Models and analysis of security protocols
1st Semester 2007-2008
Active Intruder

Pascal Lafourcade

Université Joseph Fourier, Verimag

Master: November 12th 2007
Thanks to Steve Kremer
Last Time (I)

Lecture

- Man in the Middle Needham-Schroeder
- Dolev Yao Model
- Notion of Locality
- Undecidability Result
- Diffie-Hellman
Last Time (I)

Exercise

- Properties of Syntactic Subterms
- Locality Result
- Security against Passive Intruder
- Logical Passive Attack on Shamir 3-Pass Protocol
Outline of Today

1. Active Intruder: Security Problem
2. Unification Notions
   - Terms and Messages
   - Unification
3. Bounded Number of Sessions
4. NP-Hardness for Bounded Number of Sessions
5. Complexity
6. Tools
7. Conclusion
Outline

1. Active Intruder: Security Problem

2. Unification Notions
   - Terms and Messages
   - Unification

3. Bounded Number of Sessions

4. NP-Hardness for Bounded Number of Sessions

5. Complexity

6. Tools

7. Conclusion
The Intruder is the Network (Worst Case)

Passive: Intruder deduction problem
The Intruder is the Network (Worst Case)

Passive: Intruder deduction problem

Active Intruder Security problem

- intercept messages (add messages to his knowledge)
- Play messages from his knowledge
- Start new sessions

Execution tree has:
- infinite branching (size of messages is not bounded)
- infinite depth (number of sessions is not bounded)
Active Intruder with bounded number of sessions

- Theoretically: *decidable*
- Interesting *practically*:
  - Find flaws
  - Usually attacks use *few sessions*!
# Dolev-Yao Deduction System

<table>
<thead>
<tr>
<th>Deduction System</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_0 \vdash ? s$</td>
<td></td>
</tr>
</tbody>
</table>

(A) \[ \frac{u \in T_0}{T_0 \vdash u} \]

(P) \[ \frac{T_0 \vdash u \quad T_0 \vdash v}{T_0 \vdash \langle u, v \rangle} \]

(C) \[ \frac{T_0 \vdash u \quad T_0 \vdash v}{T_0 \vdash \{u\}_v} \]

(UL) \[ \frac{T_0 \vdash \langle u, v \rangle}{T_0 \vdash u} \]

(UR) \[ \frac{T_0 \vdash \langle u, v \rangle}{T_0 \vdash v} \]

(D) \[ \frac{T_0 \vdash \{u\}_v \quad T_0 \vdash v}{T_0 \vdash u} \]
Model: actions, roles and protocol

Definition (Action)

An action is a couple \((\text{recv}(u), \text{send}(v))\) such that \(u \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{\text{init}\}\), \(v \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{\text{stop}\}\). Denoted \((u \rightarrow v)\).
Model: actions, roles and protocol

Definition (Action)

An action is a couple \((\text{recv}(u), \text{send}(v))\) such that \(u \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \text{init}\), \(v \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \text{stop}\). Denoted \((u \rightarrow v)\).

Definition (Role)

A role is a finite sequence of actions:

\[(u_1 \rightarrow v_1), \ldots, (u_n \rightarrow v_n)\]

such that \(\text{vars}(v_i) \subseteq \bigcup_{1 \leq j \leq i} \text{vars}(u_j)\).
**Model: actions, roles and protocol**

**Definition (Action)**

An *action* is a couple \((\text{recv}(u), \text{send}(v))\) such that \(u \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{\text{init}\}, \ v \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \cup \{\text{stop}\}\). Denoted \((u \to v)\).

**Definition (Role)**

A *role* is a finite sequence of actions:

\[(u_1 \to v_1), \ldots, (u_n \to v_n)\]

such that \(\text{vars}(v_i) \subseteq \bigcup_{1 \leq j \leq i} \text{vars}(u_j)\).

**Definition (Protocol)**

A *protocol* \(P\) is a finite set of roles: \(P = \{R_1, \ldots, R_k\}\)
1st Example

Example (Needham-schroeder)

1. $A \rightarrow B : \{N_a, A\}_{pk(B)}$
2. $B \rightarrow A : \{N_a, N_b\}_{pk(A)}$
3. $A \rightarrow B : \{N_b\}_{pk(B)}$

Write down each agent’s role description.
1st Example

Example (Needham-schroeder)

1. $A \rightarrow B : \{N_a, A\}_{pk(B)}$
2. $B \rightarrow A : \{N_a, N_b\}_{pk(A)}$
3. $A \rightarrow B : \{N_b\}_{pk(B)}$

Write down each agent’s role description.

$$R_A = (init \rightarrow \{N_a, A\}_{pk(B)}),$$
$$\quad (\{N_a, y_b\}_{pk(A)} \rightarrow \{y_b\}_{pk(B)}),$$

$$R_B = (\{x_a, z\}_{pk(B)} \rightarrow \{x_a, N_b\}_{pk(z)}),$$
$$\quad (\{N_b\}_{pk(B)} \rightarrow stop)$$
Scyther Notation

A:  const Na: Nonce;
    var Nb: Nonce;

    send(A, B, {Na, A}pk(B));
    recv(B, A, {Na, Nb}pk(A));
    send(A, B, {Nb}pk(B));

B:  const Nb: Nonce;
    var Na: Nonce;

    recv(A, B, {Na, A}pk(B));
    send(B, A, {Na, Nb}pk(A));
    recv(A, B, {Nb}pk(B));
Exercise

Denning-Sacco Protocol

1. \( A \rightarrow S : \langle A, B \rangle \)
2. \( S \rightarrow A : \{\langle \langle B, N_{AB} \rangle, \langle N_s, \{\langle N_{AB}, \langle A, N_s \rangle \rangle \rangle K_{BS} \rangle \rangle \} K_{AS} \)
3. \( A \rightarrow B : \{\langle N_{AB}, \langle A, N_s \rangle \rangle \} K_{BS} \)
4. \( B \rightarrow A : \{S_{AB}\} N_{AB} \)

\( P_{DS} = \{R_A, R_B, R_S\} \) models one session of \( A, B \) and \( S \).
Exercise

Denning-Sacco Protocol

1. $A \rightarrow S : \langle A, B \rangle$
2. $S \rightarrow A : \{\langle\langle B, N_{AB}\rangle, \langle N_s, \{\langle N_{AB}, \langle A, N_s\rangle\}\rangle K_{BS}\rangle\}\} K_{AS}$
3. $A \rightarrow B : \{\langle N_{AB}, \langle A, N_s\rangle\rangle\} K_{BS}$
4. $B \rightarrow A : \{S_{AB}\} N_{AB}$

$P_{DS} = \{R_A, R_B, R_S\}$ models one session of $A, B$ and $S$.

