Models and Techniques for Symbolic Analysis of Security Protocols

Véronique Cortier [1]

July 2017

All documents (slides, useful links, etc) available here :
`https://members.loria.fr/VCortier/school-2017/`

Summer School : Models and Tools for Cryptographic Proofs

1. LORIA, CNRS - INRIA Pesto project, Université de Lorraine

Outline of the course

Part 1 Protocols

Part 2 Model : the applied-pi calculus
$\rightarrow$ The ProVerif tool

Part 3 Analysis : protocols as Horn clauses

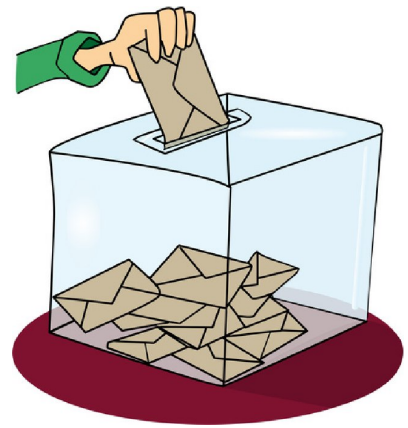
Part 1

Protocols

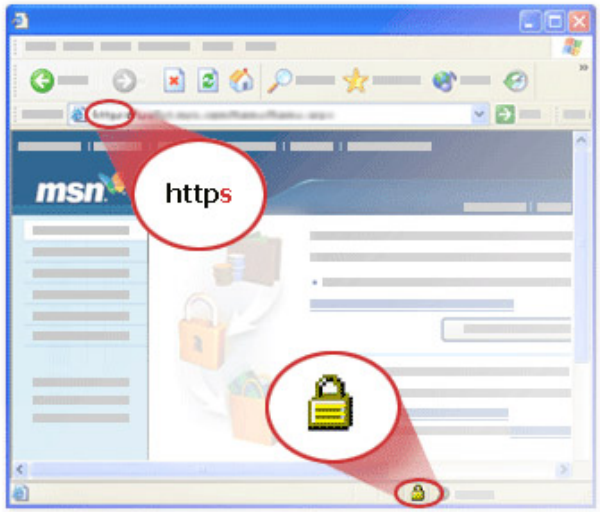
Context : cryptographic protocols

Cryptographic protocols are widely used in everyday life.

$\rightarrow$ They aim at securing communications over public or insecure networks.



Example : HTTPS



- TLS : Transport Security Layer, depuis 1994
- various implementations : OpenSSL, SecureTransport, JSSE, ...
- Lots of bugs and attacks, with fixes every month

Memory overflow

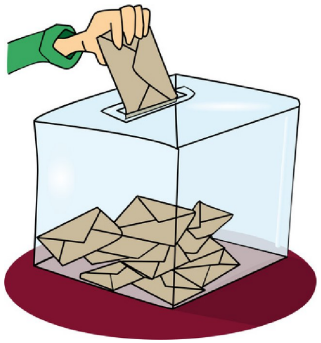
`Go to fail`

Missing checks
(MACs, signatures, ...)

FREAK attack - February 2015

Bhargavan et al.

Electronic voting



- The result corresponds to the votes.
- Each vote is confidential.
- No partial result is leaked before the end of the election
- Only voters can vote and at most once
- Coercion resistance

Security goals

Cryptographic protocols aim at

- preserving confidentiality of data
  (e.g. pin code, medical files, ...)
- ensuring authenticity
  (Are you really talking to your bank ? ?)
- ensuring anonymous communications
  (for e-voting protocols, ...)
- protecting against repudiation
  (I never sent this message ! !)
- ...

$\Rightarrow$ Cryptographic protocols vary depending on the application.

How does this work ?

How does this work ?

Protocol : describes how each participant should behave in order to get e.g. a common key.

Cryptography : makes uses of cryptographic primitives

- encryption
- signature
- hash
- ...

How to exchange a secret with commutative encryption

First : a small challenge for your nephews / nieces / cousins / children.

A completely fictitious town

Two types of inhabitants :

Sedentary inhabitants stay at their home

Post office workers deliver boxes between sedentary inhabitants

A completely fictitious town

Two types of inhabitants :

Sedentary inhabitants stay at their home

Post office workers deliver boxes between sedentary inhabitants

Axiom 1 Post office workers may steal any unlocked box
(Reminder : this scenario is entirely fictitious !)

Axiom 2 The content of locked boxes CANNOT be stolen.

Challenge

How Alice (sedentary) can send a gift to Bob (also sedentary) ?

Commutative Symmetric encryption

Symmetric encryption, denoted by $\{m\}_k$

The same key is used for encrypting and decrypting.

Commutative (symmetric) encryption

$$\{\{m\}_{k_1}\}_{k_2} = \{\{m\}_{k_2}\}_{k_1}$$

Exchanging a secret with commutative encryption (RSA)

$\{\text{pin} : 3443\}_{k_{\text{alice}}}$ ⟶

Exchanging a secret with commutative encryption (RSA)

$$\{\text{pin} : 3443\}_{k_{\text{alice}}} \longrightarrow$$

$$\left\{\{\text{pin} : 3443\}_{k_{\text{alice}}}\right\}_{k_{\text{bob}}} \longleftarrow$$

Exchanging a secret with commutative encryption (RSA)

$$\{\text{pin} : 3443\}_{k_{\text{alice}}} \longrightarrow$$

$$\leftarrow \left\{\{\text{pin} : 3443\}_{k_{\text{alice}}}\right\}_{k_{\text{bob}}}$$

$$\{\text{pin} : 3443\}_{k_{\text{bob}}} \longrightarrow$$

$$\text{Since } \left\{\{\text{pin} : 3443\}_{k_{\text{alice}}}\right\}_{k_{\text{bob}}} = \left\{\{\text{pin} : 3443\}_{k_{\text{bob}}}\right\}_{k_{\text{alice}}}$$

Exchanging a secret with commutative encryption (RSA)

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

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

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

$\rightarrow$ It does not work ! (Authentication problem)

Exchanging a secret with commutative encryption (RSA)

$$\{\text{pin} : 3443\}_{k_{\text{alice}}} \longrightarrow$$

$$\left\{\{\text{pin} : 3443\}_{k_{\text{alice}}}\right\}_{k_{\text{bob}}} \longleftarrow$$

$$\{\text{pin} : 3443\}_{k_{\text{bob}}} \longrightarrow$$

$\rightarrow$ It does not work ! (Authentication problem)

$$\{\text{pin} : 3443\}_{k_{\text{alice}}} \longrightarrow$$

$$\left\{\{\text{pin} : 3443\}_{k_{\text{alice}}}\right\}_{k_{\text{intruder}}} \longleftarrow$$

$$\{\text{pin} : 3443\}_{k_{\text{intruder}}} \longrightarrow$$

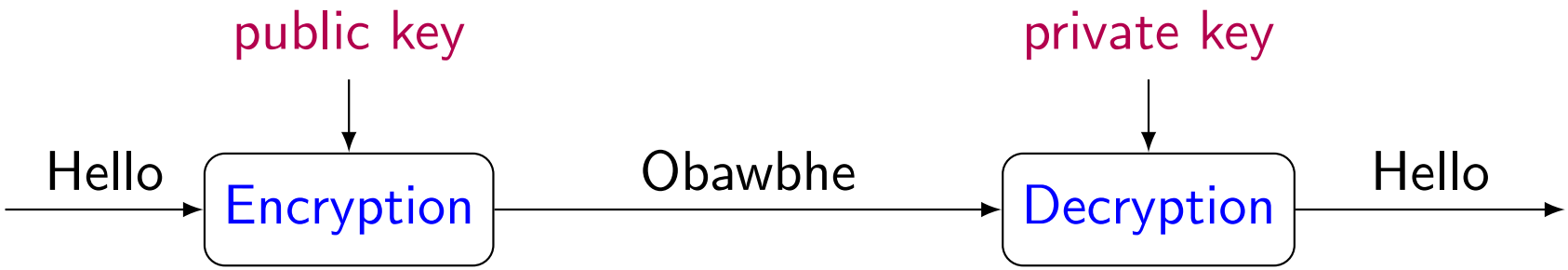
Another example

The "famous" Needham-Schroeder public key protocol

(and its associated Man-In-The-Middle Attack)

Public key encryption

Public key : $\mathrm{pk}(A)$
Encryption : $\{m\}_{\mathrm{pk}(A)}$



Encryption with the public key and decryption with the private key.

Invented only in the late 70's !

Needham-Schroeder public key protocol

N_a Random number (called nonce) generated by A.
N_b Random number (called nonce) generated by B.

- $A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$
  $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$
  $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$

Needham-Schroeder public key protocol

N_a Random number (called nonce) generated by A.
N_b Random number (called nonce) generated by B.

- $A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$
- $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$
- $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$

Needham-Schroeder public key protocol

N_a Random number (called nonce) generated by A.
N_b Random number (called nonce) generated by B.

