Formal verification of secured routing protocols

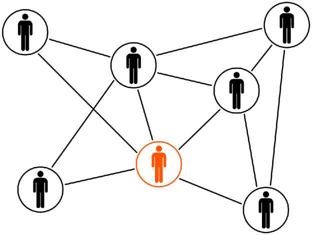
Mathilde Arnaud, Véronique Cortier, Stéphanie Delaune

LSV, ENS Cachan & CNRS & INRIA Saclay Île de France

LORIA, CNRS & INRIA Nancy Grand Est, France

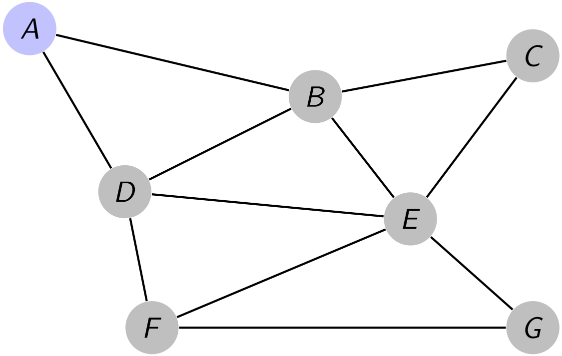
Loria
Laboratoire lorrain de recherche
en informatique et ses applications

Ad Hoc Networks

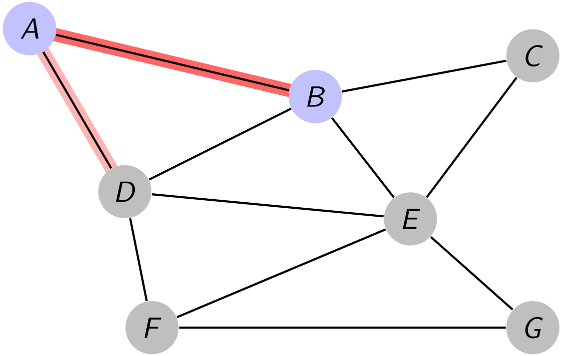


- Networks with little or no infrastructure
- Open infrastructure
- Agents can only **communicate directly with their immediate neighbors**

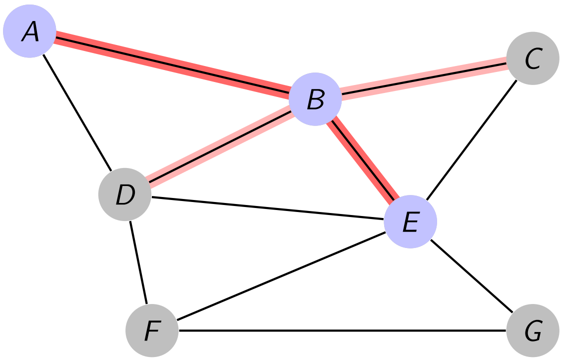
Propagation of a message in the network



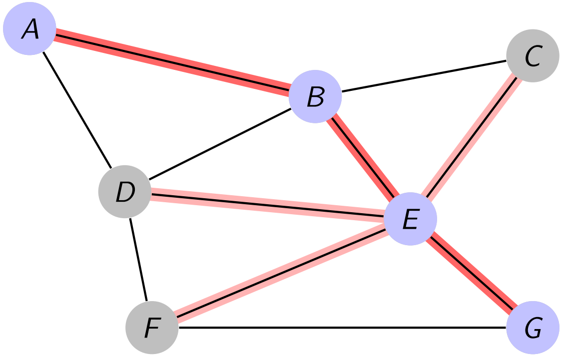
Propagation of a message in the network



Propagation of a message in the network



Propagation of a message in the network



Routing Protocols

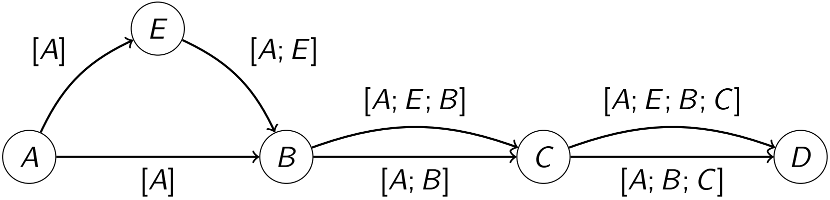
Protocol series of rules describing how each participant should behave in order to achieve a common goal