$R_A = (init \rightarrow \langle A, B \rangle), (\{\langle B, x_A\rangle, \langle y_A, z_A\rangle\} K_{AS} \rightarrow z_A), (\{w_A\} x_A \rightarrow stop)$

$R_B = (\{x_B, \langle a, y_B\rangle\} K_{BS} \rightarrow \{S_{AB}\} x_B)$

$R_S = (\langle A, B \rangle \rightarrow \{\langle B, N_{AB}, \langle N_s, \langle A, N_s\rangle\rangle\rangle K_{BS}\} K_{AS})$
Semantic

**Definition (States and Transitions)**

A **state** is a couple \((T, P)\) where \(T\) is a set of ground terms (intruder knowledge) and \(P\) a protocol. We define a **transition relation** between states \((T, P) \rightarrow (T', P')\) by:

- \(R_i \in P, R_i = (u \rightarrow v)\)
- \(T \vdash u\sigma\)  \((\text{dom}(\sigma) = \text{vars}(u))\)
- \(T' = T \cup \{v\sigma\}\)
- \(R'_i \in P', R'_i = (P \setminus \{R_i\}) \cup R_i\sigma\)
Example

Let \( T = \{a, b, k_I}\) and \( P = \{R\}\) where
\( R = (\langle x, y \rangle \rightarrow \langle \{y\}_k, x \rangle), (z \rightarrow \langle x, \langle y, z \rangle \rangle). \)

- \((T, P) \rightarrow (T \cup \{\langle \{b\}_k, a \rangle\}, \{(z \rightarrow \langle a, \langle b, z \rangle \rangle)\})\)
- \((T, P) \rightarrow (T \cup \{\langle \{a\}_{k_I} \}_k, a \rangle, \{(z \rightarrow \langle a, \langle \{a\}_{k_I}, z \rangle \rangle)\})\)
- \((T, P) \not\rightarrow (T \cup \{\langle \{a\}_k \}_k, a \rangle, \{(z \rightarrow \langle a, \langle \{a\}_k, z \rangle \rangle)\})\)
Example

Let $T = \{a, b, k_I\}$ and $P = \{R\}$ where

$R = (\langle x, y \rangle \rightarrow \langle \{y\}_k, x \rangle), (z \rightarrow \langle x, \langle y, z \rangle \rangle)$.

- $(T, P) \rightarrow (T \cup \{\langle \{b\}_k, a \rangle\}, \{(z \rightarrow \langle a, \langle b, z \rangle \rangle)\})$
- $(T, P) \rightarrow (T \cup \{\langle \{a\}_k \rangle_k, a \rangle\}, \{(z \rightarrow \langle a, \langle \{a\}_k \rangle_k, z \rangle)\})$
- $(T, P) \not\rightarrow (T \cup \{\langle \{a\}_k \rangle_k, a \rangle\}, \{(z \rightarrow \langle a, \langle \{a\}_k \rangle_k, z \rangle)\})$

Each branch has a finite depth, but possibly an infinite branching.
### Definition (Secrecy)

Let $T_1$ be a ground set of terms (Initial knowledge of the intruder). A protocol $P$ does not preserve the secrecy of a ground term $s$ for $T_1$ if there does not exist a state $(T', P')$, such that

- $T' \vdash s$
- $(T_1, P) \rightarrow^* (T', P')$

where $\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$.

If there does not exist a such state $(T', P')$ we say that $P$ preserves the secrecy of $s$ for the initial intruder knowledge $T_1$. 

**Preservation of the secrecy**
Interleaving

**Definition (Partial Order \( \prec_P \))**

A protocol \( P \) define a partial order \( \prec_P \) on actions of \( P \), s.t

\[
(u_i \rightarrow v_i) \prec_P (u_j \rightarrow v_j)
\]

if \( R \in P, \ R = (u_1 \rightarrow v_1) \cdots (u_i \rightarrow v_i) \cdots (u_j \rightarrow v_j) \cdots (u_n \rightarrow v_n) (1 \leq i \leq j \leq n). \)
**Interleaving**

<table>
<thead>
<tr>
<th>Definition (Partial Order (\prec_P))</th>
</tr>
</thead>
<tbody>
<tr>
<td>A protocol (P) define a partial order (\prec_P) on actions of (P), s.t.</td>
</tr>
<tr>
<td>((u_i \rightarrow v_i) \prec_P (u_j \rightarrow v_j))</td>
</tr>
<tr>
<td>if (R \in P), (R = (u_1 \rightarrow v_1) \ldots (u_i \rightarrow v_i) \ldots (u_j \rightarrow v_j) \ldots (u_n \rightarrow v_n)) ((1 \leq i \leq j \leq n)).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Definition (Execution Order (\prec_E))</th>
</tr>
</thead>
<tbody>
<tr>
<td>An execution order (\prec_E) of (P) is a total order on the subset (A) of actions of (P), compatible with (\prec_P) and stable by predecessor, i.e.</td>
</tr>
<tr>
<td>if (b \in A) et (a \prec_P b) then (a \in A) and (a \prec_E b)</td>
</tr>
</tbody>
</table>

It corresponds to an interleaving of roles.
Secrecy

Definition (Secrecy over \(<_E\))

Let an execution order \(<_E\) of \(P\). We assume that

\[(u_1 \rightarrow v_1) <_E \ldots <_E (u_n \rightarrow v_n)\]

\(<_E\) does not preserve the secrecy of \(s\), given \(T_1\) if there exists \(\sigma_1, \ldots, \sigma_n\) such that

\[(P, T_1) \rightarrow (P_1, T_1 \cup \{v_1\sigma_1\}) \rightarrow \ldots \rightarrow (P_n, T_1 \cup \{v_1\sigma_1, \ldots, v_n\sigma_n\})\]

and \(T_1 \cup \{v_1\sigma_1, \ldots, v_n\sigma_n\} \vdash s\).
Outline

1. Active Intruder: Security Problem
2. Unification Notions
   - Terms and Messages
   - Unification
3. Bounded Number of Sessions
4. NP-Hardness for Bounded Number of Sessions
5. Complexity
6. Tools
7. Conclusion
Arity

Definition

- \( \mathcal{F} \) is a finite set
- \( \text{Arity} \) is a mapping from \( \mathcal{F} \) into \( \mathbb{N} \)
- \( (\mathcal{F}, \text{Arity}) \) is a \textbf{ranked alphabet} or \textbf{signature} denoted \( \Sigma \)

- The \textbf{arity} of a symbol \( f \in \mathcal{F} \) is \( \text{Arity}(f) \)
- The set of symbols of arity \( p \) is denoted by \( \mathcal{F}_p \).
- Elements of arity 0, 1, \ldots \, p \) are respectively called constants, unary, \ldots \, p-ary symbols.
Example

Let \( F = \{ \text{enc}, \text{pair}, k_1, k_2, 0, 1 \} \)

\[
\begin{align*}
\text{Arity}(\text{enc}) &= \text{Arity}(\text{pair}) = 2 \\
\text{Arity}(k_1) &= \text{Arity}(k_2) = \text{Arity}(0) = \text{Arity}(1) = 0
\end{align*}
\]

We also denote \( F = \{ \text{enc}/2, \text{pair}/2, k_1/0, k_2/0, 0/0, 1/0 \} \)
Terms

Let $\mathcal{X}$ be a set of symbols called **variables**.

