How does ProVerif work?

Véronique Cortier

How to analyse security protocols ?

$\models^{?}$

confidentiality
authenticity
non-repudiation

Methodology

1. Proposing accurate models
   - symbolic models
   - cryptographic/computational models
2. Proving security
   - decidability/undecidability results
   - tools

Difficulty

Presence of an attacker

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

$\Rightarrow$ The system is infinitely branching

How to decide security for unlimited sessions ?

$\rightarrow$ In general, it is undecidable !

(i.e. there exists no algorithm for checking e.g. secrecy)

How to prove undecidability ?

How to decide security for unlimited sessions ?

$\rightarrow$ In general, it is undecidable !

(i.e. there exists no algorithm for checking e.g. secrecy)

How to prove undecidability ?

Post correspondence problem (PCP)

input $\{(u_i, v_i)\}_{1 \leq i \leq n}$, $u_i, v_i \in \Sigma^*$

output $\exists n, i_1, \ldots, i_n \quad u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$

Example : $\{(bab, b), (ab, aba), (a, baba)\}$

Solution ?

How to decide security for unlimited sessions ?

$\rightarrow$ In general, it is undecidable !

(i.e. there exists no algorithm for checking e.g. secrecy)

How to prove undecidability ?

Post correspondence problem (PCP)

input $\{(u_i, v_i)\}_{1 \leq i \leq n}$, $u_i, v_i \in \Sigma^*$

output $\exists n, i_1, \ldots, i_n \quad u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}$

Example : $\{(bab, b), (ab, aba), (a, baba)\}$

Solution ? $\rightarrow$ Yes, 1,2,3,1.

$bababab ab$

$bababab ab$

How to circumvent undecidability ?

- Find decidable subclasses of protocols.
- Design semi-decision procedure, that works in practice
- ...

How to model an unbounded number of sessions ?

"*For any* x, *if the agent* A *receives* $\mathrm{enc}(x, k_a)$ *then* A *responds with* x."

$\rightarrow$ Use of first-order logic.

Intruder

Horn clauses perfectly reflects the attacker symbolic manipulations on terms.

$$
\begin{array}{llcll}
\forall x \forall y & I(x), I(y) & \Rightarrow & I(\{x\}_y) & \text{encryption} \\
\forall x \forall y & I(\{x\}_y), I(y) & \Rightarrow & I(x) & \text{decryption} \\
\forall x \forall y & I(x), I(y) & \Rightarrow & I(< x, y >) & \text{concatenation} \\
\forall x \forall y & I(< x, y >) & \Rightarrow & I(x) & \text{first projection} \\
\forall x \forall y & I(< x, y >) & \Rightarrow & I(y) & \text{second projection}
\end{array}
$$

Protocol as Horn clauses

secret : 3443

$$\xrightarrow{\{\text{secret} : 3443\}_{k_{\text{alice}}}}$$

$$\xleftarrow{\left\{\{\text{secret} : 3443\}_{k_{\text{alice}}}\right\}_{k_{\text{bob}}}}$$

$$\xrightarrow{\{\text{secret} : 3443\}_{k_{\text{bob}}}}$$

secret : 3443

Each action of the protocol is expressed by a logical implication.

$$\begin{array}{lrcl} & & \Rightarrow & I(\{\text{secret}\}_{k_a}) \\ \forall x & I(x) & \Rightarrow & I(\{x\}_{k_b}) \\ \forall x & I(\{x\}_{k_a}) & \Rightarrow & I(x) \end{array}$$

Security reduces to consistency

secure ?

$$
\begin{array}{lrcl}
\forall x \forall y & I(x), I(y) & \Rightarrow & I(< x, y >) \\
\forall x \forall y & I(x), I(y) & \Rightarrow & I(\{x\}_y) \\
\forall x \forall y & I(\{x\}_y), I(y) & \Rightarrow & I(x) \\
\forall x \forall y & I(< x, y >) & \Rightarrow & I(x) \\
\forall x \forall y & I(< x, y >) & \Rightarrow & I(y)
\end{array}
$$

$$
\begin{array}{lrcl}
 & & & I(\{\text{secret}\}_{k_a}) \\
\forall x & I(x) & \Rightarrow & I(\{x\}_{k_b}) \\
\forall x & I(\{x\}_{k_a}) & \Rightarrow & I(x)
\end{array}
$$

Security reduces to consistency

secure ?

