Cryptographic protocols

- small programs designed to secure communication
- use cryptographic primitives (e.g., encryption, hash function, ...)

cliquez ici pour accéder à la signature de votre déclaration
Goals

- **Secrecy**: May an intruder learn some secret message between two honest participants?

- **Authentication**: Is the agent Alice really talking to Bob?
Goals

- **Secrecy:** May an intruder learn some secret message between two honest participants?

- **Authentication:** Is the agent Alice really talking to Bob?

- **Fairness:** Alice and Bob want to sign a contract. Alice initiates the protocol. May Bob obtain some advantage?

- **Privacy:** Alice participate to an election. May a participant learn something about the vote of Alice?

- **Receipt-Freeness:** Alice participate to an election. Does Alice gain any information (a receipt) which can be used to prove to a coercer that she voted in a certain way?

- ...
Needham-Schroeder’s Protocol (1978)

- \(A \rightarrow B : \{A, N_a\}_{\text{pub}(B)}\)
- \(B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)}\)
- \(A \rightarrow B : \{N_b\}_{\text{pub}(B)}\)
Needham-Schroeder’s Protocol (1978)

\[ A \rightarrow B : \{ A, N_a \}_{\text{pub}(B)} \]
\[ B \rightarrow A : \{ N_a, N_b \}_{\text{pub}(A)} \]
\[ A \rightarrow B : \{ N_b \}_{\text{pub}(B)} \]
Needham-Schroeder’s Protocol (1978)

\[
\begin{align*}
A & \rightarrow B : \quad \{A, N_a\}_{\text{pub}(B)} \\
B & \rightarrow A : \quad \{N_a, N_b\}_{\text{pub}(A)} \\
\bullet \quad A & \rightarrow B : \quad \{N_b\}_{\text{pub}(B)}
\end{align*}
\]
Needham-Schroeder’s Protocol (1978)

\[ A \rightarrow B : \{A, N_a\}_{\text{pub}(B)} \]

\[ B \rightarrow A : \{N_a, N_b\}_{\text{pub}(A)} \]

\[ A \rightarrow B : \{N_b\}_{\text{pub}(B)} \]
Needham-Schroeder’s Protocol (1978)

\[\begin{align*}
A \rightarrow B : & \quad \{A, N_a\}_{\text{pub}(B)} \\
B \rightarrow A : & \quad \{N_a, N_b\}_{\text{pub}(A)} \\
A \rightarrow B : & \quad \{N_b\}_{\text{pub}(B)}
\end{align*}\]

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’s Protocol (1978)

\[
\begin{align*}
A & \rightarrow B : \quad \{A, N_a\}_{\text{pub}(B)} \\
B & \rightarrow A : \quad \{N_a, N_b\}_{\text{pub}(A)} \\
A & \rightarrow B : \quad \{N_b\}_{\text{pub}(B)}
\end{align*}
\]

Questions

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

Attack

An attack was found 17 years after its publication! [Lowe 96]
Verification of cryptographic protocols

How cryptographic protocols can be attacked?

Breaking encryption
Verification of cryptographic protocols

How cryptographic protocols can be attacked?

Breaking encryption

Logical attack
Verification of cryptographic protocols

How cryptographic protocols can be attacked?

Breaking encryption

Logical attack

Logical attacks

- can be mounted even assuming perfect cryptography,
  \[ \rightarrow \text{replay attack, man-in-the-middle attack}, \ldots \]

- are numerous, see SPORE, Security Protocols Open REpository
  \[ \rightarrow \text{http://www.lsv.ens-cachan.fr/spore/} \]

- subtle and hard to detect by “eyeballing” the protocol
Example: Man in the Middle Attack

Agent A  Intrus I  Agent B

Attack

- involving 2 sessions in parallel,
- an honest agent has to initiate a session with I.

\[
\begin{align*}
A \rightarrow B &: \{A, N_a\}_{\text{pub}(B)} \\
B \rightarrow A &: \{N_a, N_b\}_{\text{pub}(A)} \\
A \rightarrow B &: \{N_b\}_{\text{pub}(B)}
\end{align*}
\]
Example: Man in the Middle Attack

\[
\{A, N_a\}_{\text{pub}(I)} \rightarrow \{A, N_a\}_{\text{pub}(B)}
\]

Agent \textit{A} \quad \text{Intrus} \textit{I} \quad \text{Agent \textit{B}}

\[
\begin{align*}
A &\rightarrow B : \{A, N_a\}_{\text{pub}(B)} \\
B &\rightarrow A : \{N_a, N_b\}_{\text{pub}(A)} \\
A &\rightarrow B : \{N_b\}_{\text{pub}(B)}
\end{align*}
\]
Example: Man in the Middle Attack

\[ \{A, N_a\}_{\text{pub}(I)} \rightarrow \{N_a, N_b\}_{\text{pub}(A)} \]

