# Verification of Security Protocols
# Part I

Véronique Cortier[1]

September, 2010

Fosad 2010

[1]LORIA, CNRS

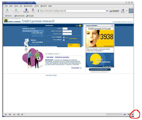
Two parts :

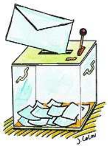
1. Analysis of security protocols with symbolic models
2. More guarantees : Analysis of security protocols with computational models

# Context : cryptographic protocols

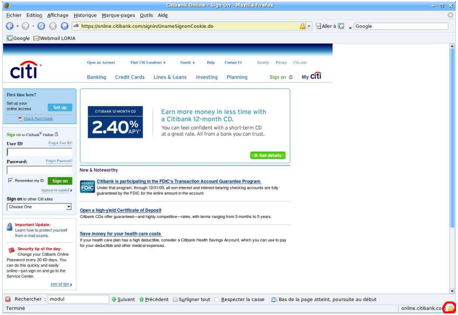
Cryptographic protocols are widely used in everyday life.

→ They aim at securing communications over public or insecure networks.

# On the web



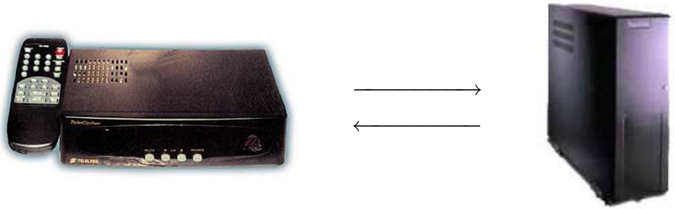

- HTTPS, i.e. the SSL protocol for ensuring confidentiality

- password-based authentication

# Credit Card payment

- It is a real card ?
- Is the pin code protected ?

# Pay-per-view devices



- Checks your identity
- You should be granted access to the movie only once
- You should not be able to broadcast the movie to other people

# 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

# Electronic purse

- It should not possible to add money without paying.
- It should not be possible to create fake electronic purse.

# 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 ! !)
- ...

⇒ Cryptographic protocols vary depending on the application.

# How does this work ?

# How does this work ?

A cryptographic protocol :

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

**Cryptographic** makes uses of cryptographic primitives (e.g. encryption, signatures, hashes, ...)

# Credit Card payment

- It is a real card ?
- Is the pin code protected ?

# Behavior in the usual case

- The waiter introduces the credit card.
- The waiter enters the amount $m$ of the transaction on the terminal.
- The terminal authenticates the card.
- The customer enters his secret code.
  If the amount $m$ is greater than 100 euros
  (and in only 20% of the cases)
  - The terminal asks the bank for authentication of the card.
  - The bank provides authentication.

# More details

4 actors : Bank, Customer, Card and Terminal.

Bank owns

- a signing key $K_B^{-1}$, secret,
- a verification key $K_B$, public,
- a secret symmetric key for each credit card $K_{CB}$, secret.

Card owns

- Data : last name, first name, card's number, expiration date,
- Signature's Value $VS = \{hash(\text{Data})\}_{K_B^{-1}}$,
- secret key $K_{CB}$.

Terminal owns the verification key $K_B$ for bank's signatures.

# Credit card payment Protocol (in short)

The terminal reads the card :

$$1. \quad Ca \quad \rightarrow \quad T : \text{Data}, \{hash(\text{Data})\}_{K_B^{-1}}$$

# Credit card payment Protocol (in short)

The terminal reads the card :

$$1. \quad Ca \quad \rightarrow \quad T : \mathrm{Data}, \{hash(\mathrm{Data})\}_{K_B^{-1}}$$

The terminal asks for the secret code :

$$\begin{array}{llll} 2. & T & \rightarrow & Cu : secret\ code? \\ 3. & Cu & \rightarrow & Ca : 1234 \\ 4. & Ca & \rightarrow & T : ok \end{array}$$

# Credit card payment Protocol (in short)

The terminal reads the card :

$$1. \quad Ca \rightarrow T : \mathrm{Data}, \{hash(\mathrm{Data})\}_{K_B^{-1}}$$

The terminal asks for the secret code :

$$\begin{array}{llll} 2. & T & \rightarrow & Cu : secret\ code? \\ 3. & Cu & \rightarrow & Ca : 1234 \\ 4. & Ca & \rightarrow & T : ok \end{array}$$

The terminal calls the bank :

$$\begin{array}{llll} 5. & T & \rightarrow & B : auth? \\ 6. & B & \rightarrow & T : N_b \\ 7. & T & \rightarrow & Ca : N_b \\ 8. & Ca & \rightarrow & T : \{N_b\}_{K_{CB}} \\ 9. & T & \rightarrow & B : \{N_b\}_{K_{CB}} \\ 10. & B & \rightarrow & T : ok \end{array}$$

# Some flaws

The security was initially ensured by :

- the cards were very difficult to reproduce,
- the protocol and the keys were secret.