$$
\begin{array}{llcl}
 & & & \neg I(\mathrm{secret}) \\
\forall x \forall y & I(x), I(y) & \Rightarrow & I(< x, y >) \\
\forall x \forall y & I(x), I(y) & \Rightarrow & I(\{x\}_y) \\
\forall x \forall y & I(\{x\}_y), I(y) & \Rightarrow & I(x) \\
\forall x \forall y & I(< x, y >) & \Rightarrow & I(x) \\
\forall x \forall y & I(< x, y >) & \Rightarrow & I(y)
\end{array}
$$

$$
\begin{array}{llcl}
 & & & I(\{\mathrm{secret}\}_{k_a}) \\
\forall x & I(x) & \Rightarrow & I(\{x\}_{k_b}) \\
\forall x & I(\{x\}_{k_a}) & \Rightarrow & I(x)
\end{array}
$$

Does not yield a contradiction ?

(i.e. consistent theory ?)

How to know if a set of formula is consistent ?

Hilbert's program (1928)
"Entscheidung Problem"

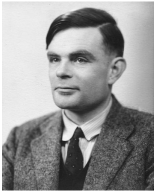
David Hilbert

How to know if a set of formula is consistent ?

Hilbert's program (1928)
"Entscheidung Problem"

David Hilbert

It is undecidable ! (1936)
→ There is no algorithm that answers this question.



Alan Turing

(at a time with no computers)

Back to our business

secure ?

$$\boxed{\text{All this for nothing ?}}$$

$$\neg I(\mathrm{secret})$$

$$
\begin{array}{llcl}
\forall x \forall y & I(x), I(y) & \Rightarrow & I(<x, y>) \\
\forall x \forall y & I(x), I(y) & \Rightarrow & I(\{x\}_y) \\
\forall x \forall y & I(\{x\}_y), I(y) & \Rightarrow & I(x) \\
\forall x \forall y & I(<x, y>) & \Rightarrow & I(x) \\
\forall x \forall y & I(<x, y>) & \Rightarrow & I(y) \\
 & & & I(\{\mathrm{secret}\}_{k_a}) \\
\forall x & I(x) & \Rightarrow & I(\{x\}_{k_b}) \\
\forall x & I(\{x\}_{k_a}) & \Rightarrow & I(x)
\end{array}
$$

Does not yield a contradiction ?

(i.e. consistent theory ?)

A standard technique : resolution

Idea : add logical consequences ...

$\forall x P(x) \Rightarrow I(s(x))$

$\forall x I(x) \Rightarrow P(s(x))$

$P(0)$

$\neg I(s(s(s(0))))$

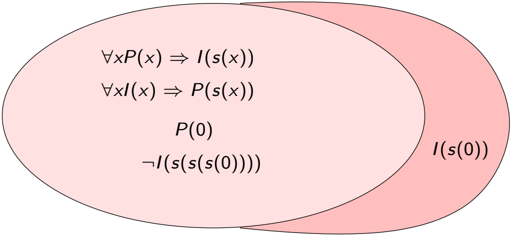
... until a contradiction is found.

We need a method (a strategy) which is :

- correct : adds formula that are indeed consequences
- complete : finds a contradiction (if it exists)
- in a finite number of steps (decidable fragment)

A standard technique : resolution

Idea : add logical consequences . . .



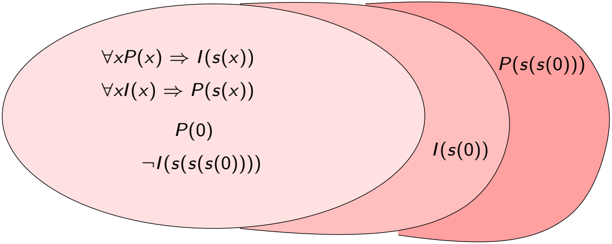
... until a contradiction is found.

We need a method (a strategy) which is :

- correct : adds formula that are indeed consequences
- complete : finds a contradiction (if it exists)
- in a finite number of steps (decidable fragment)

A standard technique : resolution

Idea : add logical consequences . . .



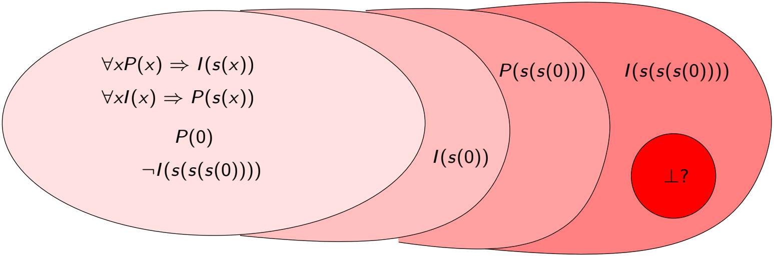
... until a contradiction is found.