- $A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}$
- $B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}$
- $A \rightarrow B : \{N_b\}_{\text{pub}(B)}$

Needham-Schroeder public key protocol

N_a Random number (called nonce) generated by A.
N_b Random number (called nonce) generated by B.

$$
\begin{array}{lll}
A & \rightarrow B: & \{A, N_a\}_{\text{pub}(B)} \\
B & \rightarrow A: & \{N_a, N_b\}_{\text{pub}(A)} \\
A & \rightarrow B: & \{N_b\}_{\text{pub}(B)}
\end{array}
$$

Questions :

- Is N_b secret between A and B ?
- When B receives $\{N_b\}_{\text{pub}(B)}$, does this message really come from A ?

Needham-Schroeder public key protocol

N_a Random number (called nonce) generated by A.
N_b Random number (called nonce) generated by B.

$$\begin{aligned} A &\rightarrow B : \quad \{A, N_a\}_{\text{pub}(B)} \\ B &\rightarrow A : \quad \{N_a, N_b\}_{\text{pub}(A)} \\ A &\rightarrow B : \quad \{N_b\}_{\text{pub}(B)} \end{aligned}$$

Questions :

- Is N_b secret between A and B ?
- When B receives $\{N_b\}_{\text{pub}(B)}$, does this message really come from A ?

$\rightarrow$ An attack was discovered in 1996, 17 years after the publication of the protocol !

Man in the middle attack

$\{A, N_a\}_{\mathrm{pub}(P)}$

$\{A, N_a\}_{\mathrm{pub}(B)}$

Man in the middle attack

$\{A, N_a\}_{\text{pub}(P)}$ →

$\{A, N_a\}_{\text{pub}(B)}$ →

← $\{N_a, N_b\}_{\text{pub}(A)}$

← $\{N_a, N_b\}_{\text{pub}(A)}$

Man in the middle attack

$\{A, N_a\}_{\text{pub}(P)}$ $\longrightarrow$

$\{A, N_a\}_{\text{pub}(B)}$ $\longrightarrow$

$\{N_a, N_b\}_{\text{pub}(A)}$ $\longrightarrow$

$\{N_a, N_b\}_{\text{pub}(A)}$ $\longrightarrow$

$\{N_b\}_{\text{pub}(P)}$ $\longrightarrow$

$\{N_b\}_{\text{pub}(B)}$ $\longrightarrow$

Man in the middle attack

$\{A, N_a\}_{\text{pub}(P)}$ →

$\{A, N_a\}_{\text{pub}(B)}$ →

← $\{B, N_a, N_b\}_{\text{pub}(A)}$

← $\{B, N_a, N_b\}_{\text{pub}(A)}$

$\{N_b\}_{\text{pub}(P)}$ →

$\{N_b\}_{\text{pub}(B)}$ →

Fixing the flaw : add the identity of B.

A symmetric key protocol

The Wide Mouthed Frog protocol (a variant)

Assumption : Each participant X shares a long term symmetric key K_{xs} with a serveur S.

Protocol :

$$\begin{array}{lll} A & \rightarrow S: & A, \{B, K_{ab}\}_{K_{as}} \\ S & \rightarrow B: & B, \{A, K_{ab}\}_{K_{bs}} \end{array}$$

Security Goal : A and B share a secret symmetric key K_{ab}.

A symmetric key protocol

The Wide Mouthed Frog protocol (a variant)

Assumption : Each participant X shares a long term symmetric key K_{xs} with a serveur S.

Protocol :

$$\begin{array}{lll} A & \rightarrow S: & A, \{B, K_{ab}\}_{K_{as}} \\ S & \rightarrow B: & B, \{A, K_{ab}\}_{K_{bs}} \end{array}$$

Security Goal : A and B share a secret symmetric key K_{ab}.

What if we consider another variant ?

$$\begin{array}{lll} A & \rightarrow S: & A, \{B\}_{K_{as}}, \{K_{ab}\}_{K_{as}} \\ S & \rightarrow B: & B, \{A\}_{K_{bs}}, \{K_{ab}\}_{K_{bs}} \end{array}$$

Part 2

Model : the applied-pi calculus

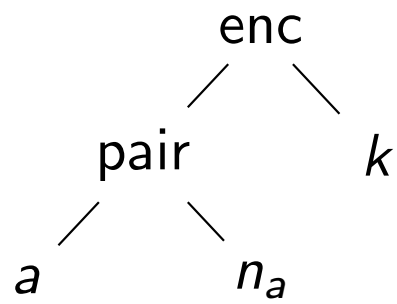
Messages

Messages are abstracted by terms.

Agents : $a, b, \ldots$ Nonces : $n_1, n_2, \ldots$
Keys : $k_1, k_2, \ldots$
Cyphertext : $\mathrm{enc}(m, k)$ Concatenation : $\mathrm{pair}(m_1, m_2)$

Example : The message $\{A, N_a\}_K$ is represented by :

$\mathrm{enc}(\mathrm{pair}(A, N_a), K)$



Intuition : only the structure of the message is kept.

Encryption-Decryption properties

$$\mathsf{dec}(\mathsf{enc}(x, y), y) = x$$

$$\pi_1(\langle x, y \rangle) = x$$

$$\pi_2(\langle x, y \rangle) = y$$

$$\mathsf{deca}(\mathsf{enca}(x, \mathsf{pub}(y)), y) = x$$

Equational theory

More generally, the cryptographic primitives are modeled by an equational theory.

Definition

An equational theory $=_E$ is a relation on terms that is closed under substitutions of terms for variables and closed by context.

- $u =_E v$ implies $u\sigma =_E v\sigma$ (for any σ)
- $u_1 =_E v_1, \ldots, u_n =_E v_n$ implies $f(u_1, \ldots, u_n) =_E f(v_1, \ldots, v_n)$ (for any $f \in \mathcal{F}$)

Other examples of theories

EXclusive Or

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

Diffie-Hellmann

$$\exp(\exp(z, x), y) = \exp(\exp(z, y), x)$$

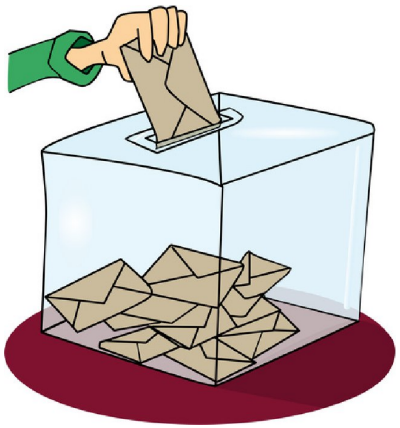
Public key encryption in ProVerif

fun pk(skey) : pkey.

fun aenc(bitstring, pkey) : bitstring.

reduc forall x : bitstring, y : skey ; adec(aenc(x,pk(y)),y) = x.

E-voting protocols



First phase :

$$V \to A : \quad \mathrm{sign}(\mathit{blind}(\mathrm{vote}, r), V)$$
$$A \to V : \quad \mathrm{sign}(\mathit{blind}(\mathrm{vote}, r), A)$$

Voting phase :

$$V \to C : \quad \mathrm{sign}(\mathrm{vote}, A)$$

...

Equational theory for blind signatures

[Kremer Ryan 05]

$$\begin{aligned} \mathsf{checksign}(\mathsf{sign}(x,y),\mathsf{pk}(y)) &= x \\ \mathsf{unblind}(\mathsf{blind}(x,y),y) &= x \\ \mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) &= \mathsf{sign}(x,z) \end{aligned}$$

Syntax for processes

The grammar of processes is as follows :

$$
\begin{aligned}
&P, Q, R := \\
&\quad 0 \\
&\quad \texttt{if } M_1 = M_2 \texttt{ then } P \texttt{ else } Q \\
&\quad \mathsf{let}\ x = M \mathsf{in}\ P \\
&\quad \mathsf{in}(c, x).P \\
&\quad \mathsf{out}(c, N).P \\
&\quad \nu n.P \\
&\quad P \mid Q \\
&\quad !P
\end{aligned}
$$

Inspired from the applied-pi calculus [Abadi-Fournet]

Example : Needham-Schroeder protocol

$$
\begin{aligned}
A &\rightarrow B : \quad \{A, N_a\}_{\mathrm{pub}(B)} \\
B &\rightarrow A : \quad \{N_a, N_b\}_{\mathrm{pub}(A)} \\
A &\rightarrow B : \quad \{N_b\}_{\mathrm{pub}(B)}
\end{aligned}
$$

N_a random number generated by A.
N_b random number generated by B.

Example : Needham-Schroeder protocol

$$\begin{aligned} A &\rightarrow B : \quad \{A, N_a\}_{\mathrm{pub}(B)} \\ B &\rightarrow A : \quad \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A &\rightarrow B : \quad \{N_b\}_{\mathrm{pub}(B)} \end{aligned}$$

N_a random number generated by A.
N_b random number generated by B.