Routing goal: allowing distant nodes to communicate

Specificities of routing protocols:

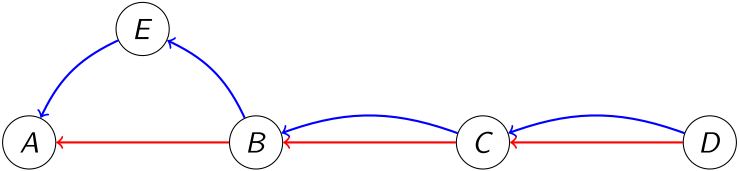
- broadcast communication
- importance of the topology of the network
- number of agents involved unknown

Example: simplified DSR



Request phase: A wants to speak to D

Example: simplified DSR

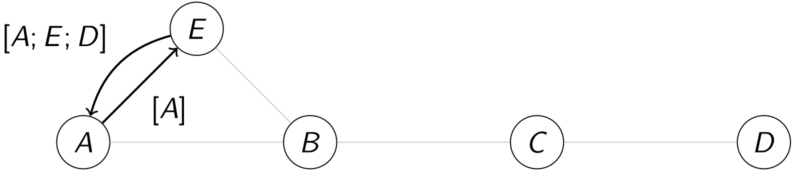


Reply phase: routes found are sent back to A

——— : route $[A; B; C; D]$

——— : route $[A; E; B; C; D]$

What if E was dishonest?



$\hookrightarrow$ A would get a false route to D !

$\Rightarrow$ Securing routing protocols

Insecure network: traditional description

Presence of an attacker

- may read every message sent on the net,
- may intercept and send new messages.

Results for cryptographic protocols

Attacks against protocols

Existence of an attack is decidable (for a bounded number of sessions).

Tools have been conceived that automatically detect logical flaws :
Proverif, AVISPA, Scyther...

but not applicable to **routing protocols** because of their specificities

Secure Routing Protocols

Goal: allowing distant nodes to communicate by finding a path between them while guaranteeing security

Notable differences with other cryptographic protocols:

- broadcast communication and topology of the network
- specific tests and security properties (e.g. route correctness)
- a form of recursivity

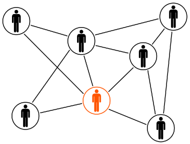
Power of the intruder

A Dolev-Yao intruder controls the network:

- hears all messages
- chooses which messages to transmit
- does not follow the protocol

Power of the intruder in an ad hoc setting:

- **broadcast** $\hookrightarrow$ cannot delete messages
- **located** $\hookrightarrow$ cannot hear distant messages



Our goals

Modeling and analysing secured routing protocols, taking into account :

- network topology
- less powerful intruder
- tests on the topology
- recursivity ?

Messages are abstracted by terms

Agents : $a, b, \ldots$

Concatenation : $\langle m_1, m_2 \rangle$

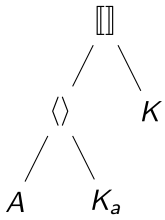
Encryption: $\{m\}_k$

Keys : $k_1, k_2, \ldots$

Lists: $[], a :: l$

Signature : $[\![m]\!]_k$

Example: The message $[\![\langle A, K_a \rangle]\!]_K$ is represented by:



Intuition: only the structure of the message is kept.