Agent A

Intrus I

\[ \{A, N_a\}_{\text{pub}(B)} \rightarrow \{N_a, N_b\}_{\text{pub}(A)} \]

Agent B

\[
\begin{align*}
A \rightarrow B & : \{A, N_a\}_{\text{pub}(B)} \\
B \rightarrow A & : \{N_a, N_b\}_{\text{pub}(A)} \\
A \rightarrow B & : \{N_b\}_{\text{pub}(B)}
\end{align*}
\]
Example: Man in the Middle Attack

\[
\begin{align*}
\text{Agent } A & \quad \{A, N_a\}_{\text{pub}(I)} & \quad \{N_a, N_b\}_{\text{pub}(A)} & \quad \{N_b\}_{\text{pub}(I)} & \quad \{A, N_a\}_{\text{pub}(B)} & \quad \{N_a, N_b\}_{\text{pub}(A)} & \quad \{N_b\}_{\text{pub}(B)} \\
\text{Intrus } I & \quad & & & & & \\
\text{Agent } B & & & & & & \\
\end{align*}
\]

\[
\begin{align*}
A \rightarrow B & : \{A, N_a\}_{\text{pub}(B)} \\
B \rightarrow A & : \{N_a, N_b\}_{\text{pub}(A)} \\
A \rightarrow B & : \{N_b\}_{\text{pub}(B)}
\end{align*}
\]
Example: Man in the Middle Attack

\[
\begin{align*}
\{A, N_a\}_{\text{pub}(I)} & \quad \rightarrow \quad \{A, N_a\}_{\text{pub}(B)} \\
\{N_a, N_b\}_{\text{pub}(A)} & \quad \leftarrow \quad \{N_a, N_b\}_{\text{pub}(A)} \\
\{N_b\}_{\text{pub}(I)} & \quad \rightarrow \quad \{N_b\}_{\text{pub}(B)}
\end{align*}
\]

Agent A  Intrus I  Agent B

**Attack**

- the intruder knows $N_b$,
- When B finishes his session (apparently with A), A has never talked with B.

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

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

A $\rightarrow$ B : $\{N_b\}_{\text{pub}(B)}$
Logical attacks - How to detect them?

Symbolic approach

- **messages** are represented by **terms** rather than bit-strings
  \[ \{ m \}_k \] encryption of the message \( m \) with key \( k \),
  \[ \langle m_1, m_2 \rangle \] pairing of messages \( m_1 \) and \( m_2 \), …

- **attacker** controls the network and can perform **specific actions**
Logical attacks - How to detect them?

Symbolic approach

- messages are represented by terms rather than bit-strings
  \[ \{m\}_k \] encryption of the message \( m \) with key \( k \),
  \[ \langle m_1, m_2 \rangle \] pairing of messages \( m_1 \) and \( m_2 \), ...

- attacker controls the network and can perform specific actions

Relevance of the approach

- numerous attacks have already been obtained,
- allows us to perform automatic verification, e.g. AVISPA, Proverif, ...
- soundness results already exist, e.g. [Micciancio & Warinschi’04]
Difficulties of the verification

Presence of an attacker ...
Difficulties of the verification

Presence of an attacker ...

who controls the communication network:

- may read every message sent on the network
- may intercept and send new messages
Difficulties of the verification

Presence of an attacker ...

who controls the communication network:

- may read every message sent on the network
- may intercept and send new messages

who has deduction capabilities (e.g. the standard Dolev-Yao model)

- encryption, decryption if he knows the decryption key,
- pairing, projection
Difficulties of the verification

Presence of an attacker ...

who controls the communication network:
- may read every message sent on the network
- may intercept and send new messages

who has deduction capabilities (e.g. the standard Dolev-Yao model)
- encryption, decryption if he knows the decryption key,
- pairing, projection

Security problem for an unbounded number of sessions is undecidable.
Difficulties of the verification

Presence of an attacker ...

who controls the communication network:
- may **read** every message sent on the network
- may **intercept** and **send** new messages

who has deduction capabilities (**e.g.** the standard Dolev-Yao model)
- encryption, decryption if he knows the decryption key,
- pairing, projection

Security problem for a **fixed** number of sessions is **decidable**.
Outline of the talk

