Modeling and Verifying Ad Hoc Routing Protocols

Mathilde Arnaud*†, Véronique Cortier* and Stéphanie Delaune†
* *LORIA, CNRS & INRIA Nancy Grand Est, France*
Email: cortier@loria.fr
† *LSV, ENS Cachan & CNRS & INRIA Saclay Île-de-France, France*
Email: arnaud@lsv.ens-cachan.fr, delaune@lsv.ens-cachan.fr

***Abstract*—Mobile ad hoc networks consist of mobile wireless devices which autonomously organize their infrastructure. In such networks, 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 malicious nodes from compromising the discovered route.**

Our contribution is twofold. We first propose 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. Our second main contribution is a decision procedure for analyzing routing protocols for any network topology. By using constraint solving techniques, we show that it is possible to automatically discover (in NPTIME) whether there exists a network topology that would allow malicious nodes to mount an attack against the protocol, for a bounded number of sessions. We also provide a decision procedure for detecting attacks in case the network topology is given a priori. We demonstrate the usage and usefulness of our approach by analyzing the protocol SRP applied to DSR.

I. Introduction

Mobile ad hoc networks consist of mobile wireless devices which autonomously organize their communication infrastructure: each node provides the function of a router and relays packets on paths to other nodes. Finding these paths 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 used in a friendly environment (e.g. [22], [15]). Recent research has recognized that this assumption is unrealistic and that attacks can be mounted [13], [19], [9]. Since an adversary can easily paralyse the operation of a whole network by attacking the routing protocol, it is crucial to prevent malicious nodes from compromising the discovered routes. Since then, secured versions of routing protocols have been developed to ensure that mobile ad hoc networks can work even in an adversarial setting [27], [13], [20]. Those routing protocols use cryptographic mechanisms such as encryption, signature, MAC, in order to prevent a malicious node to insert and delete nodes inside a path.

Formal modeling and analysis techniques are well-adapted for checking correctness of security protocols. Formal methods have for example been successfully used for authentication or key establishment security protocols and a multitude of effective frameworks have been proposed (e.g. the Paulson inductive model [21], the strand spaces model [25], the applied-pi calculus [1] or constraints systems [23] to cite only a few). While secrecy and authentication properties have been shown undecidable in the general case [12], many decision procedures have been proposed. For example, secrecy and authentication become NP-complete for a bounded number of sessions [23] and Bruno Blanchet has developed a procedure for security protocols encoded as Horn clauses [7]. This yielded various efficient tools for detecting flaws and proving security (e.g. ProVerif [8] or Avispa [5]).

While key-exchange protocols are well-studied in traditional networks, there are very few attempts to develop formal techniques allowing an automated analysis of secured routing protocols. Up to our knowledge, tools that would allow the security analysis of routing protocols are also missing. Those protocols indeed involve several subtleties that cannot be reflected in existing work. For example, the underlying network topology is crucial to define who can receive the messages sent by a node and the intruder is localized to some specific nodes (possibly several nodes). Moreover, the security properties include e.g. the validity of a route, which differ from the usual secrecy and authenticity properties.

Our contributions. The first main contribution of this paper is the proposition of a calculus, inspired from CBS# [19], which allows ad hoc networks and their security properties to be formally described and analyzed. We do not take mobility into account in the sense that the topology of the network does not change during our analysis. We propose in particular a logic to express the tests performed by the nodes at each step. It allows for example checking whether a route is "locally" valid, given the information known by the node. We model cryptography as a black box (the perfect

This work has been partially supported by the ANR-07-SESU-002 AVOTÉ.

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 model reflects the fact that a malicious node can interfere directly only with his neighbors.

We would like to emphasize that our model is not strictly dedicated to routing protocols but can be used to model many other classes of protocols. In particular, by considering a special network topology where the attacker is at the center of the network, we retrieve the classical modeling where the attacker controls all the communications. We thus can model as usual all the key exchange and authenticity protocols presented e.g. in the Clark & Jacob library [10]. Since we also provide each node with a memory, our model can also capture protocols where a state global to all sessions is assumed for each agent. It is the case of protocols where an agent should check that a key has not been already accepted in a previous session, in order to protect the protocol against replay attacks.

Our second main contribution is to provide two NP decision procedures for analyzing routing protocols for a bounded number of sessions. For a fixed set of roles and sessions, our first decision procedure allows to discover whether there exists a network topology and a malicious behavior of some nodes that yield an attack. Using similar ingredients, we can also decide whether there exists an attack, for a network topology chosen by the user. Our two procedures hold for any property that can be expressed in our logic, which includes classical properties such as secrecy as well as properties more specific to routing protocols such as route validity.

The main ingredients of our decision procedures are as follows. We first propose a symbolic semantics for our execution model and show how the analysis of routing protocols can be reduced to (generalized) constraint systems solving. We then adapt and generalize existing techniques [11] for solving our more general constraint systems. We show in particular that minimal attacks (whether the underlying network topology is fixed or not) require at most a polynomially bounded number of nodes. We demonstrate the usage and usefulness of our model and techniques by analyzing SRP (Secured Routing Protocol) [20] applied on the protocol DSR (Dynamic Source Routing Protocol) [16]. This allows us to retrieve an attack presented first in [9].

Related work. Recently, some frameworks have been proposed to model wireless communication and/or routing protocols in a more accurate way. For example, Yang and Baras [26] provide a first symbolic model for routing protocols based on strand spaces, modeling the network topology but not the cryptographic primitives that can be used for securing communications. They also implement a semi-decision procedure to search for attacks. Buttyán and Vadja [9] provide a cryptographic model for routing protocols, in a cryptographic setting. They provide a security proof (by hand) for a fixed protocol. Ács [3] uses a similar cryptographic model and provides proofs of security (or insecurity) for several protocols, but his method is not automatic. Nanz and Hankin [19] propose a process calculus to model the network topology and broadcast communications. They analyze scenarios with special topologies and attacker configuration by computing an over-approximation of reachable states. Their analysis is safe in the sense that it does not find flaws if the protocol is secure. The model proposed in this paper is inspired from their work, adding in particular a logic for specifying the tests performed at each step by the nodes on the current route and to specify the security properties. Schaller *et al* [24] propose a symbolic model that allows an accurate representation of the physical properties of the network, in particular the speed of the communication. This allows in particular to study distance bounding protocols. Several security proofs are provided for some fixed protocols, formalized in Isabelle/HOL. However, no generic procedure is proposed for proving security.

To our knowledge, our paper presents the first decidability and complexity result for routing protocols, for arbitrary intruders and network topologies. Moreover, since we reuse existing techniques on solving constraint systems, our decision procedure seems amenable to implementation, re-using existing tools (such as Avispa [5]).

Outline. Section II presents our formal model for routing protocol. It is illustrated with the modeling of the SRP protocol. We then give an alternative symbolic semantics in Section III, more amenable to automation, and we show its correctness and completeness w.r.t. the concrete semantics. Finally, we state our main decidability result in Section IV and present a sketch of its proof. Some concluding remarks can be found in Section V. Detailed proofs of our result can be found in [6].

II. Model for protocols

A. 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 an 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:

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

where $ar(f) = s_1 \times \ldots \times s_k \to s$ and t_i is a term of some sort s_i.

We assume a special sort terms that subsumes all the other sorts and such that any term is of sort terms. We write $var(t)$ the set of variables occurring in a term t and $St(t)$ the syntactic subterms of 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 $\mathcal{F}_1 = \{\mathsf{hmac}, \langle\rangle, ::, \perp, \{\_\}\_\}$, with the following arities:

- $\mathsf{hmac} : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$,
- $\langle\_,\_\rangle : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$,
- $:: \, : \mathsf{loc} \times \mathsf{lists} \to \mathsf{lists}$,
- $\perp : \to \mathsf{lists}$,
- $\{\_\}_\_ : \mathsf{terms} \times \mathsf{terms} \to \mathsf{terms}$.

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 $\mathsf{hmac}(m, k)$ represents the keyed hash message authentication code computed over message m with key k while $\langle\rangle$ is a pairing operator. The term $\{m\}_k$ represents the message m encrypted by the key k. 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 $dom(\sigma) = \{x_1, \ldots, x_n\}$. We only consider *well-sorted* substitutions, that is substitutions for which x_i and t_i have the same sort. The substitution 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 $T \vdash t$ represents the fact that the term t is computable from the set of terms T.

$$\frac{T \vdash a \quad T \vdash l}{T \vdash a :: l} \qquad \frac{T \vdash a :: l}{T \vdash a} \qquad \frac{T \vdash a :: l}{T \vdash l}$$

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u, v\rangle} \qquad \frac{T \vdash \langle u, v\rangle}{T \vdash u} \qquad \frac{T \vdash \langle u, v\rangle}{T \vdash v}$$

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \{u\}_v} \qquad \frac{T \vdash \{u\}_v \quad T \vdash v}{T \vdash u}$$

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \mathsf{hmac}(u, v)} \qquad \frac{u \in T}{T \vdash u}$$

Figure 1. Example of deduction system.

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, to build lists, to encrypt and decrypt when he knows the key. Moreover, he is able to retrieve components of a pair or a list.

B. Process calculus

Several calculi already exist to model security protocols (e.g. [2], [1]). However, modeling ad-hoc routing protocoles require several features. For instance, a node of the network may store 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# [19], which allows mobile wireless networks and their security properties to be formally described and analyzed. However, we extend this calculus to allow nodes to perform some sanity checks on the routes they receive, such as 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
0	null process
$\mathsf{out}(u).P$	emission
$\mathsf{in}\ u[\Phi].P$	reception, $\Phi \in \mathcal{L}$
$\mathsf{store}(u).P$	storage
$\mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q$	reading
$\mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q$	conditional, $\Phi \in \mathcal{L}$
$P \mid Q$	parallel composition
$!P$	replication
$\mathsf{new}\ m.P$	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 σ is such that $m = u\sigma$. If Φ is the true formula, we simply write 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 σ is such that $m = u\sigma$. If no element of the form u is found, then it behaves like Q. Sometimes, for the sake of clarity, we will omit the null process. We also omit the else part when $Q = 0$. We write $fv(P)$ for the set of free variables of P. A process P is *ground* when $fv(P) = \emptyset$.