We need to model two processes :

- one corresponding to the role of A
- one corresponding to the role of B

Role of A

$$\begin{aligned} A &\to B: & \{A, N_a\}_{\mathrm{pub}(B)} \\ B &\to A: & \{N_a, N_b\}_{\mathrm{pub}(A)} \\ A &\to B: & \{N_b\}_{\mathrm{pub}(B)} \end{aligned}$$

$P_A(\mathrm{priv}_A, \mathrm{pub}_B, A, B, N_A) :=$

Role of A

$$\begin{aligned} A &\to B: \quad \{A, N_a\}_{\mathsf{pub}(B)} \\ B &\to A: \quad \{N_a, N_b\}_{\mathsf{pub}(A)} \\ A &\to B: \quad \{N_b\}_{\mathsf{pub}(B)} \end{aligned}$$

$$P_A(\mathsf{priv}_A, \mathsf{pub}_B, A, B, N_A) := \mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(A, N_A), \mathsf{pub}_B)).$$

Role of A

$$
\begin{aligned}
A &\rightarrow B: \quad \{A, N_a\}_{\mathsf{pub}(B)} \\
B &\rightarrow A: \quad \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A &\rightarrow B: \quad \{N_b\}_{\mathsf{pub}(B)}
\end{aligned}
$$

$$
\begin{aligned}
P_A(\mathsf{priv}_A, \mathsf{pub}_B, A, B, N_A) := \\
&\mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(A, N_A), \mathsf{pub}_B)). \\
&\mathsf{in}(c, x).
\end{aligned}
$$

Role of A

$$
\begin{array}{lll}
A & \rightarrow B: & \{A, N_a\}_{\mathsf{pub}(B)} \\
B & \rightarrow A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A & \rightarrow B: & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

$$
\begin{aligned}
P_A(\mathsf{priv}_A, \mathsf{pub}_B, A, B, N_A) := \\
&\mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(A, N_A), \mathsf{pub}_B)). \\
&\mathsf{in}(c, x). \\
&\quad \mathsf{let}\ z = \mathsf{deca}(x, \mathsf{priv}_A)\ \mathsf{in}
\end{aligned}
$$

Role of A

$$
\begin{array}{lll}
A & \rightarrow B: & \{A, N_a\}_{\mathsf{pub}(B)} \\
B & \rightarrow A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A & \rightarrow B: & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

$$
\begin{aligned}
P_A(\mathsf{priv}_A, \mathsf{pub}_B, A, B, N_A) := \\
&\mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(A, N_A), \mathsf{pub}_B)). \\
&\mathsf{in}(c, x). \\
&\quad \text{let } z = \mathsf{deca}(x, \mathsf{priv}_A) \text{ in} \\
&\quad\quad \texttt{if } N_A = \pi_1(z) \texttt{ then let } y = \pi_2(z) \text{ in}
\end{aligned}
$$

Role of A

$$
\begin{array}{lll}
A & \rightarrow B: & \{A, N_a\}_{\mathsf{pub}(B)} \\
B & \rightarrow A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A & \rightarrow B: & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

$$
\begin{aligned}
P_A(\mathsf{priv}_A, \mathsf{pub}_B, A, B, N_A) := \\
&\mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(A, N_A), \mathsf{pub}_B)). \\
&\mathsf{in}(c, x). \\
&\quad \text{let } z = \mathsf{deca}(x, \mathsf{priv}_A) \text{ in} \\
&\quad\quad \texttt{if } N_A = \pi_1(z) \texttt{ then let } y = \pi_2(z) \text{ in} \\
&\quad\quad\quad \mathsf{out}(c, \mathsf{enca}(y, \mathsf{pub}_B))
\end{aligned}
$$

Role of B

$$
\begin{array}{lll}
A & \rightarrow B: & \{A, N_a\}_{\mathsf{pub}(B)} \\
B & \rightarrow A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A & \rightarrow B: & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

$$
P_B(\mathsf{priv}_B, \mathsf{pub}_A, A, B, N_B) := \mathsf{in}(c, x).
$$

Role of B

$$\begin{aligned} A &\rightarrow B: \quad \{A, N_a\}_{\mathsf{pub}(B)} \\ B &\rightarrow A: \quad \{N_a, N_b\}_{\mathsf{pub}(A)} \\ A &\rightarrow B: \quad \{N_b\}_{\mathsf{pub}(B)} \end{aligned}$$

$$\begin{aligned} P_B(\mathsf{priv}_B, \mathsf{pub}_A, A, B, N_B) := \\ \mathsf{in}(c, x). \\ \mathsf{let}\ (= A, y) = \mathsf{deca}(x, \mathsf{priv}_B)\ \mathsf{in} \end{aligned}$$

Role of B

$$
\begin{array}{lll}
A & \rightarrow B: & \{A, N_a\}_{\mathsf{pub}(B)} \\
B & \rightarrow A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A & \rightarrow B: & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

$$
\begin{aligned}
P_B(\mathsf{priv}_B, \mathsf{pub}_A, A, B, N_B) := \\
&\mathsf{in}(c, x). \\
&\quad \mathsf{let}\ (= A, y) = \mathsf{deca}(x, \mathsf{priv}_B)\ \mathsf{in} \\
&\quad\ \mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(y, N_B), \mathsf{pub}_A)).
\end{aligned}
$$

Role of B

$$
\begin{array}{lll}
A & \rightarrow B: & \{A, N_a\}_{\mathsf{pub}(B)} \\
B & \rightarrow A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A & \rightarrow B: & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

$$
\begin{aligned}
&P_B(\mathsf{priv}_B, \mathsf{pub}_A, A, B, N_B) := \\
&\qquad \mathsf{in}(c, x). \\
&\qquad\quad \mathsf{let}\ (= A, y) = \mathsf{deca}(x, \mathsf{priv}_B)\ \mathsf{in} \\
&\qquad\qquad \mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(y, N_B), \mathsf{pub}_A)). \\
&\qquad\qquad \mathsf{in}(c, z).
\end{aligned}
$$

Role of B

$$
\begin{array}{lll}
A & \rightarrow B: & \{A, N_a\}_{\mathsf{pub}(B)} \\
B & \rightarrow A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A & \rightarrow B: & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

$$
\begin{array}{l}
P_B(\mathsf{priv}_B, \mathsf{pub}_A, A, B, N_B) := \\
\qquad \mathsf{in}(c, x). \\
\qquad\quad \mathsf{let}\ (= A, y) = \mathsf{deca}(x, \mathsf{priv}_B)\ \mathsf{in} \\
\qquad\qquad \mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(y, N_B), \mathsf{pub}_A)). \\
\qquad\qquad \mathsf{in}(c, z). \\
\qquad\qquad \texttt{if}\ \mathsf{deca}(z, \mathsf{priv}_B) = N_B\ \texttt{then}\ 0\ \texttt{else}\ 0
\end{array}
$$

Complete process

Then, the complete process representing the Needham-Schroeder protocol is :

$P :=$

$$!(\nu N_A.P_A(\text{priv}_A, \text{pub}(\text{priv}_B), A, B, N_A)) \mid$$
$$!(\nu N_B.P_B(\text{priv}_B, \text{pub}(\text{priv}_A), A, B, N_B))$$

Complete process

Then, the complete process representing the Needham-Schroeder protocol is :

$$
\begin{aligned}
P := \quad & \\
& \nu\,\mathrm{priv}_A\,.\nu\,\mathrm{priv}_B\,. \\
& \mathrm{out}(c, \mathrm{pub}(\mathrm{priv}_A)).\mathrm{out}(c, \mathrm{pub}(\mathrm{priv}_B)). \\
& \quad !(\nu N_A.P_A(\mathrm{priv}_A, \mathrm{pub}(\mathrm{priv}_B), A, B, N_A)) \mid \\
& \quad !(\nu N_B.P_B(\mathrm{priv}_B, \mathrm{pub}(\mathrm{priv}_A), A, B, N_B))
\end{aligned}
$$

Complete process

Then, the complete process representing the Needham-Schroeder protocol is :

$$
\begin{aligned}
P := \quad & \\
& \nu\,\mathsf{priv}_A\,.\nu\,\mathsf{priv}_B\,. \\
& \mathsf{out}(c, \mathsf{pub}(\mathsf{priv}_A)).\mathsf{out}(c, \mathsf{pub}(\mathsf{priv}_B)). \\
& \quad !(\nu N_A.P_A(\mathsf{priv}_A, \mathsf{pub}(\mathsf{priv}_B), A, B, N_A)) \mid \\
& \quad !(\nu N_B.P_B(\mathsf{priv}_B, \mathsf{pub}(\mathsf{priv}_A), A, B, N_B))
\end{aligned}
$$

satisfied ?