But

- cryptographic flaw : 320 bits keys can be broken (1988),
- logical flaw : no link between the secret code and the authentication of the card,
- fake cards can be build.

# Some flaws

The security was initially ensured by :

- the cards were very difficult to reproduce,
- the protocol and the keys were secret.

But

- cryptographic flaw : 320 bits keys can be broken (1988),
- logical flaw : no link between the secret code and the authentication of the card,
- fake cards can be build.

→ "YesCard" build by Serge Humpich (1998 in France).

# How does the "YesCard" work ?

## Logical flaw

$$
\begin{array}{llcll}
1. & Ca & \rightarrow & T & : \mathrm{Data}, \{hash(\mathrm{Data})\}_{K_B^{-1}} \\
2. & T & \rightarrow & Ca & : secret\ code? \\
3. & Cu & \rightarrow & Ca & : 1234 \\
4. & Ca & \rightarrow & T & : ok
\end{array}
$$

# How does the "YesCard" work ?

## Logical flaw

$$
\begin{array}{llcll}
1. & Ca & \rightarrow & T & : \mathrm{Data}, \{hash(\mathrm{Data})\}_{K_B^{-1}} \\
2. & T & \rightarrow & Ca & : secret\ code? \\
3. & Cu & \rightarrow & Ca' & : 2345 \\
4. & Ca' & \rightarrow & T & : ok
\end{array}
$$

# How does the "YesCard" work ?

## Logical flaw

$$
\begin{array}{llll}
1. & Ca & \rightarrow T & : \mathrm{Data}, \{hash(\mathrm{Data})\}_{K_B^{-1}} \\
2. & T & \rightarrow Ca & : secret\ code? \\
3. & Cu & \rightarrow Ca' & : 2345 \\
4. & Ca' & \rightarrow T & : ok
\end{array}
$$

Remark : there is always somebody to debit.
$\longrightarrow$ creation of a fake card

# How does the "YesCard" work ?

## Logical flaw

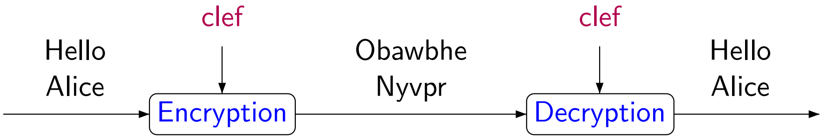
$$\begin{array}{llcll} 1. & Ca & \rightarrow & T & : \mathrm{Data}, \{hash(\mathrm{Data})\}_{K_B^{-1}} \\ 2. & T & \rightarrow & Ca & : secret\ code? \\ 3. & Cu & \rightarrow & Ca' & : 2345 \\ 4. & Ca' & \rightarrow & T & : ok \end{array}$$

Remark : there is always somebody to debit.
→ creation of a fake card

$$\begin{array}{llcll} 1. & Ca' & \rightarrow & T & : \mathrm{XXX}, \{hash(\mathrm{XXX})\}_{K_B^{-1}} \\ 2. & T & \rightarrow & Cu & : secret\ code? \\ 3. & Cu & \rightarrow & Ca' & : 0000 \\ 4. & Ca' & \rightarrow & T & : ok \end{array}$$

# Commutative Symmetric encryption

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




The same key is used for encrypting and decrypting.

Commutative (symmetric) encryption (e.g. RSA)

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

# Exchanging a secret with commutative encryption (RSA)

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

# Exchanging a secret with commutative encryption (RSA)

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

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

# Exchanging a secret with commutative encryption (RSA)

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

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

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


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

# Exchanging a secret with commutative encryption (RSA)

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

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

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


→ 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)

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

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

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

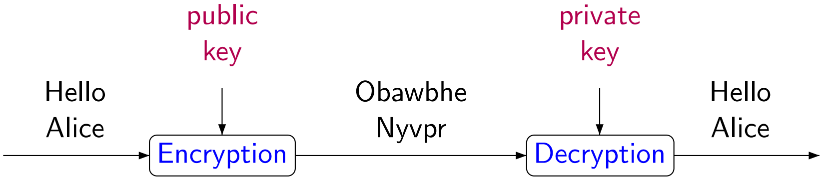
# Another example

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 $B$.