1. Introduction

2. How to deal with trace properties? (e.g. secrecy, authentication)

3. Work in progress: Equivalence based security properties (e.g. anonymity)
Outline of the talk

1. Introduction

2. How to deal with trace properties? (e.g. secrecy, authentication)

3. Work in progress: Equivalence based security properties (e.g. anonymity)
$m_1$, $m_2$ and $A$ are messages (terms)
$T$ a finite set of messages (intruder’s knowledge)

**Ax. (A)**

\[
\frac{}{T \vdash m_1 \quad m_1 \in T}
\]

**Enc. (E)**

\[
\frac{T \vdash m_1 \quad T \vdash \text{pub}(A)}{T \vdash \{m_1\}_{\text{pub}(A)}}
\]

**Dec. (D)**

\[
\frac{T \vdash \{m_1\}_{\text{pub}(A)} \quad T \vdash \text{priv}(A)}{T \vdash m_1}
\]

**Pair (P)**

\[
\frac{T \vdash m_1 \quad T \vdash m_2}{T \vdash \langle m_1, m_2 \rangle}
\]

**Proj. (Prj$_2$)**

\[
\frac{T \vdash \langle m_1, m_2 \rangle}{T \vdash m_2}
\]

**Proj. (Prj$_1$)**

\[
\frac{T \vdash \langle m_1, m_2 \rangle}{T \vdash m_1}
\]
Deducibility problem

<table>
<thead>
<tr>
<th>Deducibility problem</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INPUT:</strong> an intruder inference system $\mathcal{I}$, a finite set of terms $T$, a term $s$ (the secret).</td>
</tr>
<tr>
<td><strong>OUTPUT:</strong> Does there exist a proof of $T \vdash s$?</td>
</tr>
</tbody>
</table>
Deducibility problem

**INPUT:** an intruder inference system $\mathcal{I}$, a finite set of terms $T$, a term $s$ (the secret).

**OUTPUT:** Does there exist a proof of $T \vdash s$?

**Example:** Is $\langle s_1, s_2 \rangle$ deducible from the set of terms $T$ which contains $s_1$, $\{s_2\}_k$ and $k$?
**Deducibility problem**

**INPUT:** an intruder inference system \( \mathcal{I} \), a finite set of terms \( T \), a term \( s \) (the secret).

**OUTPUT:** Does there exist a proof of \( T \vdash s \)?

**Example:** Is \( \langle s_1, s_2 \rangle \) deducible from the set of terms \( T \) which contains \( s_1 \), \( \{s_2\}_k \) and \( k \)?

\[
\begin{align*}
  & s_1 \in T \\
  \frac{\{s_2\}_k \in T}{T \vdash \{s_2\}_k} \quad (A) \\
  \frac{\{s_2\}_k \in T \quad k \in T}{T \vdash k} \quad (A) \\
  \frac{T \vdash k}{T \vdash s_2} \quad (D) \\
  \frac{T \vdash s_1}{T \vdash \langle s_1, s_2 \rangle} \quad (P)
\end{align*}
\]
Deducibility problem - Some existing results

→ depends on the deduction capabilities of the intruder

Dolev-Yao intruder

The deducibility problem is decidable in polynomial time.
Deducibility problem - Some existing results

→ depends on the deduction capabilities of the intruder

Dolev-Yao intruder

The deducibility problem is decidable in polynomial time.

Prefix Intruder (e.g. Cipher Block Chaining)

\[
\begin{align*}
T & \vdash \{<m_1, m_2>\}_{\text{pub}(A)} \\
T & \vdash \{m_1\}_{\text{pub}(A)}
\end{align*}
\]
Deducibility problem - Some existing results

→ depends on the deduction capabilities of the intruder

Dolev-Yao intruder

The deducibility problem is decidable in polynomial time.

Prefix Intruder (e.g. Cipher Block Chaining)

\[
T \vdash \{\langle m_1, m_2 \rangle\}_{pub(A)} \quad T \vdash \{m_1\}_{pub(A)}
\]

Taking into account algebraic properties of the cryptographic primitives (e.g. RSA encryption)

\[
E := \begin{cases} 
    \text{dec(\text{enc}(x, pub(y)), priv(y))} = x \\
    \text{enc(\text{dec}(x, priv(y)), pub(y))} = x 
\end{cases}
\]

\[
\begin{array}{c}
T \vdash m \\
T \vdash k
\end{array} \quad f \in \{\text{dec, enc}\} \quad \begin{array}{c}
T \vdash m_1 \\
T \vdash m_2
\end{array} \quad m_1 \equiv_E m_2
\]
In presence of an active attacker

Insecurity problem (bounded number of sessions)

Let $I$ be an inference system modelling the attacker.

