A Coordination-based Methodology for Security Protocol Verification

Giacomo Baldi, Andrea Bracciali, Gianluigi Ferrari, Emilio Tuosto

Dipartimento di Informatica
Università di Pisa
{baldig, braccia, giangi, etuosto}@di.unipi.it

work partially supported by the PROFUNDIS project (IST-2001-33100) and by the MIUR project SP4
Analysis of Security Protocols: Outline
Outline of the problem

(1) $A \rightarrow B : \{m\}_k$
Outline of the problem

(1) \[ A \rightarrow B : \{m\}_k \]

1. Implicit assumptions: secrets’ sharing
(= resource sharing, i.e. ▷ interaction topology ◀)
Outline of the problem

(1) \[ A \rightarrow B : \{ m \}_k \]

1. Implicit assumptions: secrets’ sharing
   (= resource sharing, i.e. \( \triangleright \) interaction topology \( \triangleleft \))
2. Model of the environment (I), i.e. the \textit{power} of the intruder
Outline of the problem

1. Implicit assumptions: secrets’ sharing
   (= resource sharing, i.e. △ interaction topology ◀)
2. Model of the environment (I), i.e. the *power* of the intruder
3. Model of the environment (II), i.e. the *context* of the protocol execution

\( (1) \ A \rightarrow B : \ \{m\}_k \)
Outline of the problem

(1) \( A \rightarrow B : \{m\}_k \)

\[
\begin{array}{c}
A_k \longrightarrow \hat{\circ} \longrightarrow B_k \\
\hat{\circ} \longrightarrow \hat{\circ} \longrightarrow B_k
\end{array}
\]

1. Implicit assumptions: secrets’ sharing
   (= resource sharing, i.e. interaction topology \( \triangleleft \))
2. Model of the environment (I), i.e. the power of the intruder
3. Model of the environment (II), i.e. the context of the protocol execution
4. Which properties? i.e. “\( m \) is secret” but also “A (not A) does send \( m \) to B”
Outline of the problem

(1) $A \rightarrow B : \{m\}_k$

1. Implicit assumptions: secrets’ sharing
   (= resource sharing, i.e. △ interaction topology ▽)
2. Model of the environment (I), i.e. the power of the intruder
3. Model of the environment (II), i.e. the context of the protocol execution
4. Which properties ? i.e. “$m$ is secret” but also “$A$ (not $A$) does send $m$ to $B$”
5. Still, not enough to make the problem easy:

   formal methodologies and automated tools may help
Technical background
cIP calculus: explicit secret sharing

(1) $A \rightarrow B : \{m\}_k$

(2) ...

$A = (K)[out(\{m\}_K), \ldots]$  
$B = (J)[in(\{?X\}_J), \ldots]$
cLP calculus: explicit secret sharing

(1) \[ A \rightarrow B : \{m\}_k \]

(2) ...

\[ A = (K)[out(\{m\}_K). \ldots] \]
\[ B = (J)[in(\{?X\}_J). \ldots] \]

finite (non-recursive), typically deterministic processes
cIP calculus: explicit secret sharing

(1) \( A \rightarrow B : \{m\}_k \)

(2) ...

\[
A = (K)[out(\{m\}_K). \ldots]
\]

\[
B = (J)[in(\{?X\}_J). \ldots]
\]

*open variable* binders
cIP calculus: explicit secret sharing

(1) \( A \rightarrow B : \{m\}_k \)

(2) ...

\[
\begin{align*}
A &= (K)[out(\{m\}_K) \ldots] \\
B &= (J)[in(\{?X\}_J) \ldots]
\end{align*}
\]

input binders
cIP calculus: explicit secret sharing

(1) \( A \rightarrow B : \{m\}_k \)

(2) ...