Intruder abilities

Composition rules

$$\frac{u_1 \quad u_2}{\langle u_1, u_2 \rangle} \qquad \frac{u_1 \quad u_2}{u_1 :: u_2} \qquad \frac{u_1 \quad \mathsf{sk}(u_2)}{[\![u_1]\!]_{\mathsf{sk}(u_2)}} \qquad \frac{u_1 \quad u_2}{\{u_1\}_{u_2}}$$

Intruder abilities

Composition rules

$$\frac{u_1 \quad u_2}{\langle u_1, u_2\rangle} \qquad \frac{u_1 \quad u_2}{u_1 :: u_2} \qquad \frac{u_1 \quad \mathrm{sk}(u_2)}{[\![u_1]\!]_{\mathrm{sk}(u_2)}} \qquad \frac{u_1 \quad u_2}{\{u_1\}_{u_2}}$$

Decomposition rules

$$\frac{\langle u_1, u_2\rangle}{u_i}\ i\in\{1,2\} \qquad \frac{u_1 :: u_2}{u_i}\ i\in\{1,2\} \qquad \frac{\{u_1\}_{u_2} \quad u_2}{u_1}$$

Optional rule: $\dfrac{[\![u_1]\!]_{\mathrm{sk}(u_2)}}{u_1}$

Intruder abilities

Composition rules

$$\frac{u_1 \quad u_2}{\langle u_1, u_2\rangle} \qquad \frac{u_1 \quad u_2}{u_1 :: u_2} \qquad \frac{u_1 \quad \mathrm{sk}(u_2)}{[\![u_1]\!]_{\mathrm{sk}(u_2)}} \qquad \frac{u_1 \quad u_2}{\{u_1\}_{u_2}}$$

Decomposition rules

$$\frac{\langle u_1, u_2\rangle}{u_i}\; i\in\{1,2\} \qquad \frac{u_1 :: u_2}{u_i}\; i\in\{1,2\} \qquad \frac{\{u_1\}_{u_2} \quad u_2}{u_1}$$

Optional rule: $\dfrac{[\![u_1]\!]_{\mathrm{sk}(u_2)}}{u_1}$

Deducibility relation

A term u is deducible from a set of terms T, denoted by $T \vdash u$, if there exists a prooftree witnessing this fact.

Calculus

Inspired from CBS$\#$, introduced by Nanz and Hankin

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

State: $\lfloor P \rfloor_n, \lfloor S \rfloor_n, \mathcal{I}$

Formulas

$\Phi ::=$	Formula
$\text{check}(a, b)$	a and b are neighbors
$\text{checkl}(c, l)$	l is locally correct for c
$\text{loop}(l)$	existence of a loop in a list
$\text{route}(l)$	validity of a route
$\Phi_1 \wedge \Phi_2$	conjunction
$\Phi_1 \vee \Phi_2$	disjunction
$\neg\Phi$	negation

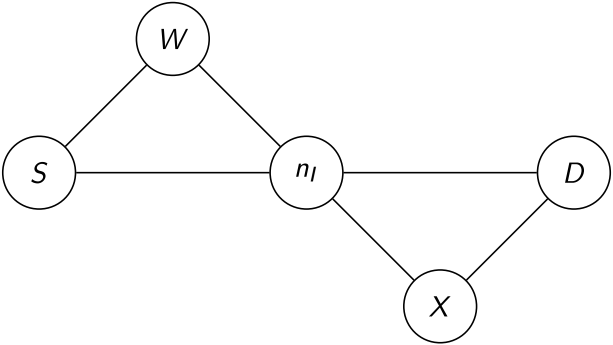
Property and tests expressed in $\mathcal{L}$

Expressiveness of the model

Concerning the specificities of routing protocols :

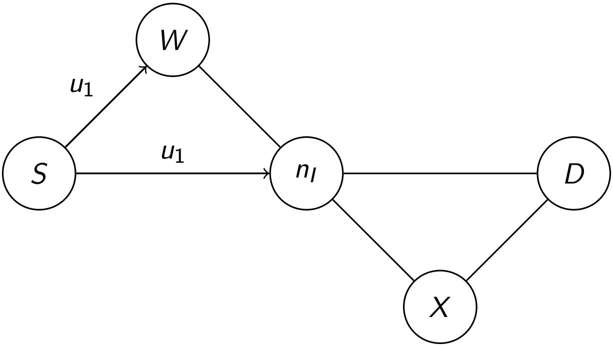
- list as a data structure
- broadcast communication and network topology
- specific tests and security properties