INPUT: a finite set $R_1, \ldots, R_m$ of instances of roles,
a finite set $T_0$ of terms (initial intruder knowledge),
a term $s$ (the secret)
In presence of an active attacker

Insecurity problem (bounded number of sessions)

Let $\mathcal{I}$ be an inference system modelling the attacker.

INPUT: a finite set $R_1, \ldots, R_m$ of instances of roles,
a finite set $T_0$ of terms (initial intruder knowledge),
a term $s$ (the secret)

OUTPUT: Does there exist an interleaving of $R_1, \ldots, R_m$
runnable from $T_0$ w.r.t. $\mathcal{I}$ at the end of which

- the intruder knowledge is $T$, and
- $s$ is deducible from $T$ in $\mathcal{I}$?
In presence of an active attacker

Insecurity problem (bounded number of sessions)

Let $\mathcal{I}$ be an inference system modelling the attacker.

INPUT: a finite set $R_1, \ldots, R_m$ of instances of roles,
a finite set $T_0$ of terms (initial intruder knowledge),
a term $s$ (the secret)

OUTPUT: Does there exist an interleaving of $R_1, \ldots, R_m$
runnable from $T_0$ w.r.t. $\mathcal{I}$ at the end of which
- the intruder knowledge is $T$, and
- $s$ is deducible from $T$ in $\mathcal{I}$?

Security properties (trace properties): e.g. secrecy, some kinds of authentication properties, …
Running example: Needham-Schroeder’s protocol

\[ A \rightarrow B \quad : \quad \{ A, N_a \}_{\text{pub}(B)} \]
\[ B \rightarrow A \quad : \quad \{ N_a, N_b \}_{\text{pub}(A)} \]
\[ A \rightarrow B \quad : \quad \{ N_b \}_{\text{pub}(B)} \]

Roles composing the protocol

\[ R_A(x_a, x_b) \quad : \quad \nu n_a. \quad \text{out}(\{ x_a, n_a \}_{\text{pub}(x_b)}); \]
\[ \quad \text{in}(\{ n_a, x_n_b \}_{\text{pub}(x_a)}); \text{out}(\{ x_n_b \}_{\text{pub}(x_b)}) \]

\[ R_B(y_b) \quad : \quad \nu n_b. \quad \text{in}(\{ y_a, y_n_a \}_{\text{pub}(y_b)}); \text{out}(\{ y_n_a, n_b \}_{\text{pub}(y_a)}) \]
Running example: Needham-Schroeder’s protocol

\[
\begin{align*}
A \rightarrow B & : \{ A, N_a \}_{\text{pub}(B)} \\
B \rightarrow A & : \{ N_a, N_b \}_{\text{pub}(A)} \\
A \rightarrow B & : \{ N_b \}_{\text{pub}(B)}
\end{align*}
\]

Roles composing the protocol

\[
\begin{align*}
R_A(x_a, x_b) & : \nu n_a. \ \text{out}(\{ x_a, n_a \}_{\text{pub}(x_b)}); \\
& \quad \text{in}(\{ n_a, x_n_b \}_{\text{pub}(x_a)}); \text{out}(\{ x_n_b \}_{\text{pub}(x_b)}) \\
R_B(y_b) & : \nu n_b. \ \text{in}(\{ y_a, y_n_a \}_{\text{pub}(y_b)}); \text{out}(\{ y_n_a, n_b \}_{\text{pub}(y_a)})
\end{align*}
\]

To retrieve the well-known man-in-the-middle attack, we consider

- \( R_A(a, l) \) and \( R_B(b) \) (running in parallel).
- \( T_0 = \{ a, b, l, \text{pub}(a), \text{pub}(b), \text{pub}(l), \text{priv}(l) \} \)
- Is \( n_b \) deducible by the intruder?
Insecurity problem via constraint solving

Protocol rules

\[ \text{in}(u_1); \text{out}(v_1) \]
\[ \text{in}(u_2); \text{out}(v_2) \]
\[ \ldots \]
\[ \text{in}(u_n); \text{out}(v_n) \]

Constraint System

\[ \mathcal{C} = \begin{cases} 
T_0 \models u_1 \\
T_0, v_1 \models u_2 \\
\ldots \\
T_0, v_1, \ldots, v_n \models s 
\end{cases} \]
Insecurity problem via constraint solving

Protocol rules

\[
\begin{align*}
in(u_1); & \quad \text{out}(v_1) \\
in(u_2); & \quad \text{out}(v_2) \quad \ldots \\
in(u_n); & \quad \text{out}(v_n)
\end{align*}
\]

Constraint System

\[
C = \left\{ \begin{array}{l}
T_0 \models u_1 \\
T_0, v_1 \models u_2 \\
\ldots \\
T_0, v_1, \ldots, v_n \models s
\end{array} \right. 
\]