Complete process - continued

Better :

$P :=$

$$\nu\,\mathrm{priv}_A\,.\nu\,\mathrm{priv}_B\,.\nu\,\mathrm{priv}_C\,.$$
$$\mathrm{out}(c, \mathrm{pub}(\mathrm{priv}_A)).\mathrm{out}(c, \mathrm{pub}(\mathrm{priv}_B)).\mathrm{out}(c, \mathrm{priv}_C).$$
$$!(\nu N_A.P_A(\mathrm{priv}_A, \mathrm{pub}(\mathrm{priv}_B), A, B, N_A)) \mid$$
$$!(\nu N_A.P_A(\mathrm{priv}_A, \mathrm{pub}(\mathrm{priv}_C), A, C, N_A)) \mid$$
$$!(\nu N_B.P_B(\mathrm{priv}_B, \mathrm{pub}(\mathrm{priv}_A), A, B, N_B)) \mid$$
$$!(\nu N_B.P_B(\mathrm{priv}_B, \mathrm{pub}(\mathrm{priv}_C), C, B, N_B))$$

Complete process - continued

Better :

$$
\begin{aligned}
P := \quad & \nu\,\mathsf{priv}_A .\nu\,\mathsf{priv}_B .\nu\,\mathsf{priv}_C . \\
& \mathsf{out}(c, \mathsf{pub}(\mathsf{priv}_A)).\mathsf{out}(c, \mathsf{pub}(\mathsf{priv}_B)).\mathsf{out}(c, \mathsf{priv}_C). \\
& \quad !(\nu N_A.P_A(\mathsf{priv}_A, \mathsf{pub}(\mathsf{priv}_B), A, B, N_A)) \mid \\
& \quad !(\nu N_A.P_A(\mathsf{priv}_A, \mathsf{pub}(\mathsf{priv}_C), A, C, N_A)) \mid \\
& \quad !(\nu N_B.P_B(\mathsf{priv}_B, \mathsf{pub}(\mathsf{priv}_A), A, B, N_B)) \mid \\
& \quad !(\nu N_B.P_B(\mathsf{priv}_B, \mathsf{pub}(\mathsf{priv}_C), C, B, N_B))
\end{aligned}
$$

and also

$$
\begin{aligned}
& !(\nu N_A.P_A(\mathsf{priv}_B, \mathsf{pub}(\mathsf{priv}_A), B, A, N_A)) \mid \\
& !(\nu N_A.P_A(\mathsf{priv}_B, \mathsf{pub}(\mathsf{priv}_C), B, C, N_A)) \mid \\
& \ldots
\end{aligned}
$$

What to remember when modeling a protocol ?

1. A process for each role of the protocol
   - identify the initial knowledge of the agent (here, $N_A, \mathsf{priv}_A, \mathsf{pub}_B, A, \ldots$)
   - identify the values learned during the protocol, modeled with variables

What to remember when modeling a protocol ?

1. A process for each role of the protocol
   - identify the initial knowledge of the agent (here, $N_A, \mathsf{priv}_A, \mathsf{pub}_B, A, \ldots$)
   - identify the values learned during the protocol, modeled with variables

Don't think you're done !

What to remember when modeling a protocol ?

1. A process for each role of the protocol
   - identify the initial knowledge of the agent (here, $N_A, \mathrm{priv}_A, \mathrm{pub}_B, A, \ldots$)
   - identify the values learned during the protocol, modeled with variables

   Don't think you're done !

2. Initial knowledge of the intruder
   - identify the public data, should be sent to the network
   - identify the private data of the corrupted agents

What to remember when modeling a protocol ?

1. A process for each role of the protocol
   - identify the initial knowledge of the agent (here, $N_A, \mathsf{priv}_A, \mathsf{pub}_B, A, \ldots$)
   - identify the values learned during the protocol, modeled with variables

   Don't think you're done !

2. Initial knowledge of the intruder
   - identify the public data, should be sent to the network
   - identify the private data of the corrupted agents

3. Finally, the complete process
   - put all the roles together
   - don't forger roles where an honest agent talks with a corrupted one !

How to express that a protocol is secure ? (in ProVerif)

Secrecy

Pretty simple :

query attacker(s)

How to express authentication ?

$\rightarrow$ a correspondence property : if B finishes a session, thinking he has talked to A with session nonce N_b then A has also finished a session, thinking she has talked to B with session nonce N_b.

Syntax - enriched

$$
\begin{aligned}
P, Q, R := \; & 0 \\
& \texttt{if } M_1 = M_2 \texttt{ then } P \texttt{ else } Q \\
& \mathsf{in}(c, x).P \\
& \mathsf{out}(c, N).P \\
& \nu n.P \\
& !P \\
& \mathsf{event}(p(u_1, \ldots, u_n)).P
\end{aligned}
$$

where p is a predicate of arity n.

Example : Needham-Schroeder (continued)

Role of A - with events

$$
\begin{aligned}
A &\rightarrow B: \quad \{A, N_a\}_{\mathsf{pub}(B)} \\
B &\rightarrow A: \quad \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A &\rightarrow B: \quad \{N_b\}_{\mathsf{pub}(B)}
\end{aligned}
$$

$P_A(\mathsf{priv}_A, \mathsf{pub}_B, A, B, N_A) :=$

$\mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(A, N_A), \mathsf{pub}_B)).$
$\mathsf{in}(c, x).$
$\quad \mathsf{let}\ (= N_A, y) = \mathsf{deca}(x, \mathsf{priv}_A)\ \mathsf{in}$
$\quad \mathsf{out}(c, \mathsf{enca}(y, \mathsf{pub}_B))$

Role of A - with events

$$
\begin{aligned}
A &\rightarrow B: \quad \{A, N_a\}_{\mathsf{pub}(B)} \\
B &\rightarrow A: \quad \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A &\rightarrow B: \quad \{N_b\}_{\mathsf{pub}(B)}
\end{aligned}
$$

$P_A(\mathsf{priv}_A, \mathsf{pub}_B, A, B, N_A) :=$

BeginA(A, B, $\mathrm{N_A}$)

$\mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(A, N_A), \mathsf{pub}_B))$.

$\mathsf{in}(c, x)$.

let $(= N_A, y) = \mathsf{deca}(x, \mathsf{priv}_A)$ in

$\mathsf{out}(c, \mathsf{enca}(y, \mathsf{pub}_B))$

EndA(A, B, y)

Role of B

$$
\begin{aligned}
A &\rightarrow B: \quad \{A, N_a\}_{\mathsf{pub}(B)} \\
B &\rightarrow A: \quad \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A &\rightarrow B: \quad \{N_b\}_{\mathsf{pub}(B)}
\end{aligned}
$$

$P_B(\mathsf{priv}_B, \mathsf{pub}_A, A, B, N_B) :=$

$\mathsf{in}(c, x).$

$\mathsf{let}\ (= A, y) = \mathsf{deca}(x, \mathsf{priv}_B)\ \mathsf{in}$

$\mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(y, N_B), \mathsf{pub}_A)).$

$\mathsf{in}(c, z).$

$\mathtt{if}\ \mathsf{deca}(z, \mathsf{priv}_B) = N_B$

$\mathtt{then}\ 0\ \ \mathtt{else}\ 0$

Role of B

$$
\begin{array}{lll}
A & \rightarrow B: & \{A, N_a\}_{\mathsf{pub}(B)} \\
B & \rightarrow A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A & \rightarrow B: & \{N_b\}_{\mathsf{pub}(B)}
\end{array}
$$

$P_B(\mathsf{priv}_B, \mathsf{pub}_A, A, B, N_B) :=$

$\mathsf{in}(c, x).$

$\mathsf{let}\ (= A, y) = \mathsf{deca}(x, \mathsf{priv}_B)\ \mathsf{in}$

$\mathsf{BeginB}(\mathsf{A}, \mathsf{B}, \mathsf{N_B})$

$\mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(y, N_B), \mathsf{pub}_A)).$

$\mathsf{in}(c, z).$

$\mathtt{if}\ \mathsf{deca}(z, \mathsf{priv}_B) = N_B$

$\mathtt{then}\ \ \mathsf{EndB}(\mathsf{A}, \mathsf{B}, \mathsf{y})\ \mathtt{else}\ 0$

Role of B

$$\begin{aligned} A &\rightarrow B: & \{A, N_a\}_{\mathsf{pub}(B)} \\ B &\rightarrow A: & \{N_a, N_b\}_{\mathsf{pub}(A)} \\ A &\rightarrow B: & \{N_b\}_{\mathsf{pub}(B)} \end{aligned}$$

$P_B(\mathsf{priv}_B, \mathsf{pub}_A, A, B, N_B) :=$

$\mathsf{in}(c, x).$

$\quad \mathsf{let}\ (= A, y) = \mathsf{deca}(x, \mathsf{priv}_B)\ \mathsf{in}$

$\quad \mathsf{BeginB}(\mathsf{A}, \mathsf{B}, \mathsf{N_B})$

$\quad \mathsf{out}(c, \mathsf{enca}(\mathsf{pair}(y, N_B), \mathsf{pub}_A)).$

$\quad \mathsf{in}(c, z).$

$\quad\quad \mathtt{if}\ \mathsf{deca}(z, \mathsf{priv}_B) = N_B$

$\quad\quad\quad \mathtt{then}\ \ \mathsf{EndB}(\mathsf{A}, \mathsf{B}, \mathsf{y})\ \mathtt{else}\ 0$