Example: source node



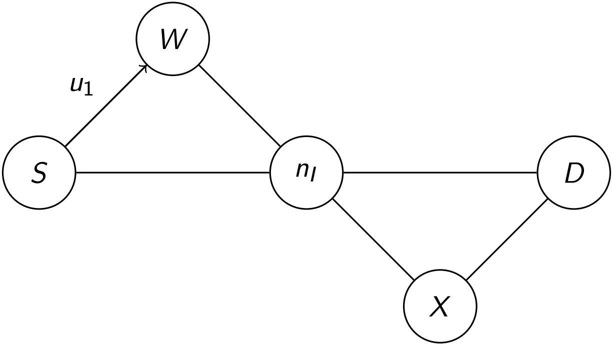
S : out(u_1).in $u_2[\Phi_S]$

Example: source node



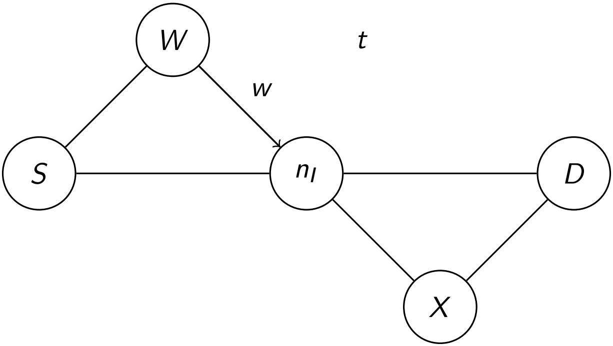
$$S : \text{out}(u_1).\text{in } u_2[\Phi_S] \rightarrow \text{in } u_2[\Phi_S]$$

Example: intermediate node



W : in $w_1[\Phi_W]$.store(t).out(w_2).0

Example: intermediate node



$$\sigma = \text{mgu}(u_1, w_1) \quad \text{and} \quad w = w_2\sigma$$

Concrete transitions

Problem with concrete transitions: infinitely many possibilities

$$\begin{array}{lcll} \lfloor \text{in } x.\text{out}(x) \rfloor_n & \rightarrow & \lfloor \text{out}(t_1) \rfloor_n & \text{if } \mathcal{I} \vdash t_1 \\ & \rightarrow & \lfloor \text{out}(t_2) \rfloor_n & \text{if } \mathcal{I} \vdash t_2 \\ & \rightarrow & \lfloor \text{out}(t_3) \rfloor_n & \text{if } \mathcal{I} \vdash t_3 \\ & & \vdots & \vdots \end{array}$$

$\hookrightarrow$ Introduction of symbolic transitions to avoid state explosion by keeping some variables

Symbolic transition: Example

Concrete transition	Symbolic transition
$\lfloor \mathsf{in}\ u[\Phi].P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}$	$\lfloor \mathsf{in}\ u[\Phi].P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}; \mathcal{C}$
$\rightarrow$	$\rightarrow_s$
$\lfloor P\sigma \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I}$	$\lfloor P \rfloor_n \cup \mathcal{P}; \mathcal{S}; \mathcal{I};$
if $\mathcal{I} \vdash t$, $[\![\Phi\sigma]\!] = 1$	$\mathcal{C} \cup \{\mathcal{I} \Vdash u;\ \Phi\}$
$(n_I, n) \in E$,	if $(n_I, n) \in E$
and $\sigma = \mathrm{mgu}(t, u)$	

All possible concrete transitions are captured in one symbolic transition

Secrecy via constraint solving [Millen et al]

Constraint systems are used to specify secrecy preservation under a particular, finite scenario.