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

# Needham-Schroeder public key protocol

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

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

# Needham-Schroeder public key protocol

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

$$
\begin{array}{llr}
A & \rightarrow B : & \{A, N_a\}_{\mathrm{pub}(B)} \\
B & \rightarrow A : & \{N_a, N_b\}_{\mathrm{pub}(A)} \\
\bullet\ A & \rightarrow B : & \{N_b\}_{\mathrm{pub}(B)}
\end{array}
$$

# Needham-Schroeder public key protocol

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

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

Questions :

- Is $N_b$ secret between $A$ and $B$ ?
- When $B$ receives $\{N_b\}_{\mathrm{pub}(B)}$, 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 $B$.

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

Questions :

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

$\rightarrow$ An attack was discovered in 1994, 15 years after the publication of the protocol !

# Man in the middle attack

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

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

# Man in the middle attack

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

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

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

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

# Man in the middle attack

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

$\xleftarrow{\{\quad N_a,N_b\}_{\mathrm{pub}(A)}}$

$\xrightarrow{\{N_b\}_{\mathrm{pub}(P)}}$

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

$\xleftarrow{\{\quad N_a,N_b\}_{\mathrm{pub}(A)}}$

$\xrightarrow{\{N_b\}_{\mathrm{pub}(B)}}$

# Man in the middle attack

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

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

$\xrightarrow{\{N_b\}_{\mathrm{pub}(P)}}$

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

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

$\xrightarrow{\{N_b\}_{\mathrm{pub}(B)}}$

Fixing the flaw : add the identity of $B$.

# Outline of the talk

# Difficulty

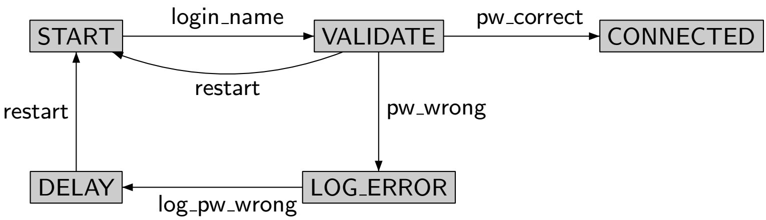
Presence of an attacker

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

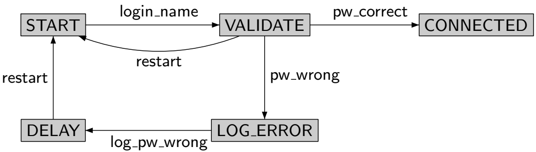
$\Rightarrow$ The system is infinitely branching

# 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.

→ It is important to have a tight modeling of the messages.

# An appropriate datastructure : Terms

Given a signature $\mathcal{F}$ of symbols with an arity

e.g. $\{\text{enc}, \text{pair}, a, b, c, n_a, n_b\}$

and a set $\mathcal{X}$ of variables,

the set of terms $T(\mathcal{F}, \mathcal{X})$ is inductively defined as follows :

- constants terms (e.g. $a, b, c, n_a, n_b$) are terms
- variables are terms
- $f(t_1, \ldots, t_n)$ is a term whenever $t_1, \ldots, t_n$ are terms.

Intuition : from words to trees.

$\rightarrow$ There exists automata on trees instead of (classical) automata on words, see e.g. TATA http ://tata.gforge.inria.fr/

# Messages

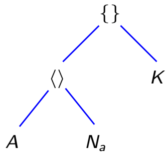
Messages are abstracted by terms.

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.

# Intruder abilities

## Composition rules

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u\ ,\, v \rangle} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathrm{enc}(u, v)} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathrm{enca}(u, v)}$$

# Intruder abilities

## Composition rules

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u\,,v \rangle} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathrm{enc}(u,v)} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathrm{enca}(u,v)}$$

## Decomposition rules

$$\frac{}{T \vdash u}\ u \in T \qquad \frac{T \vdash \langle u\,,v \rangle}{T \vdash u} \qquad \frac{T \vdash \langle u\,,v \rangle}{T \vdash v}$$

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

# Intruder abilities

## Composition rules

$$\frac{T \vdash u \quad T \vdash v}{T \vdash \langle u\,, v\rangle} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathrm{enc}(u, v)} \qquad \frac{T \vdash u \quad T \vdash v}{T \vdash \mathrm{enca}(u, v)}$$

## Decomposition rules

$$\frac{}{T \vdash u}\; u \in T \qquad \frac{T \vdash \langle u\,, v\rangle}{T \vdash u} \qquad \frac{T \vdash \langle u\,, v\rangle}{T \vdash v}$$