Solution of a constraint system in \( \mathcal{I} \)

A substitution \( \sigma \) such that

\[
\text{for every } T \models u \in C, \ u\sigma \text{ is deducible from } T\sigma \text{ in } \mathcal{I}.
\]
Running example: Needham-Schroeder’s protocols

\[ R_A(a, l) \text{ and } R_B(b) \text{ (running in parallel)} \]

\[
in\left(\{n_a, x_{n_b}\}_{\text{pub}(a)}\right) ; \quad \text{out}\left(\{x_{n_b}\}_{\text{pub}(l)}\right) \]

\[
in\left(\{y_a, y_{n_a}\}_{\text{pub}(b)}\right) ; \quad \text{out}\left(\{y_{n_a}, n_b\}_{\text{pub}(y_a)}\right) \]
Running example: Needham-Schroeder’s protocols

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>in(${n_a, x_{nb}}<em>{\text{pub}(a)}$) ; out(${x</em>{nb}}_{\text{pub}(I)}$)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>in(${y_a, y_{na}}<em>{\text{pub}(b)}$) ; out(${y</em>{na}, n_b}_{\text{pub}(y_a)}$)</td>
<td></td>
</tr>
</tbody>
</table>
### Running example: Needham-Schroeder’s protocols

<table>
<thead>
<tr>
<th>1</th>
<th>( \text{out}({a, n_a}_{\text{pub}(l)}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>( \text{in}({y_a, y_{n_a}}<em>{\text{pub}(b)}) ) ; ( \text{out}({y</em>{n_a}, n_b}_{\text{pub}(y_a)}) )</td>
</tr>
<tr>
<td>3</td>
<td>( \text{in}({n_a, x_{n_b}}<em>{\text{pub}(a)}) ) ; ( \text{out}({x</em>{n_b}}_{\text{pub}(l)}) )</td>
</tr>
</tbody>
</table>

### Constraints System
Running example: Needham-Schroeder’s protocols

<table>
<thead>
<tr>
<th>1</th>
<th>( \text{out} \left( {a, n_a}_{\text{pub}(l)} \right) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>( \text{in} \left( {n_a, x_{n_b}}<em>{\text{pub}(a)} \right) ); ( \text{out} \left( {x</em>{n_b}}_{\text{pub}(l)} \right) )</td>
</tr>
<tr>
<td>2</td>
<td>( \text{in} \left( {y_a, y_{n_a}}<em>{\text{pub}(b)} \right) ); ( \text{out} \left( {y</em>{n_a}, n_b}_{\text{pub}(y_a)} \right) )</td>
</tr>
</tbody>
</table>

Constraints System

\[ T_0, \{a, n_a\}_{\text{pub}(l)} \]
Running example: Needham-Schroeder’s protocols

\( R_A(a, l) \) and \( R_B(b) \) (running in parallel)

1. \( \text{out(} \{a, n_a\}_{\text{pub}(l)} \) \\
3. \( \text{in(} \{n_a, x_{n_b}\}_{\text{pub}(a)} \); \text{ out(} \{x_{n_b}\}_{\text{pub}(l)} \)

2. \( \text{in(} \{y_a, y_{n_a}\}_{\text{pub}(b)} \); \text{ out(} \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \)

Constraints System

\[ T_0, \{a, n_a\}_{\text{pub}(l)} \vdash \{y_a, y_{n_a}\}_{\text{pub}(b)} \]
Running example: Needham-Schroeder’s protocols

### $R_A(a, l)$ and $R_B(b)$ (running in parallel)

<table>
<thead>
<tr>
<th>Step</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>out(${a, n_a}_{\text{pub}(l)}$)</td>
</tr>
<tr>
<td>2</td>
<td>in(${y_a, y_{na}}<em>{\text{pub}(b)}$) ; out(${y</em>{na}, n_b}_{\text{pub}(y_a)}$)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>in(${n_a, x_{nb}}<em>{\text{pub}(a)}$) ; out(${x</em>{nb}}_{\text{pub}(l)}$)</td>
<td></td>
</tr>
</tbody>
</table>

### Constraints System

\[ T_0, \ \{a, n_a\}_{\text{pub}(l)} \ |\ - \ \{y_a, y_{na}\}_{\text{pub}(b)} \]
\[ T_0, \ \{a, n_a\}_{\text{pub}(l)}, \ {y_{na}, n_b}_{\text{pub}(y_a)} \]
Running example: Needham-Schroeder’s protocols

$R_A(a, I)$ and $R_B(b)$ (running in parallel)

1
\[
\text{out}(\{a, n_a\}_{\text{pub}(I)})
\]