Scenario

$$\begin{array}{ccc} \mathrm{rcv}(u_1) & \xrightarrow{N_1} & \mathrm{snd}(v_1) \\ \mathrm{rcv}(u_2) & \xrightarrow{N_2} & \mathrm{snd}(v_2) \\ & \cdots & \\ \mathrm{rcv}(u_n) & \xrightarrow{N_n} & \mathrm{snd}(v_n) \end{array}$$

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} u_1 \\ T_0, v_1 \overset{?}{\vdash} u_2 \\ \quad \cdots \\ T_0, v_1, .., v_n \overset{?}{\vdash} s \end{cases}$$

where T_0 is the initial knowledge of the attacker.

Remark: Constraint Systems may be used more generally for trace-based properties, e.g. authentication.

Secrecy via constraint solving [Millen et al]

Constraint systems are used to specify secrecy preservation under a particular, finite scenario.

Scenario

$$\begin{array}{ccc} \mathrm{rcv}(u_1) & \xrightarrow{N_1} & \mathrm{snd}(v_1) \\ \mathrm{rcv}(u_2) & \xrightarrow{N_2} & \mathrm{snd}(v_2) \\ & \dots & \\ \mathrm{rcv}(u_n) & \xrightarrow{N_n} & \mathrm{snd}(v_n) \end{array}$$

Constraint System

$$\mathcal{C} = \begin{cases} T_0 \overset{?}{\vdash} u_1 \\ T_0, v_1 \overset{?}{\vdash} u_2 \\ \quad \dots \\ T_0, v_1, .., v_n \overset{?}{\vdash} s \end{cases}$$

Solution of a constraint system

A substitution σ such that for every $T \Vdash u \in \mathcal{C}$, $u\sigma$ is deducible from $T\sigma$, that is $T\sigma \vdash u\sigma$.

How do we show decidability ?

Step 1. Simplifying the constraint system
$\longrightarrow$ common step to all our results

Step 2. Bounding solutions
$\longrightarrow$ specific techniques for

1. routing protocols with topology tests
2. protocols with recursivity

Goal of the simplification: obtain simpler constraint systems

Definition (Solved constraint systems)

Solved constraint systems are of the form
$\mathcal{C} = T_1 \overset{?}{\vdash} x_1 \wedge \cdots \wedge T_n \overset{?}{\vdash} x_n$, where the x_i are variables.

We show that we can simplify constraint system and obtain solutions to solved constraint systems.

Simplification rules

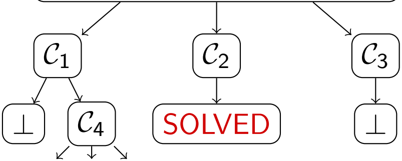
$$\mathrm{R}_{\mathrm{ax}}: \quad \mathcal{C} \wedge T \Vdash u \quad \rightsquigarrow \quad \mathcal{C}$$
$$\text{if } T \cup \{x \mid T' \Vdash x \in \mathcal{C}, T' \subsetneq T\} \vdash u$$

$$\mathrm{R}_{\mathrm{unif}}: \quad \mathcal{C} \wedge T \Vdash u \quad \rightsquigarrow_{\sigma} \quad \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma$$
$$\text{if } \sigma = \mathrm{mgu}(t_1, t_2) \text{ where } t_1, t_2 \in st(T, u), \text{ and } t_1 \neq t_2$$

$$\mathrm{R}_{\mathrm{fail}}: \quad \mathcal{C} \wedge T \Vdash u \quad \rightsquigarrow \quad \perp$$
$$\text{if } vars(T, u) = \emptyset \text{ and } T \nvdash u$$

$$\mathrm{R}_{\mathrm{f}}: \quad \mathcal{C} \wedge T \Vdash \mathrm{f}(u, v) \quad \rightsquigarrow \quad \mathcal{C} \wedge T \Vdash u \wedge T \Vdash v$$

$$\mathcal{C} = \begin{cases} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \quad \ldots \\ T_0, v_1, .., v_n \Vdash u_{n+1} \end{cases}$$



Theorem

$\mathcal{C}$ has a solution iff $\mathcal{C} \rightsquigarrow^*_\sigma \mathcal{C}'$ with $\mathcal{C}'$ in solved form.