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

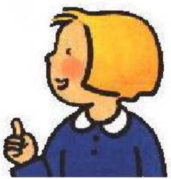
**Deducibility relation**

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

# A simple protocol


$$\xrightarrow{\langle \mathsf{Bob}, \mathsf{k} \rangle}$$

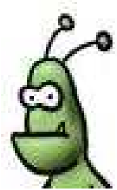
$$\xleftarrow{\langle \mathsf{Alice}, \mathsf{enc}(\mathsf{s}, \mathsf{k}) \rangle}$$



# A simple protocol


$$\xrightarrow{\langle \mathsf{Bob}, \mathsf{k} \rangle}$$

$$\xleftarrow{\langle \mathsf{Alice}, \mathsf{enc}(\color{red}{\mathsf{s}}, \mathsf{k}) \rangle}$$



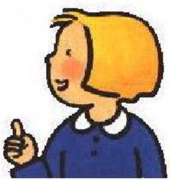
**Question ?**

Can the attacker learn the secret s ?

# A simple protocol

$$\xrightarrow{\quad \langle \mathrm{Bob}, \mathrm{k} \rangle \quad}$$

$$\xleftarrow{\quad \langle \mathrm{Alice}, \mathrm{enc}(\mathrm{s}, \mathrm{k}) \rangle \quad}$$



Answer : Of course, Yes !

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

# 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.

Exercise : (medium) Prove it.

# 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.

Exercise : (medium) Prove it.

**Lemma (Locality)**

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

# Protocol description

Protocol :

$$\begin{array}{rcl} A \rightarrow B & : & \{\text{pin}\}_{k_a} \\ B \rightarrow A & : & \{\{\text{pin}\}_{k_a}\}_{k_b} \\ A \rightarrow B & : & \{\text{pin}\}_{k_b} \end{array}$$

A protocol is a finite set of roles :

- role $\Pi(1)$ corresponding to the 1[st] participant played by $a$ talking to $b$ :

$$\begin{array}{rcl} \text{init} & \xrightarrow{k_a} & \text{enc}(\text{pin}, k_a) \\ \text{enc}(x, k_a) & \rightarrow & x. \end{array}$$

# Protocol description

Protocol :

$$
\begin{array}{rcl}
A \rightarrow B & : & \{\mathrm{pin}\}_{k_a} \\
B \rightarrow A & : & \{\{\mathrm{pin}\}_{k_a}\}_{k_b} \\
A \rightarrow B & : & \{\mathrm{pin}\}_{k_b}
\end{array}
$$

A protocol is a finite set of roles :

- role $\Pi(1)$ corresponding to the $1^{st}$ participant played by $a$ talking to $b$ :

$$
\begin{array}{rcl}
\mathrm{init} & \xrightarrow{k_a} & \mathrm{enc}(\mathrm{pin}, k_a) \\
\mathrm{enc}(x, k_a) & \rightarrow & x.
\end{array}
$$

- role $\Pi(2)$ corresponding to the $2^{nd}$ participant played by $b$ with $a$ :

$$
\begin{array}{rcl}
x & \xrightarrow{k_b} & \mathrm{enc}(x, k_b) \\
\mathrm{enc}(y, k_b) & \rightarrow & \mathrm{stop}.
\end{array}
$$

## Secrecy via constraint solving [Millen et al]

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

**Scenario**

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

**Constraint System**

$$\mathcal{C} = \left\{ \begin{array}{l} T_0 \Vdash u_1 \\ T_0, v_1 \Vdash u_2 \\ \quad \ldots \\ T_0, v_1, .., v_n \Vdash s \end{array} \right.$$

where $T_0$ is the initial knowledge of the attacker.

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

# Secrecy via constraint solving [Millen et al]

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

**Scenario**

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

**Constraint System**

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

where $T_0$ is the initial knowledge of the attacker.

## Solution of a constraint system

A substitution $\sigma$ such that

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

# Example of a system constraint

$$\begin{array}{lcl} A \rightarrow B & : & \{\mathrm{pin}\}_{k_a} \\ B \rightarrow A & : & \{\{\mathrm{pin}\}_{k_a}\}_{k_b} \\ A \rightarrow B & : & \{\mathrm{pin}\}_{k_b} \end{array}$$

and the attacker initially knows $T_0 = \{\mathrm{init}\}$.

One possible associated constraint system is :

$$\mathcal{C} = \begin{cases} \{\mathrm{init}\} \Vdash \mathrm{init} \\ \{\mathrm{init}, \{\mathrm{pin}\}_{k_a}\} \Vdash \{\mathrm{x}\}_{k_a} \\ \{\mathrm{init}, \{\mathrm{pin}\}_{k_a}, x\} \Vdash \mathrm{pin} \end{cases}$$

Is there a solution ?