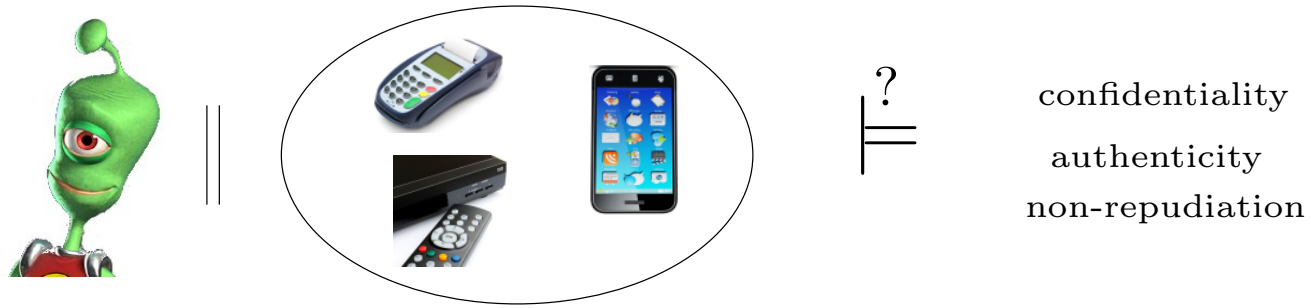
Authentication properties :

$$\begin{aligned} \forall x\ \mathsf{EndB}(A, B, x) &\Rightarrow \mathsf{BeginA}(A, B, x) \\ \forall x\ \mathsf{EndA}(A, B, x) &\Rightarrow \mathsf{BeginB}(A, B, x) \end{aligned}$$

Part 3

Analysis : protocols as Horn clauses

How to analyse security protocols ?



Methodology

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

Difficulty

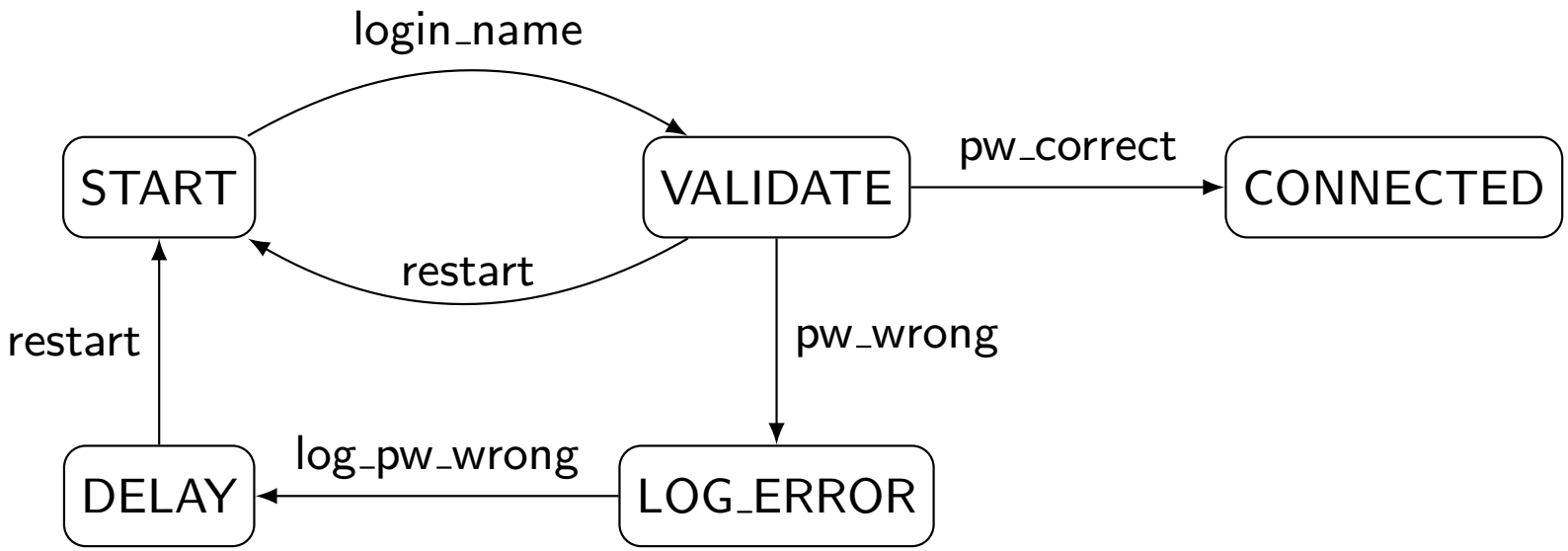
Presence of an attacker

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

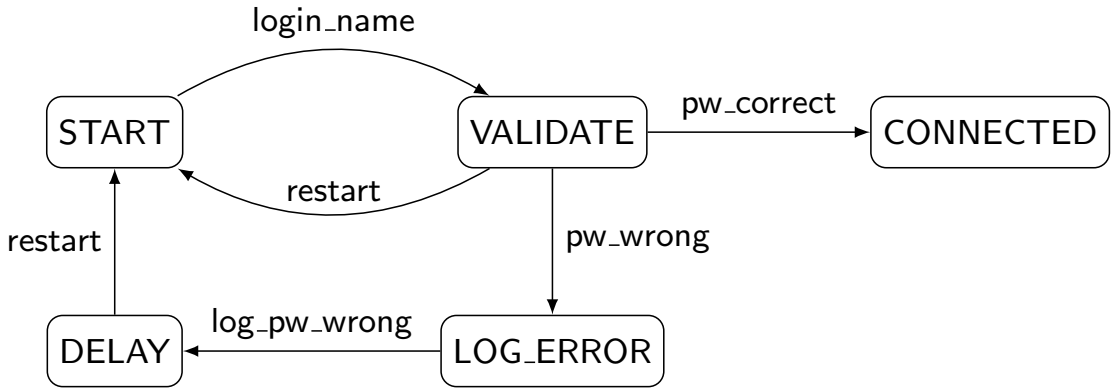
$\Rightarrow$ The system is infinitely branching

A first approach

Why not modeling security protocol using a (possibly extended) automata ?



How to model a security protocol ?



- The output of each participants strongly depends on the data received inside the message.
- At each step, a malicious user (called the adversary) may create arbitrary messages.
- The output of the adversary strongly depends on the messages sent on the network.

$\rightarrow$ It is important to have a tight modeling of the messages.

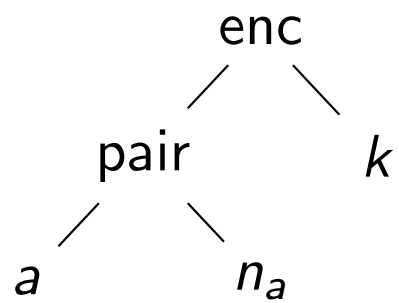
Messages

Messages are abstracted by terms.

Agents : $a, b, \ldots$ Nonces : $n_1, n_2, \ldots$
Keys : $k_1, k_2, \ldots$
Cyphertext : $\mathrm{enc}(m, k)$ Concatenation : $\mathrm{pair}(m_1, m_2)$

Example : The message $\{A, N_a\}_K$ is represented by :

$\mathrm{enc}(\mathrm{pair}(A, N_a), K)$



Intuition : only the structure of the message is kept.

A simple protocol

$\langle Alice, k \rangle$ →

← $\langle Bob, \mathrm{enc}(s, k) \rangle$

A simple protocol

$\langle Alice, k \rangle$ →

← $\langle Bob, \mathrm{enc}(s, k) \rangle$

Question ?

Can the attacker learn the secret s ?

A simple protocol

$\langle Alice, k \rangle$ →

← $\langle Bob, \mathrm{enc}(s, k) \rangle$

Question ?

Can the attacker learn the secret s ?

Answer : Of course, Yes !

$$\frac{\dfrac{\langle \mathrm{Bob}, \mathrm{enc}(s, k) \rangle}{\mathrm{enc}(s, k)} \qquad \dfrac{\langle \mathrm{Alice}, k \rangle}{k}}{s}$$

Intruder abilities

Composition rules

$$\frac{u \quad v}{\mathsf{pair}(u, v)} \qquad \frac{u \quad v}{\mathsf{enc}(u, v)} \qquad \frac{u \quad v}{\mathsf{enca}(u, v)}$$

Intruder abilities

Composition rules

$$\frac{u \quad v}{\mathrm{pair}(u,v)} \qquad \frac{u \quad v}{\mathrm{enc}(u,v)} \qquad \frac{u \quad v}{\mathrm{enca}(u,v)}$$

Decomposition rules

$$\frac{}{u}\ u \in T \qquad \frac{\mathrm{pair}(u,v)}{u} \qquad \frac{\mathrm{pair}(u,v)}{v}$$

$$\frac{\mathrm{enc}(u,v) \quad v}{u} \qquad \frac{\mathrm{enca}(u,\mathrm{pub}(v)) \quad \mathrm{priv}(v)}{u}$$

Deducibility relation

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.

