An introduction to formal symbolic models

for verifying security protocols

Stéphanie Delaune

Univ Rennes, CNRS, IRISA

Saturday, April 14th, 2018

Verifying security protocols: a difficult task

- testing their resilience against well-known attacks is not sufficient;
- manual security analysis is error-prone.

Verifying security protocols: a difficult task

- testing their resilience against well-known attacks is not sufficient;
- manual security analysis is error-prone.

Security

Defects in e-passports allow real-time tracking

This threat brought to you by RFID

The register - Jan. 2010

Lifestyle › Tech › News

Contactless card theft: Users warned to watch out for 'digital pickpockets'

Independent - Feb. 2016

A sucessful approach: formal symbolic verification

$\longrightarrow$ provides a rigorous framework and automatic tools to analyse security protocols and find their logical flaws.

A sucessful approach: formal symbolic verification

$\longrightarrow$ provides a rigorous framework and automatic tools to analyse security protocols and find their logical flaws.

TAMARIN — Tamarin prover interactive mode

ProVerif

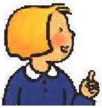
AVANTSSAR

Some examples of logical flaws:

- 2008: Authentication flaw in the Single Sign-On protocol used *e.g.* in GMail
  $\longrightarrow$ Armando *et al.* using Avantssar

- 2010: a flaw in the french implementation of the BAC protocol
  $\longrightarrow$ Chothia & Smirnov

Logical flaw on an example



$$\xrightarrow{\mathsf{aenc}(\mathsf{sign}(k_{AB}, \mathsf{prv}(A)), \mathsf{pub}(B))}$$

Is the Denning Sacco protocol a good key exchange protocol?

Logical flaw on an example

$$\xrightarrow{\mathsf{aenc}(\mathsf{sign}(k_{AB}, \mathsf{prv}(A)), \mathsf{pub}(B))}$$

Is the Denning Sacco protocol a good key exchange protocol? No !

Logical flaw on an example

$$\xrightarrow{\mathsf{aenc}(\mathsf{sign}(k_{AB}, \mathsf{prv}(A)), \mathsf{pub}(B))}$$

Is the Denning Sacco protocol a good key exchange protocol? No !

Description of a possible attack:

$$\xrightarrow{\mathsf{aenc}(\mathsf{sign}(k_{AC}, \mathsf{prv}(A)), \mathsf{pub}(C))}$$

Logical flaw on an example

$$\xrightarrow{\mathrm{aenc}(\mathrm{sign}(k_{AB}, \mathrm{prv}(A)), \mathrm{pub}(B))}$$

Is the Denning Sacco protocol a good key exchange protocol? No !

Description of a possible attack:

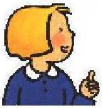
$$\xrightarrow{\mathrm{aenc}(\mathrm{sign}(k_{AC}, \mathrm{prv}(A)), \mathrm{pub}(C))}$$

$\mathrm{sign}(k_{AC}, \mathrm{prv}(A))$

k_{AC}

$$\xrightarrow{\mathrm{aenc}(\mathrm{sign}(k_{AC}, \mathrm{prv}(A)), \mathrm{pub}(B))}$$

Logical flaw on an example



$\mathrm{aenc}(\mathrm{sign}(k_{AB}, \mathrm{prv}(A)), \mathrm{pub}(B)) \longrightarrow$

Is the Denning Sacco protocol a good key exchange protocol? No !

Description of a possible attack:

$\mathrm{aenc}(\mathrm{sign}(k_{AC}, \mathrm{prv}(A)), \mathrm{pub}(C)) \longrightarrow$

$\mathrm{sign}(k_{AC}, \mathrm{prv}(A))$

k_{AC}

$\mathrm{aenc}(\mathrm{sign}(k_{AC}, \mathrm{prv}(A)), \mathrm{pub}(B)) \longrightarrow$

A possible fix: $\mathrm{aenc}(\mathrm{sign}(\langle B, k_{AB}\rangle, \mathrm{prv}(A)), \mathrm{pub}(B))$

Two major families of models ...