3
\[
in(\{n_a, x_{n_b}\}_{\text{pub}(a)}) \quad ; \quad \text{out}(\{x_{n_b}\}_{\text{pub}(I)})
\]

2
\[
in(\{y_a, y_{n_a}\}_{\text{pub}(b)}) \quad ; \quad \text{out}(\{y_{n_a}, n_b\}_{\text{pub}(y_a)})
\]

Constraints System

\[
T_0, \ \{a, n_a\}_{\text{pub}(I)} \vdash \{y_a, y_{n_a}\}_{\text{pub}(b)}
\]

\[
T_0, \ \{a, n_a\}_{\text{pub}(I)}, \ \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \vdash \{n_a, x_{n_b}\}_{\text{pub}(a)}
\]
### Running example: Needham-Schroeder’s protocols

<table>
<thead>
<tr>
<th>Step</th>
<th>Action 1</th>
<th>Action 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\text{out}({a, n_a}_{\text{pub}(l)})$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\text{in}({y_a, y_{n_a}}_{\text{pub}(b)})$</td>
<td>$\text{out}({y_{n_a}, n_b}_{\text{pub}(y_a)})$</td>
</tr>
<tr>
<td>3</td>
<td>$\text{in}({n_a, x_{n_b}}_{\text{pub}(a)})$</td>
<td>$\text{out}({x_{n_b}}_{\text{pub}(l)})$</td>
</tr>
</tbody>
</table>

### Constraints System

\[
T_0, \quad \{a, n_a\}_{\text{pub}(l)} \vdash \{y_a, y_{n_a}\}_{\text{pub}(b)}
\]

\[
T_0, \quad \{a, n_a\}_{\text{pub}(l)}, \quad \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \vdash \{n_a, x_{n_b}\}_{\text{pub}(a)}
\]

\[
T_0, \quad \{a, n_a\}_{\text{pub}(l)}, \quad \{y_{n_a}, n_b\}_{\text{pub}(y_a)}, \quad \{x_{n_b}\}_{\text{pub}(l)}
\]
Running example: Needham-Schroeder’s protocols

**$R_A(a, I)$ and $R_B(b)$ (running in parallel)**

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>$\text{out}({a, n_a}_{\text{pub}(I)})$</td>
</tr>
<tr>
<td>2</td>
<td>$\text{in}({y_a, y_{n_a}}_{\text{pub}(b)})$</td>
<td>$\text{out}({y_{n_a}, n_b}_{\text{pub}(y_a)})$</td>
</tr>
<tr>
<td>3</td>
<td>$\text{in}({n_a, x_{n_b}}_{\text{pub}(a)})$</td>
<td>$\text{out}({x_{n_b}}_{\text{pub}(I)})$</td>
</tr>
</tbody>
</table>

**Constraints System**

\[
T_0, \ \{a, n_a\}_{\text{pub}(I)} \vdash \{y_a, y_{n_a}\}_{\text{pub}(b)} \\
T_0, \ \{a, n_a\}_{\text{pub}(I)}, \ \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \vdash \{n_a, x_{n_b}\}_{\text{pub}(a)} \\
T_0, \ \{a, n_a\}_{\text{pub}(I)}, \ \{y_{n_a}, n_b\}_{\text{pub}(y_a)}, \ \{x_{n_b}\}_{\text{pub}(I)} \vdash \ n_b
\]
Running example: Needham-Schroeder’s protocols

**$R_A(a, l)$ and $R_B(b)$ (running in parallel)**

<table>
<thead>
<tr>
<th>Step</th>
<th>Message</th>
<th>Out Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\text{in}({n_a, x_{n_b}}_{\text{pub}(l)})$</td>
<td>$\text{out}({a, n_a}_{\text{pub}(l)})$</td>
</tr>
<tr>
<td>2</td>
<td>$\text{in}({y_a, y_{n_a}}_{\text{pub}(b)})$</td>
<td>$\text{out}({y_{n_a}, n_b}_{\text{pub}(y_a)})$</td>
</tr>
<tr>
<td>3</td>
<td>$\text{in}({n_a, x_{n_b}}_{\text{pub}(a)})$</td>
<td>$\text{out}({x_{n_b}}_{\text{pub}(l)})$</td>
</tr>
</tbody>
</table>

**Constraints System**

$$T_0, \{a, n_a\}_{\text{pub}(l)} \vdash \{y_a, y_{n_a}\}_{\text{pub}(b)}$$
$$T_0, \{a, n_a\}_{\text{pub}(l)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \vdash \{n_a, x_{n_b}\}_{\text{pub}(a)}$$
$$T_0, \{a, n_a\}_{\text{pub}(l)}, \{y_{n_a}, n_b\}_{\text{pub}(y_a)}, \{x_{n_b}\}_{\text{pub}(l)} \vdash n_b$$

