How does ProVerif work?

Véronique Cortier\textsuperscript{1}

\textsuperscript{1}LORIA, CNRS - INRIA Cassis project, Université de Lorraine
How to decide security for unlimited sessions?

→ 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?

→ 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?

→ 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  \(\exists n, i_1, \ldots, i_n \ u_{i_1} \cdots u_{i_n} = v_{i_1} \cdots v_{i_n}\)

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

Solution ? → Yes, \(1, 2, 3, 1\).

\[babababab\]
\[babababab\]
How to encode PCP in protocols?

Given $\{(u_i, v_i)\}_{1 \leq i \leq n}$, we construct the following protocol $P$:

\[
A \rightarrow B : \ \{\langle u_1, v_1 \rangle\}_{K_{ab}}, \ldots, \{\langle u_k, v_k \rangle\}_{K_{ab}}
\]

\[
B : \{\langle x, y \rangle\}_{K_{ab}} \rightarrow A : \ \{\langle x, u_1, y, v_1 \rangle\}_{K_{ab}}, \{s\}\{\langle x, u_1, x, u_1 \rangle\}_{K_{ab}},
\]

\[
\ldots, \{\langle x, u_k, y, v_k \rangle\}_{K_{ab}}, \{s\}\{\langle x, u_k, x, u_k \rangle\}_{K_{ab}}
\]

where $a_1 \cdot a_2 \cdots a_n$ denotes the term $\langle \cdots \langle a_1, a_2 \rangle, a_3, \rangle \cdots a_n \rangle$. 


How to encode PCP in protocols?

Given \( \{(u_i, v_i)\}_{1 \leq i \leq n} \), we construct the following protocol \( P \):

\[
A \rightarrow B : \{\langle u_1, v_1 \rangle\}_K^{ab}, \ldots, \{\langle u_k, v_k \rangle\}_K^{ab}
\]
\[
B : \{\langle x, y \rangle\}_K^{ab} \rightarrow A : \{\langle x, u_1, y, v_1 \rangle\}_K^{ab}, \{s\}\{\langle x, u_1, x, u_1 \rangle\}_K^{ab}, \ldots, \{\langle x, u_k, y, v_k \rangle\}_K^{ab}, \{s\}\{\langle x, u_k, x, u_k \rangle\}_K^{ab}
\]

where \( a_1 \cdot a_2 \cdots a_n \) denotes the term \( \langle \cdots \langle \langle a_1, a_2 \rangle, a_3, \rangle \cdots a_n \rangle \).

Then there is an attack on \( P \) iff there is a solution to the Post Correspondence Problem with entry \( \{(u_i, v_i)\}_{1 \leq i \leq n} \).
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 $\text{enc}(x, k_a)$ then $A$ responds with $x$.”

→ Use of first-order logic.
Some vocabulary

First order logic

Atoms $P(t_1, \ldots, t_n)$ where $t_i$ are terms, $P$ is a predicate

Literals $P(t_1, \ldots, t_n)$ or $\neg P(t_1, \ldots, t_n)$

closed under $\lor, \land, \neg, \exists, \forall$

Clauses: Only universal quantifiers

Horn Clauses: at most one positive literal

$$A_1, \ldots, A_n \Rightarrow B$$

where $A_i, B$ are atoms.
Intruder

Horn clauses perfectly reflects the attacker symbolic manipulations on terms.

\[
\begin{align*}
I(x), I(y) & \Rightarrow I(<x,y>) & \text{pairing} \\
I(x), I(y) & \Rightarrow I({x}y) & \text{encryption} \\
I({x}y), I(y) & \Rightarrow I(x) & \text{decryption} \\
I(<x,y>) & \Rightarrow I(x) & \text{projection} \\
I(<x,y>) & \Rightarrow I(y) & \text{projection}
\end{align*}
\]
Protocol:

\[
\begin{align*}
A \rightarrow B & : \ \{ \text{pin} \}_{k_a} \\
B \rightarrow A & : \ \{ \{ \text{pin} \}_{k_a} \}_{k_b} \\
A \rightarrow B & : \ \{ \text{pin} \}_{k_b}
\end{align*}
\]