... with some advantages and some drawbacks.

Computational model

- $+$ messages are bitstring, a general and powerful adversary
- $-$ manual proofs, tedious and error-prone

Symbolic model

- $-$ abstract model, e.g. messages are terms
- $+$ automatic proofs

Two major families of models ...

... with some advantages and some drawbacks.

Computational model

- $+$ messages are bitstring, a general and powerful adversary
- $-$ manual proofs, tedious and error-prone

Symbolic model

- $-$ abstract model, e.g. messages are terms
- $+$ automatic proofs

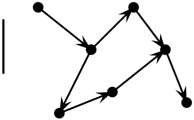
Some results allowed to make a link between these two very different models.

$\longrightarrow$ Abadi & Rogaway 2000

Formal (symbolic) verification in a nutshell

Does the protocol *satisfy* a security property?

Modelling

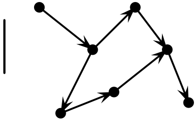


$\models \qquad \varphi$

Formal (symbolic) verification in a nutshell

Does the protocol *satisfy* a security property?

Modelling



$\models \varphi$

Two main tasks

1. Modelling protocols, security properties, and the attacker
2. Designing verification algorithms and tools

Part I

Modelling protocols, security properties and the attacker

Symbolic models in a nutshell

Some well-known existing models:

- strand spaces [Guttman *et al.*, 99],
- Multiset Rewriting [Durgin *et al.*, 99] - Tamarin tool
- spi-calculus [Abadi & Gordon, 97],
- applied-pi calculus [Abadi & Fournet, 01] - ProVerif tool

Symbolic models in a nutshell

Some well-known existing models:

- strand spaces [Guttman *et al.*, 99],
- Multiset Rewriting [Durgin *et al.*, 99] - Tamarin tool
- spi-calculus [Abadi & Gordon, 97],
- applied-pi calculus [Abadi & Fournet, 01] - ProVerif tool

They share some common ingredients:

- messages are abstracted by terms (perfect cryptography)
- the Dolev-Yao attacker who controls the entire network
- language with constructs for concurrency and communication

Messages as first-order terms

Terms are built over a set of names $\mathcal{N}$, and a signature $\mathcal{F}$.

$$\begin{array}{lcll} \mathrm{t} & ::= & n & \text{name } n \\ & | & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F} \end{array}$$

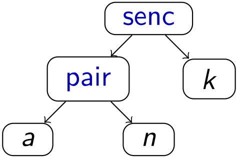
Messages as first-order terms

Terms are built over a set of names $\mathcal{N}$, and a signature $\mathcal{F}$.

$$
\begin{array}{lcll}
\mathrm{t} & ::= & n & \text{name } n \\
 & \mid & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F}
\end{array}
$$

Example: representation of $\{a, n\}_k$

- Names: n, k, a
- constructors: senc, pair,



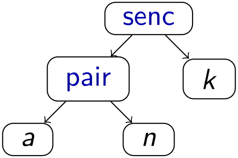
Messages as first-order terms

Terms are built over a set of names $\mathcal{N}$, and a signature $\mathcal{F}$.

$$
\begin{array}{lcll}
\mathrm{t} & ::= & n & \text{name } n \\
& | & f(t_1, \ldots, t_k) & \text{application of symbol } f \in \mathcal{F}
\end{array}
$$

Example: representation of $\{a, n\}_k$

- Names: n, k, a
- constructors: senc, pair,
- destructors: sdec, proj_1, proj_2.