We need a method (a strategy) which is :

- correct : adds formula that are indeed consequences
- complete : finds a contradiction (if it exists)
- in a finite number of steps (decidable fragment)

A standard technique : resolution

Idea : add logical consequences . . .



... until a contradiction is found.

We need a method (a strategy) which is :

- correct : adds formula that are indeed consequences
- complete : finds a contradiction (if it exists)
- in a finite number of steps (decidable fragment)

Binary resolution

A, B are atoms and C, D are clauses.

An intuitive rule
$$\frac{A \Rightarrow C \quad A}{C}$$

In other words
$$\frac{\neg A \vee C \quad A}{C}$$

Binary resolution

A, B are atoms and C, D are clauses.

An intuitive rule
$$\frac{A \Rightarrow C \quad A}{C}$$

In other words
$$\frac{\neg A \vee C \quad A}{C}$$

Generalizing
$$\frac{\neg A \vee C \quad B}{C\theta}\ \theta = mgu(A, B) \quad \text{(i.e. } A\theta = B\theta\text{)}$$

Binary resolution

A, B are atoms and C, D are clauses.

An intuitive rule

$$\frac{A \Rightarrow C \quad A}{C}$$

In other words

$$\frac{\neg A \vee C \quad A}{C}$$

Generalizing

$$\frac{\neg A \vee C \quad B}{C\theta} \; \theta = mgu(A, B) \quad \text{(i.e. } A\theta = B\theta\text{)}$$

Generalizing a bit more

$$\frac{\neg A \vee C \quad B \vee D}{C\theta \vee D\theta} \; \theta = mgu(A, B) \quad \text{Binary resolution}$$

Binary resolution and Factorization

$$\frac{\neg A \vee C \quad B \vee D}{C\theta \vee D\theta}\ \theta = \mathrm{mgu}(A, B) \qquad \text{Binary resolution}$$

$$\frac{A \vee B \vee C}{A\theta \vee C\theta}\ \theta = \mathrm{mgu}(A, B) \qquad \text{Factorisation}$$

Theorem (Soundness and Completeness)

Binary resolution and factorisation are sound and refutationally complete,
i.e. a set of clauses C is not satisfiable if and only if $\perp$ (the empty clause) can be obtained from C by binary resolution and factorisation.

Exercise : Why do we need the factorisation rule ?

Example

$$\mathcal{C} = \{\neg I(s), \quad I(k_1), \quad I(\{s\}_{\langle k_1, k_1\rangle}), \quad I(\{x\}_y), I(y) \Rightarrow I(x), \quad I(x), I(y) \Rightarrow I(\langle x, y\rangle)$$

$$\dfrac{\neg I(s) \qquad \dfrac{\dfrac{I(\{s\}_{\langle k_1,k_1\rangle}) \qquad I(\{x\}_y), I(y) \Rightarrow I(x)}{I(\langle k_1, k_1\rangle) \Rightarrow I(s)} \qquad \dfrac{I(k_1) \qquad \dfrac{I(k_1) \qquad I(x), I(y) \Rightarrow I(\langle x, y\rangle)}{I(y) \Rightarrow I(\langle k_1, y\rangle)}}{I(\langle k_1, k_1\rangle)}}{I(s)}}{\perp}$$

But it is not terminating !

$$
\dfrac{I(y) \Rightarrow I(\langle s, y\rangle) \qquad \dfrac{I(y) \Rightarrow I(\langle s, y\rangle) \qquad \dfrac{I(s) \qquad \dfrac{I(s) \quad I(x), I(y) \Rightarrow I(\langle x, y\rangle)}{I(y) \Rightarrow I(\langle s, y\rangle)}}{I(\langle s, s\rangle)}}{I(\langle s, \langle s, s\rangle\rangle)}}{\dfrac{I(\langle s, \langle s, \langle s, s\rangle\rangle\rangle)}{\ldots}}
$$

$\rightarrow$ This does not yield any decidability result.