**Definition**

The set $\mathcal{T}(\mathcal{F}, \mathcal{X})$ of **terms** over the ranked alphabet $\mathcal{F}$ and the set of variables $\mathcal{X}$ is the smallest set defined by:

- $\mathcal{F}_0 \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
- $\mathcal{X} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$
- if $p \geq 1$, $f \in \mathcal{F}_p$ and $t_1, \ldots, t_p \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, then $f(t_1, \ldots, t_p) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

- If $\mathcal{X} = \emptyset$ then $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is also written $\mathcal{T}(\mathcal{F})$. Terms in $\mathcal{T}(\mathcal{F})$ are called **ground terms**.
- A term in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is **linear** if each variable occurs at most once in $t$. 
Example

Let \( \mathcal{F} = \{\text{enc}/2, \text{pair}/2, k_1/0, k_2/0, 0/0, 1/0\} \) and \( \mathcal{X} = \{x, y, z\} \). \( \text{pair}(x, 1), \text{enc}(\text{pair}(y, z), k_1) \) and \( \text{enc}(0, k_1) \) are terms in \( \mathcal{T}(\mathcal{F}, \mathcal{X}) \). \( \text{pair}(0, 1), \text{enc}(0, k_1) \) are terms in \( \mathcal{T}(\mathcal{F}) \), i.e., ground terms.

We also denote \( \{\_\}_\_ \) for \( \text{enc}(\_ , \_) \) and \( \langle \_ , \_ \rangle \) for \( \text{pair}(\_ , \_) \).
Substitution

Definition

- A **substitution** (respectively a **ground substitution**) $\sigma$ is a mapping from $\mathcal{X}$ into $\mathcal{I}(\mathcal{F}, \mathcal{X})$ (respectively into $\mathcal{I}(\mathcal{F})$) where there are only finitely many variables not mapped to themselves.

- Substitutions can be extended to $\mathcal{I}(\mathcal{F}, \mathcal{X})$ in such a way that:

  $$\forall f \in \mathcal{F}_n, \forall t_1, \ldots, t_n \in \mathcal{I}(\mathcal{F}, \mathcal{X}) :$$

  $$\sigma(f(t_1, \ldots, t_n)) = f(\sigma(t_1), \ldots, \sigma(t_n)).$$

The **domain** of a substitution $\sigma$ is the subset of variables $x \in \mathcal{X}$ such that $\sigma(x) \neq x$. 
Example:

Let $\sigma = \{ x \leftarrow N_A, y \leftarrow \{ \langle N_A, N_B \rangle \}_{k_B} \}$ and $t = \langle x, \langle y, \langle x, x \rangle \rangle \rangle$. Then,

$$\sigma(t) = \langle N_A, \langle \{ \langle N_A, N_B \rangle \}_{k_B}, \langle N_A, N_A \rangle \rangle \rangle$$
Unification

Definition

Two $t$ and $u$ are unifiable if there exists a substitution $\sigma$ such that $\sigma s = \sigma t$

Examples:
Unification

Definition

Two $t$ and $u$ are unifiable if there exists a substitution $\sigma$ such that $\sigma s = \sigma t$.

Examples:
$s = a \quad t = X$
Unification

Definition

Two $t$ and $u$ are unifiable if there exists a substitution $\sigma$ such that $\sigma s = \sigma t$

Examples:
$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$
Unification

Definition

Two $t$ and $u$ are unifiable if there exists a substitution $\sigma$ such that $\sigma s = \sigma t$

Examples:

$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$

$s = a \quad t = p(X)$
Unification

Definition

Two $t$ and $u$ are unifiable if there exists a substitution $\sigma$ such that $\sigma s = \sigma t$

Examples:

$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$

$s = a \quad t = p(X) \quad$ No unifier
Unification

Definition

Two \( t \) and \( u \) are unifiable if there exists a substitution \( \sigma \) such that \( \sigma s = \sigma t \)

Examples:

\[
\begin{align*}
  s &= a & t &= X & \sigma &= \{X \leftarrow a\} \\
  s &= a & t &= p(X) & \text{No unifier} \\
  s &= p(a, X) & t &= p(Y, b)
\end{align*}
\]
Unification

Definition

Two $t$ and $u$ are unifiable if there exists a substitution $\sigma$ such that $\sigma s = \sigma t$

Examples:

$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$

$s = a \quad t = p(X) \quad$ No unifier

$s = p(a, X) \quad t = p(Y, b) \quad \sigma = \{X \leftarrow b; Y \leftarrow a\}$
Unification

Definition

Two $t$ and $u$ are unifiable if there exists a substitution $\sigma$ such that $\sigma s = \sigma t$

Examples:

- $s = a$, $t = X$, $\sigma = \{X \leftarrow a\}$
- $s = a$, $t = p(X)$ No unifier
- $s = p(a, X)$, $t = p(Y, b)$, $\sigma = \{X \leftarrow b; Y \leftarrow a\}$
- $s = p(f(X), g(Z))$, $t = p(f(a), Y)$
Unification

Definition

Two $t$ and $u$ are unifiable if there exists a substitution $\sigma$ such that $\sigma s = \sigma t$

Examples:

$s = a \quad t = X \quad \sigma = \{X \leftarrow a\}$
$s = a \quad t = p(X) \quad \text{No unifier}$
$s = p(a, X) \quad t = p(Y, b) \quad \sigma = \{X \leftarrow b; Y \leftarrow a\}$
$s = p(f(X), g(Z)) \quad t = p(f(a), Y) \quad \sigma = \{X \leftarrow a; Y \leftarrow g(Z)\}$
Unification

Definition

Two \( t \) and \( u \) are unifiable if there exists a substitution \( \sigma \) such that \( \sigma s = \sigma t \)

Examples:
\[
\begin{align*}
  s &= a \quad t = X \quad \sigma = \{ X \leftarrow a \} \\
  s &= a \quad t = p(X) \quad \text{No unifier} \\
  s &= p(a, X) \quad t = p(Y, b) \quad \sigma = \{ X \leftarrow b; Y \leftarrow a \} \\
  s &= p(f(X), g(Z)) \quad t = p(f(a), Y) \\
      &\quad \sigma = \{ X \leftarrow a; Y \leftarrow g(Z) \} \text{ or } \sigma = \{ X \leftarrow a; Y \leftarrow g(b); Z \leftarrow b \}
\end{align*}
\]
Most General Unifier