\[
\begin{align*}
A &= (K)[out(\{m\}_K), \ldots] \\
B &= (J)[in(\{?X\}_J), \ldots]
\end{align*}
\]

security by means of communication matching
Multi-session protocol runs

Principal instances

\[ A_1 = (K_1)[out(\{m_1\}K_1). \ldots] \]
\[ B_2 = (J_2)[in(\{?X_2\}J_2). \ldots] \]

+ Mappings

\[ \gamma = \{K_1 \rightarrow k, J_1 \rightarrow k\} \]

= Contexts

\[ \text{join}(A_1, B_2, \gamma, \emptyset) = \begin{cases} A_1 = [out(\{m_1\}_k). \ldots] \\ B_2 = [in(\{?X_2\}_k). \ldots] \end{cases} \]
Protocol runs: context traces

Intruder (Dolev-Yao):
- can not guess keys
- generates all the messages received
- receives all the messages sent
- acquires a knowledge $\kappa$. 
Protocol runs: context traces

Intruder (Dolev-Yao):
- can not guess keys
- receives all the messages sent
- generates all the messages received
- acquires a knowledge $\kappa$.

\[
\kappa \ni m : \exists \gamma \text{ ground s.t. } d\gamma \sim m \quad (\text{in})
\]
\[
\langle (\hat{X}_i)[in(d)].E_i \cup C, \chi, \kappa \rangle \leftrightarrow \langle (\hat{X}_i)[E_i\gamma] \cup C, \chi \gamma, \kappa \rangle
\]

\[
\langle (\hat{X}_i)[out(m)].E_i \cup C, \chi, \kappa \rangle \leftrightarrow \langle (\hat{X}_i)[E_i'] \cup C, \chi, \kappa \cup m \rangle \quad (\text{out})
\]

\[
C' = \text{join}(A_i, \gamma, C) \quad A \triangleq (\hat{X})[E] \quad \text{i new} \\
\langle C, \chi, \kappa \rangle \leftrightarrow \langle C', \chi \gamma, \kappa \cup \{A_i, A_i^+\} \rangle \quad (\text{join})
\]
Protocol runs: context traces

Intruder (Dolev-Yao):
- can not guess keys
- receives all the messages sent
- generates all the messages received
- acquires a knowledge \( \kappa \).

\[
\kappa \triangleright m : \exists \gamma \text{ ground s.t. } d \gamma \sim m
\]

\[
\langle (\tilde{X}_i)[in(d)].E_i \cup C, \chi, \kappa \rangle \leftrightarrow \langle (\tilde{X}_i)[E_i\gamma] \cup C, \chi \gamma, \kappa \rangle \quad (in)
\]

\[
\langle (\tilde{X}_i)[out(m)].E_i \cup C, \chi, \kappa \rangle \leftrightarrow \langle (\tilde{X}_i)[E_i'] \cup C, \chi, \kappa \cup m \rangle \quad (out)
\]

\[
C' = join(A_i, \gamma, C) \quad A \triangleq (\tilde{X})[E] \quad i \text{ new}
\]

\[
\langle C, \chi, \kappa \rangle \leftrightarrow \langle C', \chi \gamma, \kappa \cup \{A_i, A_i^+\} \rangle \quad (join)
\]

Protocol (symbolic) runs:
\[
\langle C, \emptyset, \kappa_{init} \rangle \leftrightarrow \langle \emptyset, \chi, \kappa \rangle
\]

(e.g. \( \chi = x_1 \rightarrow x_1(\kappa^27), \kappa = \{m_2,\{x_2(\kappa^25)\}_k,\ldots\} \))
Expressing properties

PL Logic: predicking over $\kappa$, variables and messages, and relations between senders and receivers (secrecy, integrity, authentication, ...)

$$m \in \kappa \mid m = n \mid \forall A.i : \phi \mid \neg \phi \mid \phi \land \psi,$$
Expressing properties

PL Logic: predicating over $\kappa$, variables and messages, and relations between senders and receivers (secrecy, integrity, authentication, ...)