Ordered Binary resolution and Factorization

Let $<$ be any order on clauses.

$$\frac{\neg A \vee C \quad B \vee D}{C\theta \vee D\theta} \quad \begin{array}{c}\theta = \mathrm{mgu}(A, B) \\ A\theta \not< C\theta \vee D\theta\end{array}$$

Ordered binary resolution

$$\frac{A \vee B \vee C}{A\theta \vee C\theta} \quad \begin{array}{c}\theta = \mathrm{mgu}(A, B) \\ A\theta \not< C\theta\end{array}$$

Ordered factorisation

Ordered Binary resolution and Factorization

Let $<$ be any order on clauses.

$$\frac{\neg A \vee C \quad B \vee D}{C\theta \vee D\theta} \quad \begin{array}{c}\theta = \mathrm{mgu}(A, B) \\ A\theta \not< C\theta \vee D\theta\end{array}$$

Ordered binary resolution

$$\frac{A \vee B \vee C}{A\theta \vee C\theta} \quad \begin{array}{c}\theta = \mathrm{mgu}(A, B) \\ A\theta \not< C\theta\end{array}$$

Ordered factorisation

Theorem (Soundness and Completeness)

Ordered binary resolution and factorisation are sound and refutationally complete provided that $<$ is liftable

$$\forall A, B, \theta \qquad A < B \Rightarrow A\theta < B\theta$$

Examples of liftable orders

$$\forall A, B, \theta \qquad A < B \Rightarrow A\theta < B\theta$$

First example : subterm order

$P(t_1, \ldots, t_n) < Q(u_1, \ldots, u_k)$ iff any t_i is a subterm of $u_1, \ldots, u_k$

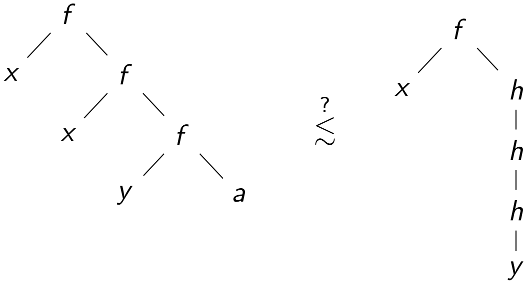
$\rightarrow$ extended to clauses as follows : $C_1 < C_2$ iff any literal of C_1 is smaller than some literal of C_2.

Exercise : Show that $\mathcal{C}$ is not satisfiable by ordered resolution (and factorisation).

Examples of liftable orders - continued

Second example : $P(t_1, \ldots, t_n) \lesssim Q(u_1, \ldots, u_k)$ iff

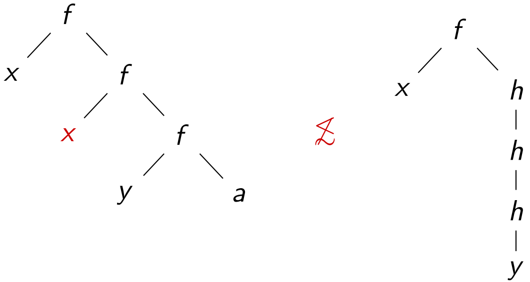
1. $\mathrm{depth}(P(t_1, \ldots, t_n)) \leq \mathrm{depth}(Q(u_1, \ldots, u_k))$
2. For any variable x,
   $\mathrm{depth}_x(P(t_1, \ldots, t_n)) \leq \mathrm{depth}_x(Q(u_1, \ldots, u_k))$



Examples of liftable orders - continued

Second example : $P(t_1, \ldots, t_n) \lesssim Q(u_1, \ldots, u_k)$ iff

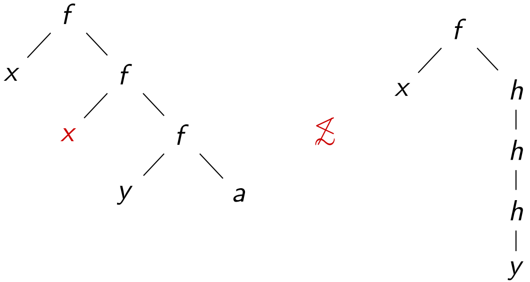
1. $\mathrm{depth}(P(t_1, \ldots, t_n)) \leq \mathrm{depth}(Q(u_1, \ldots, u_k))$
2. For any variable x,
   $\mathrm{depth}_x(P(t_1, \ldots, t_n)) \leq \mathrm{depth}_x(Q(u_1, \ldots, u_k))$



Examples of liftable orders - continued

Second example : $P(t_1, \ldots, t_n) \lesssim Q(u_1, \ldots, u_k)$ iff

1. $\text{depth}(P(t_1, \ldots, t_n)) \leq \text{depth}(Q(u_1, \ldots, u_k))$
2. For any variable x,
   $\text{depth}_x(P(t_1, \ldots, t_n)) \leq \text{depth}_x(Q(u_1, \ldots, u_k))$



Exercise : Show that $\forall A, B, \theta \qquad A \lesssim B \Rightarrow A\theta \lesssim B\theta$

Back to protocols

Intruder clauses are of the form

$$\pm I(f(x_1,\ldots,x_n)),\ \pm I(x_i),\ \pm I(x_j)$$

Protocol clauses

$$\begin{array}{rcl} & \Rightarrow & I(\{\mathrm{pin}\}_{k_a}) \\ I(x) & \Rightarrow & I(\{x\}_{k_b}) \\ I(\{x\}_{k_a}) & \Rightarrow & I(x) \end{array}$$

At most one variable per clause !