Definition

The most general unification between two terms \( s \) and \( t \), denoted by \( \text{mgu}(s, t) \) if: \( \forall \sigma \) such that \( s\sigma = t\sigma, \exists \theta \) such that \( \sigma = \text{mgu}(s, t)\theta \)

\[
s = p(f(X), g(Z)) \quad t = p(f(a), Y)
\]

\[
\sigma_1 = \{X \leftarrow a; Y \leftarrow g(Z)\} \quad \sigma_2 = \{X \leftarrow a; Y \leftarrow g(b); Z \leftarrow b\}
\]
Goal

Design an algorithm that for a given unification problem \( s =? t \)

- returns an mgu of \( s \) and \( t \) if they are unifiable.
- reports failure otherwise.
Naive Algorithm

Write down two terms and set markers at the beginning of the terms. Then:

1. Move the markers simultaneously, one symbol at a time, until both move off the end of the term (success), or until they point to two different symbols;

2. If the two symbols are both non-variables, then fail; otherwise, one is a variable (call it $x$) and the other one is the first symbol of a subterm (call it $t$):
   - If $x$ occurs in $t$, then fail;
   - Otherwise, replace $x$ everywhere by $t$ (including in the solution), write down "$x \leftarrow t" as a part of the solution, and return to 1.
Example: \( f(x, g(a), g(z)) =? f(g(y), g(y), g(g(x))) \)

\[ f(x, g(a), g(z)) \]

\[ f(g(y), g(y), g(g(x))) \]
Example: \( f(x, g(a), g(z)) =? f(g(y), g(y), g(g(x))) \)

\[
f(x, g(a), g(z))
\]

\[
f(g(y), g(y), g(g(x)))
\]
Example: \( f(x, g(a), g(z)) =? f(g(y), g(y), g(g(x))) \)

\[ f(x, g(a), g(z)) \]

\[ f(g(y), g(y), g(g(x))) \]

\( \sigma = \{ x \leftarrow g(y) \} \)
Example: \( f(x, g(a), g(z)) \equiv? f(g(y), g(y), g(g(x))) \)

\[ f(g(y), g(a), g(z)) \]

\[ f(g(y), g(y), g(g(g(y)))) \]

\[ \sigma = \{ x \leftarrow g(y) \} \]
Unification

Example: \( f(x, g(a), g(z)) =? f(g(y), g(y), g(g(x))) \)

\[ f(g(y), g(a), g(z)) \]

\[ f(g(y), g(y), g(g(g(y)))) \]

\[ \sigma = \{ x \leftarrow g(y) \} \]
Example: \( f(x, g(a), g(z)) =? f(g(y), g(y), g(g(x))) \)

\[ f(g(a), g(a), g(z)) \]

\[ f(g(a), g(a), g(g(g(a)))) \]

\[ \sigma = \{ x \leftarrow g(a), y \leftarrow a \} \]
Example: \( f(x, g(a), g(z)) =? f(g(y), g(y), g(g(x))) \)

\[
f(g(a), g(a), g(z))
\]

\[
f(g(a), g(a), g(g(a)))
\]

\[\sigma = \{ x \leftarrow g(a), y \leftarrow a \}\]
Example: \( f(x, g(a), g(z)) = ? f(g(y), g(y), g(g(x))) \)

\[
f(g(a), g(a), g(g(g(a))))
\]

\[
\sigma = \{ x \leftarrow g(a), y \leftarrow a, z \leftarrow g(g(a)) \}
\]
Questions

1 Correctness:
   - Does the algorithm always terminate?
   - Does it always produce an mgu for two unifiable terms, and fail for non-unifiable terms?
   - Do these answers depend on the order of operations?

2 Complexity:
   - How much space does this take, and how much time?

3 Extension with equational theory, e.g., \( ab = ba \).
Syntactic Unification is Unitary

Theorem (Robinson)

*Without equational theory there exists an unique mgu for syntactic unification (modulo renaming). Unification is called unitary.*

Outline

1. Active Intruder: Security Problem
2. Unification Notions
   - Terms and Messages
   - Unification
3. Bounded Number of Sessions
4. NP-Hardness for Bounded Number of Sessions
5. Complexity
6. Tools
7. Conclusion
Constraints System

Symbolic representation of execution tree by constraints system.

**Definition (Constraints System)**

A constraint is an expression \( T \models u \) where \( T \) is a set of terms and \( u \) a term.

A constraints system \( C \) is a finite set of constraints \( \bigcup_{1 \leq i \leq n} T_i \models u_i \) such that

- \( T_i \subseteq T_{i+1} \) \hspace{1cm} (1 \leq i \leq n)
- if \( T_i \models u_i \in C \) and \( x \in \text{vars}(T_i) \) then
  \( T_j = \text{min}\{T' \mid T' \models v \in C, x \in \text{vars}(v)\} \) exists and \( j < i \)

A substitution \( \sigma \) is a solution of \( C \) if \( T\sigma \models u\sigma \) for all \( T \models u \in C \).

We denote by \( \perp \) a constraints system unsatisfiable.
From Protocols to Constraints system

Let $P$ a protocol, $<_E$ an execution order of $P$ and $s$ a secret term.

$$(u_1 \rightarrow v_1) <_E (u_2 \rightarrow v_2) <_E \ldots <_E (u_n \rightarrow v_n)$$

We associate $C$:

\[
\begin{align*}
T_1 & \vdash u_1 \\
T_2 = T_1 \cup \{v_1\} & \vdash u_2 \\
& \vdots \\
T_n = T_{n-1} \cup \{v_{n-1}\} & \vdash u_n \\
T_{n+1} = T_n \cup \{v_n\} & \vdash s
\end{align*}
\]

We show that $C$ has a solution iff $<_E$ does not preserve the secret of the term $s$. 
Exercises

Exercise 1

\[
\begin{align*}
A &\rightarrow B : \langle A, N_A \rangle \\
B &\rightarrow A : \{ \langle N_A, N_B \rangle \} K_{ab} \\
A &\rightarrow B : N_B \\
B &\rightarrow A : \{ \langle K, N_B \rangle \} K_{ab} \\
A &\rightarrow B : \{ s \} K
\end{align*}
\]

Intruder knows only identities of \( A \) and \( B \).