\[ m \in \kappa \mid m = n \mid \forall A.i : \phi \mid \neg \phi \mid \phi \land \psi, \]

$< \kappa, \chi >$ are (symbolic) models of PL:

\[
\frac{x_i \chi = m \chi}{\kappa \models_\chi x_i = m} (=) \quad \frac{\kappa \ni m \chi}{\kappa \models_\chi m \in \kappa} (\in) \quad \frac{\kappa \not\models_\chi \phi}{\kappa \models_\chi \neg \phi} (\neg) \quad \frac{\kappa \models_\chi \phi \quad \kappa \models_\chi \psi}{\kappa \models_\chi \phi \land \psi} (\wedge)
\]

\[
\frac{\kappa \models_\chi \phi \{^j/i\}}{\kappa \models_\chi \forall A.i : \phi} (\forall) \quad \left[ \begin{array}{c}
\kappa \ni m \\
_{\models_{sim}} \\
x(\kappa) = m
\end{array} \right]
\]
Expressing properties

PL Logic: predicating over \( \kappa \), variables and messages, and relations between senders and receivers (secrecy, integrity, authentication, ...)

\[
m \in \kappa \mid m = n \mid \forall A.i : \phi \mid \neg \phi \mid \phi \land \psi,
\]

\(< \kappa, \chi >\) are (symbolic) models of PL:

\[
\frac{x_i \chi = m \chi}{\kappa} \models_{\chi} x_i = m \quad (=) \quad \frac{\kappa \nsim m \chi}{\kappa} \models_{\chi} m \in \kappa \quad (\in) \quad \frac{\kappa \not\models_{\chi} \phi}{\kappa} \models_{\chi} \neg \phi \quad (\neg) \quad \frac{\kappa \models_{\chi} \phi}{\kappa} \models_{\chi} \phi \land \psi \quad (\land)
\]

\[
\frac{\kappa \models_{\chi} \phi \{ j / i \} \text{ for all } A_j : \kappa \nsim A_j}{\kappa} \models_{\chi} \forall A.i : \phi \quad (\forall) \quad \frac{\kappa \nsim m}{x(\kappa) = m} \quad (\nsim_{\text{sim}})
\]

\[
\chi = \{ x_1 \rightarrow x_1(\kappa_2) \}, \kappa_2 \cap \kappa = \emptyset, \kappa = \{ A_1, ... \}
\]

\[
\kappa \models_{\chi} \forall A.i : \neg x_i \in \kappa
\]
Expressing properties

PL Logic: predicking over $\kappa$, variables and messages, and relations between senders and receivers (secrecy, integrity, authentication, ...)

$m \in \kappa \mid m = n \mid \forall A.i : \phi \mid \neg \phi \mid \phi \land \psi,$

$< \kappa, \chi >$ are (symbolic) models of PL:

\[
\begin{align*}
x_i \chi = m \chi & \Rightarrow \kappa \models_\chi x_i = m \\
\kappa \models_\chi m \chi & \Rightarrow \kappa \models_\chi m \in \kappa \\
\kappa \models_\chi \neg \phi & \Rightarrow \kappa \models_\chi \phi \land \psi \\
\kappa \models_\chi \phi \{^j_i\} & \Rightarrow \kappa \models_\chi \forall A.i : \phi \\
x(\kappa) = m & \Rightarrow (\triangle_{sim})
\end{align*}
\]

\[
\chi = \{x_1 \rightarrow x_1(\kappa_2)\}, \kappa_2 \cap \kappa = \emptyset, \kappa = \{A_1, ...\}
\]

$\kappa \models_\chi \neg x_1(\kappa_2) \in \kappa$

$\kappa \models_\chi \forall A.i : \neg x_i \in \kappa$
Our verification methodology
Methodology

1. Protocol formalisation: cIP calculus and $\mathcal{PL}$
2. Initial secret sharing: a $\mathcal{PL}$ connection formula
3. Intruder knowledge definition
4. Automatic verification phase: $\text{ASPA}_y\text{A}$