# Example of a system constraint

$$\begin{array}{lcl} A \to B & : & \{\text{pin}\}_{k_a} \\ B \to A & : & \{\{\text{pin}\}_{k_a}\}_{k_b} \\ A \to B & : & \{\text{pin}\}_{k_b} \end{array}$$

and the attacker initially knows $T_0 = \{\text{init}\}$.

One possible associated constraint system is :

$$\mathcal{C} = \left\{ \begin{array}{l} \{\text{init}\} \Vdash \text{init} \\ \{\text{init}, \{\text{pin}\}_{k_a}\} \Vdash \{\text{x}\}_{k_a} \\ \{\text{init}, \{\text{pin}\}_{k_a}, x\} \Vdash \text{pin} \end{array} \right.$$

Is there a solution ?

Of course yes, simply consider $x = \text{pin}$ !

# Example of a system constraint

$$\begin{array}{lcl} A \rightarrow B & : & \{\text{pin}\}_{k_a} \\ B \rightarrow A & : & \{\{\text{pin}\}_{k_a}\}_{k_b} \\ A \rightarrow B & : & \{\text{pin}\}_{k_b} \end{array}$$

and the attacker initially knows $T_0 = \{\text{init}\}$.

One possible associated constraint system is :

$$\mathcal{C} = \begin{cases} \{\text{init}\} \Vdash \text{init} \\ \{\text{init}, \{\text{pin}\}_{k_a}\} \Vdash \{\text{x}\}_{k_a} \\ \{\text{init}, \{\text{pin}\}_{k_a}, x\} \Vdash \text{pin} \end{cases}$$

Is there a solution ?

Of course yes, simply consider $x = \text{pin}$ !

Exercise : (easy) Propose the constraint system associated to the (non-corrected) Needham-Schroeder protocol (for a reasonable choice of sessions) and exhibit a solution.

# How to solve constraint system ?

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

Question Is there a solution $\sigma$ of $\mathcal{C}$ ?

# An easy case : "solved constraint systems"

## General case

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

Question Is there a solution $\sigma$ of $\mathcal{C}$ ?

# An easy case : "solved constraint systems"

## General case

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

Question Is there a solution $\sigma$ of $\mathcal{C}$ ?

## Solved constraint systems

Given $\mathcal{C} = \begin{cases} T_0 \Vdash x_1 \\ T_0, v_1 \Vdash x_2 \\ \quad \dots \\ T_0, v_1, .., v_n \Vdash x_{n+1} \end{cases}$

Question Is there a solution $\sigma$ of $\mathcal{C}$ ?

# An easy case : "solved constraint systems"

## General case

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

Question Is there a solution $\sigma$ of $\mathcal{C}$ ?

## Solved constraint systems

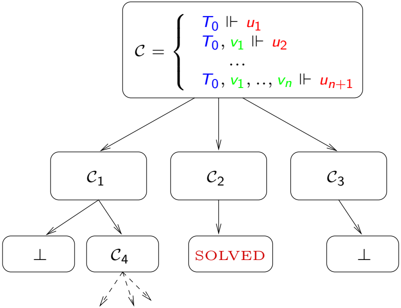
$$\text{Given } \mathcal{C} = \begin{cases} T_0 \Vdash x_1 \\ T_0, v_1 \Vdash x_2 \\ \quad \dots \\ T_0, v_1, .., v_n \Vdash x_{n+1} \end{cases}$$

Question Is there a solution $\sigma$ of $\mathcal{C}$ ?

Of course yes !

# Decision procedure [Millen / Comon-Lundh]

Goal : Transformation of the constraints in order to obtain a solved constraint system.




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

# Transformation rules

$R_1$ : $\mathcal{C} \wedge T \Vdash u \rightsquigarrow \mathcal{C}$ if $T \cup \{x \mid T' \Vdash x \in \mathcal{C}, T' \subsetneq T\} \vdash u$

$R_2$ : $\mathcal{C} \wedge T \Vdash u \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \Vdash u\sigma$ $u' \in st(T)$ if $\sigma = \mathrm{mgu}(u, u')$

$R_3$ : $\mathcal{C} \wedge T \Vdash v \rightsquigarrow_\sigma \mathcal{C}\sigma \wedge T\sigma \Vdash v\sigma$ $u, u' \in st(T)$ if $\sigma = \mathrm{mgu}(u, u')$

$R_4$ : $\mathcal{C} \wedge T \Vdash u \rightsquigarrow \perp$ if $\mathrm{var}(T, u) = \emptyset$ and $T \not\vdash u$

$R_5$ : $\mathcal{C} \wedge T \Vdash f(u, v) \rightsquigarrow \mathcal{C} \wedge T \Vdash u \wedge T \Vdash v$ for $f \in \{\langle\rangle, \mathrm{enc}\}$

