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

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

Outline of the problem

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

A_k $\qquad\qquad$ B_k

1. Implicit assumptions: secrets' sharing
(= resource sharing, i.e. ▷ interaction topology ◁)

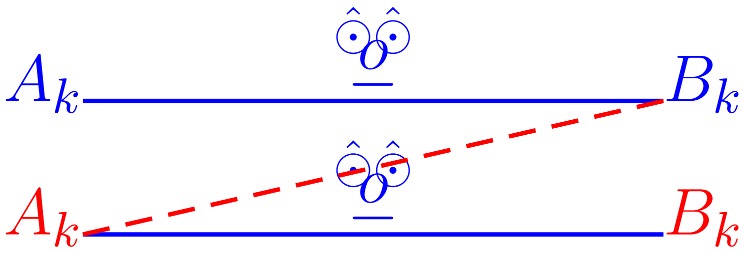
Outline of the problem

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

1. Implicit assumptions: secrets' sharing
(= resource sharing, i.e. ▷ interaction topology ◁)
2. Model of the environment (I), i.e. the *power* of the intruder

Outline of the problem

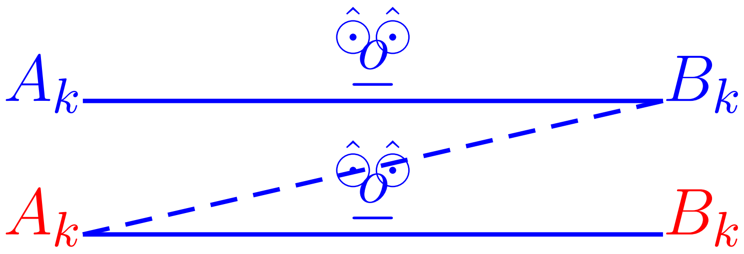
$$(1) \quad A \quad \rightarrow \quad B: \quad \{m\}_k$$



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

Outline of the problem

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



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"*

Outline of the problem

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

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

$$
\begin{array}{llcll}
(1) & A & \rightarrow & B: & \{m\}_k \\
(2) & \ldots & & &
\end{array}
$$

$$
\begin{array}{lcl}
A & = & (K)[out(\{m\}_K).\ \ldots] \\
B & = & (J)[in(\{?X\}_J).\ \ldots]
\end{array}
$$

cIP calculus: explicit secret sharing

$$
\begin{array}{llcl}
(1) & A & \to & B: \ \{m\}_k \\
(2) & \ldots & &
\end{array}
$$

$$
\begin{array}{rcl}
A & = & (K)[out(\{m\}_K). \ \ldots] \\
B & = & (J)[in(\{?X\}_J). \ \ldots]
\end{array}
$$

fi nite (non-recursive), typically deterministic processes

cIP calculus: explicit secret sharing

$$
\begin{array}{llll}
(1) & A \rightarrow B: & \{m\}_k \\
(2) & \ldots
\end{array}
$$

$$
\begin{array}{lll}
A & = & (K)[out(\{m\}_K). \ldots] \\
B & = & (J)[in(\{?X\}_J). \ldots]
\end{array}
$$

open variable binders

cIP calculus: explicit secret sharing

$$\begin{array}{ll} (1) & A \;\rightarrow\; B : \;\; \{m\}_k \\ (2) & \ldots \end{array}$$

$$\begin{array}{lcl} A & = & (K)[out(\{m\}_K).\ \ldots] \\ B & = & (J)[in(\{?X\}_J).\ \ldots] \end{array}$$

input binders

cIP calculus: explicit secret sharing

$$\begin{array}{lllll} (1) & A & \rightarrow & B: & \{m\}_k \\ (2) & \ldots & & & \end{array}$$

$$\begin{array}{lll} A & = & (K)[out(\{m\}_K).\ \ldots] \\ B & = & (J)[in(\{?X\}_J).\ \ldots] \end{array}$$

security by means of communication matching

Multi-session protocol runs

Principal instances

$$
\begin{aligned}
A_1 &= (K_1)[out(\{m_1\}_{K_1}). \ldots] \\
B_2 &= (J_2)[in(\{?X_2\}_{J_2}). \ldots]
\end{aligned}
$$

+

Mappings

$$\gamma = \{K_1 \rightarrow k, J_1 \rightarrow k\}$$

=

Contexts

$$join(A_1, B_2, \gamma, \emptyset) = \left\{ \begin{aligned} A_1 &= [out(\{m_1\}_k). \ldots] \\ B_2 &= [in(\{?X_2\}_k). \ldots] \end{aligned} \right\}$$

Protocol runs: context traces

Intruder (Dolev-Yao):

- can not guess keys
- receives all the messages sent
- generates all the messages received
- acquires a knowledge κ.