- Give role specification of this protocol of an instance of execution between \( A \) and \( B \).
- Give a constraint system associated to this protocol between \( A \) and \( B \).
Solution

\[ A \rightarrow B : \langle A, N_A \rangle \]
\[ B \rightarrow A : \{ \langle N_A, N_B \rangle \} K_{ab} \]
\[ A \rightarrow B : N_B \]
\[ B \rightarrow A : \{ \langle K, N_B \rangle \} K_{ab} \]
\[ A \rightarrow B : \{ s \} K \]

\[ T_1 = \{ A, B, \langle A, N_A \rangle, \{ \langle N_A, N_B \rangle \} K_{ab}, N_B, \{ \langle K, N_B \rangle \} K_{ab}, \{ s \} K, \text{init}, \text{stop} \} \]

Roles

\[ R_A = (\text{init} \rightarrow \langle A, N_A \rangle), \]
\[ (\{ \langle N_A, X_{N_B} \rangle \} K_{(A,X_B)} \rightarrow X_{N_B}), \]
\[ (\{ \langle X_K, X_{N_B} \rangle \} K_{(A,X_B)} \rightarrow \{ s \} X_K) \]

\[ R_B = (\langle X_A, X_{N_A} \rangle \rightarrow \{ \langle X_{N_A}, N_B \rangle \} K_{(X_A,B)}), \]
\[ (N_B \rightarrow \{ \langle K, N_B \rangle \} K_{(X_A,B)}), \]
\[ (\{ X_s \} K \rightarrow \text{stop}) \]
Solution

\[ A \rightarrow B : \langle A, N_A \rangle \]
\[ B \rightarrow A : \{\langle N_A, N_B \rangle\} K_{ab} \]
\[ A \rightarrow B : N_B \]
\[ B \rightarrow A : \{\langle K, N_B \rangle\} K_{ab} \]
\[ A \rightarrow B : \{s\}_K \]

\[ T_1 = \{A, B, \langle A, N_A \rangle, \{\langle N_A, N_B \rangle\} K_{ab}, N_B, \{\langle K, N_B \rangle\} K_{ab}, \{s\}_K, init, stop\} \]

Constraint System

\[ T_1 \]
\[ T_2 = T_1 \cup \{\langle A, N_A \rangle\} \]
\[ T_3 = T_2 \cup \{\{\langle X_{N_A}, N_B \rangle\} K_{(X_{N_A}, B)}\} \]
\[ T_4 = T_3 \cup \{X_{N_B}\} \]
\[ T_5 = T_4 \cup \{\{\langle K, N_B \rangle\} K_{(X_{K}, B)}\} \]
\[ T_6 = T_5 \cup \{\{s\} X_K\} \]
\[ T_7 = T_6 \cup \{\text{stop}\} \]
\[ \vdash init \]
\[ \vdash \langle X_A, X_{N_A} \rangle \]
\[ \vdash \{\langle N_A, X_{N_B} \rangle\} K_{(A, X_B)} \]
\[ \vdash N_B \]
\[ \vdash \{\langle X_K, X_{N_B} \rangle\} K_{(A, X_B)} \]
\[ \vdash \{X_s\}_K \]
\[ \vdash s \]
Resolution of Constraints systems

Definition (Rules of simplification: $C \leadsto_{\sigma} C'$)

$R_1$  $C \cup \{T \vdash u\} \leadsto C$  if $T \cup \{x \mid T' \vdash x \in C, T' \subseteq T\} \vdash u$

$R_2$  $C \cup \{T \vdash u\} \leadsto_{\sigma} C\sigma \cup \{T\sigma \vdash u\sigma\}$  $\sigma = \text{mgu}(t, u), t \in \text{st}(T), t, u \text{ no variables}$

$R_3$  $C \cup \{T \vdash u\} \leadsto_{\sigma} C\sigma \cup \{T\sigma \vdash u\sigma\}$  $\sigma = \text{mgu}(t_1, t_2), t_1, t_2 \in \text{st}(T), t_1, t_2 \text{ no variables}$

$R_4$  $C \cup \{T \vdash \{u\}_v\} \leadsto C \cup \{T \vdash u, T \vdash v\}$

$R_5$  $C \cup \{T \vdash \langle u, v \rangle\} \leadsto C \cup \{T \vdash u, T \vdash v\}$

$R_6$  $C \cup \{T \vdash u\} \leadsto \bot$  if $T = \emptyset$ or $\text{var}(T) = \text{var}(u) = \emptyset$ and $T \not\vdash u$
Properties of simplification rules

Lemma (Preservation)

*Simplification rules transform a constraints system into a constraints system.*
Properties of simplification rules

Lemma (Preservation)

Simplification rules transform a constraints system into a constraints system.

Lemma (Correctness)

If $C \leadsto_\sigma C'$ then if $\theta$ is a solution of $C'$, $\sigma\theta$ is also a solution of $C$. 
Properties of simplification rules

Lemma (Preservation)

*Simplification rules transform a constraints system into a constraints system.*

Lemma (Correctness)

*If \( C \sim_{\sigma} C' \) then if \( \theta \) is a solution of \( C' \), \( \sigma \theta \) is also a solution of \( C \).*

Lemma (Termination)

*Simplification rules always terminate: There does not exist infinite chain \( C \sim_{\sigma_1} C_1 \sim_{\sigma_2} C_2 \sim_{\sigma_3} \ldots \).*
Properties

Definition (Solved Form)

A constraints system $C$ is in **solved form** if $C = \bot$ or if each constraint is of the following form $T \models x$ where $x$ is a variable $T \neq \emptyset$.

Lemma

*All constraints systems in solved form different of $\bot$ has at least one solution.*
Properties

Definition (Solved Form)

A constraints system $C$ is in solved form if $C = \perp$ or if each constraint is of the following form $T \vDash x$ where $x$ is a variable $T \neq \emptyset$.

Lemma

All constraints systems in solved form different of $\perp$ has at least one solution.

Lemma (Completeness)

If $C$ is a constraint system not in solved form and if $\sigma$ is a solution of $C$ then there exists $\theta, \tau$ such that $C \rightsquigarrow_\theta C'$, $\sigma = \theta \tau$ and $\tau$ is a solution of $C'$. 
Decidability

Theorem

Preservation of the secrecy for protocol with bounded number of sessions is decidable.

- Guess an interleaving and build constraints system associated.
- Using previous lemma $C$ has a solution iff there exists $C'$ in solved form such that $C' \neq \bot$ and $C \leadsto \tau C'$
- Using termination lemma to conclude.

We also can show that the problem is in co-NP.
Exercises

Exercise 1

\[ A \rightarrow B : \langle A, N_A \rangle \]
\[ B \rightarrow A : \{\langle N_A, N_B \rangle\} K_{ab} \]
\[ A \rightarrow B : N_B \]
\[ B \rightarrow A : \{\langle K, N_B \rangle\} K_{ab} \]
\[ A \rightarrow B : \{s\}_K \]

Intruder knows only identities of \( A \) and \( B \).

