang and J. S. Baras. Modeling vulnerabilities of ad hoc routing protocols. In *Proc. 1st ACM Workshop on Security of ad hoc and Sensor Networks (SASN'03)*, pages 12–20, 2003.
18. M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proc. 1st ACM workshop on Wireless SEcurity (WiSE'02)*, pages 1–10, New York, NY, USA, 2002. ACM.