Protocol runs: context traces

Intruder (Dolev-Yao):

- can not guess keys
- receives all the messages sent
- generates all the messages received
- acquires a knowledge κ.

$$\frac{\kappa \bowtie m : \ \exists\gamma \text{ ground s.t. } d\gamma \sim m}{\langle(\tilde{X}_i)[in(d).E_i] \cup \mathcal{C}, \chi, \kappa\rangle \longmapsto \langle(\tilde{X}_i)[E_i\gamma] \cup \mathcal{C}, \chi\gamma, \kappa\rangle} \ (in)$$

$$\frac{}{\langle(\tilde{X}_i)[out(m).E_i] \cup \mathcal{C}, \chi, \kappa\rangle \longmapsto \langle(\tilde{X}_i)[E_i'] \cup \mathcal{C}, \chi, \kappa \cup m\rangle} \ (out)$$

$$\frac{\mathcal{C}' = join(A_i, \gamma, \mathcal{C}) \qquad A \triangleq (\tilde{X})[E] \qquad i \text{ new}}{\langle\mathcal{C}, \chi, \kappa\rangle \longmapsto \langle\mathcal{C}', \chi\gamma, \kappa \cup \{A_i, A_i^+\}\rangle} \ (join)$$

Protocol runs: context traces

Intruder (Dolev-Yao):

- can not guess keys
- receives all the messages sent
- generates all the messages received
- acquires a knowledge κ.

$$\frac{\kappa \bowtie m : \ \exists\gamma \text{ ground s.t. } d\gamma \sim m}{\langle(\tilde{X}_i)[in(d).E_i] \cup \mathcal{C}, \chi, \kappa\rangle \longmapsto \langle(\tilde{X}_i)[E_i\gamma] \cup \mathcal{C}, \chi\gamma, \kappa\rangle} \ (in)$$

$$\frac{}{\langle(\tilde{X}_i)[out(m).E_i] \cup \mathcal{C}, \chi, \kappa\rangle \longmapsto \langle(\tilde{X}_i)[E_i'] \cup \mathcal{C}, \chi, \kappa \cup m\rangle} \ (out)$$

$$\frac{\mathcal{C}' = join(A_i, \gamma, \mathcal{C}) \qquad A \triangleq (\tilde{X})[E] \qquad i \text{ new}}{\langle \mathcal{C}, \chi, \kappa\rangle \longmapsto \langle \mathcal{C}', \chi\gamma, \kappa \cup \{A_i, A_i^+\}\rangle} \ (join)$$

Protocol (symbolic) runs: $\langle \mathcal{C}, \emptyset, \kappa_{init}\rangle \longmapsto \langle \emptyset, \chi, \kappa\rangle$

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

Expressing properties

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

$$m \in \kappa \quad | \quad m = n \quad | \quad \forall A.i : \phi \quad | \quad \neg\phi \quad | \quad \phi \wedge \psi,$$

Expressing properties

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

$$m \in \kappa \quad | \quad m = n \quad | \quad \forall A.i : \phi \quad | \quad \neg\phi \quad | \quad \phi \wedge \psi,$$

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

$$\frac{x_i\chi = m\chi}{\kappa \models_\chi x_i = m}(=) \qquad \frac{\kappa \bowtie \ m\chi}{\kappa \models_\chi m \in \kappa}(\in) \qquad \frac{\kappa \not\models_\chi \phi}{\kappa \models_\chi \neg\phi}(\neg) \qquad \frac{\kappa \models_\chi \phi \quad \kappa \models_\chi \psi}{\kappa \models_\chi \phi \wedge \psi}(\wedge)$$

$$\frac{\kappa \models_\chi \phi\{^j/_i\} \quad \text{for all } A_j : \kappa \bowtie A_j}{\kappa \models_\chi \forall A.i : \phi}(\forall) \qquad \left[\frac{\kappa \bowtie m}{x(\kappa) = m}(\bowtie_{sim})\right]$$