- Use simplification rules to transform the system in solved form.
- There exists an easy attack, can you find it?
Solution

\[ T_1 = \{ A, B, \langle A, N_A \rangle, \{ \langle N_A, N_B \rangle \}_{K_{ab}}, N_B, \{ \langle K, N_B \rangle \}_{K_{ab}}, \{ s \}_K, \text{init, stop} \} \]

\[
\begin{align*}
C_1 & \quad T_1 \quad \vdash \quad \text{init} \\
C_2 & \quad T_2 = T_1 \cup \{ \langle A, N_A \rangle \} \quad \vdash \quad \langle X_A, X_{N_A} \rangle \\
C_3 & \quad T_3 = T_2 \cup \{ \{ \langle X_{N_A}, N_B \rangle \}_{K_{(X_{A}, B)}} \} \quad \vdash \quad \{ \langle N_A, X_{N_B} \rangle \}_{K_{(A, X_B)}} \\
C_4 & \quad T_4 = T_3 \cup \{ X_{N_B} \} \quad \vdash \quad N_B \\
C_5 & \quad T_5 = T_4 \cup \{ \{ \langle K, N_B \rangle \}_{K_{(X_{A}, B)}} \} \quad \vdash \quad \{ \langle X_K, X_{N_B} \rangle \}_{K_{(A, X_B)}} \\
C_6 & \quad T_6 = T_5 \cup \{ \{ s \}_X \} \quad \vdash \quad \{ X_s \}_K \\
C_7 & \quad T_7 = T_6 \cup \{ \text{stop} \} \quad \vdash \quad s
\end{align*}
\]

Road book


\[
R_2 \quad C \cup \{ T \vdash u \} \stackrel{\sim}{\rightarrow}_\sigma \quad C\sigma \cup \{ T\sigma \vdash u\sigma \} \quad \sigma = mgu(t, u), t \in \text{st}(T), \quad t, u \text{ no variables}
\]

- Apply nothing on \( C_1 \), already in resolved form.
- Apply \( R_2 \) on \( C_2 \) give \( \sigma_1 = \{ X_{N_A} \leftarrow N_A, X_A \leftarrow A \} \) and \( R_1 \)
Solution

\[ T_1 = \{ A, B, \langle A, N_A \rangle, \{ \langle N_A, N_B \rangle \}_k, N_B, \{ \langle K, N_B \rangle \}_k, \{ s \}_K, \text{init, stop} \} \]

\[ C_3 \sigma_1 \quad T_3 = T_2 \cup \{ \{ \langle N_A, N_B \rangle \}_k \} \upharpoonright \{ \langle N_A, X_{N_B} \rangle \}_{K(A,X_B)} \]

\[ C_4 \sigma_1 \quad T_4 = T_3 \cup \{ X_{N_B} \} \upharpoonright \{ N_B \} \]

\[ C_5 \sigma_1 \quad T_5 = T_4 \cup \{ \{ \langle K, N_B \rangle \}_k \} \upharpoonright \{ \langle X_K, X_{N_B} \rangle \}_{K(A,X_B)} \]

\[ C_6 \sigma_1 \quad T_6 = T_5 \cup \{ \{ s \}_K \} \upharpoonright \{ X_s \}_K \]

\[ C_7 \sigma_1 \quad T_7 = T_6 \cup \{ \text{stop} \} \upharpoonright \{ s \} \]

Road book \( \sigma_1 = \{ X_{N_A} \leftarrow N_A, X_A \leftarrow A \} \)

- Apply \( R_2 \) on \( C_3 \) give \( \sigma_2 = \{ X_{N_B} \leftarrow N_B, X_B \leftarrow B \} \) (or \( N_A \)) and \( R_1 \)
Solution

\[ T_1 = \{ A, B, \langle A, N_A \rangle, \{ \langle N_A, N_B \rangle \}_K, N_B, \{ \langle K, N_B \rangle \}_K, \{ s \}_K, \text{init, stop} \} \]

\[ C_5 \sigma_1 \sigma_2 \quad T_5 = T_4 \cup \{ \{ \langle K, N_B \rangle \}_K \} \quad \vdash \quad \{ \langle X_K, N_B \rangle \}_K \]

\[ C_6 \sigma_1 \sigma_2 \quad T_6 = T_5 \cup \{ \{ s \}_K \} \quad \vdash \quad \{ X_s \}_K \]

\[ C_7 \sigma_1 \sigma_2 \quad T_7 = T_6 \cup \{ \text{stop} \} \quad \vdash \quad s \]

Road book \( \sigma_1 = \{ X_{N_A} \leftarrow N_A, X_A \leftarrow A \} \) \( \sigma_2 = \{ X_{N_B} \leftarrow N_B, X_B \leftarrow B \} \)

- Apply \( R_2 \) on \( C_5 \sigma_1 \sigma_2 \) give \( \sigma_3 = \{ X_K \leftarrow N_A \} \)
- Apply \( R_2 \), on \( \sigma_1 \sigma_2 \sigma_3 C_6 \) give \( \sigma_4 = \{ X_S \leftarrow s \} \)
Solution

1. \( A \rightarrow B : \langle A, N_A \rangle \)
2. \( B \rightarrow A : \{\langle N_A, N_B \rangle \}_{K_{ab}} \)
3. \( A \rightarrow B : N_B \)
4. \( B \rightarrow A : \{\langle K, N_B \rangle \}_{K_{ab}} \)
5. \( A \rightarrow B : \{s\}_K \)

The resolution of constraint system gives the following attack:
Send 2nd message \( \{\langle N_A, N_B \rangle \}_{K_{ab}} \) instead of the 4th message \( \{\langle K, N_B \rangle \}_{K_{ab}} \). Hence \( A \) will replay \( \{s\}_{N_A} \) because intruder knows \( N_A \)
Exercises

Exercise 2

\[ A \rightarrow B : \{\langle A, K\rangle\}_{K_{ab}} \]

\[ B \rightarrow A : \{s\}_{K_{ab}} \]

Intruder knows only identities of A and B. Show that the secret data s is preserved by one single session between A and B.
Solution

\[ A \rightarrow B : \{\langle A, K \rangle\}_{K_{ab}} \]
\[ B \rightarrow A : \{s\}_{K_{ab}} \]

\[ T_1 = \{A, B, \{\langle A, K \rangle\}_{K_{ab}}, \{s\}_{K_{ab}}\} \]