**Solution**

$$\sigma = \{y_a \mapsto , y_{n_a} \mapsto , x_{n_b} \mapsto \}$$
Running example: Needham-Schroeder’s protocols

\( R_A(a, I) \) and \( R_B(b) \) (running in parallel)

<table>
<thead>
<tr>
<th>Step</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>( \text{out}({a, n_a}_{pub(I)}) )</td>
</tr>
<tr>
<td>3</td>
<td>( \text{in}({n_a, x_{n_b}}_{pub(a)}) )</td>
<td>( \text{out}({x_{n_b}}_{pub(I)}) )</td>
</tr>
<tr>
<td>2</td>
<td>( \text{in}({y_a, y_{n_a}}_{pub(b)}) )</td>
<td>( \text{out}({y_{n_a}, n_b}_{pub(y_a)}) )</td>
</tr>
</tbody>
</table>

Constraints System

\[
T_0, \quad \{a, n_a\}_{pub(I)} \vdash \{y_a, y_{n_a}\}_{pub(b)} \\
T_0, \quad \{a, n_a\}_{pub(I)}, \quad \{y_{n_a}, n_b\}_{pub(y_a)} \vdash \{n_a, x_{n_b}\}_{pub(a)} \\
T_0, \quad \{a, n_a\}_{pub(I)}, \quad \{y_{n_a}, n_b\}_{pub(y_a)}, \quad \{x_{n_b}\}_{pub(I)} \vdash n_b
\]

Solution \( \sigma = \{y_a \mapsto a, \ y_{n_a} \mapsto n_a, \ x_{n_b} \mapsto \} \)

Stéphanie Delaune () Security via constraint solving October 30, 2006 17 / 25
Running example: Needham-Schroeder’s protocols

$R_A(a,l)$ and $R_B(b)$ (running in parallel)

1. out($\{a, n_a\}_{\text{pub}(l)}$)
2. in($\{y_a, y_{n_a}\}_{\text{pub}(b)}$) $; \text{ out}(\{y_{n_a}, n_b\}_{\text{pub}(y_a)})$
3. in($\{n_a, x_{n_b}\}_{\text{pub}(a)}$) $; \text{ out}(\{x_{n_b}\}_{\text{pub}(l)})$

Constraints System

$T_0, \{a, n_a\}_{\text{pub}(l)} \vdash \{y_a, y_{n_a}\}_{\text{pub}(b)}$
$T_0, \{a, n_a\}_{\text{pub}(l)} \vdash \{y_{n_a}, n_b\}_{\text{pub}(y_a)} \vdash \{n_a, x_{n_b}\}_{\text{pub}(a)}$
$T_0, \{a, n_a\}_{\text{pub}(l)} \vdash \{y_{n_a}, n_b\}_{\text{pub}(y_a)}, \{x_{n_b}\}_{\text{pub}(l)} \vdash n_b$

Solution $\sigma = \{y_a \mapsto a, y_{n_a} \mapsto n_a, x_{n_b} \mapsto n_b\}$
Existing results

Many theoretical results for different intruder models

- to take into account \textit{algebraic properties} of cryptographic primitives (exclusive or, cipher block chaining, ...)
- to take into account the fact that some data are \textit{poorly-chosen} (\textit{e.g.} passwords)
Existing results

Many theoretical results for different intruder models
- to take into account algebraic properties of cryptographic primitives (exclusive or, cipher block chaining, ...)
- to take into account the fact that some data are poorly-chosen (e.g. passwords)

Few generic results
- procedure to solve constraint systems for a class of intruder
  \(\Rightarrow\) e.g. any intruder who can be described by a subterm convergent reriting system
- combination result for disjoint intruder models.
Existing results

Many theoretical results for different intruder models
- to take into account algebraic properties of cryptographic primitives (exclusive or, cipher block chaining, ...)
- to take into account the fact that some data are poorly-chosen (e.g. passwords)

Few generic results
- procedure to solve constraint systems for a class of intruder
  $\rightarrow$ e.g. any intruder who can be described by a subterm convergent rewriting system

- combination result for disjoint intruder models.

Some tools
- AVISPA tool (Atse, OFMC)
Outline of the talk

1 Introduction

2 How to deal with trace properties? (e.g. secrecy, authentication)

3 Work in progress: Equivalence based security properties (e.g. anonymity)
Motivation: Electronic voting

Advantages:

- Convenient,
- Efficient facilities for tallying votes.