Possible iteration of 2, 3 and 4.
The Algorithm

Initialization of the intruder knowledge

Joining Principals

Formula normalisation

Open variables connection

Invariant pruning

Security property check

Invariant check

Intruder rebuilding
(1) $A \rightarrow B : \ na, A$
(2) $B \rightarrow S : \ na, A, nb, B$
(3) $S \rightarrow B : \ \{nb, A, kab\}_{kbs}, \{na, B, kab\}_{kas}$
(4) $B \rightarrow A : \ \{na, B, kab\}_{kas}, \{Tb, A, kab\}_{kbb}, nc, \{na\}_{kab}$
(5) $A \rightarrow B : \ \{nc\}_{kab}$
The methodology: refinement steps

Implicit assumptions (from the previous phase):

- A and B share a session key $k_{ab}$
- A has a ticket issued by B
- The intruder has a copy of the ticket

(1) $A \rightarrow B : ma, \{T_b, A, k_{ab}\}_{k_{bb}}$
(2) $B \rightarrow A : mb, \{ma\}_{k_{ab}}$
(3) $A \rightarrow B : \{mb\}_{k_{ab}}$
The methodology: refinement steps

Implicit assumptions (from the previous phase):
- A and B share a session key $kab$
- A has a ticket issued by B
- The intruder has a copy of the ticket

1. Modeling the protocol:

   $A : (b, sk, tk)$ [ $\text{out}(nma, \{b, A, sk\}_{tk}). \text{in}(\?mb, \{nma\}_{sk}). \text{out}(\{mb\}_{sk})$ ]

   $B : (sk, tk)$ [ $\text{in}(\?ma, \{B, ?u, sk\}_{tk}). \text{out}(nmb, \{ma\}_{sk}). \text{in}(\{nmb\}_{sk})$ ]

   $\forall B.i : \exists A.j : b_j = B_i \rightarrow ma_i = nma_j \land mb_j = nmb_i$. 
The methodology: refinement steps

Implicit assumptions (from the previous phase):

- A and B share a session key $kab$
- A has a ticket issued by B
- The intruder has a copy of the ticket

1. Modeling the protocol:

   $A : (b, sk, tk) \ [out(nma, \{b, A, sk\}_{tk}).\ in(?mb, \{nma\}_{sk}).\ out(\{mb\}_{sk})]$

   $B : (sk, tk) \ [in(?ma, \{B, ?u, sk\}_{tk}). \ out(nmb, \{ma\}_{sk}). \ in(\{nmb\}_{sk})]$

   $\forall B.i : \exists A.j : b_j = B_i \rightarrow ma_i = nma_j \land mb_j = nmb_i.$

2. Connections:

   $\forall A.i : \exists B.j : tk_j = tk_i \rightarrow b_i = B_j \land sk_j = sk_i$
Implicit assumptions (from the previous phase):

- A and B share a session key \( kab \)
- A has a ticket issued by B
- The intruder has a copy of the ticket

1. **Modeling the protocol:**

   \[
   A : (b, s_k, t_k) \quad [ \text{out}(nma, \{b, A, s_k\}_{t_k}). \text{in}(?mb, \{nma\}_{s_k}). \text{out}(\{mb\}_{s_k})] \\
   B : (s_k, t_k) \quad [ \text{in}(?ma, \{B, ?u, s_k\}_{t_k}). \text{out}(nmb, \{ma\}_{s_k}). \text{in}(\{nmb\}_{s_k})] \\
   \]

   \[
   \forall B.i : \exists A.j : b_j = B_i \rightarrow ma_i = nma_j \land mb_j = nmb_i. 
   \]

2. **Connections:**

   - \( \forall A.i : \exists B.j : t_kj = t_ik \rightarrow b_i = B_j \land s_kj = s_ki \)