Constraint System

\[
\begin{align*}
C_1 & \quad T_1 & \vdash & \{\langle A, X_{K} \rangle\}_{K_{ab}} \\
C_2 & \quad T_2 = T_1 \cup \{\langle A, X_{K} \rangle\}_{K_{ab}} & \vdash & \{s\}_{X_{Kab}} \\
C_3 & \quad T_3 = T_2 \cup \{s\}_{X_{Kab}} & \vdash & s
\end{align*}
\]
Solution

\[ C_1 \quad T_1 \quad \vdash \quad \{\langle A, X_K \rangle\} x_{K_{ab}} \]
\[ C_2 \quad T_2 = T_1 \cup \{\langle A, X_K \rangle\} x_{K_{ab}} \quad \vdash \quad \{s\} x_{K_{ab}} \]
\[ C_3 \quad T_3 = T_2 \cup \{s\} x_{K_{ab}} \quad \vdash \quad s \]

\[ T_1 = \{A, B, \{\langle A, K \rangle\} K_{ab}, \{s\} K_{ab}\} \]

Road book

- Apply nothing or \( R_4 \) or \( R_5 \) and \( R_2 \) on \( C_1 \) give
  \[ \sigma_0 = \{X_K \leftarrow K, X_{K_{ab}} \leftarrow K_{ab}\} \]
- Apply \( R_5 \) or nothing and \( R_2 \), on \( \sigma_0 C_2 \) give \( \sigma_1 = \{X_{N_B} \leftarrow N_B\} \)
  (or \( N_A \))

Each time you meet a solved form of the form \( \perp \) with \( R_6 \).
Outline

1. Active Intruder: Security Problem
2. Unification Notions
   Terms and Messages
   Unification
3. Bounded Number of Sessions
4. NP-Hardness for Bounded Number of Sessions
5. Complexity
6. Tools
7. Conclusion
NP-hardness

Theorem

Decide if a protocol $P$ does not preserve the secrecy of a ground term $s$ from an initial knowledge $T_1$ is NP-difficult.
Recall 3-SAT Problem

**Definition**

**Input:** set of propositional variables \( \{x_1, \ldots, x_n\} \) and a conjunction of clauses with 3 literals.

\[
f(\vec{x}) = \bigwedge_{1 \leq i \leq l} (x_{i,1}^{e_{i,1}} \lor x_{i,2}^{e_{i,2}} \lor x_{i,3}^{e_{i,3}})
\]

where \( e_{i,j} \in \{+, -\} \) and \( x^+ = x, x^- = \neg x \).

**Question:** Does exist a valuation \( V \) of \( \{x_1, \ldots, x_n\} \), such that \( V(f(\vec{x})) = \top \).

**Theorem**

*3-SAT problem is NP-complete.*
**NP-difficulty**

We build a protocol such that an intruder can deduce $s$ iff $f(\vec{x})$ is satisfiable.

$$g(x_{i,j}^{\epsilon_{i,j}}) = \begin{cases} x_{i,j} & \text{if } \epsilon_{i,j} = + \\ \{x_{i,j}\}_K & \text{if } \epsilon_{i,j} = - \end{cases}$$

$$\forall 1 \leq i \leq l : f_i(\vec{x}) = \langle g(x_{i,1}^{\epsilon_{i,1}}), g(x_{i,2}^{\epsilon_{i,2}}), g(x_{i,3}^{\epsilon_{i,3}}) \rangle$$

We suppose Initial intruder knowledge is $\{\bot, \top\}$.

$$A : \langle x_1, \ldots, x_n \rangle \rightarrow \{\langle f_1(\vec{x}), \langle f_2(\vec{x}), \ldots, \langle f_n(\vec{x}), \text{end} \rangle \ldots \rangle \}_p$$

$$\forall 1 \leq i \leq l :$$

$$B_i : \{\langle \{\top\}, \langle x, y \rangle \rangle, z \}_p \rightarrow \{z\}_p$$

$$\overline{B}_i : \{\langle \{\bot\}_K, \langle x, y \rangle \rangle, z \}_p \rightarrow \{z\}_p$$

$$C_i : \{\langle x, \langle \top, y \rangle \rangle, z \}_p \rightarrow \{z\}_p$$

$$\overline{C}_i : \{\langle x, \{\bot\}_K, y \rangle \rangle, z \}_p \rightarrow \{z\}_p$$

$$D_i : \{\langle x, \langle y, \top \rangle \rangle, z \}_p \rightarrow \{z\}_p$$

$$\overline{D}_i : \{\langle x, \langle y, \{\bot\}_K \rangle \rangle, z \}_p \rightarrow \{z\}_p$$

$$E : \{\text{end}\}_p \rightarrow s$$
Outline

1. Active Intruder: Security Problem
2. Unification Notions
   - Terms and Messages
   - Unification
3. Bounded Number of Sessions
4. NP-Hardness for Bounded Number of Sessions
5. Complexity
6. Tools
7. Conclusion
Complexity

Complexity depends of intruder capabilities. In classical Dolev-Yao intruder model we (pair + encryption) we have the following results:

- **Passive Intruder**
  Problem is *polynomial*

- **Bounded Number of sessions**
  Problem is *NP-complete*
  Tools can verify 3-4 sessions: useful to *finds flaws*! OFMC, Cl-Atse, SATMC, FDR, Athena...

- **Unbounded Number of sessions**
  Problem is in general *undecidable*
  Tools are *corrects, but uncomplete* (can find false attacks, can not terminate) Proverif, TA4SP, Scyther.
Outline

1. Active Intruder: Security Problem
2. Unification Notions
   - Terms and Messages
   - Unification
3. Bounded Number of Sessions
4. NP-Hardness for Bounded Number of Sessions
5. Complexity
6. Tools
7. Conclusion
Hermes

Hermes does not proceed by estimating the knowledge of the intruder!

- It computes a set of safe messages and uses a symbolic representation based on patterns to approximate the infinite set of safe messages.
- Hermes checks protocols for bounded and unbounded number of sessions.
  - TACAS’03 - **Pattern-based Abstraction for verifying Secrecy in Protocols**. L. Bozga, Y. Lakhnech and M. Perin
  - CAV’03 - **Hermes, a tool verifying secrecy properties of unbounded security protocols**. L. Bozga, Y. Lakhnech and M. Perin

http://www-verimag.imag.fr/~async/hermes/prouve/hermes.php
developped at Verimag by Y. Bouzouzou, L. Bozga, C. Ene, R. Janvier, Y. Lakhnech, L. Mazaré and M. Périn.
Welcome to Hermes!