Horn clauses:

\[
\begin{align*}
I(\{ \text{pin} \}_{k_a}) & \\
I(\{ x \}_{k_b}) & \\
I(\{ x \}_{k_a}) & \Rightarrow I(x)
\end{align*}
\]
Protocol

**Protocol:**

\[ A \rightarrow B : \{\text{pin}\}_{k_a} \]

\[ B \rightarrow A : \{\{\text{pin}\}_{k_a}\}_{k_b} \]

\[ A \rightarrow B : \{\text{pin}\}_{k_b} \]

**Horn clauses:**

\[ \Rightarrow I(\{\text{pin}\}_{k_a}) \]

\[ I(x) \Rightarrow I(\{x\}_{k_b}) \]

\[ I(\{x\}_{k_a}) \Rightarrow I(x) \]

**Secrecy property** is a **reachability** (accessibility) property

\[ \neg I(\text{pin}) \]

Then there exists an attack if and only if the set of formula corresponding to Intruder manipulations + protocol + property is **NOT** satisfiable.
How to decide satisfiability?

→ Resolution techniques
Binary resolution

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

An intuitive rule

$$A \Rightarrow C \quad A \quad C$$

In other words

$$\neg A \lor C \quad A \quad C$$
Binary resolution

\( A, B \) are atoms and \( C, D \) are clauses.

An intuitive rule

\[
\frac{A \Rightarrow C}{\frac{A}{C}}
\]

In other words

\[
\frac{\neg A \lor C}{\frac{A}{C}}
\]

Generalizing

\[
\frac{\neg A \lor C}{\frac{B}{\theta \in mgu(A, B)}}\quad (\text{i.e. } A\theta = B\theta)
\]
Binary resolution

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

An intuitive rule

$$
\begin{array}{c}
A \Rightarrow C \\
A \\
\hline \\
C \\
\end{array}
$$

In other words

$$
\begin{array}{c}
\neg A \lor C \\
A \\
\hline \\
C \\
\end{array}
$$

Generalizing

$$
\begin{array}{c}
\neg A \lor C \\
B \\
\hline \\
C \theta \\
\end{array} \\
\theta = mgu(A, B) \text{ (i.e. } A\theta = B\theta) \\
$$

Generalizing a bit more

$$
\begin{array}{c}
\neg A \lor C \\
B \lor D \\
\hline \\
C \theta \lor D \theta \\
\end{array} \\
\theta = mgu(A, B) \text{ Binary resolution}
$$
Binary resolution and Factorization

\[-A \lor C \quad B \lor D\]
\[\frac{}{C\theta \lor D\theta} \quad \theta = \text{mgu}(A, B) \quad \text{Binary resolution}\]

\[A \lor B \lor C\]
\[\frac{}{A\theta \lor C\theta} \quad \theta = \text{mgu}(A, B) \quad \text{Factorisation}\]

**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 $\bot$ (the empty clause) can be obtained from $C$ by binary resolution and factorisation.*

**Exercise:** Why do we need the factorisation rule?
Example

\[ C = \{-I(s), \ I(k_1), \ I(\{s\} \langle k_1, k_1 \rangle), \ I(\{x\}_y), I(y) \Rightarrow I(x), \ I(x), I(y) \Rightarrow I(\langle x, y \rangle)\} \]

\[ \begin{align*}
I(\{s\} \langle k_1, k_1 \rangle) & \quad I(\{x\}_y), I(y) \Rightarrow I(x) \\
I(\langle k_1, k_1 \rangle) & \Rightarrow s \\
\neg I(s) & \quad I(s)
\end{align*} \]

\[ \begin{align*}
I(k_1) & \quad I(x), I(y) \Rightarrow I(\langle x, y \rangle) \\
I(k_1) & \quad I(y) \Rightarrow I(\langle k_1, y \rangle) \\
I(\langle k_1, k_1 \rangle) & \quad I(\langle k_1, k_1 \rangle)
\end{align*} \]
But it is not terminating!