Examples

$$S = \left\{ \begin{array}{l} \mathsf{enc}(\mathsf{pair}(\mathsf{pair}(a, k_3), k_4),\ \mathsf{pair}(k_1, k_2)), \\ a, \\ k_1, \\ \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1)) \end{array} \right\}$$

$$S \stackrel{?}{\vdash} k_1,$$

Examples

$$S = \left\{ \begin{array}{l} \mathsf{enc}(\mathsf{pair}(\mathsf{pair}(a, k_3), k_4),\ \mathsf{pair}(k_1, k_2)), \\ a, \\ k_1, \\ \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1)) \end{array} \right\}$$

$$S \overset{?}{\vdash} k_1, S \overset{?}{\vdash} k_3,$$

Examples

$$S = \left\{ \begin{array}{l} \mathsf{enc}(\mathsf{pair}(\mathsf{pair}(a, k_3), k_4),\ \mathsf{pair}(k_1, k_2)), \\ a, \\ k_1, \\ \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1)) \end{array} \right\}$$

$$S \overset{?}{\vdash} k_1, S \overset{?}{\vdash} k_3,$$

$$\dfrac{\dfrac{k_1 \quad k_1}{\mathsf{pair}(k_1, k_1)} \qquad \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1))}{k_3}$$

Examples

$$S = \left\{ \begin{array}{l} \mathsf{enc}(\mathsf{pair}(\mathsf{pair}(a, k_3), k_4),\ \mathsf{pair}(k_1, k_2)), \\ a, \\ k_1, \\ \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1)) \end{array} \right\}$$

$$S \overset{?}{\vdash} k_1, S \overset{?}{\vdash} k_3, S \overset{?}{\vdash} \mathsf{pair}(a, k_3),$$

Examples

$$S = \left\{ \begin{array}{l} \mathsf{enc}(\mathsf{pair}(\mathsf{pair}(a, k_3), k_4),\ \mathsf{pair}(k_1, k_2)), \\ a, \\ k_1, \\ \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1)) \end{array} \right\}$$

$$S \overset{?}{\vdash} k_1, S \overset{?}{\vdash} k_3, S \overset{?}{\vdash} \mathsf{pair}(a, k_3),$$

$$\dfrac{\dfrac{k_1 \quad k_1}{\mathsf{pair}(k_1, k_1)} \qquad \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1))}{k_3}$$

Examples

$$S = \left\{ \begin{array}{l} \mathsf{enc}(\mathsf{pair}(\mathsf{pair}(a, k_3), k_4),\ \mathsf{pair}(k_1, k_2)), \\ a, \\ k_1, \\ \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1)) \end{array} \right\}$$

$$S \overset{?}{\vdash} k_1, S \overset{?}{\vdash} k_3, S \overset{?}{\vdash} \mathsf{pair}(a, k_3),$$

$$\dfrac{\dfrac{\dfrac{k_1 \quad k_1}{\mathsf{pair}(k_1, k_1)} \quad \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1))}{k_3} \quad a}{\mathsf{pair}(a, k_3)}$$

Examples

$$S = \left\{ \begin{array}{l} \mathsf{enc}(\mathsf{pair}(\mathsf{pair}(a, k_3), k_4),\ \mathsf{pair}(k_1, k_2)), \\ a, \\ k_1, \\ \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1)) \end{array} \right\}$$

$$S \stackrel{?}{\vdash} k_1, S \stackrel{?}{\vdash} k_3, S \stackrel{?}{\vdash} \mathsf{pair}(a, k_3), S \stackrel{?}{\vdash} k_4,$$

Examples

$$S = \left\{ \begin{array}{l} \mathsf{enc}(\mathsf{pair}(\mathsf{pair}(a, k_3), k_4),\ \mathsf{pair}(k_1, k_2)), \\ a, \\ k_1, \\ \mathsf{enc}(k_3, \mathsf{pair}(k_1, k_1)) \end{array} \right\}$$

$$S \stackrel{?}{\vdash} k_1, S \stackrel{?}{\vdash} k_3, S \stackrel{?}{\vdash} \mathsf{pair}(a, k_3), S \stackrel{?}{\vdash} k_4, S \stackrel{?}{\vdash} \mathsf{pair}(a, k_4)$$

Decision of the intruder problem

Given A set of messages S and a message m

Question Can the intruder learn m from S that is $S \vdash m$?

This problem is decidable in polynomial time. (left as exercice)

Decision of the intruder problem

Given A set of messages S and a message m

Question Can the intruder learn m from S that is $S \vdash m$?

This problem is decidable in polynomial time. (left as exercice)

Lemma (Locality)

If there is a proof of $S \vdash m$ then there is a proof that only uses the subterms of S and m.

Decision of the intruder problem

Given A set of messages S and a message m

Question Can the intruder learn m from S that is $S \vdash m$?

This problem is decidable in polynomial time. (left as exercice)

Lemma (Locality)

If there is a proof of $S \vdash m$ then there is a proof that only uses the subterms of S and m.

Induction hypothesis : If there is a (length minimal) proof of $S \vdash m$ then there is a proof that only uses the subterms of S and m,
Moreover
If there is a (length minimal) proof of $S \vdash m$ that ends with a decomposition rule then there is a proof that only uses the subterms of S.

How to decide security in the active case ?

How to decide security in the active case ?

- In general, it is undecidable !
  Easily encode two-counters machines, Post correspondence problem, etc (even for a passive adversary)

How to decide security in the active case ?

- In general, it is undecidable !
  Easily encode two-counters machines, Post correspondence problem, etc (even for a passive adversary)

- Bounded number of sessions
  - secrecy is (co)NP-complete [Rusinowitch, Turuani 2001]
  - tools : Avispa, Scyther (bounded and unbounded), ...

How to decide security in the active case ?

- In general, it is undecidable !
  Easily encode two-counters machines, Post correspondence problem, etc (even for a passive adversary)

- Bounded number of sessions
  - secrecy is (co)NP-complete [Rusinowitch, Turuani 2001]
  - tools : Avispa, Scyther (bounded and unbounded), ...

- Unbounded number of sessions
  How to circumvent undecidability ?

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

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

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

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

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 ?

$$\forall x \forall y \quad I(x), I(y) \Rightarrow I(< x, y >)$$
$$\forall x \forall y \quad I(x), I(y) \Rightarrow I(\{x\}_y)$$
$$\forall x \forall y \quad I(\{x\}_y), I(y) \Rightarrow I(x)$$
$$\forall x \forall y \quad I(< x, y >) \Rightarrow I(x)$$
$$\forall x \forall y \quad I(< x, y >) \Rightarrow I(y)$$

$$I(\{\text{secret}\}_{k_a})$$
$$\forall x \quad I(x) \Rightarrow I(\{x\}_{k_b})$$
$$\forall x \quad I(\{x\}_{k_a}) \Rightarrow I(x)$$

Security reduces to consistency

$\rightsquigarrow$

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

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

$$I(\{\text{secret}\}_{k_a})$$

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

secure ?

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"

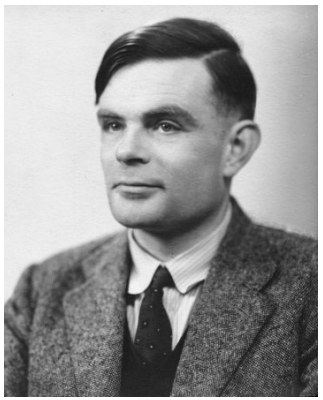
David Hilbert

How to know if a set of formula is consistent ?

Hilbert's program (1928)
"Entscheidung Problem"

David Hilbert

It is undecidable ! (1936)
$\rightarrow$ There is no algorithm that answers this question.



Alan Turing

(at a time with no computers)

Back to our business

secure ?

All this for nothing ?

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

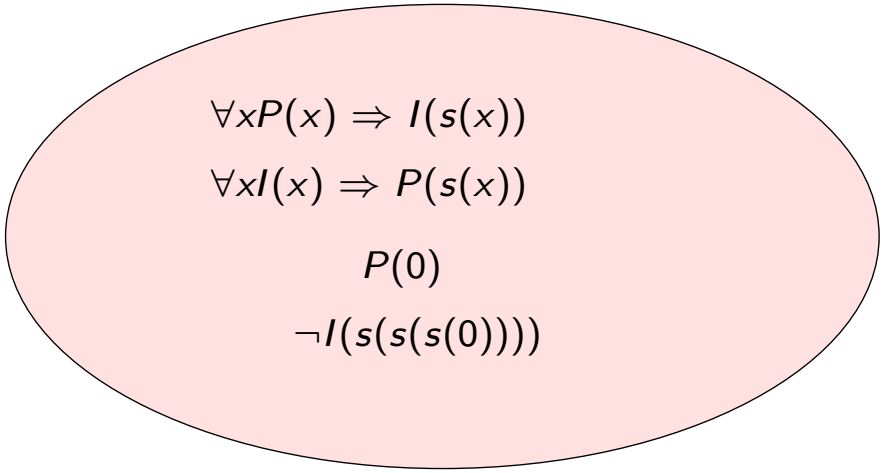
$$
\begin{array}{lrcl}
\forall x \forall y & I(x), I(y) & \Rightarrow & I(< x, y >) \\
\forall x \forall y & I(x), I(y) & \Rightarrow & I(\{x\}_y) \\
\forall x \forall y & I(\{x\}_y), I(y) & \Rightarrow & I(x) \\
\forall x \forall y & I(< x, y >) & \Rightarrow & I(x) \\
\forall x \forall y & I(< x, y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

Does not yield a contradiction ?