The term algebra is equipped with an equational theory E.

$$
\mathrm{sdec}(\mathrm{senc}(x, y), y) = x \qquad
\begin{array}{rcl}
\mathrm{proj}_1(\mathrm{pair}(x, y)) & = & x \\
\mathrm{proj}_2(\mathrm{pair}(x, y)) & = & y
\end{array}
$$

Example: $\mathrm{proj}_1(\mathrm{sdec}(\mathrm{senc}(\langle a, n \rangle, k), k)) =_{\mathrm{E}} a$.

Protocols as processes

$\longrightarrow$ the applied pi calculus [Abadi & Fournet, 2001]

P, Q	$:=$	0	null process
		$\mathrm{in}(c, x).P$	input
		$\mathrm{out}(c, u).P$	output
		$\text{if } u = v \text{ then } P \text{ else } Q$	conditional
		$P \mid Q$	parallel composition
		$!P$	replication
		$\text{new } n.P$	fresh name generation

Protocols as processes

$\longrightarrow$ the applied pi calculus [Abadi & Fournet, 2001]

P, Q	$:=$	0	null process
		$\mathsf{in}(c, x).P$	input
		$\mathsf{out}(c, u).P$	output
		$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$	conditional
		$P \mid Q$	parallel composition
		$!P$	replication
		$\mathsf{new}\ n.P$	fresh name generation

Semantics $\rightarrow$:

COMM	$\mathsf{out}(c, u).P \mid \mathsf{in}(c, x).Q \rightarrow P \mid Q\{u/x\}$
THEN	$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \rightarrow P$ when $u =_{\mathsf{E}} v$
ELSE	$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q \rightarrow Q$ when $u \neq_{\mathsf{E}} v$
REPL	$!P \rightarrow P \mid !P$

Going back to the Denning Sacco protocol (1/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{prv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What symbols and equations do we need to model this protocol?

Going back to the Denning Sacco protocol (1/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{prv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What symbols and equations do we need to model this protocol?

1. symmetric encryption: senc and sdec

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

Going back to the Denning Sacco protocol (1/3)

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{prv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

What symbols and equations do we need to model this protocol?

1. symmetric encryption: senc and sdec

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

2. asymmetric encryption: aenc, adec, and pk

$$\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$$

Going back to the Denning Sacco protocol (1/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{prv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

What symbols and equations do we need to model this protocol?

1. symmetric encryption: senc and sdec

$$\mathsf{sdec}(\mathsf{senc}(x, y), y) = x$$

2. asymmetric encryption: aenc, adec, and pk

$$\mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$$

3. signature: ok, sign, check, getmsg, and pk

$$\mathsf{check}(\mathsf{sign}(x, y), \mathsf{pk}(y)) = \mathsf{ok} \text{ and } \mathsf{getmsg}(\mathsf{sign}(x, y)) = x$$

Going back to the Denning Sacco protocol (2/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{prv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

Going back to the Denning Sacco protocol (2/3)

$$\begin{array}{rcl} A \rightarrow B & : & \mathrm{aenc}(\mathrm{sign}(k, \mathrm{prv}(A)), \mathrm{pub}(B)) \\ B \rightarrow A & : & \mathrm{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{rcl} P_A(sk_a, pk_b) & = & \mathrm{new}\ k. \\ & & \mathrm{out}(c, \mathrm{aenc}(\mathrm{sign}(k, sk_a), pk_b)). \\ & & \mathrm{in}(c, x_a).\ \ldots \end{array}$$

Going back to the Denning Sacco protocol (2/3)

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{prv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{lcl} P_A(sk_a, pk_b) & = & \mathsf{new}\ k. \\ & & \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \mathsf{in}(c, x_a).\ \ldots \end{array}$$

$$\begin{array}{lcl} P_B(sk_b, pk_a) & = & \mathsf{in}(c, x_b). \\ & & \quad \mathsf{if}\ \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok}\ \mathsf{then} \\ & & \qquad\qquad \mathsf{new}\ s. \\ & & \qquad\qquad \mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b)))) \end{array}$$

Going back to the Denning Sacco protocol (3/3)

$P_A(sk_a, pk_b) =$
new k.
$\text{out}(c, \text{aenc}(\text{sign}(k, sk_a), pk_b))$.
$\text{in}(c, x_a)$. ...