# Intruder step

The intruder can built messages

$$R_5 : \quad \mathcal{C} \wedge T \Vdash f(u,v) \quad \rightsquigarrow \quad \mathcal{C} \wedge T \Vdash u \wedge T \Vdash v \quad \text{for } f \in \{\langle\rangle, \mathsf{enc}\}$$

# Intruder step

The intruder can built messages

$$R_5 : \quad \mathcal{C} \wedge T \Vdash f(u,v) \quad \rightsquigarrow \quad \mathcal{C} \wedge T \Vdash u \wedge T \Vdash v \quad \text{for } f \in \{\langle\rangle, \mathrm{enc}\}$$

Example :

$$a, k \Vdash \mathrm{enc}(\langle x, y\rangle, k) \quad \rightsquigarrow \quad \begin{array}{l} a, k \Vdash k \\ a, k \Vdash \langle x, y\rangle \end{array}$$

# Unsolvable constraints

$$R_4 : \mathcal{C} \wedge T \Vdash u \rightsquigarrow \perp \qquad \text{if } \mathrm{var}(T, u) = \emptyset \text{ and } T \not\vdash u$$

Example :

$$\begin{array}{lcl} \ldots & & \\ a, \mathrm{enc}(s, k) \Vdash s & \rightsquigarrow & \perp \\ \ldots & & \end{array}$$

## Guessing equalities

1. Example : $k, \mathrm{enc}(\mathrm{enc}(x, k'), k) \Vdash \mathrm{enc}(a, k')$

$$R_2 : \mathcal{C} \ \wedge \ T \Vdash u \rightsquigarrow_\sigma \mathcal{C}\sigma \ \wedge \ T\sigma \Vdash u\sigma \qquad u' \in st(T)$$
$$\text{if } \sigma = \mathrm{mgu}(u, u'),\ u, u' \notin \mathcal{X},\ u \neq u'$$

# Guessing equalities

1. Example : $k, \mathrm{enc}(\mathrm{enc}(x, k'), k) \Vdash \mathrm{enc}(a, k')$

$$R_2 : \mathcal{C} \;\wedge\; T \Vdash u \rightsquigarrow_\sigma \mathcal{C}\sigma \;\wedge\; T\sigma \Vdash u\sigma \qquad u' \in st(T)$$
$$\text{if } \sigma = \mathrm{mgu}(u, u'),\ u, u' \notin \mathcal{X},\ u \neq u'$$

2. Example : $\mathrm{enc}(s, \langle a, x\rangle), \mathrm{enc}(\langle y, b\rangle, k), k \Vdash s$

$$R_3 : \mathcal{C} \;\wedge\; T \Vdash v \rightsquigarrow_\sigma \mathcal{C}\sigma \;\wedge\; T\sigma \Vdash v\sigma \qquad u, u' \in st(T)$$
$$\text{if } \sigma = \mathrm{mgu}(u, u'),\ u, u' \notin \mathcal{X},\ u \neq u'$$

# Eliminating redundancies

$$k \Vdash x$$
$$k, \mathsf{enc}(s, x) \Vdash s$$

The constraint $\mathsf{enc}(s, x) \Vdash s$ will be satisfied as soon as $k \Vdash x$ is satisfied.

# Eliminating redundancies

$$k \Vdash x$$
$$k, \mathrm{enc}(s, x) \Vdash s$$

The constraint $\mathrm{enc}(s, x) \Vdash s$ will be satisfied as soon as $k \Vdash x$ is satisfied.

$$R_1 : \mathcal{C} \wedge T \Vdash u \rightsquigarrow \mathcal{C} \quad \text{if } T \cup \{x \mid T' \Vdash x \in \mathcal{C}, T' \subsetneq T\} \vdash u$$

# Soundness and completeness

**Theorem**

**Soundness** *If* $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$ *and* $\theta$ *solution of* $\mathcal{C}'$ *then* $\sigma\theta$ *is a solution of* $\mathcal{C}$.

**Completeness** *If* $\theta$ *solution of* $\mathcal{C}$ *then there exists* $\mathcal{C}', \sigma, \theta'$ *such that* $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$, $\theta = \sigma\theta'$ *and* $\theta'$ *is a solution of* $\mathcal{C}$.

**Termination** $\rightsquigarrow$ *is terminating in polynomial time in the size of* $\mathcal{C}$.

# Soundness and completeness

**Theorem**

Soundness *If* $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$ *and* $\theta$ *solution of* $\mathcal{C}'$ *then* $\sigma\theta$ *is a solution of* $\mathcal{C}$.