3. **Intruder knowledge**

   - \( B_1, A_3, \text{and } B_2, A_3 \text{ may share the same session key (ticket) «} \):
   - \( \{ B_2, A_3, s_{kB_2} \}_{tk_{B_2}}, \{ B_1, A_3, s_{kB_1} \}_{tk_{B_1}} \)
4. Discovering an attack

(1) \( A_3 \rightarrow B_2 : \ nma_3, \{B_2, A_3, kab\}_{kb2} \)
(2) \( B_2 \rightarrow I : \ nmb_2, \{nma_3\}_{kab} \)
(3) \( I \rightarrow B_1 : \ nmb_2, \{B_1, A_3, kab\}_{kb1} \)
(4) \( B_1 \rightarrow I : \ nmb_1, \{nmb_2\}_{kab} \)
(5) \( I \rightarrow B_2 : \ \{nmb_2\}_{kab} \)
(6) \( I \rightarrow A_3 : \ nmb_1, \{nma_3\}_{kab} \)
(7) \( A_3 \rightarrow I : \ \{nmb_1\}_{kab} \)
(8) \( I \rightarrow B_1 : \ \{nmb_1\}_{kab} \)

- \( A_3 \) requests authentication to \( B_2 \), which encrypts \( nma_3 \) and proposes \( nmb_2 \)
4. Discovering an attack

(1) \( A_3 \rightarrow B_2 : \ nma_3, \{B_2, A_3, \text{kab}\}_{kb2} \)
(2) \( B_2 \rightarrow I : \ nmb_2, \{nma_3\}_{kab} \)
(3) \( I \rightarrow B_1 : \ nmb_2, \{B_1, A_3, \text{kab}\}_{kb1} \)
(4) \( B_1 \rightarrow I : \ nmb_1, \{nmb_2\}_{kab} \)
(5) \( I \rightarrow B_2 : \ {nmb_2\}_{kab} \)
(6) \( I \rightarrow A_3 : \ nmb_1, \{nma_3\}_{kab} \)
(7) \( A_3 \rightarrow I : \ {nmb_1\}_{kab} \)
(8) \( I \rightarrow B_1 : \ {nmb_1\}_{kab} \)

• \( I \) asks authentication to \( B_1 \), which encrypts \( nmb_2 \) and proposes \( nmb_1 \)
4. Discovering an attack

(1) \( A_3 \rightarrow B_2 : nma_3, \{B_2, A_3, kab\}_{kb2} \)
(2) \( B_2 \rightarrow I : nmb_2, \{nma_3\}_{kab} \)
(3) \( I \rightarrow B_1 : nmb_2, \{B_1, A_3, kab\}_{kb1} \)
(4) \( B_1 \rightarrow I : nmb_1, \{nmb_2\}_{kab} \)
(5) \( I \rightarrow B_2 : \{nmb_2\}_{kab} \)
(6) \( I \rightarrow A_3 : nmb_1, \{nma_3\}_{kab} \)
(7) \( A_3 \rightarrow I : \{nmb_1\}_{kab} \)
(8) \( I \rightarrow B_1 : \{nmb_1\}_{kab} \)

- \( I \) let \( B_2 \) terminate, by means of \( nmb_2 \), that has not been encrypted by \( A_3 \) with whom \( B_2 \) believes to speak.
4. Discovering an attack

(1) $A_3 \rightarrow B_2 : nma_3, \{B_2, A_3, kab\}_{kb2}$

(2) $B_2 \rightarrow I : nmb_2, \{nma_3\}_{kab}$

(3) $I \rightarrow B_1 : nmb_2, \{B_1, A_3, kab\}_{kb1}$

(4) $B_1 \rightarrow I : nmb_1, \{nmb_2\}_{kab}$

(5) $I \rightarrow B_2 : \{nmb_2\}_{kab}$

(6) $I \rightarrow A_3 : nmb_1, \{nma_3\}_{kab}$

(7) $A_3 \rightarrow I : \{nmb_1\}_{kab}$

(8) $I \rightarrow B_1 : \{nmb_1\}_{kab}$

- $I(B_2)$ replays to $A_3$, proposing $nmb_1$, $A_3$ encrypts $nmb_1$, originally proposed by $B_1$ for $I(A_3)$
4. Discovering an attack