$P_B(sk_b, pk_a) =$
$\text{in}(c, x_b)$.
if $\text{check}(\text{adec}(x_b, sk_b), pk_a) = \text{ok}$ then
new s.
$\text{out}(c, \text{senc}(s, \text{getmsg}(\text{adec}(x_b, sk_b))))$

Going back to the Denning Sacco protocol (3/3)

$P_A(sk_a, pk_b) =$
$\text{new } k.$
$\text{out}(c, \text{aenc}(\text{sign}(k, sk_a), pk_b)).$
$\text{in}(c, x_a).\ \ldots$

$P_B(sk_b, pk_a) =$
$\text{in}(c, x_b).$
$\text{if check}(\text{adec}(x_b, sk_b), pk_a) = \text{ok then}$
$\quad \text{new } s.$
$\quad \text{out}(c, \text{senc}(s, \text{getmsg}(\text{adec}(x_b, sk_b))))$

Example: a simple scenario

$$P_{\text{DS}} = \text{new } sk_a, sk_b.(P_A(sk_a, \text{pk}(sk_b)) \mid P_B(sk_b, \text{pk}(sk_a))$$

Going back to the Denning Sacco protocol (3/3)

$P_A(sk_a, pk_b) \;=$
$\mathsf{new}\ k.$
$\mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)).$
$\mathsf{in}(c, x_a).\ \ldots$

$P_B(sk_b, pk_a) \;=$
$\mathsf{in}(c, x_b).$
$\mathsf{if}\ \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) = \mathsf{ok}\ \mathsf{then}$
$\quad \mathsf{new}\ s.$
$\quad \mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(x_b, sk_b))))$

Example: a simple scenario

$$P_{\mathsf{DS}} = \mathsf{new}\ sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\big)$$

$$\xrightarrow{(\textsc{Comm})} \mathsf{new}\ sk_a, sk_b, k.(\ \mathsf{in}(c, x_a).\ \ldots$$

$$\mid\ \mathsf{if}\ \mathsf{check}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b), pk_a) = \mathsf{ok}\ \mathsf{then}$$

$$\mathsf{new}\ s.\mathsf{out}(c, \mathsf{senc}(s, \mathsf{getmsg}(\mathsf{adec}(\mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b), sk_b)))))$$

Going back to the Denning Sacco protocol (3/3)

$P_A(sk_a, pk_b) \;=$
$\text{new } k.$
$\text{out}(c, \text{aenc}(\text{sign}(k, sk_a), pk_b)).$
$\text{in}(c, x_a).\ \ldots$

$P_B(sk_b, pk_a) \;=$
$\text{in}(c, x_b).$
$\text{if check}(\text{adec}(x_b, sk_b), pk_a) = \text{ok then}$
$\quad \text{new } s.$
$\quad \text{out}(c, \text{senc}(s, \text{getmsg}(\text{adec}(x_b, sk_b))))$

Example: a simple scenario

$$P_{\mathrm{DS}} = \text{new } sk_a, sk_b.(P_A(sk_a, \text{pk}(sk_b)) \mid P_B(sk_b, \text{pk}(sk_a))$$

$$\xrightarrow{(\textsc{Comm})} \text{new } sk_a, sk_b, k.(\ \text{in}(c, x_a).\ \ldots$$
$$\mid\ \text{if check}(\text{adec}(\text{aenc}(\text{sign}(k, sk_a), pk_b), sk_b), pk_a) = \text{ok then}$$
$$\text{new } s.\text{out}(c, \text{senc}(s, \text{getmsg}(\text{adec}(\text{aenc}(\text{sign}(k, sk_a), pk_b), sk_b)))))$$

$$\xrightarrow{(\textsc{Then})} \text{new } sk_a, sk_b, k.(\ \text{in}(c, x_a).\ \ldots$$
$$\text{new } s.\text{out}(c, \text{senc}(s, \underline{\text{getmsg}(\text{adec}(\text{aenc}(\text{sign}(k, sk_a), pk_b), sk_b))})))$$

Going back to the Denning Sacco protocol (3/3)

$P_A(sk_a, pk_b) \ =$
new k.
$\mathrm{out}(c, \mathrm{aenc}(\mathrm{sign}(k, sk_a), pk_b))$.
$\mathrm{in}(c, x_a)$. ...