The store and read primitives are particularly important when modeling routing protocols, in order to avoid multiple answers to a single request or to allow nodes to store and retrieve already known routes. These primitives can also be used to represent other classes of protocols, where a global state is assumed for each agent, in order to store some information (black list, already used keys. *etc.*) throughout the sessions.

Secured routing protocols typically perform some checks on the message they received before accepting it. 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 an undirected graph $G = (V, E)$ with $V \subseteq \mathcal{N}_{\mathsf{loc}}$, the semantics $[\![\Phi]\!]_G$ of a formula $\Phi \in \mathcal{L}_{\mathsf{route}}$ is recursively defined as follows:

- $[\![\mathsf{check}(a, b)]\!]_G = 1$ iff $(a, b) \in E$ with a, b of sort loc,
- $[\![\mathsf{checkl}(c, l)]\!]_G = 1$ 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 $(b, c) \in E$.
- $[\![\mathsf{route}(l)]\!]_G = 1$ iff l is of sort lists, $l = a_1 :: \ldots :: a_n$, for every $1 \le i < n$, $(a_i, a_{i+1}) \in E$, and for every $1 \le i, j \le n$, $i \ne j$ implies that $a_i \ne a_j$.
- $[\![\mathsf{loop}(l)]\!]_G$ iff l is of sort lists and there exists an element appearing at least twice in l,
- $[\![\Phi_1 \wedge \Phi_2]\!]_G = [\![\Phi_1]\!]_G \wedge [\![\Phi_2]\!]_G$,
- $[\![\Phi_1 \vee \Phi_2]\!]_G = [\![\Phi_1]\!]_G \vee [\![\Phi_2]\!]_G$, and
- $[\![\neg\Phi]\!]_G = \neg[\![\Phi]\!]_G$.

C. Example: modeling the SRP protocol

We consider the secured routing protocol SRP introduced in [20], assuming that each node already knows his neighbors (running e.g. some neighbor discovery protocol). SRP is not a routing protocol by itself, it describes a generic way for securing source-routing protocols. We model here its application to the DSR protocol [16]. 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 by checking 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 source node S initiating the search of a route towards a destination node D is $P_{\mathsf{init}}(S, D) = \mathsf{new}\ id.\mathsf{out}(u_1).\mathsf{in}\ u_2[\Phi_S].0$ where:

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

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:

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

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

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, the destination 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:

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

where:

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

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

$$P_{\mathsf{rep}}(V) = \mathsf{in}\ w'[\Phi'_V].\mathsf{out}(w').0$$

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

D. 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 his neighbors. More specifically, we assume that the topology of the network is represented by an undirected graph $G = (V, E)$ with $V \subseteq \mathcal{N}_{\mathsf{loc}}$, where an edge in the graph models the fact that two nodes are neighbors. We also assume that we have a set of nodes $\mathcal{M} \subseteq V$ that are controlled by the attacker. These nodes are then called *malicious*. Our model is not restricted to a single malicious node. Our results allow us to consider the case of several compromised nodes that collaborate by sharing their knowledge. However, it is well-known that the presence of several colluding malicious nodes often yields straightforward attacks [14], [17].

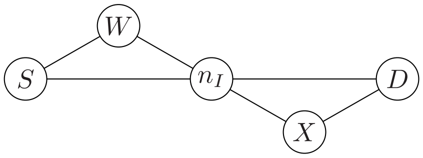
A (ground) *concrete configuration* of the network is a triplet $(\mathcal{P}; \mathcal{S}; \mathcal{I})$ where:

- $\mathcal{P}$ 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 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 ground terms representing the messages seen by the intruder.

Example 2: Continuing our modeling of SRP , a possible initial configuration for the SRP protocol is

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

where both the source node S and the destination node D wish to communicate. A more realistic configuration would include intermediary nodes but as shown in the next examples, this initial configuration is already sufficient to present an attack. We assume that each node has an empty storage list and that the initial knowledge of the intruder is given by an infinite set of terms $\mathcal{I}_0$. A possible network configuration is modeled by the graph G_0 below. We assume that there is a single malicious node, i.e. $\mathcal{M}_0 = \{n_I\}$. The nodes W and X are two extra (honest) nodes. We do not need to assume that the intermediate nodes W and X execute the routing protocol.



Each honest node broadcasts its messages to all its neighbors. To capture more malicious behaviors, we allow the nodes 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 parameterized by the underlying graph G and the set of malicious nodes $\mathcal{M}$.

The relation $\to^*_{G,\mathcal{M}}$ is the reflexive and transitive closure of $\to_{G,\mathcal{M}}$. We may write $\to$, $\to_G$, $\to_{\mathcal{M}}$ instead of $\to_{G,\mathcal{M}}$ when the underlying network topology G or the underlying set $\mathcal{M}$ is clear from the context.

Note that in case we assume that there is a single malicious node and each honest node is connected to it (and only to it), we retrieve the model where the attacker is assumed to control all the communications.

Example 3: Continuing the example developed in Section II-C, the following sequence of transitions is enabled from the initial configuration K_0:

$$K_0 \to^*_{G_0,\mathcal{M}_0} (\lfloor \mathsf{in}\ u_2[\Phi_S].0\rfloor_S \cup \lfloor P_{\mathsf{dest}}(D,S)\rfloor_D; \emptyset; \mathcal{I}_0 \cup \{u_1\})$$

where u_1, u_2, Φ_S have already been defined in the previous section:

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

During this transition, S broadcasts to its neighbors a request for finding a route to D. The intruder n_I is a neighbor of S in G_0, so he learns the request message. Assuming that the intruder knows the names of his neighbors, i.e. $W, X \in \mathcal{I}_0$, he can then build the following fake message request:

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

and broadcast it. Since $(X, D) \in E$, the node D accepts this message and the resulting configuration is

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

Comm
$$(\{\lfloor \mathsf{in}\ u_j[\Phi_j].P_j\rfloor_{n_j} \mid \mathsf{mgu}(t,u_j) \neq \bot, [\![\Phi_j\sigma_j]\!]_G = 1, (n,n_j) \in E\} \cup \lfloor \mathsf{out}(t).P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\{\lfloor P_j\sigma_j\rfloor_{n_j}\} \cup \lfloor P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}')$$
where $\sigma_j = \mathsf{mgu}(t,u_j)$, $\mathcal{I}' = \mathcal{I} \cup \{t\}$ if $(n,n_I) \in E$ for some $n_I \in \mathcal{M}$ and $\mathcal{I}' = \mathcal{I}$ otherwise.

In
$$(\lfloor \mathsf{in}\ u[\Phi].P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P\sigma\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I})$$
if $(n_I,n) \in E$ for some $n_I \in \mathcal{M}$, $\mathcal{I} \vdash t$, $\sigma = \mathsf{mgu}(t,u)$ and $[\![\Phi\sigma]\!]_G = 1$

Store
$$(\lfloor \mathsf{store}(t).P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P\rfloor_n \cup \mathcal{P};\lfloor t\rfloor_n \cup \mathcal{S};\mathcal{I})$$

Read-Then
$$(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\lfloor t\rfloor_n \cup \mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P\sigma\rfloor_n \cup \mathcal{P};\lfloor t\rfloor_n \cup \mathcal{S};\mathcal{I})$$
where $\sigma = \mathsf{mgu}(t,u)$

Read-Else
$$(\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I})$$
if for all t such that $\lfloor t\rfloor_n \in \mathcal{S}$, $\mathsf{mgu}(t,u) = \bot$

If-Else
$$(\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \quad \text{if } [\![\Phi]\!]_G = 1$$

If-Then
$$(\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \quad \text{if } [\![\Phi]\!]_G = 0$$

Par
$$(\lfloor P_1 \mid P_2\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P_1\rfloor_n \cup \lfloor P_2\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I})$$

Repl
$$(\lfloor !P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P\alpha\rfloor_n \cup \lfloor !P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I})$$
where α is a renaming of the bound variables of P

New
$$(\lfloor \mathsf{new}\ m.P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I}) \to_{G,\mathcal{M}} (\lfloor P\{^{m'}/_{m}\}\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I})$$
where m' is a fresh name

Figure 2. Concrete transition system.

$$\text{where } \left\{\begin{array}{l} v_2 = \langle \mathsf{rep}, D, S, x_{id}, x_a :: x_l, \\ \qquad\qquad \mathsf{hmac}(\langle D, S, x_{id}, x_a :: x_l\rangle, K_{SD})\rangle \\ \sigma = \{^{id}/_{x_{id}}, {}^{X}/_{x_a}, {}^{[W;S]}/_{x_l}\} \end{array}\right.$$

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

Definition 1: Let G be a graph and $\mathcal{M}$ be a set of nodes. There is an *$\mathcal{M}$-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:

$$(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})];\mathcal{S};\mathcal{I}) \to^{*}_{G,\mathcal{M}} (\lfloor \mathsf{out}(\mathsf{error})\rfloor_n \cup \mathcal{P}', \mathcal{S}', \mathcal{I}')$$

where error is a special symbol not occurring in the configuration $(\mathcal{P}[\_];\mathcal{S};\mathcal{I})$.

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 $(\mathcal{P};\mathcal{S};\mathcal{I})$ for a graph $G = (V,E)$ can be defined by the (non) existence of an $\{n_I\}$-attack on the configuration $(\mathcal{P} \cup \lfloor W\rfloor_n;\mathcal{S};\mathcal{I})$ and the formula true for the graph $G' = (V \cup \{n\}, E \cup \{(n,n_I)\})$.

Example 4: For the SRP protocol, 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 checking whether the formula $\neg\mathsf{route}(x_L)$ holds. Let $P'_{\mathsf{init}}(S,D)$ be the resulting process.