Completeness *If* $\theta$ *solution of* $\mathcal{C}$ *then there exists* $\mathcal{C}', \sigma, \theta'$ *such that* $\mathcal{C} \rightsquigarrow_\sigma \mathcal{C}'$, $\theta = \sigma\theta'$ *and* $\theta'$ *is a solution of* $\mathcal{C}$.

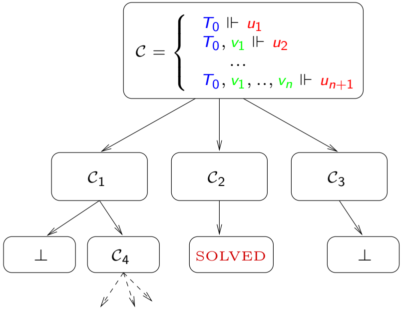
Termination $\rightsquigarrow$ *is terminating in polynomial time in the size of* $\mathcal{C}$.

Exercise (easy) : show correctness

Exercise (easy) : show termination using the lexicographic order (number of var, size of $\mathcal{C}$). What complexity do you get ?
(More involved) : show termination in polynomial time

Full proofs in [TOCL 2010]

# NP-procedure for solving constraint systems

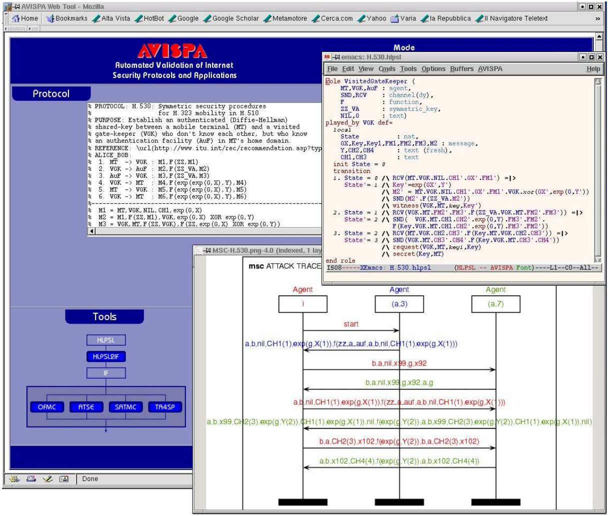



## Corollary

*Checking secrecy for a bounded number of sessions is NP.*

NP-hardness can be shown by encoding 3-SAT.

# Example of tool : Avispa Platform




www.avispa-project.org

Collaborators

- LORIA, France
- DIST, Italy
- ETHZ, Switzerland
- Siemens, Germany

# Limitations of this approach ?

Are you ready to use any protocol verified with this technique ?

# Limitations of this approach ?

Are you ready to use any protocol verified with this technique ?

- Only a finite scenario is checked.
  $\longrightarrow$ What happens if the protocol is used one more time ?
- The underlying mathematical properties of the primitives are abstracted away.
- The specification of the protocol is analysed, but not its implementation.

# How to decide security for unlimited sessions ?

$\longrightarrow$ 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 ?

$\rightarrow$ 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.

$bababab$

$bababab$

# How to encode PCP in protocols ?

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

$$
\begin{array}{rl}
A \quad \rightarrow B: & \{\langle\overline{u_1}, \overline{v_1}\rangle\}_{K_{ab}}, \ldots, \{\langle\overline{u_k}, \overline{v_k}\rangle\}_{K_{ab}} \\
B: \{\langle x, y\rangle\}_{K_{ab}} \quad \rightarrow A: & \{\langle\overline{x, u_1}, \overline{y, v_1}\rangle\}_{K_{ab}}, \{s\}_{\{\langle\overline{x,u_1},\overline{x,u_1}\rangle\}_{K_{ab}}}, \\
& \ldots, \{\langle\overline{x, u_k}, \overline{y, v_k}\rangle\}_{K_{ab}}, \{s\}_{\{\langle\langle\overline{x,u_k},\overline{x,u_k}\rangle\}_{K_{ab}}}
\end{array}
$$

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

# How to encode PCP in protocols ?

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

$$
\begin{array}{rll}
A & \to B : & \{\langle\overline{u_1}, \overline{v_1}\rangle\}_{K_{ab}}, \ldots, \{\langle\overline{u_k}, \overline{v_k}\rangle\}_{K_{ab}} \\
B : \{\langle x, y\rangle\}_{K_{ab}} & \to A : & \{\langle\overline{x, u_1}, \overline{y, v_1}\rangle\}_{K_{ab}}, \{s\}_{\{\langle\overline{x, u_1}, \overline{x, u_1}\rangle\}_{K_{ab}}}, \\
& & \ldots, \{\langle\overline{x, u_k}, \overline{y, v_k}\rangle\}_{K_{ab}}, \{s\}_{\{\langle\langle\overline{x, u_k}, \overline{x, u_k}\rangle\}_{K_{ab}}}
\end{array}
$$

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

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* $\mathrm{enc}(x, k_a)$ *then* $A$ *responds with* $x$*."*

$\longrightarrow$ Use of first-order logic.