$P_B(sk_b, pk_a) \ =$
$\mathrm{in}(c, x_b)$.
if $\mathrm{check}(\mathrm{adec}(x_b, sk_b), pk_a) = \mathrm{ok}$ then
new s.
$\mathrm{out}(c, \mathrm{senc}(s, \mathrm{getmsg}(\mathrm{adec}(x_b, sk_b))))$

Example: a simple scenario

$$P_{\mathrm{DS}} \ = \text{new } sk_a, sk_b.(P_A(sk_a, \mathrm{pk}(sk_b)) \mid P_B(sk_b, \mathrm{pk}(sk_a))$$

$$\xrightarrow{(\textsc{Comm})} \text{new } sk_a, sk_b, k.(\ \mathrm{in}(c, x_a).\ \ldots$$
$$\mid\ \text{if } \mathrm{check}(\mathrm{adec}(\mathrm{aenc}(\mathrm{sign}(k, sk_a), pk_b), sk_b), pk_a) = \mathrm{ok} \text{ then}$$
$$\text{new } s.\mathrm{out}(c, \mathrm{senc}(s, \mathrm{getmsg}(\mathrm{adec}(\mathrm{aenc}(\mathrm{sign}(k, sk_a), pk_b), sk_b)))))$$

$$\xrightarrow{(\textsc{Then})} \text{new } sk_a, sk_b, k.(\ \mathrm{in}(c, x_a).\ \ldots$$
$$\text{new } s.\mathrm{out}(c, \mathrm{senc}(s, \underline{\mathrm{getmsg}(\mathrm{adec}(\mathrm{aenc}(\mathrm{sign}(k, sk_a), pk_b), sk_b))})))$$

this represents a normal execution between two honest participants

Trace-based security properties

Confidentiality (as non-deducibility)

For all processes A, for all execution $A \mid P \rightarrow^* Q$, we have that Q is not of the form $\mathtt{new}\,\tilde{n}.(\mathrm{out}(c, s).Q' \mid Q'')$ with c public.

Trace-based security properties

Confidentiality (as non-deducibility)

For all processes A, for all execution $A \mid P \rightarrow^{*} Q$, we have that Q is not of the form $\texttt{new}\,\tilde{n}.(\mathrm{out}(c, s).Q' \mid Q'')$ with c public.

Authentication (as a correspondence property)

1. add events of the form endB(...) or beginA(...) in processes
2. write a query:
   $\forall x_B, x_A, x_K.\mathrm{endB}(x_B, x_A, x_K) \Rightarrow \mathrm{beginA}(x_A, x_B, x_K)$.

For all processes A, for all execution $A \mid P \rightarrow^{*} Q$ that goes through the event $\mathrm{endB}(b, a, k)$, the event $\mathrm{beginA}(a, b, k)$ has been executed before.