New characterization property

if $T_i\theta \vdash u$, we have that $T_i\theta \vdash u$ using composition rules.

$T_1, \ldots, T_n$ represent a basis for deducible terms

Results for Routing protocols

We show decidability in two cases:

- decidability of an attack for a given topology
- existence of a topology leading to an attack

Fixed topology

Theorem

Let P be a protocol without replication and Φ a property. Deciding whether there is an attack on P and Φ for a given topology is NP-complete.

1. guess a path of symbolic execution
2. reduce to solved constraint systems (extension of the approach of Millen, Shmatikov and Comon-Lundh with an infinity of names and extended signature)
3. decide existence of a solution

Existence of a topology leading to an attack

Theorem

Let P be a protocol and Φ a property. Deciding whether there exists a topology such that there is an attack on P and Φ is NP-complete.

- Guess the edges between the nodes appearing in the protocol
- As in the case of a fixed topology, reduce the problem to solving a constraint system in solved form
- Bounding the size of the solution (in the size of the initial configuration) allows to bound the size of the graph

Protocols with recursivity tests

Aim: Analysing protocols that involve iterative or recursive operations.

- group protocols
- certification paths for public keys
- delegation rights
- secured source routing protocols

Example 1: Certificate Chains

Public keys need to be **certified**

Example (X.509 public key certificates)

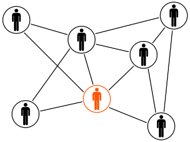
$$[[\![\langle A_1, \mathrm{pub}(A_1)\rangle]\!]_{\mathrm{sk}(A_2)}; [\![\langle A_2, \mathrm{pub}(A_2)\rangle]\!]_{\mathrm{sk}(A_3)}; \ldots$$
$$\ldots; [\![\langle A_{n-1}, \mathrm{pub}(A_{n-1})\rangle]\!]_{\mathrm{sk}(A_n)}; [\![\langle A_n, \mathrm{pub}(A_n)\rangle]\!]_{\mathrm{sk}(S)}]$$

where

- S is some trusted server, and
- each agent A_{i+1} certifies the public key $\mathrm{pub}(A_i)$ of agent A_i.

Example 2: Secured Source Routing

To certify the route, each node signs the fact that it belongs to it.



Example (SMNDP)

The route is represented by $l_{route} = [A_n; \ldots; A_1]$. The expected message is of the form

$$[[\![\langle A_n, A_0, l_{route}\rangle]\!]_{\mathsf{sk}(A_1)}; [\![\langle A_n, A_0, l_{route}\rangle]\!]_{\mathsf{sk}(A_2)}; \ldots \\ \ldots; [\![\langle A_n, A_0, l_{route}\rangle]\!]_{\mathsf{sk}(A_n)}].$$

Remark: $[\![\langle A_n, A_0, l_{route}\rangle]\!]_{\mathsf{sk}(A_i)}$ both depends on the list l_{route} and on its i-th element.

Our decidability results

We prove decidability for constraint systems with recursive tests:

- for link-based recursive languages $\mathcal{L}_{link}$

Example (Certificate chains)

$$\mathcal{L}_1 = \{[[\![\langle A_1, \mathrm{pub}(A_1)\rangle]\!]_{\mathrm{sk}(A_2)}; [\![\langle A_2, \mathrm{pub}(A_2)\rangle]\!]_{\mathrm{sk}(A_3)}; \ldots \\ \ldots; [\![\langle A_n, \mathrm{pub}(A_n)\rangle]\!]_{\mathrm{sk}(S)}] \mid A_1, \ldots, A_n \text{ agent names}, n \in \mathbb{N}\}$$

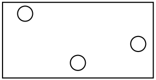
- for mapping-based languages $\mathcal{L}_{mapping}$

Example (SMNDP)