\[
\begin{align*}
I(s) & \quad I(x), I(y) \Rightarrow I(\langle x, y \rangle) \\
I(y) & \Rightarrow I(\langle s, y \rangle) \\
I(\langle s, s \rangle) & \Rightarrow I(\langle s, s \rangle)
\end{align*}
\]

\[
\begin{align*}
I(y) & \Rightarrow I(\langle s, y \rangle) \\
I(\langle s, s \rangle) & \Rightarrow I(\langle s, s \rangle)
\end{align*}
\]

\[
\begin{align*}
I(\langle s, \langle s, s \rangle \rangle) & \Rightarrow I(\langle s, \langle s, s \rangle \rangle)
\end{align*}
\]

\[\ldots\]

\[\rightarrow \text{This does not yield any decidability result.}\]
Ordered Binary resolution and Factorization

Let $<$ be any order on clauses.

\[
\begin{align*}
\neg A \lor C & \quad B \lor D \\
\hline
C \theta \lor D \theta & \quad \theta = \text{mgu}(A, B) \\
\end{align*}
\]

Ordered binary resolution

\[
\begin{align*}
A \lor B \lor C & \quad \theta = \text{mgu}(A, B) \\
\hline
A \theta \lor C \theta & \quad A \theta \not< C \theta \\
\end{align*}
\]

Ordered factorisation
Ordered Binary resolution and Factorization

Let \(<\) be any order on clauses.

\[
\begin{align*}
\neg A \lor C & \quad B \lor D \\
\hline
C\theta \lor D\theta \\
\end{align*}
\]

\[\theta = \text{mgu}(A, B)\]

Ordered binary resolution

\[A\theta \not< C\theta \lor D\theta\]

\[A \lor B \lor C\]

\[\theta = \text{mgu}(A, B)\]

Ordered factorisation

\[A\theta \lor C\theta\]

\[A\theta \not< C\theta\]

Theorem (Soundness and Completeness)

Ordered binary resolution and factorisation are sound and refutationally complete provided that \(<\) is liftable

\[\forall A, B, \theta \quad A < B \implies A\theta < B\theta\]
Examples of liftable orders

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

First example: subterm order

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

→ 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 $$C$$ is not satisfiable by ordered resolution (and factorisation).
Examples of liftable orders - continued

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

1. \( \text{depth}(P(t_1, \ldots, t_n)) \leq \text{depth}(Q(u_1, \ldots, u_k)) \)

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

\[
\begin{array}{c}
    f \\
    \text{?} \\
    \text{\preceq} \\
    x \\
    f \\
    x \\
    f \\
    x \\
    y \\
    a \\
\end{array}
\begin{array}{c}
    f \\
    x \\
    h \\
    h \\
    h \\
    y \\
\end{array}
\]
Examples of liftable orders - continued

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

1. $\text{depth}(P(t_1, \ldots, t_n)) \leq \text{depth}(Q(u_1, \ldots, u_k))$

2. For any variable $x$,
   $\text{depth}_x(P(t_1, \ldots, t_n)) \leq \text{depth}_x(Q(u_1, \ldots, u_k))$
Examples of liftable orders - continued

Second example: $P(t_1, \ldots, t_n) \preceq Q(u_1, \ldots, u_k)$ 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 $\forall A, B, \theta \quad A \preceq B \implies A\theta \preceq B\theta$
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

$$\Rightarrow \quad I(\{\text{pin}\}_{k_a})$$

$$I(x) \Rightarrow I(\{x\}_{k_b})$$

$$I(\{x\}_{k_a}) \Rightarrow I(x)$$

At most one variable per clause!
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

\[ \Rightarrow I(\{\text{pin}\}_{k_a}) \]
\[ I(x) \Rightarrow I(\{x\}_{k_b}) \]
\[ I(\{x\}_{k_a}) \Rightarrow I(x) \]

At most one variable per clause!

Theorem

Given a set \( C \) of clauses such that each clause of \( C \)

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

Then ordered (\( \langle \)) 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)
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] → 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]
  → 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]
  → some tools for proving security (ProVerif, Scyther)