Equivalence-based security properties

Vote privacy

the fact that a particular voter voted in a particular way is not revealed to anyone

$$V_A(yes) \mid V_B(no) \stackrel{?}{\approx} V_A(no) \mid V_B(yes)$$

Equivalence-based security properties

Vote privacy

the fact that a particular voter voted in a particular way is not revealed to anyone

$$V_A(\textit{yes}) \mid V_B(\textit{no}) \stackrel{?}{\approx} V_A(\textit{no}) \mid V_B(\textit{yes})$$

Unlinkability

the fact that a user may make multiple uses of a service or a resource without others being able to link these uses together.

$$!\,\mathrm{new}\,k.!\,P(k) \stackrel{?}{\approx} !\,\mathrm{new}\,k.P(k)$$

Equivalence-based security properties

Vote privacy

the fact that a particular voter voted in a particular way is not revealed to anyone

$$V_A(\textit{yes}) \mid V_B(\textit{no}) \overset{?}{\approx} V_A(\textit{no}) \mid V_B(\textit{yes})$$

Unlinkability

the fact that a user may make multiple uses of a service or a resource without others being able to link these uses together.

$$!\,\mathrm{new}\ k.!\ P(k) \ \overset{?}{\approx}\ !\,\mathrm{new}\ k.P(k)$$

Testing equivalence $P \approx Q$

$$P \approx Q \quad \text{iff} \quad (P \mid A){\Downarrow_c} \ \Leftrightarrow\ (Q \mid A){\Downarrow_c} \ \text{for any process } A$$

where $R \Downarrow_c$ means that R can evolve and emits on public channel c.

Part II

Designing verification algorithms and tools

State of the art in a nutshell

for analysing confidentiality/authentication properties

Unbounded number of sessions

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]
- decidable for restricted classes [Lowe, 99] [Rammanujam & Suresh, 03] [D'Osualdo *et al.*, 17]

$\longrightarrow$ tools: ProVerif, Tamarin, Maude-NPA, ...

State of the art in a nutshell

for analysing confidentiality/authentication properties

Unbounded number of sessions

- undecidable in general [Even & Goldreich, 83; Durgin *et al*, 99]
- decidable for restricted classes [Lowe, 99]
  [Rammanujam & Suresh, 03] [D'Osualdo *et al.*, 17]

$\longrightarrow$ tools: ProVerif, Tamarin, Maude-NPA, ...

Bounded number of sessions

- a decidability result (NP-complete)
  [Rusinowitch & Turuani, 01; Millen & Shmatikov, 01]

$\longrightarrow$ tools: AVANTSSAR platform, ...

ProVerif [Blanchet, 01]

ProVerif is a verifier for cryptographic protocols that may prove that a protocol is secure or exhibit attacks.

`http://proverif.inria.fr`

Advantages

- fully automatic, and quite efficient
- a rich process algebra: replication, else branches, ...
- handles many cryptographic primitives
- various security properties: secrecy, correspondences, equivalences

ProVerif [Blanchet, 01]

ProVerif is a verifier for cryptographic protocols that may prove that a protocol is secure or exhibit attacks.

`http://proverif.inria.fr`

Advantages

- fully automatic, and quite efficient
- a rich process algebra: replication, else branches, ...
- handles many cryptographic primitives
- various security properties: secrecy, correspondences, equivalences

No miracle

- the tool can say "can not be proved";
- termination is not guaranteed

ProVerif

ProVerif implements a resolution strategy well-adapted to protocols.

Approximation of the translation in Horn clauses:

- the freshness of nonces is partially modeled;
- the number of times a message appears is ignored, only the fact that is has appeared is taken into account;
- the state of the principals is not fully modeled.

$\longrightarrow$ These approximations are keys for an efficient verification.

Experimental results

$\longrightarrow$ ProVerif works well in practice.

Protocol	Result	ms
Needham-Schroeder shared key	Attack	52
Needham-Schroeder shared key corrected	Secure	109
Denning-Sacco	Attack	6
Denning-Sacco corrected	Secure	7
Otway-Rees	Secure	10
Otway-Rees, variant of Paulson98	Attack	12
Yahalom	Secure	10
Simpler Yahalom	Secure	11
Main mode of Skeme	Secure	23

Pentium III, 1 GHz.