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

where $P = \mathsf{if}\ \neg\mathsf{route}(x_L)\ \mathsf{then}\ \mathsf{out}(\mathsf{error})$. Then, we recover the attack mentioned in [9] with the topology G_0 given in Example 2, 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:

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

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

III. 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. [4]).

A. Constraint systems

As in [18], [11], [23], 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 formulas 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: A *constraint system* $\mathcal{C}$ is a finite conjunction of *constraints* of the form $v = u$ (unification constraint), $\mathcal{I} \Vdash u$ (deduction constraint), $\forall X.\, v \neq u$ (disequality constraint), and Φ (formula of $\mathcal{L}_{\mathsf{route}}$), where v, u are terms, $\mathcal{I}$ is a non empty set of terms, and X is a set of variables. Moreover, we assume that the constraints in $\mathcal{C}$ can be ordered $C_1, \ldots, C_n$ in such a way that the following properties hold:

- (monotonicity) If $C_i = (\mathcal{I}_i \Vdash u_i)$ and $C_j = (\mathcal{I}_j \Vdash u_j)$ with $i < j$ then $\mathcal{I}_i \subseteq \mathcal{I}_j$;
- (origination) If $C_i = (\mathcal{I}_i \Vdash u_i)$ (resp. $C_i = (v_i = u_i)$) then for all $x \in \mathit{var}(\mathcal{I}_i)$ (resp. $x \in \mathit{var}(v_i)$), there exists $j < i$ such that
  - either $C_j = (\mathcal{I}_j \Vdash u_j)$ with $x \in \mathit{var}(u_j)$;
  - or $C_j = (v_j = u_j)$ with $x \in \mathit{var}(u_j)$.

Lastly, we assume that $\mathit{var}(\mathcal{C}) \subseteq \mathit{rvar}(\mathcal{C})$ where $\mathit{rvar}(\mathcal{C})$ represents the set of variables introduced in $\mathcal{C}$ in the right-hand-side of a unification constraint or a deduction constraint.

The origination property ensures that variables are always introduced by a unification constraint or a deduction constraint, which is always the case when modeling protocols. Actually, the set $\mathit{rvar}(\mathcal{C})$ represents all the free variables. This means that all the variables are introduced in the right-hand-side of a unification constraint or a deduction constraint.

Note that our disequality constraints are rather general since they do not simply allow to check that two terms are different ($u \neq v$), but they also allow to ensure that no unification was possible at a certain point of the execution, which is a necessary check due to our broadcast primitive.

A *solution* to a constraint system $\mathcal{C}$ for a graph G is a ground substitution θ such that $\mathit{dom}(\theta) = \mathit{rvar}(\mathcal{C})$ and:

- $t\theta = u\theta$ for all $t = u \in \mathcal{C}$;
- $T\theta \vdash u\theta$ for all $T \Vdash u \in \mathcal{C}$;
- for all $(\forall X.\, t \neq u) \in \mathcal{C}$, then the terms $t\theta$ and $u\theta$ are not unifiable (even renaming the variables of X with fresh variables); and
- $[\![\Phi\theta]\!]_G = 1$ for every formula $\Phi \in \mathcal{C}$.

Example 5: Consider the following set of constraints:

$$\mathcal{C} = \left\{ \begin{array}{l} \mathcal{I}_0 \cup \{u_1\} \Vdash v_1 \ \wedge\ \Phi_D \ \wedge \\ \mathcal{I}_0 \cup \{u_1, v_2\} \Vdash u_2 \ \wedge\ \Phi_S \ \wedge\ \neg\mathsf{route}(x_L) \end{array} \right\}$$

with:

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

We have that $\mathcal{C}$ is a constraint system, and the substitution

$$\sigma = \{{}^{id}/_{x_{id}}, {}^{X}/_{x_a}, {}^{[W;S]}/_{x_l}, {}^{[X;W;S]}/_{x_L}\}$$

is a solution of the constraint system $\mathcal{C}$ for graph G_0.

B. 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 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 (not necessarily ground) representing the messages seen by the intruder.
- $\mathcal{C}$ is a constraint system such that $T \subseteq \mathcal{I}$ for every constraint $T \Vdash u \in \mathcal{C}$.

Such a configuration is *ground* when:

$$\mathit{fv}(\mathcal{P}) \cup \mathit{var}(\mathcal{S}) \cup \mathit{var}(\mathcal{I}) \subseteq \mathit{rvar}(\mathcal{C}).$$

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.

Whenever $(\mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}) \rightarrow^{s}_{G,\mathcal{M}} (\mathcal{P}'; \mathcal{S}'; \mathcal{I}'; \mathcal{C}')$ where $(\mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C})$ is a (ground) symbolic configuration then $(\mathcal{P}'; \mathcal{S}'; \mathcal{I}'; \mathcal{C}')$ is still a (ground) symbolic configuration.

Example 6: Executing the same transitions as in Example 4 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 $\mathcal{C}, u_1, v_2$ are defined as in Example 5.

C. 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 symbolic executions. More precisely, a concrete configuration

$$\textsc{Comm}_s \quad \begin{array}{r}(\lfloor \mathsf{out}(t).P\rfloor_n \cup \{\lfloor \mathsf{in}\ u_i[\Phi_i].P'_i\rfloor_{n_i} \mid i \in I\} \\ \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C})\end{array} \rightarrow^s_{G,\mathcal{M}} \begin{array}{r}(\lfloor P\rfloor_n \cup \{\lfloor \mathsf{in}\ u_k[\Phi_k].P'_k\rfloor_{n_k} \mid k \in K \cup L\} \\ \cup \{\lfloor P'_j\rfloor_{n_j} \mid j \in J\} \cup \mathcal{P};\mathcal{S};\mathcal{I}';\mathcal{C}')\end{array}$$

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 $(n_i, n) \in E$ for every $i \in I$,
- $\mathcal{C}' = \mathcal{C} \wedge \{t = u_j \wedge \Phi_j \mid j \in J\} \wedge \{\forall (var(u_k) \smallsetminus rvar(\mathcal{C})) \,.\, t \neq u_k \mid k \in K\} \wedge \{t = u_l\alpha_l \wedge \neg\Phi_l\alpha_l \mid l \in L\}$ where α_l is a renaming of $var(u_l) \smallsetminus rvar(\mathcal{C})$ by fresh variables,
- $\mathcal{I}' = \mathcal{I} \cup \{t\}$ when $(n, n_I) \in E$ for some $n_I \in \mathcal{M}$, and $\mathcal{I}' = \mathcal{I}$ otherwise.

$$\textsc{In}_s \quad (\lfloor \mathsf{in}\ u[\Phi].P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C} \wedge \mathcal{I} \Vdash u \wedge \Phi\}) \quad \text{if } (n_I, n) \in E \text{ for some } n_I \in \mathcal{M}$$

$$\textsc{Store}_s \quad (\lfloor \mathsf{store}(t).P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P\rfloor_n \cup \mathcal{P};\lfloor t\rfloor_n \cup \mathcal{S};\mathcal{I};\mathcal{C})$$

$$\textsc{Read-Then}_s \quad (\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C} \wedge t = u) \quad \text{where } \lfloor t\rfloor_n \in \mathcal{S}$$

$$\textsc{Read-Else}_s \quad (\lfloor \mathsf{read}\ u\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C} \wedge \{\forall X \,.\, t \neq u \mid \lfloor t\rfloor_n \in \mathcal{S}\}) \quad \text{where } X = var(u) \smallsetminus rvar(\mathcal{C})$$

$$\textsc{If-Then}_s \quad (\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C} \wedge \Phi)$$

$$\textsc{If-Else}_s \quad (\lfloor \mathsf{if}\ \Phi\ \mathsf{then}\ P\ \mathsf{else}\ Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor Q\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C} \wedge \neg\Phi)$$

$$\textsc{Par}_s \quad (\lfloor P_1 \mid P_2\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P_1\rfloor_n \cup \lfloor P_2\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C})$$

$$\textsc{Repl}_s \quad \lfloor !P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C} \rightarrow^s_{G,\mathcal{M}} \lfloor P\alpha\rfloor_n \cup \lfloor !P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}$$

where α is a renaming of the bound variables of P that are not in $rvar(\mathcal{C})$.

$$\textsc{New}_s \quad (\lfloor \mathsf{new}\ m.P\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \rightarrow^s_{G,\mathcal{M}} (\lfloor P\{^{m'}/_m\}\rfloor_n \cup \mathcal{P};\mathcal{S};\mathcal{I};\mathcal{C}) \quad \text{where } m' \text{ is a fresh name}$$

Figure 3. Symbolic transition system.

is represented by a symbolic configuration if it is one of its instances, called *concretization*.

Definition 3 (θ-concretization): A concretization of a symbolic configuration $K_s = (\mathcal{P}_s;\mathcal{S}_s;\mathcal{I}_s;\mathcal{C})$ is a concrete configuration $K_c = (\mathcal{P};\mathcal{S};\mathcal{I})$ such that there exists θ a solution of $\mathcal{C}$ and, furthermore, $\mathcal{P}_s\theta = \mathcal{P}$, $\mathcal{S}_s\theta = \mathcal{S}$, $\mathcal{I}_s\theta = \mathcal{I}$. We say that K_c is a θ-concretization of K_s.

Note that the θ-concretization of a ground symbolic configuration is a ground concrete configuration.

Each concrete transition can be matched by a symbolic one.

Lemma 1 (Completeness): Let G be a graph and $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$. Let $K_s = (\mathcal{P}_s;\mathcal{S}_s;\mathcal{I}_s;\mathcal{C})$ be a ground symbolic configuration and θ be a solution of $\mathcal{C}$. Let K_c be the θ-concretization of K_s. Let K'_c be a concrete configuration such that $K_c \rightarrow_{G,\mathcal{M}} K'_c$. Then there exists a ground symbolic configuration K'_s and a substitution θ' such that:

- K'_c is the θ'-concretization of K'_s, and
- $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$.

The proof is performed by studying each rule of the concrete transition system, showing that the corresponding symbolic rule covers all possible cases. In particular, disequality constraints allow to faithfully model cases where nodes reject a message because the message does not match the expected pattern.

Conversely, each symbolic transition can be instantiated in a concrete one.