(1) \[ A_3 \rightarrow B_2 : \quad nma_3, \{B_2, A_3, kab\}_{kB2} \]
(2) \[ B_2 \rightarrow I : \quad nmb_2, \{nma_3\}_{kab} \]
(3) \[ I \rightarrow B_1 : \quad nmb_2, \{B_1, A_3, kab\}_{kB1} \]
(4) \[ B_1 \rightarrow I : \quad nmb_1, \{nmb_2\}_{kab} \]
(5) \[ I \rightarrow B_2 : \quad \{nmb_2\}_{kab} \]
(6) \[ I \rightarrow A_3 : \quad nmb_1, \{nma_3\}_{kab} \]
(7) \[ A_3 \rightarrow I : \quad \{nmb_1\}_{kab} \]
(8) \[ I \rightarrow B_1 : \quad \{nmb_1\}_{kab} \]

- I let \( B_1 \) terminate and believe it has spoken with \( A_3 \) (which does not receive what sent by \( B_2 \))

\[ \forall B.i : \exists A.j : b_j = B_i \rightarrow ma_i = nma_j \land mb_j = nmb_i \]
\[ b_3 = B_2 \not\rightarrow nma_3 = nma_3 \land nmb_1 = nmb_2 \]
Discussion

- Known attack (within known scenario)
- Connection formula + $\kappa$ for “reconstructing” initial hypothesis
- Attack due to a not expected condition (quite unlucky duplication of the same session key), to foresee all the desired conditions is known to be difficult
- A new run with a “more precise” connection formula allow us to tune analysis, by cutting-off this condition
Experimentation

<table>
<thead>
<tr>
<th>Join</th>
<th>Configurations</th>
<th>Time (s)</th>
<th>Attacks</th>
<th>Configurations</th>
<th>Time (s)</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>10240</td>
<td>58</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\phi_{KSL}$</td>
<td>550</td>
<td>12</td>
<td>0</td>
<td>13218</td>
<td>4:21</td>
<td>0</td>
</tr>
<tr>
<td>$\phi'_{KSL}$</td>
<td>590</td>
<td>34</td>
<td>0</td>
<td>15723</td>
<td>5:07</td>
<td>0</td>
</tr>
</tbody>
</table>

Attack report for the first phase of KSL
## Experimentation

| Join/Knowl. | 2 Instances | | 3 Instances | | 4 Instances | |
|-------------|-------------|-------------|-------------|-------------|-------------|
|              | Conf. | Time (s) | Attacks | Conf. | Time (s) | Attacks | Conf. | Time (s) | Attacks |
| true, $\kappa_0$ | 104 | 0.69 | 0 | 3878 | 1.53 | 8 | – | – | – |
| true, $\bar{\kappa}_0$ | 104 | 0.85 | 0 | 3878 | 1.89 | 8 | 130870 | 2:27 | 16 |
| $\phi_{KSL}, \kappa_0$ | 71 | 0.64 | 0 | 3220 | 1.50 | 6 | – | – | – |
| $\bar{\phi}_{KSL}, \bar{\kappa}_0$ | 71 | 0.80 | 0 | 3220 | 1.85 | 6 | 52692 | 1:16 | 12 |

Attack report for KSL repeated authentication part
### Experimentation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Number of states</th>
<th>Times</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \text{ASPAS}_Y \text{A} )</td>
<td>( \text{TRUST} )</td>
</tr>
<tr>
<td>NS (2 instances)</td>
<td>55</td>
<td>328</td>
</tr>
<tr>
<td>KSL (2 instances)</td>
<td>39</td>
<td>135</td>
</tr>
<tr>
<td>KSL (4 instances)</td>
<td>21742</td>
<td>69875</td>
</tr>
</tbody>
</table>