Main limitations

Dolev-Yao attacker

As any participant, the attacker can intercept, build, and send messages without introducing any delay.
$\longrightarrow$ not suitable to analyse distance bounding protocols

We need a model that takes into account:

- the fact that transmitting a message takes time,
- the location of participants.

How existing symbolic models/tools can be extended/adapted to analyse distance bounding protocols?

$\longrightarrow$ see talks given by T. Chothia, J. Toro-Pozo, and A. Debant

Handling low-level operators

Distance bounding protocols often rely on some low-level operators.

Single bit message: Symbolic models do not allow one to reason at this level.

$\longrightarrow$ this is a problem to model rapid phases in distance bounding.

Algebraic properties of low level operators: A faithful model need to take into account the algebraic properties of those operators:

Example: exclusive-or operator

$$\begin{array}{rclcrcl} (x \oplus y) \oplus z & = & x \oplus (y \oplus z) & \quad & x \oplus 0 & = & x \\ x \oplus y & = & y \oplus x & \quad & x \oplus x & = & 0 \end{array}$$

$\longrightarrow$ those operators are only partially supported in existing verification tools.

Towards probabilistic models

Existing symbolic verification tools do not allow one to model probabilistic behaviours.

the protocol is declared unsecure as soon as there is a behaviour of the attacker that allows one to reach a bad state.

Towards probabilistic models

Existing symbolic verification tools do not allow one to model probabilistic behaviours.

the protocol is declared unsecure as soon as there is a behaviour of the attacker that allows one to reach a bad state.

To say that a bad state is reachable with probability at most p, we need to introduce probability in our modelling
$\longrightarrow$ *e.g.* partially observable Markov decision processes

Some recent works by R. Chadha et al.

- Verification of randomized security protocols — LICS, 2017
- Modular Verification of Protocol Equivalence in the Presence of Randomness — ESORICS, 2017

Privacy-type properties

In comparison to trace-based security properties

- a more recent research area
- more difficult to analyse (we have to compare sets of traces).

Privacy-type properties

In comparison to trace-based security properties

- a more recent research area
- more difficult to analyse (we have to compare sets of traces).

State-of-the art for traditional protocols

- ProVerif (and Tamarin) consider a strong form of equivalence, namely diff-equivalence.
  $\longrightarrow$ not suitable to analyse *e.g.* unlinkability of the BAC protocol.
- Verification tools for a bounded number of sessions suffer from the well-known state explosion problem
  $\longrightarrow$ only able to analyse very few sessions of the protocol, e.g. 2 or 3 processes in parallel.

Privacy-type properties

In comparison to trace-based security properties

- a more recent research area
- more difficult to analyse (we have to compare sets of traces).

State-of-the art for traditional protocols

- ProVerif (and Tamarin) consider a strong form of equivalence, namely diff-equivalence.
  $\longrightarrow$ not suitable to analyse *e.g.* unlinkability of the BAC protocol.
- Verification tools for a bounded number of sessions suffer from the well-known state explosion problem
  $\longrightarrow$ only able to analyse very few sessions of the protocol, e.g. 2 or 3 processes in parallel.

Open challenge: extending existing verification tools to be able to analyse privacy-type properties on distance bounding protocols.

POPSTAR in a nutshell (2017-2022)

Reasoning about **Physical properties** Of **security Protocols** with an Application To **contactless Systems**

Main issues:

- specificities of contactless systems are not well understood;
- a lack of formal model to reason about these systems.

Main outcomes:

- solid foundations to reason about physical properties;
- new algorithms and tools to analyse the security and privacy of modern protocols;
- make the upcoming generation of nomadic contactless devices more secure.

POPSTAR in a nutshell (2017-2022)

Reasoning about **Physical properties** Of
security Protocols
with an Application To **contactless Systems**

`https://project.inria.fr/popstar/`

Advertisement - Regular job offers:

- PhD positions and Post-doc positions;
- One research associate position (up to 3 years).

$\longrightarrow$ contact me: `stephanie.delaune@irisa.fr`