Back to protocols

Intruder clauses are of the form

$$\pm I(f(x_1, \dots, x_n)),\ \pm I(x_i),\ \pm I(x_j)$$

Protocol clauses

$$\begin{aligned} &\Rightarrow\ I(\{\text{pin}\}_{k_a}) \\ I(x) &\Rightarrow\ I(\{x\}_{k_b}) \\ I(\{x\}_{k_a}) &\Rightarrow\ I(x) \end{aligned}$$

At most one variable per clause !

Theorem

Given a set $\mathcal{C}$ of clauses such that each clause of $\mathcal{C}$

- *either contains at most one variable*
- *or is of the form $\pm I(f(x_1, \dots, x_n)),\ \pm I(x_i),\ \pm I(x_j)$*

Then ordered ($\lesssim$) binary resolution and factorisation is terminating.

Decidability for an unbounded number of sessions

Corollary

For any protocol that can be encoded with clauses of the previous form, then checking secrecy is decidable.

But how to deal with protocols that need more than one variable per clause ?

ProVerif

Developed by Bruno Blanchet, Paris, France.

- No restriction on the clauses
- Implements a sound semi-decision procedure (that may not terminate).
- Based on a resolution strategy well adapted to protocols.
- performs very well in practice !
  - Works on most of existing protocols in the literature
  - Is also used on industrial protocols (e.g. certified email protocol, JFK, Plutus filesystem)

Resolution strategy with selection

Definition

A selection function is any function *sel* such that $sel(H \Rightarrow C) \subseteq H$.

$$\frac{\neg A \vee C \quad B \vee D}{C\theta \vee D\theta} \quad \begin{array}{c} \theta = \mathrm{mgu}(A, B) \\ A \in sel(\neg A \vee C) \text{ or } sel(\neg A \vee C) = \emptyset \\ sel(B \vee D) = \emptyset \end{array}$$

Resolution strategy with selection

Definition

A selection function is any function sel such that $sel(H \Rightarrow C) \subseteq H$.

$$\frac{\neg A \vee C \qquad B \vee D}{C\theta \vee D\theta} \quad \begin{array}{c} \theta = \mathrm{mgu}(A, B) \\ A \in sel(\neg A \vee C) \text{ or } sel(\neg A \vee C) = \emptyset \\ sel(B \vee D) = \emptyset \end{array}$$

Theorem

Resolution and factorisation with selection are sound and refutationally complete for any selection function.

Limitations of ProVerif

What is the gap between processes and Horn clauses ?

Limitations of ProVerif

What is the gap between processes and Horn clauses ?

- The order of actions is abstracted
- Impossible to specify "just once"
  $\rightarrow$ P and $!P$ have the same translation in Horn clauses.
- Nonces are abstracted by function of the inputs.
  Example :
  $$\mathrm{in}(x).\mathrm{new}\, n.\mathrm{out}(\mathrm{enc}((x, n); k))$$
  is intuitively translated into the clause
  $$I(x) \Rightarrow I(\mathrm{enc}((x, n(x)); k))$$

What formal methods allow to do ?

- In general, secrecy preservation is undecidable.

What formal methods allow to do ?

- In general, secrecy preservation is undecidable.
- For a bounded number of sessions, secrecy is co-NP-complete [RusinowitchTuruani CSFW01]
  $\rightarrow$ several tools for detecting attacks (Casper, Avispa, Scyther, ...)

What formal methods allow to do ?

- In general, secrecy preservation is undecidable.
- For a bounded number of sessions, secrecy is co-NP-complete [RusinowitchTuruani CSFW01]
  $\rightarrow$ several tools for detecting attacks (Casper, Avispa, Scyther, ...)
- For an unbounded number of sessions
  - for one-copy protocols, secrecy is DEXPTIME-complete [CortierComon RTA03] [SeildVerma LPAR04]
  - for message-length bounded protocols, secrecy is DEXPTIME-complete [Durgin et al FMSP99] [Chevalier et al CSL03]

  $\rightarrow$ some tools for proving security (ProVerif, Scyther)