Comparing \( \text{ASPAS}_Y \text{A} \)
A different approach

Init(a,b):

\[ a \neq b \land a \neq Id0 \]
write \( <a,b> \)
read e
\( <kb,b2> \leftarrow pdecrypt(e,\text{Pub}(S)) \)
\[ b2 = b \]
fresh na
write \( E_p(<na,a>,kb) \)
read e2
\( <na2,nb> \leftarrow pdecrypt(e2,\text{Priv}(a)) \)
\[ na2 = na \]
write \( E_p(nb,kb) \)
assert(secret(nb) or b=Id0)
nil

Which is the protocol part?
Which is the join formula?
Which is the security property?
Init(a,b):
   [a!=b] ; [a!=Id0]
   write <a,b>
   read e
   <kb,b2> <- pdecrypt(e,Pub(S))
   [b2=b]
   fresh na
   write Ep(<na,a>,kb)
   read e2
   <na2,nb> <- pdecrypt(e2,Priv(a))
   [na2=na]
   write Ep(nb,kb)
   assert(secret(nb) or b=Id0)
   nil

Which is the protocol part?
Which is the join formula?
Which is the security property?
Concluding remarks

• A refinement-based verification methodology
  • formal and (semi-) automated
  • supporting fine tuning of specification (separation of concerns)
  • practically usable
  • inspired by open system verification

Future developments:
- extending expressiveness (e.g., time)
- improving efficiency (formulas as heuristic for state exploration)
- better understanding (other) properties and logic
- extending the approach to verification of open system:
  - e.g., connection conditions imply behavioral properties
    (not to allow a given sharing of keys entails safety)
Concluding remarks

- A refinement-based verification methodology
  - formal and (semi-) automated
  - supporting fine tuning of specification (separation of concerns)
  - practically usable
  - inspired by *open system* verification
- Future developments:
Concluding remarks

- A refinement-based verification methodology
  - formal and (semi-) automated
  - supporting fine tuning of specification (separation of concerns)
  - practically usable
  - inspired by open system verification

- Future developments:
  - extending expressiveness (e.g. time)
  - improving efficiency (formulas as heuristic for state exploration)
Concluding remarks

• A refinement-based verification methodology
  • formal and (semi-) automated
  • supporting fine tuning of specification (separation of concerns)
  • practically usable
  • inspired by open system verification

• Future developments:
  • extending expressiveness (e.g. time)
  • improving efficiency (formulas as heuristic for state exploration)
  • better understanding (other) properties and logic
Concluding remarks

- A refinement-based verification methodology
  - formal and (semi-) automated
  - supporting fine tuning of specification (separation of concerns)
  - practically usable
  - inspired by open system verification
- Future developments:
  - extending expressiveness (e.g. time)
  - improving efficiency (formulas as heuristic for state exploration)
  - better understanding (other) properties and logic
  - extending the approach to verification of open system:
    e.g. connection conditions imply behavioural properties
    (not to allow a given sharing of keys entails safety)
Some close approaches

• Murϕ [MMS97] is a very early model checker for security protocols. Security properties and open systems in [MART03]
  • no open variables
  • non-symbolic
  • wrt [MART03] join is a coordination mechanism
• STA [BB02] & TRUST [VAN02] symbolically check security properties on protocols describes as “spi”-like processes
  • ad-hoc logic (i.e., correspondence assertions)
  • in TRUST properties hard-wired in protocols
  • no support for multiple sessions
• Similar languages/different analysis techniques: [CW01] & [BDNN00(A)]
  • in [CW01] for defining events which also relates PN to strand spaces
  • in [BDNN00(A)] for reducing complexity of static analysis
A short bibliography

- ASPASYA is available at http://www.di.unipi.it/~etuosto/aspasya/aspasya.html

Thank you