Lemma 2 (Soundness): Let G be a graph and $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$. Let $K_s = (\mathcal{P}_s;\mathcal{S}_s;\mathcal{I}_s;\mathcal{C})$ and $K'_s = (\mathcal{P}'_s;\mathcal{S}'_s;\mathcal{I}'_s;\mathcal{C}')$ be two ground symbolic configurations such that $K_s \rightarrow^s_{G,\mathcal{M}} K'_s$. Let θ' be a solution of $\mathcal{C}'$ and θ be the restriction of θ' to $rvar(\mathcal{C})$. Let K_c be the θ-concretization of K_s. There exists a ground concrete configuration K'_c such that:

- $K_c \rightarrow_{G,\mathcal{M}} K'_c$, and
- K'_c is the θ'-concretization of K'_s.

We deduce that checking for a concrete attack can be reduced to checking for a symbolic one.

Proposition 1: Let G be a graph and $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$. Let $K = (\mathcal{P}[\_];\mathcal{S};\mathcal{I})$ be a ground concrete configuration with a hole, and Φ be a formula. There is an $\mathcal{M}$-attack on K and Φ for graph G if, and only if,

$$(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})];\mathcal{S};\mathcal{I};\emptyset) \rightarrow^{s*}_{G,\mathcal{M}} (\lfloor \mathsf{out}(u)\rfloor_n \cup \mathcal{P}_s;\mathcal{S}_s;\mathcal{I}_s;\mathcal{C})$$

and the constraint system $\mathcal{C} \wedge u = \mathsf{error}$ has a solution.

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.

Example 7: 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 $\mathcal{C}$ has a solution σ for graph G_0 (cf. Example 5), so there is an $\{n_I\}$-attack on K_0 for G_0.

IV. Decidability result

In the remaining of the paper, we assume the fixed signature $(\mathcal{S}_1, \mathcal{F}_1)$ (defined in Example 1) for list, concatenation, mac and encryption. We also assume its associated deduction system $\vdash$ defined in Figure 1.

Simple properties like secrecy are undecidable when considering an unbounded number of role executions, even for classical protocols [12]. Since our class of processes encompasses classical protocols, the existence of an attack is also undecidable. In what follows, we thus consider a finite number of sessions, that is processes without replication. In most existing frameworks, the intruder is given as initial knowledge a finite number of messages (e.g. some of the secret keys or messages learned in previous executions). However, in the context of routing protocols, it is important to give an a priori unbounded number of node names to the attacker that he can use as its will, in particular for possibly passing some disequality constraints and for creating false routes.

We say that a process is *finite* if it does not contain the replication operator. A concrete configuration $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ is said *initial* if K is ground, $\mathcal{P}$ is finite, $\mathcal{S}$ is a finite set of terms and $\mathcal{I} = \mathcal{N}_{\mathsf{loc}} \cup \mathcal{I}'$ where $\mathcal{I}'$ a finite set of terms (the intruder is given all the node names in addition to its usual initial knowledge).

Our second main contribution is to show that accessibility properties are decidable for finite processes of our process algebra, which models secured routing protocols, for a bounded number of sessions. We actually provide two decision procedures, according to whether the network is *a priori* given or not. In case the network topology is not fixed in advance, our procedure allows to automatically discover whether there exists a (worst-case) topology that would yield an attack.

Theorem 1: Let $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ be an initial concrete configuration with a hole, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a property. Deciding whether there exists a graph G such that there is an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.

Theorem 2: Let $K = (\mathcal{P}[\_]; \mathcal{S}; \mathcal{I})$ be an initial concrete configuration with a hole, G be a graph, $\mathcal{M} \subseteq \mathcal{N}_{\mathsf{loc}}$ be a finite set of nodes, and $\Phi \in \mathcal{L}_{\mathsf{route}}$ be a property. Deciding whether there exists an $\mathcal{M}$-attack on K and Φ for the topology G is NP-complete.

Note that Theorem 1 does not imply Theorem 2 and reciprocally. Theorems 1 and 2 ensure in particular that we can decide whether a routing protocol like SRP can guarantee that any route accepted by the source is indeed a route (a path) in the network (which can be fixed by the user or discovered by the procedure). The NP-hardness of the existence of an attack comes from the NP-hardness of the existence of a solution for deduction constraint systems [23]. The (NP) decision procedures proposed for proving Theorems 1 and 2 involve several steps, with many common ingredients.

Step 1. Applying Proposition 1, it is sufficient to decide whether there exists a sequence of symbolic transitions (and a graph G if G is not fixed)

$$(\mathcal{P}[\mathsf{if}\ \Phi\ \mathsf{then}\ \mathsf{out}(\mathsf{error})]; \mathcal{S}; \mathcal{I}; \emptyset) \rightarrow^{s*}_{G,\mathcal{M}} (\lfloor \mathsf{out}(u) \rfloor_n \cup \mathcal{P}_s; \mathcal{S}_s; \mathcal{I}_s; \mathcal{C})$$

such that $\mathcal{C} \wedge u = \mathsf{error}$ admits a solution for the graph G. Since processes contain no replication and involve communication between a finite number of nodes, it is possible to guess the sequence of symbolic transitions yielding an attack (by guessing also the edges between the nodes that are either in $\mathcal{M}$ or involved in a communication step) and the resulting configuration remains of size polynomially bounded by the size of the initial configuration. Moreover, any left-hand-side of a deduction constraint in $\mathcal{C}$ is of the form $T \cup \mathcal{N}_{\mathsf{loc}}$ where T is a finite set of terms. It then remains to decide the existence of a solution for our class of constraint systems.

Step 2. It has been shown in [11] 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 extend this result to our generalized notion of constraint systems, *i.e.* with disequality tests and formula of $\mathcal{L}_{\mathsf{route}}$, and for an intruder knowledge with an infinite number of names.

Step 3. We then show how to decide the existence of a solution for a constraint system, where each deduction constraint is solved, that is of the form $T \Vdash x$. It is not straightforward like in [11] since we are left with (non solved) disequality constraints and formulas. The key step consists in showing that we can bound (polynomially) the size of the lists in a minimal attack.

The two last steps are developed in the two following subsections.

A. Solving constraint systems with an infinite number of names

A constraint system $\mathcal{C}$ can be split into four disjoint sets $\mathcal{C} = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4$ where $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ are respectively the sets of unification constraints, deduction constraints, disequality constraints and formulas.

Unification constraints can be easily encoded into deduction constraints by replacing each equality $u = v$ by the constraint $\{u\}_k \Vdash \{v\}_k$. More precisely, instead of $\mathcal{C}$, we consider the constraint system $\mathcal{C}'$ where each constraint $C_i \in \mathcal{C}$ of the form $u_i = v_i$ has been replaced by $\mathcal{I}_j, \{u_i\}_{k_i} \Vdash \{v_i\}_{k_i}$ where k_i is a fresh key and j is the greatest index such that $j < i$ and C_j is a deduction constraint of the form $\mathcal{I}_j \Vdash t_j$ (or $\mathcal{I}_j$ is empty if such a j does not exist). Moreover, the term $\{u_i\}_{k_i}$ is added in the left-hand side of all deduction constraints greater than C_i. It is immediate to see that $\mathcal{C}$ and $\mathcal{C}'$ admits the same set of solutions and that $\mathcal{C}'$ remains a constraint system.

Thus we are left to decide the existence of a solution for $\mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4$ where $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ are respectively the sets of deduction constraints, disequality constraints and formulas. We say that a constraint system $\mathcal{C}$ is a *deduction constraint system* if all its constraints are deduction constraints. Such a system is in *solved form* if $\mathcal{C} = T_1 \Vdash x_1 \wedge \ldots \wedge T_n \Vdash x_n$ where $x_1, \ldots, x_n$ are distinct variables. The goal of this section is to show that we can assume that $\mathcal{C}_2$ is in *solved form*.

It has been shown in [11] that the existence of a solution of a deduction constraint system can be reduced to the existence of a solution of a *solved* deduction constraint system by applying (a variant of) the transformation rules presented in Figure 4.

All the rules are indexed by a substitution. When there is no index then the identity substitution is implicitly assumed. We write $\mathcal{C} \rightsquigarrow^n_\sigma \mathcal{C}'$ if there are $\mathcal{C}_1, \ldots, \mathcal{C}_n$ with $n \geq 1$, $\mathcal{C}' = \mathcal{C}_n$, $\mathcal{C} \rightsquigarrow_{\sigma_1} \mathcal{C}_1 \rightsquigarrow_{\sigma_2} \cdots \rightsquigarrow_{\sigma_n} \mathcal{C}_n$, and $\sigma = \sigma_n \circ \sigma_{n-1} \circ \cdots \circ \sigma_1$. We write $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$ if $\mathcal{C} \rightsquigarrow^n_\sigma \mathcal{C}'$ for some $n \geq 1$, or if $\mathcal{C}' = \mathcal{C}$ and σ is the identity substitution.

Getting a polynomial bound on the length of simplification sequences can be achieved by considering a (complete) strategy in order to avoid getting twice the same constraint. It has been shown in [11] that a deduction constraint system admits a solution if, and only if, the transformation rules yield (in non-deterministically polynomial time) a solved constraint.

However, the result of [11] assumes that the deduction constraints are of the form $T \Vdash u$ where T is a *finite* set of terms. We have extended this result to the case where T contains an infinite set of names. Moreover, we have adapted the simplification rules to our signature with mac and lists.

Definition 4: Let $\mathcal{C}$ be a deduction constraint system where all left hand sides of constraints are finite, and $\mathcal{I}_0$ be a (possibly infinite) set of names. We say that $(\mathcal{C}, \mathcal{I}_0)$ is a *special* constraint system if $St(\mathcal{C}) \cap \mathcal{I}_0 = \emptyset$. The deduction constraint system $\overline{\mathcal{C}}^{\mathcal{I}_0}$ associated to $(\mathcal{C}, \mathcal{I}_0)$ is inductively defined by

$$\overline{\mathcal{C} \wedge T \Vdash u}^{\mathcal{I}_0} = \overline{\mathcal{C}}^{\mathcal{I}_0} \wedge ((\mathcal{I}_0 \cup T) \Vdash u).$$

A substitution θ is a solution of a special constraint system $(\mathcal{C}, \mathcal{I}_0)$ if for every $T \Vdash u \in \mathcal{C}$, $(T \cup \mathcal{I}_0)\theta \Vdash u\theta$, i.e. θ is a solution of $\overline{\mathcal{C}}^{\mathcal{I}_0}$.