Online Examples

- Needham Schroeder bounded scenario
- Needham Schroeder unbounded scenario
- Needham Schroeder iterative scenario
- Needham Schroeder Lowe bounded scenario
- Needham Schroeder Lowe unbounded scenario
- Needham Schroeder Lowe iterative scenario
- Electronic Purse symmetric keys bounded scenario
- Electronic Purse symmetric keys unbounded scenario
- Electronic Purse symmetric keys iterative scenario
- Electronic Vote bounded scenario
- Electronic Vote unbounded scenario

Execute:

```
hermes | Options: hi 
not strong secret 
all abstract executions
```

```
# The Needham-Schroeder Protocole
# A, B : Principal
# Na, Nb : Nonce
# PKa, PKb, PKs, SKa, SKb, SKs : Key
# PKa, SKa : is a key pair
# PKb, SKb : is a key pair
# PKs, SKs : is a key pair
#
# 1. A -> B : {Na, A}PKb
# 2. B -> A : {Na, Nb}PKa
# 3. A -> B : {Nb}PKb

signature
alice, bob, intruder : principal;
PK : principal -> pubkey;
end

role Alice (A: principal; B: principal; SKa: privkey)
declare
Na, v Nb: message;
bEGIN
new(Na);
send(crypt(asym, PK(B), [Na, A]));
recy(crypt(asym, PK(A), [Na, v Nb]));
send(crypt(asym, PK(B), v Nb));

public
alice, bob, intruder, PK

initial
x_M = [ inv(PK(intruder)) ]
```
• **Avispa**
  
  **OFMC**: On-the-fly Model-Checker employs several symbolic techniques to explore the state space in a demand-driven way.

  **CL-AtSe**: Constraint-Logic-based Attack Searcher applies constraint solving with simplification heuristics and redundancy elimination techniques.

  **SATMC**: SAT-based Model-Checker builds a propositional formula encoding all the possible traces (of bounded length) on the protocol and uses a SAT solver.

  **TA4SP**: Tree Automata based on Automatic Approximations for the Analysis of Security Protocols approximates the intruder knowledge by using regular tree languages and rewriting to produce under and over approximations.

• **Proverif**: Analyses unbounded number of session using over-approximation with Horn Clauses.

• **Scyther**: Verifies bounded and unbounded number of session with backwards search based on partially ordered patterns.
On-the-Fly Model-Checker (OFMC)

- Common language for specifying protocols and security properties.
- Supports symmetric and asymmetric keys, cryptographic hash functions, key-tables, user-definable algebraic functions, etc.

<table>
<thead>
<tr>
<th>Input</th>
<th>Output (&lt;1 second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROTOCOL Needham-Schroeder;</td>
<td></td>
</tr>
<tr>
<td>Identifiers</td>
<td></td>
</tr>
<tr>
<td>A, B: user;</td>
<td>A -&gt; Spy: {A,Na}Kspy</td>
</tr>
<tr>
<td>Na, Nb: nonce;</td>
<td>Spy -&gt; B: {A,Na}Kb</td>
</tr>
<tr>
<td>Ka, Kb: public_key;</td>
<td>B -&gt; A: {Na,Nb}Ka</td>
</tr>
<tr>
<td>Messages</td>
<td>A -&gt; Spy: {Nb}Kspy</td>
</tr>
<tr>
<td>1. A -&gt; B: {A,Na}Kb</td>
<td>Spy -&gt; B: {Nb}Kb</td>
</tr>
<tr>
<td>2. B -&gt; A: {Na,Nb}Ka</td>
<td></td>
</tr>
<tr>
<td>3. A -&gt; B: {Nb}Kb</td>
<td></td>
</tr>
<tr>
<td>Intruder_knowledge Spy, b, ka, kb, kspy;</td>
<td></td>
</tr>
<tr>
<td>Goal correspondence_between A B;</td>
<td></td>
</tr>
</tbody>
</table>
Scyther

- Alternative: backwards search based on patterns
  - Security properties represented by claim events in the protocol.
  - Supports symmetric and asymmetric keys, cryptographic hash functions, key-tables, multiple protocols in parallel, composed keys, etc (but no user-definable algebraic functions)
  - Can perform unbounded verification of protocols
  - Provides complete characterization of protocol roles: Answer to: “after execution of a protocol role, what events must also have occurred?”

- Also state-of-art. Freely available for download for Windows, Linux and Mac OS X.
- Will be used in the exercise sessions.
models and analysis of security protocols 1st semester 2007-2008 active intruder tools

input

protocol ns3(I,R) {
  role I {
    const ni: Nonce;
    var nr: Nonce;
    send_1(I,R, {ni,I}pk(R) );
    read_2(R,I, {ni,nr}pk(I) );
    send_3(I,R, {nr}pk(R) );

    claim_i1(I,Secret,ni);
    claim_i2(I,Nisynch);
  }

  role R {
    var ni: Nonce;
    const nr: Nonce;
    read_1(I,R, {ni,I}pk(R) );
    send_2(R,I, {ni,nr}pk(I) );
    read_3(I,R, {nr}pk(R) );

    claim_r1(R,Secret,ni);
    claim_r2(R,Nisynch);
  }
}

output (<0.02 seconds)

[Id 1] Protocol ns3, role R, claim type Secret

Run #1
Alice in role R
I -> Bob
R -> Alice
Const nr#1
Var ni#1
read_1 from Bob
{ ni#2,Bob }pk(Alice)
send_2 to Bob
{ ni#2,nr#1 }pk(Bob)
decrypt
nr#1
encrypt
ni#2

Run #2
Bob in role I
I -> Bob
R -> Eve
Const ni#2
Var nr#1
send_1 to Eve
{ ni#2,Bob }pk(Eve)
read_2 from Eve
{ ni#2,nr#1 }pk(Bob)
decrypt

Initial intruder knowledge
pk(Alice)
sk(Eve)

62 / 68
Proverif uses Horn Clauses

A Horn clause is a logical formula of the form

\[
\frac{L_1, \ldots, L_n}{L} \quad (\equiv \neg L_1 \lor \ldots \lor \neg L_n \lor L)
\]

Formalism simple and homogeneous for
- modeling intruder capabilities
- modeling protocol rules
- checking an unbounded number of sessions

This formalism is used like intermediary representation (translation from high level language “Pi-calculus like”) in the Tool ProVerif [Blanchet2001]

http://www.di.ens.fr/~blanchet/crypto.html
Approximations

- **Nonces** are modeled by functions of previous received messages. If intruder sends the same messages, then the same nonces will be used.

- **One step of the protocol can be executed several times if previous steps are executed at least once.**

  Example:
  1. Intruder sends to A the message $M_1$
  2. A answers by $M_2$
  3. Intruder sends to A the message $M_3$
  4. A answers by $M_4$
  5. Intruder sends to A the message $M'_3$ *(without executing the 2 first steps)*
  6. A replies with $M'_4$
Outline

1 Active Intruder: Security Problem

2 Unification Notions
   Terms and Messages
   Unification

3 Bounded Number of Sessions

4 NP-Hardness for Bounded Number of Sessions

5 Complexity

6 Tools

7 Conclusion
Summary

Today

- Active Intruder
- Bounded Number of Sessions
- NP-Hardness
- Tools
Next Time

- Playing with Tools:
  - Scyther
  - Avispa: OFMC, Cl-Atse, SATMC, TA4SP
  - Proverif
Thank you for your attention

Questions?