# Intruder

Horn clauses perfectly reflects the attacker symbolic manipulations on terms.

$$
\begin{array}{rcll}
I(x), I(y) & \Rightarrow & I(<x, y>) & \text{pairing} \\
I(x), I(y) & \Rightarrow & I(\{x\}_y) & \text{encryption} \\
I(\{x\}_y), I(y) & \Rightarrow & I(x) & \text{decryption} \\
I(<x, y>) & \Rightarrow & I(x) & \text{projection} \\
I(<x, y>) & \Rightarrow & I(y) & \text{projection}
\end{array}
$$

# Protocol

Protocol :

$$A \rightarrow B \quad : \quad \{pin\}_{k_a}$$
$$B \rightarrow A \quad : \quad \{\{pin\}_{k_a}\}_{k_b}$$
$$A \rightarrow B \quad : \quad \{pin\}_{k_b}$$

Horn clauses :

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

# Protocol

Protocol :

$$A \rightarrow B \quad : \quad \{\mathrm{pin}\}_{k_a}$$
$$B \rightarrow A \quad : \quad \{\{\mathrm{pin}\}_{k_a}\}_{k_b}$$
$$A \rightarrow B \quad : \quad \{\mathrm{pin}\}_{k_b}$$

Horn clauses :

$$\Rightarrow \quad I(\{\mathrm{pin}\}_{k_a})$$
$$I(x) \quad \Rightarrow \quad I(\{x\}_{k_b})$$
$$I(\{x\}_{k_a}) \quad \Rightarrow \quad I(x)$$

Secrecy property is a reachability (accessibility) property

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

Then there exists an attack iff the set of formula corresponding to
Intruder manipulations + protocol + property
is NOT satisfiable.

How to decide satisfiability ?

$\longrightarrow$ Resolution techniques

# 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 $\vee, \wedge, \neg, \exists, \forall$

Clauses : Only universal quantifiers

Horn Clauses : at most one positive literal

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

where $A_i, B$ are atoms.

# 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

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

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

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

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

# But it is not terminating !

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

$\rightarrow$ 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.

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

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 $u_1, \ldots, u_k$

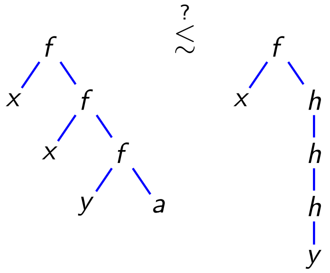
$\rightarrow$ extended to clauses as follows : $C_1 < C_2$ iff any literal of $C_1$ is smaller than some literal of $C_2$.

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

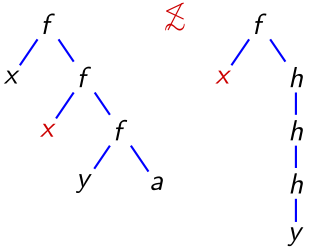
1. $\mathrm{depth}(P(t_1, \ldots, t_n)) \leq \mathrm{depth}(Q(u_1, \ldots, u_k))$
2. For any variable $x$,
   $\mathrm{depth}_x(P(t_1, \ldots, t_n)) \leq \mathrm{depth}_x(Q(u_1, \ldots, u_k))$



# Examples of liftable orders - continued

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

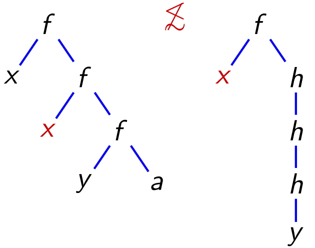
1. $\mathrm{depth}(P(t_1, \ldots, t_n)) \leq \mathrm{depth}(Q(u_1, \ldots, u_k))$
2. For any variable $x$,
   $\mathrm{depth}_x(P(t_1, \ldots, t_n)) \leq \mathrm{depth}_x(Q(u_1, \ldots, u_k))$



# Examples of liftable orders - continued

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

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




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

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

Protocol clauses

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

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

*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)

# 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]
  $\longrightarrow$ several tools for detecting attacks (Casper, Avispa platform... )

# 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]
  $\longrightarrow$ several tools for detecting attacks (Casper, Avispa platform... )
- 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]

  $\longrightarrow$ some tools for proving security (ProVerif, EVA Platform)