We show that when solving a special constraint system $(\mathcal{C}, \mathcal{I}_0)$, it is sufficient to apply the transformation rules to $\mathcal{C}$, following a well-chosen strategy in order to get a polynomial bound on the length of simplification sequences. We consider the following strategy $\mathcal{S}$:

- apply eagerly R_4 and postpone R_1 as much as possible
- apply the substitution rules eagerly (as soon as they are enabled). This implies that all substitution rules are applied at once, since the rules $\mathsf{R}_1, \mathsf{R}_4, \mathsf{R}_f$ cannot enable a substitution.
- when R_4 and substitution rules are not enabled, apply R_f to the constraint whose right hand side is maximal (in size).

For ordinary constraint systems, the strategy $\mathcal{S}$ is complete and yields derivations of polynomial length (see Section 4.7 in [11]). It remains to show that the procedure also works for special constraint systems.

Theorem 3: Let $(\mathcal{C}_0, \mathcal{I}_0)$ be a special constraint system, Φ a set of formulas and disequality constraints, and θ be a substitution.

1) (Correctness) If $\mathcal{C}_0 \rightsquigarrow^*_\sigma \mathcal{C}'$ by a derivation in $\mathcal{S}$ for some $\mathcal{C}'$ and some substitution σ, and if θ is a solution for $\Phi\sigma$ and $(\mathcal{C}', \mathcal{I}_0)$, then $\sigma\theta$ is a solution for Φ and $(\mathcal{C}_0, \mathcal{I}_0)$.
2) (Completeness) If θ is a solution for $(\mathcal{C}_0, \mathcal{I}_0)$ and Φ, then there exists a deduction constraint system $\mathcal{C}'$ in solved form and substitutions σ, θ' such that $\theta = \sigma\theta'$, $\mathcal{C}_0 \rightsquigarrow^*_\sigma \mathcal{C}'$ by a derivation in $\mathcal{S}$, and θ' is a solution for $(\mathcal{C}', \mathcal{I}_0)$ and $\Phi\sigma$.
3) (Termination) If $\mathcal{C}_0 \rightsquigarrow^n_\sigma \mathcal{C}'$ by a derivation in $\mathcal{S}$ for some deduction constraint system $\mathcal{C}'$ and some substitution σ, then n is polynomially bounded in the size of $\mathcal{C}_0$.

The proof of Theorem 3 is mainly an adaptation of the result in [11] and relies on the following lemma, which intuitively states that adding an infinite set of disjoint names does not provide an additional deduction power to the intruder.

Lemma 3: Let T be a set of terms that contains at least one constant, u a term and E a set of constants such that $St(T \cup \{u\}) \cap E = \emptyset$. If $T \cup E \vdash u$, then we have that $T \vdash u$.

$$
\begin{array}{llcll}
\mathsf{R}_1 & \mathcal{C} \wedge T \Vdash u & \rightsquigarrow & \mathcal{C} & \text{if } T \cup \{x \mid (T' \Vdash x) \in \mathcal{C}, T' \subsetneq T\} \vdash u \\
\mathsf{R}_2 & \mathcal{C} \wedge T \Vdash u & \rightsquigarrow_\sigma & \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma & \text{if } \sigma = \mathsf{mgu}(t, v), t \in St(T), v \in St(u), t \neq v, t, v \text{ not variables} \\
\mathsf{R}_3 & \mathcal{C} \wedge T \Vdash u & \rightsquigarrow_\sigma & \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma & \text{if } \sigma = \mathsf{mgu}(t_1, t_2), t_1, t_2 \in St(T), t_1 \neq t_2, t_1, t_2 \text{ not variables} \\
\mathsf{R}_4 & \mathcal{C} \wedge T \Vdash u & \rightsquigarrow & \perp & \text{if } var(T, u) = \emptyset \text{ and } T \not\vdash u \\
\mathsf{R}_f & \mathcal{C} \wedge T \Vdash f(u, v) & \rightsquigarrow & \mathcal{C} \wedge T \Vdash u \wedge T \Vdash v & \text{for } f \in \{\langle\rangle, ::, \mathsf{hmac}, \mathsf{enc}\}
\end{array}
$$

Figure 4. Simplification rules

B. Bounding the size of minimal attacks

Applying the technique described in the previous section, we are left to decide the existence of a solution for a constraint system $\mathcal{C} = \mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4$ where $\mathcal{C}_2$ is a solved deduction constraint system, $\mathcal{C}_3$ contains disequality constraints, and $\mathcal{C}_4$ contains formulas of $\mathcal{L}_{\mathsf{route}}$. We can show that the size of the constraint system $\mathcal{C}$ obtained after the successive transformations remains polynomially bounded in the size of the initial configuration.

We first prove that given any solution of $\mathcal{C}$, the variables which are not of sort loc or lists can be instantiated by any fresh name, still preserving the solution.

Lemma 4: Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system in solved form, Φ_1 be a formula of $\mathcal{L}_{\mathsf{route}}$, Φ_2 be a set of disequality constraints, and G be a graph. Consider σ a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G. There is a solution σ' of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G such that:

- $x\sigma' = x\sigma$ for every variable x of sort loc or lists;
- $x\sigma' \in \mathcal{I}$ otherwise.

We then show that it is possible to find a solution in which lists are polynomially bounded. We need to prove two separate lemmas, according to whether the network topology is fixed or not. In case the network topology is not fixed, we show that we can bound the size of an attack, possibly by changing the graph.

Lemma 5: Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system in solved form, Φ_1 be a conjunction of atomic formulas of $\mathcal{L}_{\mathsf{route}}$, Φ_2 be a set of disequality constraints. If there is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for the graph G, then there exists a graph G' and a substitution σ such that σ is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G', and σ is polynomially bounded in the size of $\mathcal{C}, \Phi_1$ and Φ_2.

In case the network topology is fixed, we show that we can bound the size of an attack, where the bound depends on the size of the graph.

Lemma 6: Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system in solved form, Φ_1 be a conjunction of atomic formulas of $\mathcal{L}_{\mathsf{route}}$, Φ_2 be a set of disequality constraints, and G be a graph. If there is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, then there exists a solution σ of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G that is polynomially bounded in the size of $\mathcal{C}, \Phi_1, \Phi_2$ and G.

The proofs of Lemma 5 and 6 use the fact that disequality constraints can be satisfied using fresh node names and that the predicates of the logic $\mathcal{L}_{\mathsf{route}}$ check only a finite number of nodes. Combining the lemmas, we get that it is possible to bound the size of a (minimal) solution. For Theorem 1, we conclude by further noticing that nodes that do not occur explicitly in a solution σ can be removed from the graph.

The complexity analysis actually requires to state slightly more precise lemmas. Assume given a constraint system $\mathcal{C} = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4$ where $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4$ are respectively the sets of unification constraints, deduction constraints, disequality constraints and formulas. We have seen that $\mathcal{C}_1$ can be polynomially encoded as deductions constraints thus we can assume $\mathcal{C}_1$ to be empty. Then, by Theorem 3, we can apply (non-deterministically) a polynomial number of simplification rules and get a constraint system $\mathcal{C}_2\sigma \wedge \mathcal{C}_3\sigma \wedge \mathcal{C}_4\sigma$ where $\mathcal{C}_2\sigma$ is in solved form. Then, as shown in the proofs of Lemma 5 and 6, we can actually bound the size of a minimal solution polynomially in $\mathcal{C}_2 \wedge \mathcal{C}_3 \wedge \mathcal{C}_4$.

V. Conclusion

Using our symbolic semantics, we have shown that, for general processes that can broadcast and perform some correctness check in addition to the usual pattern matching, existence of attacks can be reduced to existence of a solution for (generalized) constraint systems. 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 result enables in particular to automatically discover whether a particular network topology may allow malicious nodes to mount an attack.

As future work, we plan to extend our results to other cryptographic primitives (e.g. signatures and public keys) in order to model more protocols. Since our results reuse existing techniques such as constraint solving, we believe that our procedure could be implemented in existing tools after a few adaptations. At the moment, if the network topology is given in advance, then the location of malicious nodes is also known before the protocol starts. We believe that it would be possible to analyze a protocol for a given network of honest nodes without knowing a priori the location of malicious nodes. Another extension would be to model mobility during the execution of the protocol. This would allows us to consider changes in the network topology and to analyze the security of route updates. This requires to model an appropriate security property.

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] G. Ács. *Secure Routing in Multi-hop Wireless Networks*. PhD thesis, Budapest University of Technology and Economics, 2009.

[4] R. Amadio, D. Lugiez, and V. Vanackère. On the symbolic reduction of processes with cryptographic functions. *Theoretical Computer Science*, 290(1):695–740, 2002.

[5] A. Armando et al. The AVISPA Tool for the automated validation of internet security protocols and applications. In *Proc. 17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *LNCS*, pages 281–285. Springer, 2005.

[6] M. Arnaud, V. Cortier, and S. Delaune. Modeling and verifying ad hoc routing protocols. Research Report LSV-10-03, Laboratoire Spécification et Vérification, ENS Cachan, France, Feb. 2010. 36 pages.

[7] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proc. 14th Computer Security Foundations Workshop (CSFW'01)*. IEEE Comp. Soc. Press, 2001.

[8] B. Blanchet. An automatic security protocol verifier based on resolution theorem proving (invited tutorial). In *Proc. 20th International Conference on Automated Deduction (CADE'05)*, 2005.

[9] L. Buttyán and I. Vajda. Towards Provable Security for Ad Hoc Routing Protocols. In *Proc. 2nd ACM workshop on Security of ad hoc and sensor networks (SASN'04)*, pages 94–105, New York, NY, USA, 2004. ACM.

[10] J. Clark and J. Jacob. A survey of authentication protocol literature. http://www.cs.york.ac.uk/~jac/papers/drareviewps.ps, 1997.