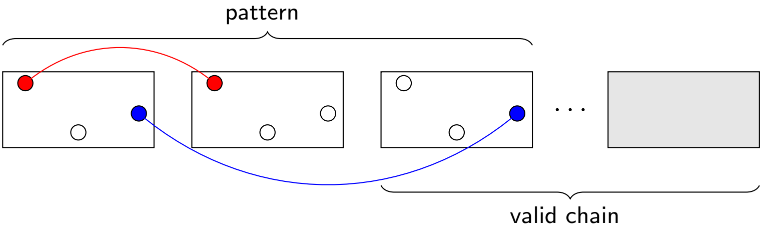
$$\mathcal{L}_2 = [[\![\langle A_n, A_0, l_{route}\rangle]\!]_{\mathrm{sk}(A_1)}; [\![\langle A_n, A_0, l_{route}\rangle]\!]_{\mathrm{sk}(A_2)}; \ldots \\ \ldots; [\![\langle A_n, A_0, l_{route}\rangle]\!]_{\mathrm{sk}(A_n)}] \mid l_{route} = [A_n, \ldots, A_1], n \in \mathbb{N}\}$$

Case of link-based recursive language

Links are terms containing variables that can be instantiated by basic terms, *e.g.* names.



Chains in such a language are lists of links recursively constrained:



Encoding the example

Certificate lists are all built from the term $\mathrm{m} = [\![\langle x, \mathrm{pub}(y)\rangle]\!]_{\mathrm{sk}(z)}$

$$[\![\langle x, \mathrm{pub}(x)\rangle]\!]_{\mathrm{sk}(y)} :: [\![\langle y, \mathrm{pub}(y)\rangle]\!]_{\mathrm{sk}(z)} :: \ldots [[\![\langle w, \mathrm{pub}(w)\rangle]\!]_{\mathrm{sk}(S)}]$$

The basic certificate chain is of the form $[[\![\langle w, \mathrm{pub}(w)\rangle]\!]_{\mathrm{sk}(S)}]$

Theorem

Let $\mathcal{L}$ be a link-based recursive language. Let $\mathcal{C}$ be a constraint system and ϕ be a conjunction of $\mathcal{L}$-language constraints. Deciding whether $\mathcal{C}$ and ϕ has a solution is in NP.

Intuitively, bounding the solution is done by limiting the number of possible links

Case of mapping-based languages

From a base shape b and a list of names $\ell = [a_0; \ldots, a_n]$, the following terms can be built:

- $b_0 = \mathsf{b}[a_0, \perp]$
- $b_1 = \mathsf{b}[a_1, [b_0]]$
- $b_2 = \mathsf{b}[a_2, [b_1; b_0]]$

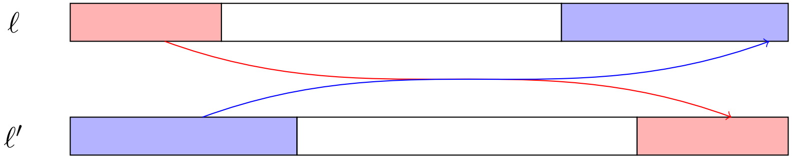
  $\vdots$
- $b_n = \mathsf{b}[a_n, [b_{n-1}; \ldots; b_0]]$

$(\ell, \ell') \in \mathcal{L}$ if and only if $\ell' = [b_n; b_{n-1} \ldots; b_0]$

Theorem

Let $\mathcal{L}$ be a mapping-based recursive language. Let $\mathcal{C}$ be a constraint system and ϕ be a conjunction of $\mathcal{L}$-language constraints. Deciding whether $\mathcal{C} \wedge \phi$ has a solution is in NP.

Intuition to bound the lists: the beginning of $\ell = [a_0; \ldots; a_n]$ constrains the end of $\ell' = [b_n; \ldots; b_0]$ and reciprocally.



Idea: cut in the middle

Conclusion

- Model for ad hoc networks
- Decidability for routing protocols for fixed and for unknown topologies
- NP decision procedures for security protocols with recursive tests for two classes of tests

Future work:

- Full analysis of recursive routing protocols
- Implementation
- Anonymous routing ?