Expressing properties

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

$$m \in \kappa \quad | \quad m = n \quad | \quad \forall A.i : \phi \quad | \quad \neg\phi \quad | \quad \phi \wedge \psi,$$

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

$$\frac{x_i\chi = m\chi}{\kappa \models_\chi x_i = m}\,(=) \qquad \frac{\kappa \bowtie \ m\chi}{\kappa \models_\chi m \in \kappa}\,(\in) \qquad \frac{\kappa \not\models_\chi \phi}{\kappa \models_\chi \neg\phi}\,(\neg) \qquad \frac{\kappa \models_\chi \phi \quad \kappa \models_\chi \psi}{\kappa \models_\chi \phi \wedge \psi}\,(\wedge)$$

$$\frac{\kappa \models_\chi \phi\{^j/_i\} \quad \text{for all } A_j : \kappa \bowtie A_j}{\kappa \models_\chi \forall A.i : \phi}\,(\forall) \qquad \left[\frac{\kappa \bowtie m}{x(\kappa) = m}\,(\bowtie_{sim})\right]$$

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

$$\kappa \models_\chi \forall A.i : \neg \ x_i \in \kappa$$

Expressing properties

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

$$m \in \kappa \quad | \quad m = n \quad | \quad \forall A.i : \phi \quad | \quad \neg\phi \quad | \quad \phi \wedge \psi,$$

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

$$\frac{x_i\chi = m\chi}{\kappa \models_\chi x_i = m}\,(=) \qquad \frac{\kappa \bowtie m\chi}{\kappa \models_\chi m \in \kappa}\,(\in) \qquad \frac{\kappa \not\models_\chi \phi}{\kappa \models_\chi \neg\phi}\,(\neg) \qquad \frac{\kappa \models_\chi \phi \quad \kappa \models_\chi \psi}{\kappa \models_\chi \phi \wedge \psi}\,(\wedge)$$

$$\frac{\kappa \models_\chi \phi\{^j/_i\} \quad \text{for all } A_j : \kappa \bowtie A_j}{\kappa \models_\chi \forall A.i : \phi}\,(\forall) \qquad \left[\frac{\kappa \bowtie m}{x(\kappa) = m}\,(\bowtie_{sim})\right]$$

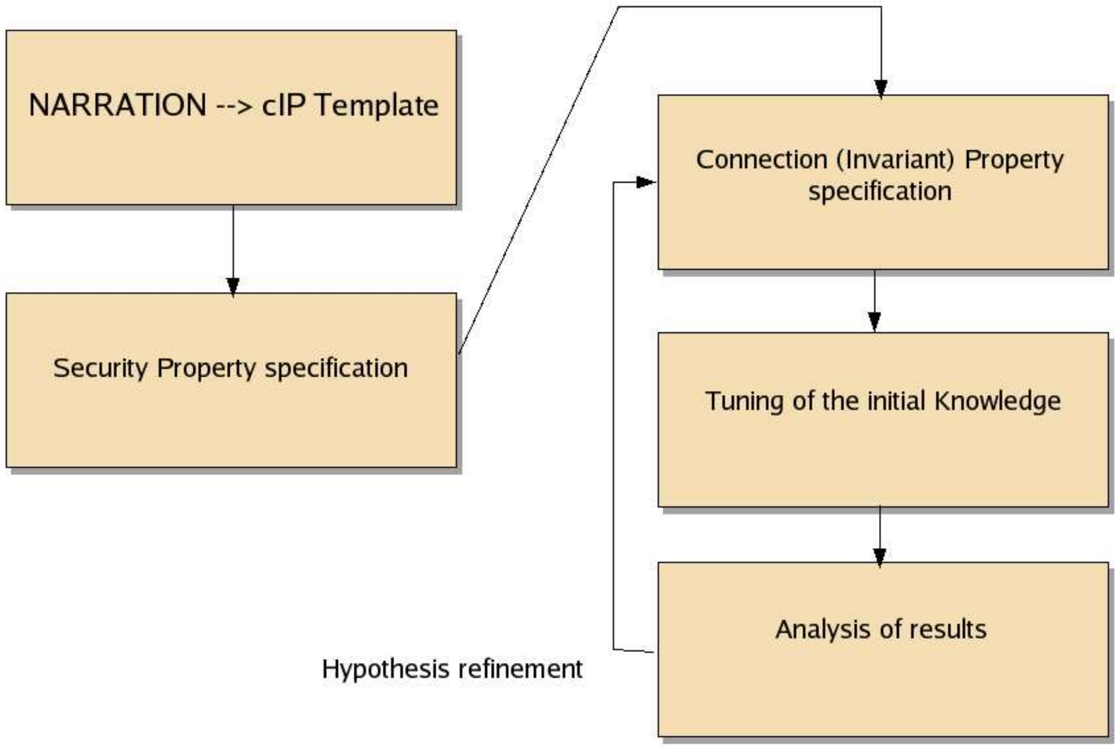
$$\chi = \{x_1 \rightarrow x_1(\kappa_2)\}, \kappa_2 \cap \kappa = \emptyset, \kappa = \{A_1, ...\} \qquad \cfrac{\cfrac{\kappa \not\models_\chi x_1(\kappa_2) \in \kappa}{\kappa \models_\chi \neg\ x_1 \in \kappa}}{\kappa \models_\chi \forall A.i : \neg\ x_i \in \kappa}$$