[11] H. Comon-Lundh, V. Cortier, and E. Zalinescu. Deciding security properties for cryptographic protocols. Application to key cycles. *ACM Transactions on Computational Logic (TOCL)*, 11(4):496–520, 2010.

[12] N. Durgin, P. Lincoln, J. Mitchell, and A. Scedrov. Undecidability of bounded security protocols. In *Proc. Workshop on Formal Methods and Security Protocols*, 1999.

[13] Y.-C. Hu, A. Perrig, and D. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. *Wireless Networks*, 11, January 2005.

[14] Y.-C. Hu, A. Perrig, and D. B. Johnson. Wormhole attacks in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(2):370–380, 2006.

[15] 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.

[16] 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.

[17] L. Lazos, R. Poovendran, C. Meadows, P. Syverson, and L. W. Chang. Preventing wormhole attacks on wireless ad hoc networks: a graph theoretic approach. In *Wireless Communications and Networking Conference*, volume 2, pages 1193–1199 Vol. 2, 2005.

[18] J. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *Proc. 8th ACM Conference on Computer and Communications Security (CCS'01)*, 2001.

[19] S. Nanz and C. Hankin. A Framework for Security Analysis of Mobile Wireless Networks. *Theoretical Computer Science*, 367(1):203–227, 2006.

[20] 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.

[21] L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6(1-2):85–128, 1998.

[22] 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, 1999.

[23] 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.

[24] 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.

[25] F. J. Thayer, J. C. Herzog, and J. D. Guttman. Strand spaces: Proving security protocols correct. *Journal of Computer Security*, 7(1), 1999.

[26] 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)*, 2003.

[27] M. G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proc. 1st ACM workshop on Wireless SEcurity (WiSE'02)*, pages 1–10. ACM, 2002.

Appendix

In this appendix, we give a full proof of Lemma 4 and Lemma 6. We also explain how to adapt the proof of Lemma 6 to prove Lemma 5. Details can be found in [6].

A. Proof of Lemma 4

Let S be a set, we denote by $\#S$ the cardinal of S.

Lemma 4: Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system in solved form, Φ_1 be a formula of $\mathcal{L}_{\mathsf{route}}$, Φ_2 be a set of disequality constraints, and G be a graph. Consider σ a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G. There is a solution σ' of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G such that:

- $x\sigma' = x\sigma$ for every variable x of sort loc or lists;
- $x\sigma' \in \mathcal{I}$ otherwise.

Proof: Since $(\mathcal{C}, \mathcal{I})$ is a special constraint system in solved form, we have that

$$\mathcal{C} = T_1 \Vdash x_1 \ \wedge \ \ldots \ \wedge \ T_n \Vdash x_n$$

where:

- $x_1, \ldots, x_n$ are distinct variables, and
- $var((\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2) = \{x_1, \ldots, x_n\} = rvar(\mathcal{C})$.

We show the result by induction on:

$$\mu(\sigma) = \#\{x \in rvar(\mathcal{C}) \mid \quad x \text{ is not of sort } \mathsf{loc} \text{ or } \mathsf{lists} \text{ and } x\sigma \notin \mathcal{I}\}.$$

Base case: $\mu(\sigma) = 0$. In such a case, since $rvar(\mathcal{C})$ contains all the variables that occur in the constraint system, we easily conclude. The substitution σ is already of the right form.

Induction step: $\mu(\sigma) > 0$. Let i_0 be the maximal index $1 \leq i_0 \leq n$ such that $x_{i_0}\sigma \notin \mathcal{I}$ and x_{i_0} is not of sort loc or lists. Let a be a name in $\mathcal{I}$ that does not occur elsewhere. Let $\sigma' = \tau \cup \{x_{i_0} \mapsto a\}$ where $\tau = \sigma|_X$ with $X = dom(\sigma) \smallsetminus \{x_{i_0}\}$. Clearly, we have that $\mu(\sigma') < \mu(\sigma)$. In order to conclude, it remains to show that σ' is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$.

1) *We show that σ' is a solution of* $(\mathcal{C}, \mathcal{I})$. For every $i < i_0$, since σ is a solution of $(\mathcal{C}, \mathcal{I})$, we have that $T_i\sigma \cup \mathcal{I} \vdash x_i\sigma$. Since x_{i_0} does not occur in this constraint, we also have that $T_i\sigma' \cup \mathcal{I} \vdash x_i\sigma'$. Since $a \in \mathcal{I}$, we have that $T_{i_0}\sigma' \cup \mathcal{I} \vdash x_{i_0}\sigma'$.
   For every $i > i_0$, according to the definition of i_0, either x_i is of sort loc or lists, or $x_i\sigma \in \mathcal{I}$. In the first case, as for every term t of sort loc or lists, $\mathcal{N}_{\mathsf{loc}} \vdash t$, we have that $\mathcal{N}_{\mathsf{loc}} \vdash x_i\sigma$. In the second case, $\mathcal{I} \vdash x_i\sigma$. Hence, in both cases, we have that $T_i\sigma' \cup \mathcal{I} \vdash x_i\sigma'$.
2) *We show that σ' is a solution of* Φ_1. All the variables appearing in Φ_1 are of type loc or lists. Hence, we have that $\Phi_1\sigma = \Phi_1\sigma'$. This allows us to conclude.
3) *Lastly, we show that σ' is a solution of* Φ_2. Let $\forall Y.u \neq v$ be a disequality constraint in Φ_2. Assume w.l.o.g. that $dom(\sigma) \cap Y = \emptyset$. Since σ is a solution of $\forall Y.u \neq v$, we know that $u\sigma$ and $v\sigma$ are not unifiable.
   Assume by contradiction that there exists a substitution θ' such that $u\sigma'\theta' = v\sigma'\theta'$ (i.e. σ' does not satisfy $\forall Y.u \neq v$). We can assume w.l.o.g. that $u\sigma'\theta'$ and $v\sigma'\theta'$ are ground terms, and $x_{i_0} \notin dom(\theta')$. In such a case, we have that:
   $$\begin{array}{l}(u\sigma')\theta' = ((u\tau)\{x_{i_0} \mapsto a\})\theta' = ((u\tau)\theta')\{x_{i_0} \mapsto a\} \\ (v\sigma')\theta' = ((v\tau)\{x_{i_0} \mapsto a\})\theta' = ((v\tau)\theta')\{x_{i_0} \mapsto a\}\end{array}$$
   Since a is fresh, we deduce that $(u\tau)\theta' = (v\tau)\theta'$. Hence, we have also that:
   $$((u\tau)\theta')\{x_{i_0} \mapsto x_{i_0}\sigma\} = ((v\tau)\theta')\{x_{i_0} \mapsto x_{i_0}\sigma\}$$
   i.e. $u\sigma\theta' = v\sigma\theta'$. This contradicts the fact that $u\sigma$ and $v\sigma$ are not unifiable.

Hence, σ' is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$. ■

B. Proofs of Lemma 6 and Lemma 5

Let u be a term. We denote by $|u|_d$ the maximal depth of a variable in u. The lemma below is useful to bound the depth of variables after application of a substitution.

Lemma 7: Let T be a set of terms, P be a set of equations between terms in T and $\sigma = \mathsf{mgu}(P)$. For every variable x, we have that:

$$|x\sigma|_d \leq \#dom(\sigma) \cdot \max\{|t|_d \mid t \in T\}.$$

Definition 5: An *extracted list from* a list $l = [a_1, \ldots, a_n]$ is a list $[a_{i_1}, \ldots, a_{i_k}]$ such that $1 \leq i_1 \leq i_2 \leq \ldots \leq i_k \leq n$ with $0 \leq k \leq n$.

A *marked lis*t is a list in which some elements are marked.

Definition 6: Let l' be marked list in which at least one of its element is not marked. A *variation* of $l' = a'_1 :: \ldots :: a'_n$ is a list $l = a_1 :: \ldots :: a_n$ such that:

- there exists $1 \leq j \leq n$ such that a'_j is not marked and a_j is a fresh name,
- for all $1 \leq i \leq n$ such that $i \neq j$, we have that $a_i = a'_i$.

We prove that we can find a solution in which lists are polynomially bounded. In case the network topology is fixed, the bound depends on the size of the graph, i.e. its number of edges. Let l be a list, we denote by $|l|$ the length of l.

Lemma 6: Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system in solved form, Φ_1 be a conjunction of atomic formulas of $\mathcal{L}_{\mathsf{route}}$, Φ_2 be a set of disequality constraints, and G be a graph. If there is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, then there exists a solution σ of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G that is polynomially bounded in the size of $\mathcal{C}, \Phi_1, \Phi_2$ and G.

Proof: We write $G = (V_G, E_G)$, $\Phi_2 = \bigwedge_n \forall Y_n . u_n \neq v_n$,

$$\Phi_1 = \bigwedge_i \pm_i \mathsf{check}(a_i, b_i) \wedge \bigwedge_j \bigwedge_k^{p_j} \pm_{j_k} \mathsf{checkl}(c_{j_k}, l_j) \wedge \bigwedge_l \pm_l \mathsf{route}(r_l) \wedge \bigwedge_h \pm_h \mathsf{loop}(p_h)$$

with $\pm \in \{+, -\}$, a_i, b_i, c_{j_k} are of sort loc, l_j, r_l, p_h are terms of type lists, u_n, v_n are terms and Y_n are sets of variables.

In the following, we denote:

- N the maximal depth of a variable in the disequality constraints,
- k the maximal number of variables in a disequality constraint,
- C the number of constraints $\pm$checkl in Φ_1,
- L the number of constraints loop in Φ_1,
- R the number of constraints $\neg$route in Φ_1, and
- $M = \max(kN + 3C + L + R + 3, |G|)$ where $|G|$ is the size of G, i.e. $|G| = \#E_G$.

We show that, if there is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G, then there exists a substitution σ such that σ is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, and

- for all variables x of sort lists, $|x\sigma| \leq M$, and
- $x\sigma \in \mathcal{I}$ otherwise.