Drawbacks:

- Risk of large-scale and undetectable fraud,
- Such protocols are extremely error-prone.

"A 15-year-old in a garage could manufacture smart cards and sell them on the Internet that would allow for multiple votes"

Avi Rubin

Possible issue: formal methods
abstract analysis of the protocol against formally-stated properties
Expected properties

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

![Vote for me]

**Receipt-freeness:** a voter cannot prove that she voted in a certain way (this is important to protect voters from coercion)

**Coercion-resistance:** same as receipt-freeness, but the coercer interacts with the voter during the protocol, e.g. by preparing messages
How to model such security properties?

[Kremer & Ryan, 2005] – Formalisation of Privacy

→ consider 2 honest voters and swap their votes

Privacy

A voting protocol respects privacy if

\[ S[V_A\{a/v\} \mid V_B\{b/v\}] \approx S[V_A\{b/v\} \mid V_B\{a/v\}] \]

[Delaune, Kremer & Ryan, 2006]

Formalisation of Receipt-freeness and Coercion-resistance in term of equivalence.
Some examples

\[ S[V_A\{a/v\} \mid V_B\{b/v\}] \approx S[V_A\{b/v\} \mid V_B\{a/v\}] \]

Naive vote protocol (version 1)

\[ V \rightarrow S : \{v\}_{\text{pub}(S)} \]

What about privacy?
Some examples

\[ S[V_A\{^a/_v\} \mid V_B\{^b/_v\}] \approx S[V_A\{^b/_v\} \mid V_B\{^a/_v\}] \]

**Naive vote protocol (version 1)**

\[ V \rightarrow S : \{v\}_{\text{pub}(S)} \]

What about privacy? **OK**
Some examples

\[ S[V_A^{a/V} \mid V_B^{b/V}] \approx S[V_A^{b/V} \mid V_B^{a/V}] \]

Naive vote protocol (version 1)

\[ V \rightarrow S : \{v\}_{pub(S)} \]

What about privacy? OK

Naive vote protocol (version 2)

\[ V \rightarrow S : Id, \{v\}_{pub(S)} \]

What about privacy?
Some examples

\[ S[V_A^{a/v} \mid V_B^{b/v}] \approx S[V_A^{b/v} \mid V_B^{a/v}] \]

**Naive vote protocol (version 1)**

\[ V \rightarrow S : \{v\}_{pub(S)} \]

What about privacy? OK

**Naive vote protocol (version 2)**

\[ V \rightarrow S : Id, \{v\}_{pub(S)} \]

What about privacy?

- **deterministic** encryption: NOT OK
- **probabilistic** encryption: OK
More formally

Labeled bisimilarity $(\cong_\ell)$

The largest symmetric relation $\mathcal{R}$ on processes, such that $A \mathcal{R} B$ implies

1. $\phi(A) \cong_s \phi(B)$ (depends on $E$),
2. if $A \rightarrow A'$, then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some $B'$,
3. if $A \xrightarrow{\alpha} A'$, then $B \rightarrow^* \xrightarrow{\alpha} B'$ and $A' \mathcal{R} B'$ for some $B'$.

This relation is in general undecidable. Why?

- unfolding tree is infinite in depth
- unfolding tree is infinititely branching (because of inputs)
- equational theories may be complex

Tool: Proverif

→ Obviously, the procedure is not complete. Proverif is not able to conclude for privacy even for naive voting protocols (version 1)
Our Goal:
to do better than Proverif in the context of a **bounded** number of sessions

- **Infinite depth:**
  ← we restrict to consider processes without replication (finite processes),

- **Infinite branching:**
  ← we define a notion of **symbolic** processes and **symbolic** bisimulation

Concrete \( in(x).out(\{x\}_k) \xrightarrow{in(m_1)} out(\{m_1\}_k) \)

Symbolic \( (in(x).out(\{x\}_k); C) \xrightarrow{in(x)} (out(\{x\}_k); C \cup \phi(P) \models x) \)
Work in Progress

Our Goal:
to do better than Proverif in the context of a bounded number of sessions

- **Infinite depth:**
  - we restrict to consider processes without replication (finite processes),

- **Infinite branching:**
  - we define a notion of symbolic processes and symbolic bisimulation

Concrete \[ in(x).out(\{x\}_k) \xrightarrow{in(m_1)} out(\{m_1\}_k) \]

Symbolic \( (in(x).out(\{x\}_k); C) \xrightarrow{in(x)} (out(\{x\}_k); C \cup \phi(P) \models x) \)

Then, we plan:

- to **design a procedure** to solve our constraint systems for a class of equational theory as larger as possible
- to implement a **tool**