Our verification methodology

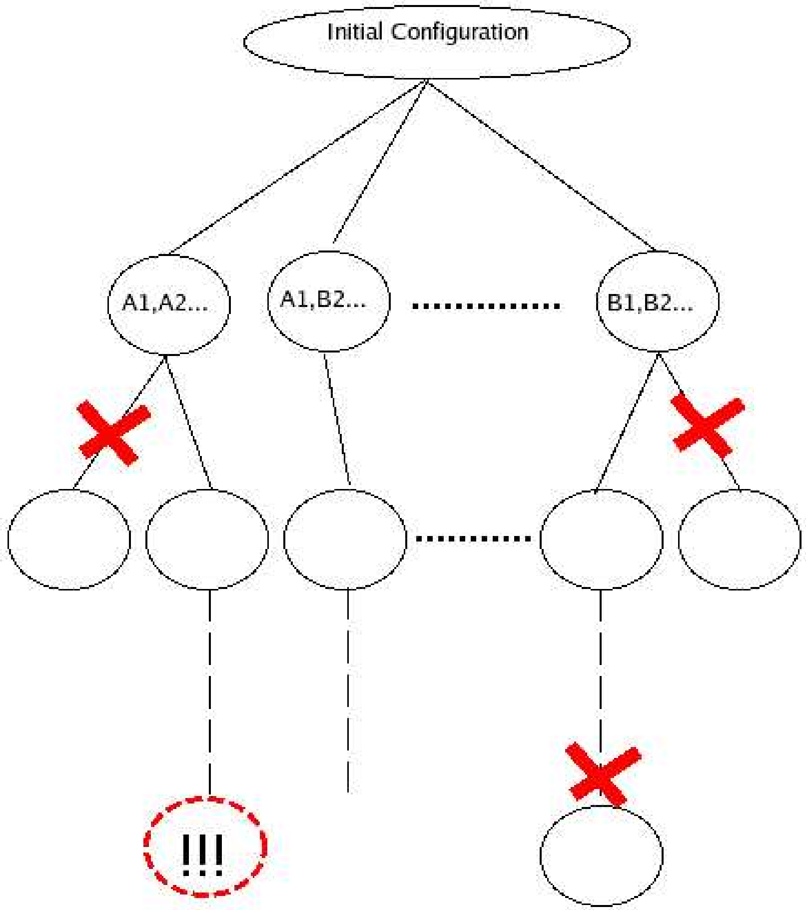
Methodology

1. Protocol formalisation:
   cIP calculus and $\mathcal{PL}$
2. Initial secret sharing:
   a $\mathcal{PL}$ *connection formula*
3. Intruder knowledge defi nition
4. Automatic verifi cation phase:
   $\mathcal{A}$SPASYA

Possible iteration of 2, 3 and 4.