First, we have that $x\sigma \in \mathcal{I}$ when x is a variable of sort loc. Moreover, thanks to Lemma 4, we can assume that $x\sigma \in \mathcal{I}$ when x is a variable that is neither of sort loc nor of type lists. Now, among these solutions, consider a smallest solution σ of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, where the size of a solution σ is given by $|\sigma| = |x_1\sigma| + \ldots + |x_n\sigma|$ where $x_1, \ldots, x_n$ are the variables of sort lists that occur in $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$.

If $|x\sigma| \leq M$ for all variables x of sort lists, then we easily conclude. Otherwise, there exists a variable x_ℓ of sort lists such that the length of $x_\ell\sigma$ is greater than M. We are going to show that we can build σ' from σ, solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G, smaller than σ. More specifically, we build σ' such that for all $x \neq x_\ell, x\sigma' = x\sigma$, and $|x_\ell\sigma'| \leq M < |x_\ell\sigma|$.

We build $x_\ell\sigma'$ by marking the names we want to keep in the list in the following manner:

$$x_\ell\sigma = \begin{array}{|c|c|c|c|c|c|}\hline a_1 & a_2 & \ldots & a_{kN} & \ldots & a_P \\ \hline\end{array}$$

We mark the first kN names in the list:

$$\begin{array}{|c|c|c|c|c|}\hline \overline{a_1} & \overline{a_2} & \ldots & \overline{a_{kN}} & \ldots \\ \hline\end{array}$$

We then mark the other names we want to keep in the list in the following way:

Case of a checkl *that occurs positively.*

If there exists c_{j_k} such that $\mathsf{checkl}(c_{j_k}, l_j)$ is a constraint that occurs positively in Φ_1, i.e. $\pm_{j_k} = +$, and $x_\ell \in \mathit{var}(l_j)$. Assume that $l_j = d_1 :: \ldots :: d_p :: x_\ell$. As σ is a solution for Φ_1, in particular we know that $c = c_{j_k}\sigma$ appears exactly once in $l_j\sigma$, and for any l' sublist of $l_j\sigma$,

- if $l' = a :: c :: l_1$, then $(a, c) \in E_G$.
- if $l' = c :: b :: l_1$, then $(b, c) \in E_G$.

Since c appears exactly once in $l_j\sigma$, either there exists n such that $c = d_n\sigma$, or there exists m such that $c = a_m$. In the first case and if $n = p$, we mark a_1. In the second case, we mark a_m, a_{m-1}(if $m > 1$) and a_{m+1}(if $m < P$). Any variation of a list extracted from $x_\ell\sigma$ containing at least the marked names plus another one satisfies the checkl condition for graph G.

$$\begin{array}{|c|c|c|c|c|c|c|}\hline a_1 & \ldots & \overline{a_{m-1}} & \overline{a_m} & \overline{a_{m+1}} & \ldots & a_P \\ \hline\end{array}$$

Case of a checkl *that occurs negatively.*

If there exists c_{j_k} such that $\mathsf{checkl}(c_{j_k}, l_j)$ is a constraint that occurs negatively in Φ_1, i.e. $\pm_{j_k} = -$, and $x_\ell \in \mathit{var}(l_j)$. Assume that $l_j = b_1 :: \ldots :: b_p :: x_\ell$. As σ is a solution for Φ_1, we can have three different cases depending on $c = c_{j_k}\sigma$:

- c does not appear in $l_j\sigma$: for every n, m, $b_n\sigma \neq c$ and $a_m \neq c$. In that case, we mark nothing.
- c appears at least twice in $l_j\sigma$. In that case, we choose two occurrences of c and we mark them when they appear in $x_\ell\sigma$.

$$\begin{array}{|c|c|c|c|c|c|c|}\hline a_1 & \ldots & \overline{c} & \ldots & \overline{c} & \ldots & a_P \\ \hline\end{array}$$

- c appears once in $l_j\sigma$, but one of his neighbors in the list is not a neighbor of his in the graph. For example, $c = a_i$ and $(a_i, a_{i+1}) \notin E_G$. We mark c and this false neighbor when they appear in $x_\ell\sigma$.

$$\begin{array}{|c|c|c|c|c|c|}\hline a_1 & \ldots & \overline{a_i} & \overline{a_{i+1}} & \ldots & a_M \\ \hline\end{array}$$

Any variation of a list extracted from $x_\ell\sigma$ containing at least the marked names plus another one satisfies the $\neg$checkl condition for graph G.

Case of a loop *that occurs positively.*

If there exists h such that $\mathsf{loop}(p_h)$ is a constraint that occurs positively in Φ_1, i.e. $\pm_h = +$, and $x_\ell \in \mathit{var}(p_h)$. Assume $p_h = b_1 :: \ldots :: b_p :: x_\ell$. Then there exists a name c repeated in $p_h\sigma$. We mark two occurrences of such a c, when they appear in $x_\ell\sigma$.

$$\begin{array}{|c|c|c|c|c|c|c|}\hline a_1 & \ldots & \overline{c} & \ldots & \overline{c} & \ldots & a_P \\ \hline\end{array}$$

Any variation of a list extracted from $x_\ell\sigma$ containing at least the marked names plus another one satisfies the loop condition for graph G. Indeed, the condition does not depend on the graph.

Case of a loop *that occurs negatively.*

If there exists h such that $\mathsf{loop}(p_h)$ occurs negatively in Φ_1, i.e. $\pm_h = -$, and $x_\ell \in \mathit{var}(p_h)$. Assume that

$p_h = b_1 :: \ldots :: b_p :: x_\ell$. Removing nodes from the list preserves this condition, so any extracted list of $x_\ell\sigma$ satisfies the $\neg\mathsf{loop}$ condition. Moreover, as a variation of a list is built with a fresh constant, any variation of a list extracted from $x_\ell\sigma$ satisfies the condition.

Case of a route *that occurs negatively.*

If there exists r_l such that $\mathsf{route}(r_l)$ occurs negatively in Φ_1, i.e. $\pm_l = -$, and $x_\ell \in \mathit{var}(r_l)$. Assume that $r_l = b_1 :: \ldots :: b_p :: x_\ell$. As σ is a solution for Φ_1, we can have two different cases:

- There exists a name c repeated in $r_l\sigma$. Then we mark two occurrences of such a c, when they appear in $x_\ell\sigma$.
- There exists a sublist l of $r_l\sigma$ such that $l = c :: d :: l_1$ and $(c, d) \notin E_G$. We mark c and d if they appear in $x_\ell\sigma$.

a_1	$\ldots$	$\overline{c}$	$\overline{d}$	$\ldots$	a_P

Any variation of a list extracted from $x_\ell\sigma$ containing at least the marked names plus another one satisfies the $\neg\mathsf{route}$ condition for G.

Case of a route *that occurs positively.*

If there exists r_l such that $\mathsf{route}(r_l)$ occurs positively in Φ_1, i.e. $\pm_l = +$, and $x_\ell \in \mathit{var}(r_l)$. Assume that $r_l = b_1 :: \ldots :: b_p :: x_\ell$. Write $r_l\sigma = c_1 :: \ldots :: c_n$. As σ is a solution for Φ_1 in G, for every $0 < i < n$, $(c_i, c_{i+1}) \in E_G$ and for every $i \neq j$, $c_i \neq c_j$. Consequently, $|r_l\sigma| \leq \#E_G$, and as $|x_\ell\sigma| \leq |r_l\sigma|$, we have that $|x_\ell\sigma| \leq |G|$. But our hypothesis tells us that $|x_\ell\sigma| > M \geq |G|$. So there is no positive route condition on x_ℓ.

We count the number of marked names. We have marked the first kN names in the list. For each constraint $\pm\mathsf{checkl}$, we mark at most 3 names in the list. Suppose there are several constraints $\neg\mathsf{route}(l)$ with x_ℓ sublist of l. Either $\neg\mathsf{route}(x_\ell\sigma)$ holds, and we can mark two names in $x_\ell\sigma$ which will make all the $\neg\mathsf{route}$ constraints true; or the constraint is satisfied by marking one name for each constraint. Thus, we need only mark $\mathsf{max}(R, 2)$ names when $R \geq 1$ and 0 otherwise. Thus, in any case, it is sufficient to mark $R+1$ names in $x_\ell\sigma$. Similarly, it is sufficient to mark $L+1$ names in $x_\ell\sigma$ to satisfy the loop constraints. The number of names marked in the list is at most $kN + 3C + (R+1) + (L+1) \leq M$.

Consider l_1 extracted from $x_\ell\sigma$ by keeping only the marked names in $x_\ell\sigma$ and the first unmarked name. Such an unmarked name exists, because $|x_\ell\sigma| \geq M$. Let l_2 be the variation of l_1 replacing the first unmarked name with a fresh constant a_ℓ. For each condition considered above, l_2 satisfies it, as it is a variation of a list extracted from $x_\ell\sigma$ containing the marked names.

Let σ_0 be the substitution such that $x\sigma_0 = x\sigma$ for every $x \in \mathit{dom}(\sigma) \smallsetminus \{x_\ell\}$, and $x\sigma = x$ otherwise. Let $\sigma' = \sigma_0 \cup \{x_\ell \mapsto l_2\}$. By hypothesis, σ is a solution of Φ_1 for G, so by construction, σ' is a solution of Φ_1 for G. Now, it remains for us to show that σ' is a solution of $(\mathcal{C}, \mathcal{I})$ and Φ_2.

Deduction constraints. Consider a right-hand side variable x_i of $\mathcal{C}$. Either x_i is of sort loc or lists, which means that $\mathcal{N}_{\mathsf{loc}} \vdash x_i\sigma'$, thus $\mathcal{I} \vdash x_i\sigma'$. Or x_i is not of sort loc or lists, so in particular $x_i \in \mathit{dom}(\sigma) \smallsetminus x_\ell$, and $x_i\sigma' = x_i\sigma \in \mathcal{I}$, so $\mathcal{I} \vdash x_i\sigma'$. Hence, in both cases, we have that $\mathcal{I} \vdash x_i\sigma'$. Consequently, σ' is a solution of $(\mathcal{C}, \mathcal{I})$.