(i.e. consistent theory ?)

A standard technique : resolution

Idea : add logical consequences . . .



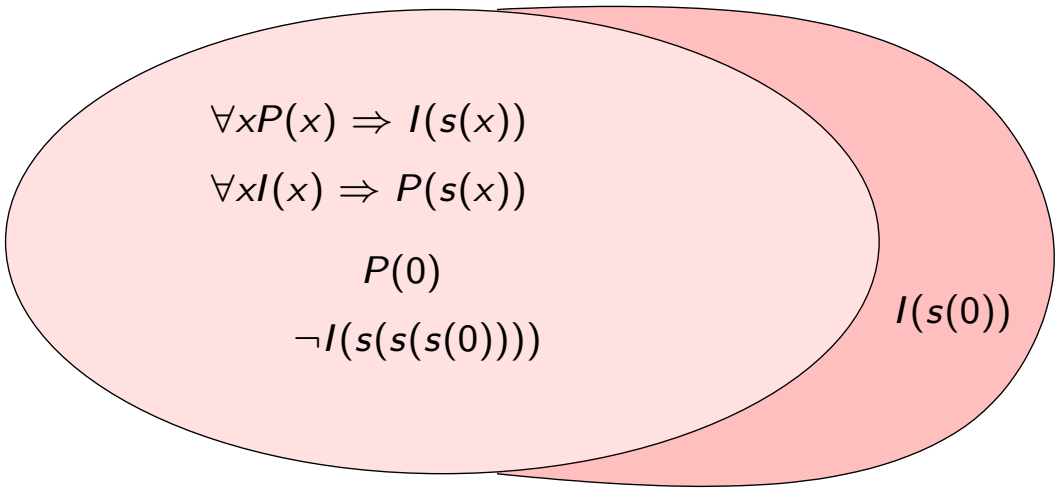
. . . until a contradiction is found.

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 . . .



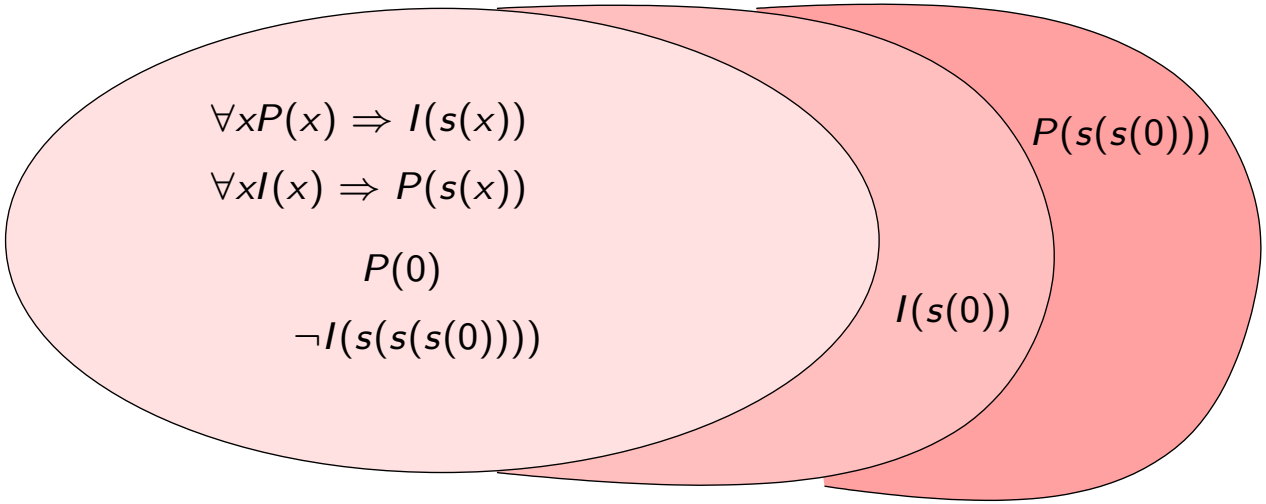
... until a contradiction is found.

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 . . .



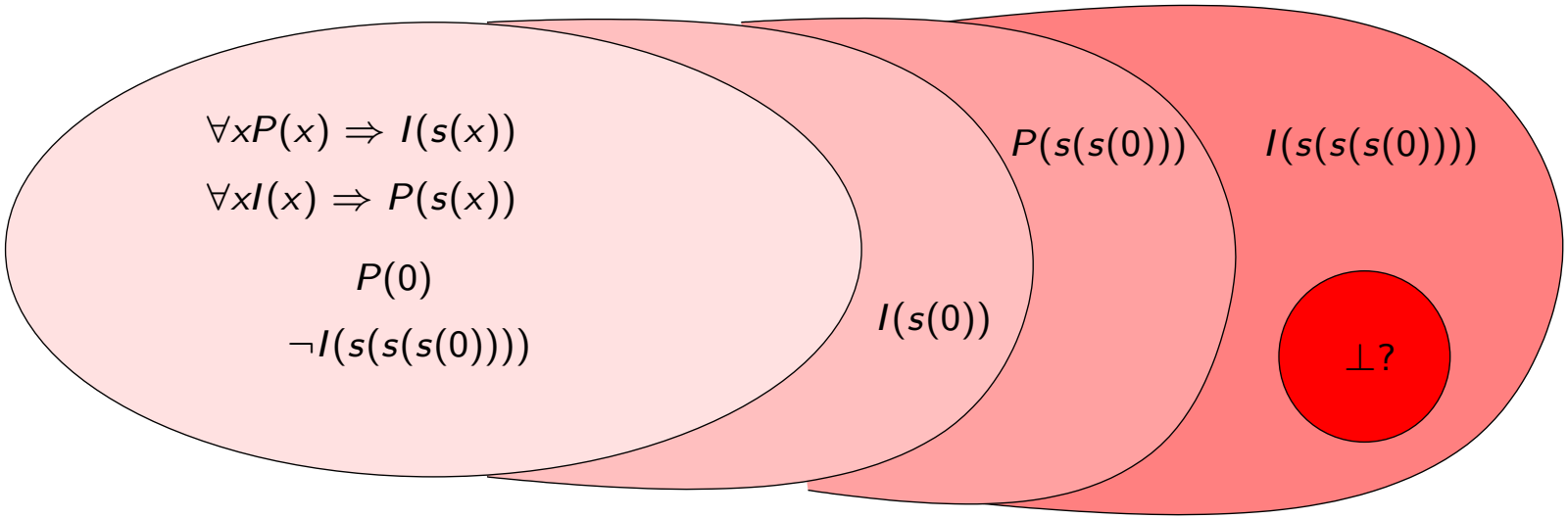
... until a contradiction is found.

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)

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 [Robinsson 1965])

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

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

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

But it is not terminating !

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

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

Ordered Binary resolution and Factorization

Let $<$ be any order on clauses.

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

Ordered binary resolution

$$\frac{A \vee B \vee C}{A\theta \vee C\theta} \quad \theta = \mathrm{mgu}(A, B), \; A\theta \not< C\theta$$

Ordered factorisation

Ordered Binary resolution and Factorization

Let $<$ be any order on clauses.

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

Ordered binary resolution

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

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 $u_1, \ldots, u_k$

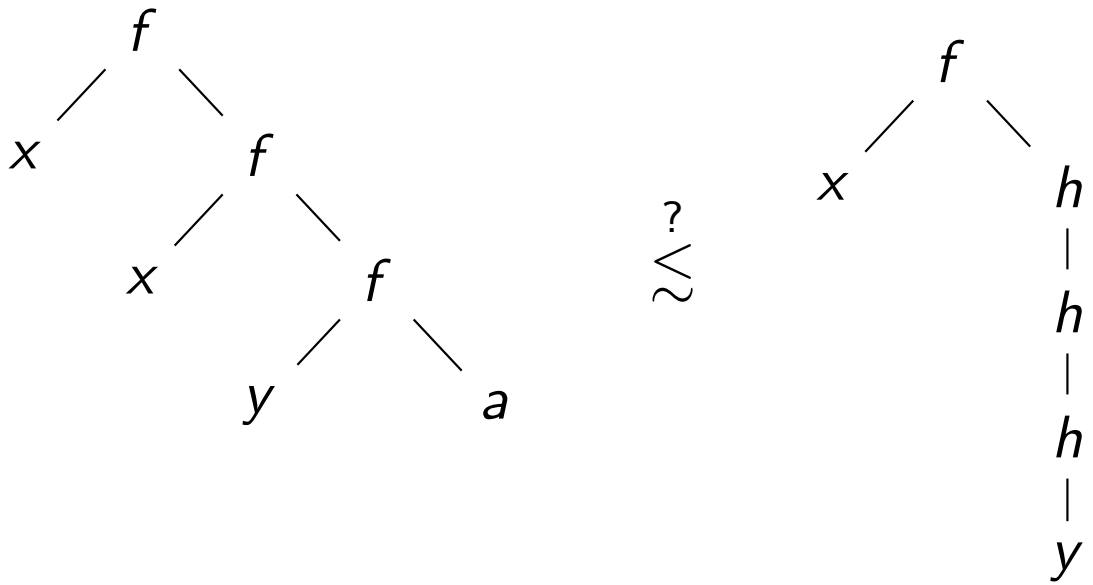
$\rightarrow$ extended to clauses as follows : $C_1 < C_2$ iff any literal of C_1 is smaller than some literal of C_2.