The Algorithm



Initialization of the intruder knowledge

Joinining Principals

Formula normalisation

Open variables connection

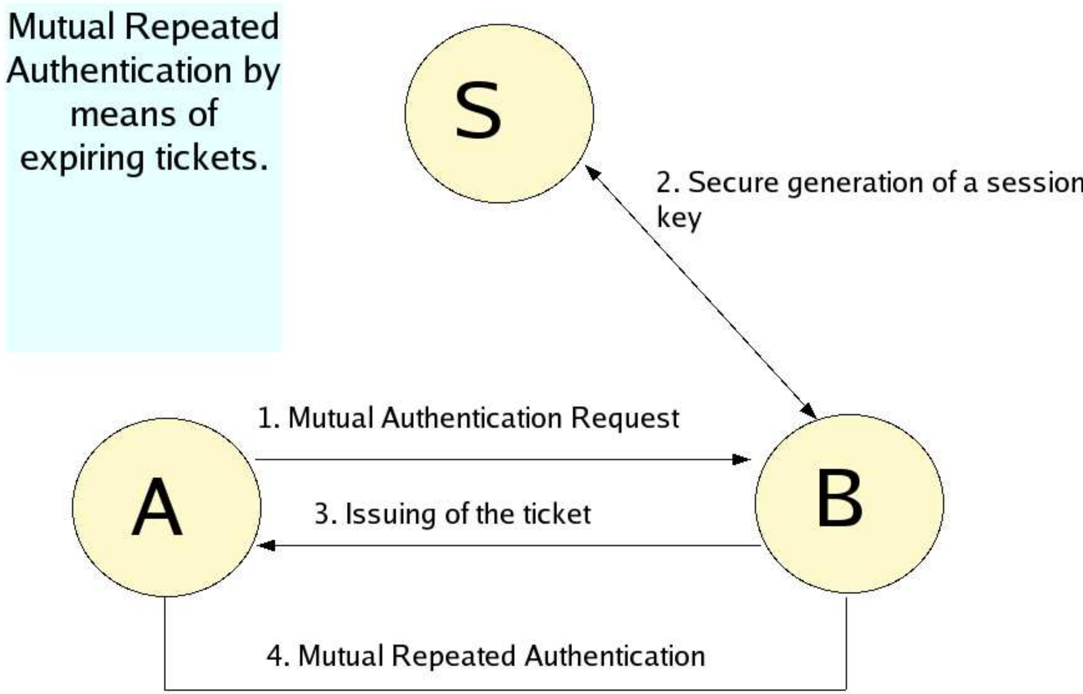
Invariant pruning

Security property check

Invariant check

Intruder rebuilding

KSL: A.Kehne, L. Schonwalder, H. Langendorfer



$$
\begin{array}{lll}
(1) & A \rightarrow B: & na, A \\
(2) & B \rightarrow S: & na, A, nb, B \\
(3) & S \rightarrow B: & \{nb, A, kab\}_{kbs}, \{na, B, kab\}_{kas} \\
(4) & B \rightarrow A: & \{na, B, kab\}_{kas}, \{Tb, A, kab\}_{kbb}, nc, \{na\}_{kab} \\
(5) & A \rightarrow B: & \{nc\}_{kab}
\end{array}
$$

The methodology: refinement steps

$$\begin{array}{lll} (1) & A \rightarrow B: & ma, \{Tb, A, kab\}_{kbb} \\ (2) & B \rightarrow A: & mb, \{ma\}_{kab} \\ (3) & A \rightarrow B: & \{mb\}_{kab} \end{array}$$

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

The methodology: refinement steps

$$
\begin{array}{lll}
(1) & A \rightarrow B: & ma, \{Tb, A, kab\}_{kbb} \\
(2) & B \rightarrow A: & mb, \{ma\}_{kab} \\
(3) & A \rightarrow B: & \{mb\}_{kab}
\end{array}
$$

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:

$$
\begin{array}{ll}
A : (b, sk, tk) & [\; out(nma, \{b, A, sk\}_{tk}).\; in(?mb, \{nma\}_{sk}).\; out(\{mb\}_{sk})] \\
B : (sk, tk) & [\; in(?ma, \{B, ?u, sk\}_{tk}.)\; out(nmb, \{ma\}_{sk}).\; in(\{nmb\}_{sk})]
\end{array}
$$

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

The methodology: refinement steps

$$
\begin{array}{lll}
(1) & A \rightarrow B: & ma, \{Tb, A, kab\}_{kbb} \\
(2) & B \rightarrow A: & mb, \{ma\}_{kab} \\
(3) & A \rightarrow B: & \{mb\}_{kab}
\end{array}
$$