Disequality constraints. Consider a disequality constraint $\forall Y.u \neq v \in \Phi_2$. We assume w.l.o.g. that $\mathit{dom}(\sigma) \cap Y = \emptyset$. We have to show that $u\sigma'$ and $v\sigma'$ are not unifiable. We distinguish two cases. Either $u\sigma_0$ and $v\sigma_0$ are not unifiable, but in such a case, we easily deduce that $u\sigma'$ and $v\sigma'$ are not unifiable too. This allows us to conclude. Otherwise, let $\mu = \mathsf{mgu}(u\sigma_0, v\sigma_0)$.

If $\mathit{dom}(\mu) \subseteq Y$, let $\tau = \{x_\ell \mapsto x\sigma\} \circ \mu$. We have that:

$$(u\sigma)\tau = ((u\sigma_0)\{x_\ell \mapsto x_\ell\sigma\})\tau = (u\sigma_0\mu)\{x_\ell \mapsto x_\ell\sigma\}$$
$$(v\sigma)\tau = ((v\sigma_0)\{x_\ell \mapsto x_\ell\sigma\})\tau = (v\sigma_0\mu)\{x_\ell \mapsto x_\ell\sigma\}.$$

Hence, we deduce that $u\sigma$ and $v\sigma$ are unifiable, and we obtain a contradiction since σ satisfies the constraint $\forall Y.u \neq v$. Hence, this case is impossible.

Otherwise, there exists a term t such that $\mu(x_\ell) = t$, and $\mathit{var}(t) \subseteq Y$. We apply Lemma 7 to the set $T = \{u\sigma_0, v\sigma_0\}$, and the set of equations $P = \{u\sigma_0 = v\sigma_0\}$. We have that $\mu = \mathsf{mgu}(P)$. Since σ_0 is ground, we get that:

$$\begin{aligned} |t|_d &\leq \#\mathit{dom}(\mu).\mathsf{max}(|u\sigma_0|_d, |v\sigma_0|_d) \\ &\leq \#\mathit{dom}(\mu).\mathsf{max}(|u|_d, |v|_d) \\ &\leq kN \end{aligned}$$

We reason by case over t:

- If t is not of sort lists, as σ' is well-sorted, $u\sigma'$ and $v\sigma'$ are not unifiable.
- Suppose $t = a_1 :: \ldots :: a_n :: \bot$, with $a_1, \ldots, a_n$ terms of type loc. We write $t = t_1 @ t_2$ with t_2 ground term of maximal size, where @ denotes the concatenation of lists. We have shown that $|t_1|_d = |t_d| \leq kN$.
  We know that $x_\ell\sigma' = b_1 :: \ldots :: b_p$ and there exists $k' > kN$ such that $b_{k'} = a_\ell$ and a_ℓ is a constant of $\mathcal{I}$ which does not appear anywhere else in the constraints. Consequently, $a_{k'} \neq a_\ell$, and so $x_\ell\sigma' \neq t\theta$ for any substitution θ.
  Now, assume by contradiction that $u\sigma'$ and $v\sigma'$ are unifiable. This means that there exists τ such that $(u\sigma')\tau = (v\sigma')\tau$. Hence, we have that $\tau \circ \{x_\ell \mapsto x_\ell\sigma'\}$ is an unifier of $u\sigma_0$ and $v\sigma_0$. By hypothesis, we have that $\mu = \mathsf{mgu}(u\sigma_0, v\sigma_0)$. Hence, we deduce that there exists θ' such that $\tau \circ \{x_\ell \mapsto x_\ell\sigma'\} = \theta' \circ \mu$. We have that:
  - $\tau \circ \{x_\ell \mapsto x_\ell\sigma'\}(x_\ell) = x_\ell\sigma'$, and

  - $\theta' \circ \mu(x_\ell) = t\theta'$.

  This leads to a contradiction.

- Suppose $t = a_1 :: \ldots :: a_n :: y_\ell$, with $y_\ell \in Y$ variable of sort lists. We know that $|t|_d \leq kN$, thus we must have $n < kN$. We reason by contradiction. Assume that there exists θ' such that $(u\sigma')\theta' = (v\sigma')\theta'$. In the remaining of the proof, we show that $u\sigma$ and $v\sigma$ are unifiable.

  By hypothesis, we have that $\theta' \circ \{x_\ell \mapsto x_\ell\sigma'\}$ is an unifier of $u\sigma_0$ and $v\sigma_0$. Since $\mu = \mathsf{mgu}(u\sigma_0, v\sigma_0)$, we deduce that there exists ρ' such that:

  $$\rho' \circ \mu \quad = \quad \theta' \circ \{x_\ell \mapsto x_\ell\sigma'\} \qquad (1)$$

  We have that $x_\ell\sigma' = (x_\ell\mu)\rho' = t\rho'$. By hypothesis, we know that the size of $x_\ell\sigma$ is greater than $M \geq kN \geq n$. Let l_t be the list obtaining from $x_\ell\sigma$ by removing its n first elements. Let ρ_0 be a substitution such that $x\rho_0 = x\rho$ for every $x \in dom(\rho) \smallsetminus \{y_\ell\}$, and $y\rho_0 = y$ otherwise. Let $\rho = \rho_0 \circ \{y_\ell \mapsto l_t\}$. In order to conclude, it remains to show that $\rho \circ \mu$ is an unifier of $u\sigma$ and $v\sigma$.

  First, as $x_\ell\sigma$ and $x_\ell\sigma'$ have the same first kN elements by construction, and $n < kN$, we have that:

  $$\begin{array}{rcl} (x_\ell\mu)\rho & = & t\rho \\ & = & (a_1 :: \ldots :: a_n :: y_\ell)\rho \\ & = & a_1\rho' :: \ldots :: a_n\rho' :: l_t \\ & = & x_\ell\sigma. \end{array}$$

  Hence, we have that $((u\sigma)\mu)\rho = ((u\sigma_0)\mu)\rho$, and $((v\sigma)\mu)\rho = ((v\sigma_0)\mu)\rho$. We easily conclude that $u\sigma$ and $v\sigma$ are unifiable since we know that $(u\sigma_0)\mu = (v\sigma_0)\mu$.

In all possible cases, σ' satisfies the disequality constraint.

As a conclusion, σ' is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$, smaller than σ, which leads to a contradiction. ∎

In case the topology is not fixed, we show that we can bound the size of an attack, possibly by changing the graph. The proof follows the same lines as the proof of Lemma 6. However, we can not consider the size of the graph to bound the size of the lists. This is used in the proof of Lemma 6 to deal with the case of route that occur positively in the formula. In Lemma 5, we rely on the fact that we can change the graph to solve this problem.

Definition 7: Let $G = (V, E)$ be a graph and V_{ubi} be a set of nodes. The graph $(V \cup V_{ubi}, E \cup E_{ubi})$ where $E_{ubi} = \{(a, b) \mid a \in V \cup V_{ubi},\ b \in V_{ubi}\}$ is called the *ubiquitous graph associated to G and V_{ubi}*.

In the proof of Lemma 5, we will consider ubiquitous variation instead of variation. This is needed to satisfy a formula route that occur positively.

Definition 8: Let l' be a marked list and n the number of unmarked elements in l'. Let V_{ubi} be a set of nodes such that $\#V_{ubi} > n$ and names in V_{ubi} do not occur in l'. A *ubiquitous variation* according to V_{ubi} of $l' = a'_1 :: \ldots :: a'_n$ is a list $l = a_1 :: \ldots :: a_n$ such that:

- for all $1 \leq i \leq n$ such that a'_i is not marked, $a_i \in V_{ubi}$,
- for all $1 \leq i \leq n$ such that a'_i is marked, $a_i = a'_i$.

Moreover we require that the ubiquitous nodes of l are all distinct.

Lemma 5: Let $(\mathcal{C}, \mathcal{I})$ be a special constraint system in solved form, Φ_1 be a conjunction of atomic formulas of $\mathcal{L}_{\mathsf{route}}$, Φ_2 be a set of disequality constraints. If there is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for the graph G, then there exists a graph G' and a substitution σ such that σ is a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for G', and σ is polynomially bounded in the size of $\mathcal{C}, \Phi_1$ and Φ_2.

Proof: We write $G = (V_G, E_G)$. We adapt the proof of Lemma 6 by showing that there exists a solution σ such that for every variable x of sort lists, we have that $|x\sigma| \leq M = 2 \times (kN + 3C + L + R + 2)$ where k, N, C, L, and R are defined as in Lemma 6.

Let σ be a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G and assume that there exists a variable x_ℓ of sort lists such that $|x_\ell\sigma| > M$. Let V_{ubi} be a set of $M/2$ fresh nodes, i.e. names in $\mathcal{I}$ that do not occur in $\mathcal{C}$, Φ_1, and Φ_2. Consider G_{ubi} the ubiquitous graph associated to G and V_{ubi}. We show that we can build σ', a solution of $(\mathcal{C}, \mathcal{I}) \wedge \Phi_1 \wedge \Phi_2$ for graph G_{ubi}, such that for $x \neq x_\ell$, $x\sigma' = x\sigma$, and $|x_\ell\sigma'| \leq M$.

We build σ' in a similar way as in the previous proof. We mark $x_\ell\sigma$ as in the previous proof. The number of names marked in the list is at most:

$$kN + 3C + (R + 1) + (L + 1) \leq M/2.$$

Consider l_1 extracted from $x_\ell\sigma$ by leaving exactly one unmarked name between sequences of marked names. Note that, we have no more than $M/2$ unmarked names in l_1. Let l_2 be the ubiquitous variation of l_1 according to V_{ubi}. The fact that we consider a ubiquitous variation allows one to satisfy the constraint route that occurs positively. Note that, we have no more than $M/2$ ubiquitous names in l_2, so $|l_2| \leq M$.

Let σ_0 be the substitution such that $x\sigma_0 = x\sigma$ for every $x \in dom(\sigma) \smallsetminus \{x_\ell\}$, and $x\sigma = x$ otherwise. Let $\sigma' = \sigma_0 \cup \{x_\ell \mapsto l_2\}$. By construction, we have that σ' satisfies Φ_1. We show that σ' is a solution of $(\mathcal{C}, \mathcal{I})$ and Φ_2 for G_{ubi} as in Lemma 6. ∎