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

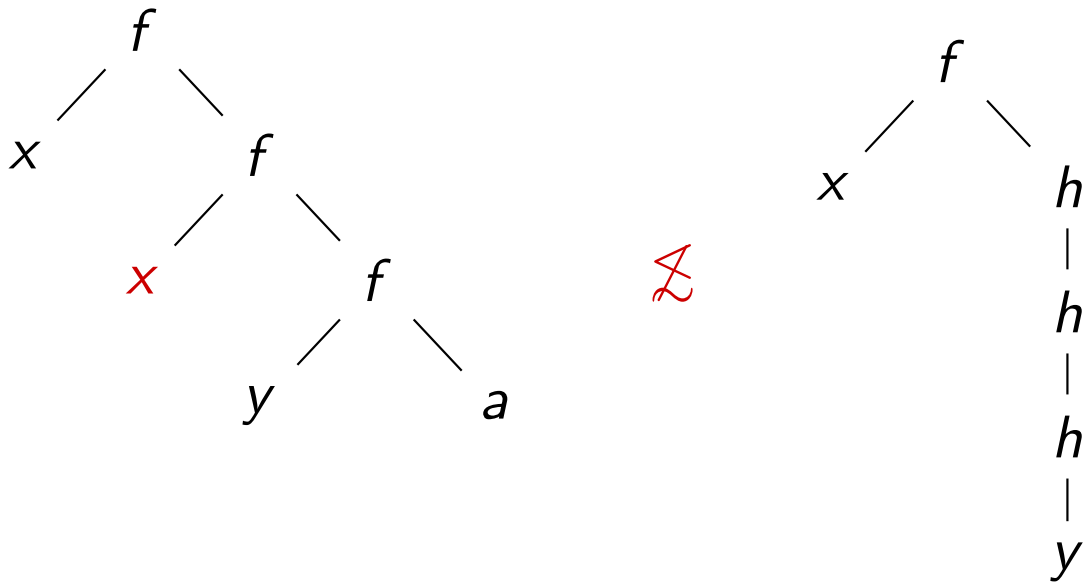
1. $\mathrm{depth}(P(t_1, \ldots, t_n)) \leq \mathrm{depth}(Q(u_1, \ldots, u_k))$
2. For any variable x,
   $\mathrm{depth}_x(P(t_1, \ldots, t_n)) \leq \mathrm{depth}_x(Q(u_1, \ldots, u_k))$



Examples of liftable orders - continued

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

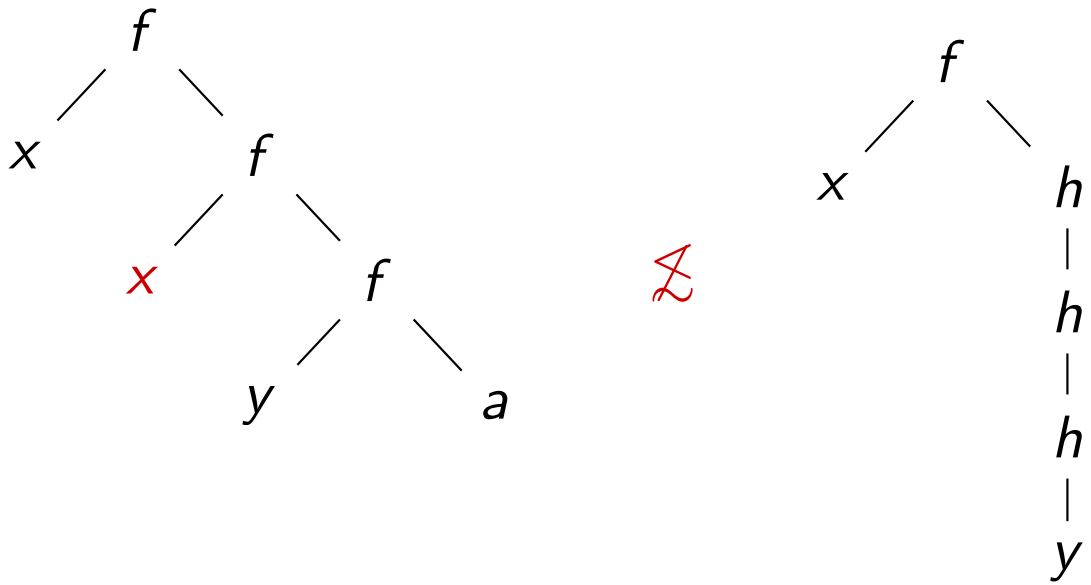
1. $\mathrm{depth}(P(t_1, \ldots, t_n)) \leq \mathrm{depth}(Q(u_1, \ldots, u_k))$
2. For any variable x,
   $\mathrm{depth}_x(P(t_1, \ldots, t_n)) \leq \mathrm{depth}_x(Q(u_1, \ldots, u_k))$



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

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

At most one variable per clause !

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

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

At most one variable per clause !

Theorem ([Comon, C. 2003])

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

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 $sel(H \Rightarrow C) \subseteq H$.

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

Theorem ([Bachmair, Ganziger 1992])

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

False attacks

Horn clauses make some abstractions w.r.t. process algebra.

which ones ?

False attacks

Horn clauses make some abstractions w.r.t. process algebra.
which ones ?

- The order of the actions is lost
- Nonces are abstracted by constants

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

False attacks

Horn clauses make some abstractions w.r.t. process algebra.

which ones ?

- The order of the actions is lost
- Nonces are abstracted by constants

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

Better :

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

(but still an over-approximation.)

Conclusion

Formal methods form a powerful approach for analyzing security protocols

- Makes use of classical techniques in formal methods : term algebra, equational theories, clauses and resolution techniques, tree automata, etc.
  $\Rightarrow$ Many decision procedures
- Several automatic tools
  - For successfully detecting attacks on protocols (e.g. Casper, Avispa)
  - For proving security for an arbitrary number of sessions (e.g. ProVerif)
- Provides cryptographic guarantees under classical assumptions on the implementation of the primitives

Some current directions of research

Enriching the symbolic model

- Considering more equational theories (e.g. theories for e-voting protocols)
- Adding more complex structures for data (list, XML, ...)
- Considering recursive protocols (e.g. group protocol) where the number of message exchanges in a session is not fixed
- Proving more complex security properties like equivalence-based properties (e.g. for anonymity or e-voting protocols)

Some bibliographical references (1/2)

A book that gathers several chapters on models and procedures for security protocols

Formal Models and Techniques for Analyzing Security Protocols. Cryptology and Information Security Series 5, IOS Press, 2011.

Notes of this course (and more) *Formal Models and Techniques for Analyzing Security Protocols : A Tutorial*. VÃ©ronique Cortier and Steve Kremer. Foundations and Trends in Programming Languages, 2014.

On protocols

- B. Schneier. *Applied Cryptography Second Edition : protocols, algorithms, and source code in C*, J. Wiley & Sons, Inc. publisher, 1996.
- A. J. Menezes and P. C. van Oorschot and S. A. Vanstone. *Handbook of applied cryptography*, CRC Press publisher, 1997.

Some bibliographical references (2/2)

Procedures on Horn clauses, for an unbounded number of sessions

- B. Blanchet. *An Efficient Cryptographic Protocol Verifier Based on Prolog Rules*. CSFW 2001.
- H. Comon-Lundh, V. Cortier. *New Decidability Results for Fragments of First-Order Logic and Application to Cryptographic Protocols*. RTA 2003.
- H. Comon, V. Cortier. *Tree automata with one memory set constraints and cryptographic protocols*. Theoretical Computer Science 2005.

Constraint solving for a bounded number of sessions

- J. K. Millen, V. Shmatikov. *Constraint solving for bounded-process cryptographic protocol analysis*. ACM Conference on Computer and Communications Security 2001.
- H. Comon-Lundh, V. Shmatikov. *Intruder Deductions, Constraint Solving and Insecurity Decision in Presence of Exclusive or*. LICS 2003