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:

$$
\begin{array}{ll}
A : (b, sk, tk) & [\ out(nma, \{b, A, sk\}_{tk}).\ in(?mb, \{nma\}_{sk}).\ out(\{mb\}_{sk})] \\
B : (sk, tk) & [\ in(?ma, \{B, ?u, sk\}_{tk}.)\ out(nmb, \{ma\}_{sk}).\ in(\{nmb\}_{sk})]
\end{array}
$$

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

2. Connections:

- $\forall A.i : \exists B.j : tk_j = tk_i \rightarrow b_i = B_j \wedge sk_j = sk_i$

The methodology: refinement steps

$$(1)\quad A \to B : \quad ma, \{Tb, A, kab\}_{kbb}$$
$$(2)\quad B \to A : \quad mb, \{ma\}_{kab}$$
$$(3)\quad A \to B : \quad \{mb\}_{kab}$$

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:

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

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

2. Connections:
   - $\forall A.i : \exists B.j : tk_j = tk_i \to b_i = B_j \wedge sk_j = sk_i$
3. Intruder knowledge
   (» B_1, A_3, and B_2, A_3 may share the same session key (ticket) «):
   - $\{ \ \{B_2, A_3, sk_{B_2}\}_{tk_{B_2}}, \ \{B_1, A_3, sk_{B_1}\}_{tk_{B_1}} \ \}$

4. Discovering an attack

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

- A_3 requests authentication to B_2, which encrypts nma_3 and proposes nmb_2

4. Discovering an attack

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

- I asks authentication to B_1, which encrypts nmb_2 and proposes nmb_1

4. Discovering an attack

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

- 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

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

- $I(B_2)$ 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

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

- I let B_1 terminate and believe it has spoken with A_3 (which does not receive what sent by B_2)
- $\forall B.i : \exists A.j : b_j = B_i \rightarrow ma_i = nma_j \ \wedge \ mb_j = nmb_i$

  $$b_3 = B_2 \not\rightarrow nma_3 = nma_3 \ \wedge \ nmb_1 = nmb_2$$

Discussion

- Known attack (within known scenario)
- Connection formula + κ 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 diffi cult
- A new run with a “more precise” connection formula allow us to tune analysis, by cutting-off this condition

Experimentation

	3 Instances			4 Instances		
Join	Configurations	Time (s)	Attacks	Configurations	Time (s)	Attacks
$true$	10240	58	0	–	–	–
ϕ_{KSL}	550	12	0	13218	4:21	0
ϕ'_{KSL}	590	34	0	15723	5:07	0

Attack report for the first phase of KSL

Experimentation

	2 Instances			3 Instances			4 Instances		
Join/Knowl.	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
$\bar{\phi}_{KSL}, \kappa_0$	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

	Number of states			Times		
Protocol	$\mathcal{A}$SPASYA	TRUST	STA	$\mathcal{A}$SPASYA	TRUST	STA
NS (2 instances)	55	328	24	0.7	0.06	0.07
KSL (2 instances)	39	135	33	0.8	0.04	0.04
KSL (4 instances)	21742	69875	-	43	1.8	-

Comparing $\mathcal{A}$SPASYA

A different approach

```
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?

A different approach

```
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

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 euristic 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 euristic 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 euristic 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 defi ning events which also relates PN to strand spaces
  - in [BDNN00(A)] for reducing complexity of static analisis

A short bibliography

- A. Bracciali, A. Brogi, G. Ferrari, E. Tuosto “Security Issues in Component Based Design”, ConCoord 2001
- A. Bracciali, A. Brogi, G. Ferrari, E. Tuosto “Security and Dynamic Compositions of Open Systems”, PDPTA 2002
- A. Bracciali “Behavioural Patterns and Software Composition” PhD Thesis Pisa, 2003
- E. Tuosto “Non-Functional Aspects of Wide Area Network Programming” PhD Thesis Pisa, 2003
- G. Baldi “Security protocol verifi cation by means of symbolic model checking” Master Thesis Pisa, 2004

- $\mathcal{A}$SPASYA is available at http://www.di.unipi.it/˜etuosto/aspasya/aspasya.html

Thank you