Protocol Verification Techniques - State Enumeration - Continuation

Design and Verification of Security Protocols and Security Ceremonies

Programa de Pós-Graduacão em Ciências da Computacão
Dr. Jean Everson Martina

August-November 2016

Programa de Pós-Graduação em Ciência da Computação UFSC

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Before we start!

Attention!

This lecture is a continuation of the previous lecture. We will start with a quick glimpse of what need to be remembered to continue with the contents.

CSP Primitives

- CSP provides two classes of primitives in its process algebra:
- Events:
  - Events represent communications or interactions;
  - Events are assumed to be indivisible and instantaneous;
  - They may be atomic names, compound names, or input/output events;
- Primitive processes:
  - Primitive processes represent fundamental behaviours;
  - STOP (the process that communicates nothing, also called deadlock);
  - SKIP (which represents successful termination).

CSP Algebraic Operators

$$
\begin{array}{rcll}
\mathit{Proc} & ::= & \mathit{STOP} & \\
& | & \mathit{SKIP} & \\
& | & e \rightarrow \mathit{Proc} & \text{(prefixing)} \\
& | & \mathit{Proc} \mathbin{\Box} \mathit{Proc} & \text{(external choice)} \\
& | & \mathit{Proc} \sqcap \mathit{Proc} & \text{(nondeterministic choice)} \\
& | & \mathit{Proc} \mathbin{|||} \mathit{Proc} & \text{(interleaving)} \\
& | & \mathit{Proc} \mathbin{|[\{X\}]|} \mathit{Proc} & \text{(interface parallel)} \\
& | & \mathit{Proc} \setminus X & \text{(hiding)} \\
& | & \mathit{Proc}; \mathit{Proc} & \text{(sequential composition)} \\
& | & \text{if } b \text{ then } \mathit{Proc} \text{ else } \mathit{Proc} & \text{(boolean conditional)} \\
& | & \mathit{Proc} \rhd \mathit{Proc} & \text{(timeout)} \\
& | & \mathit{Proc} \mathbin{\triangle} \mathit{Proc} & \text{(interrupt)}
\end{array}
$$

Needham-Schroeder Public Key Protocol

1. A $\rightarrow$ B: $\{|N_a, A|\}_{K_b}$
2. B $\rightarrow$ A: $\{|N_a, N_b|\}_{K_a}$
3. A $\rightarrow$ B: $\{|N_b|\}_{K_b}$

NSPKP Goals

- The goal of the protocol is to establish mutual authentication between two parties A and B in the presence of adversary;
- A and B obtain a secret shared key though direct communication using public key cryptography;
- This adversary can intercept messages, delay messages, read and copy messages and generate messages;
- This adversary can not learn the privates keys of principals.

Lowe's Specification using CSP

- $MSG1 \equiv \{Msg1.a.b.Encrypt.k.n_a.a' \mid a \in Initiator, a' \in Initiator, b \in Responder, k \in Key, n_a \in Nonce\}$,
- $MSG2 \equiv \{Msg2.b.a.Encrypt.k.n_a.n_b \mid a \in Initiator, b \in Responder, k \in Key, n_a \in Nonce, n_b \in Nonce\}$,
- $MSG3 \equiv \{Msg3.a.b.Encrypt.k.n_b \mid a \in Initiator, b \in Responder, k \in Key, n_b \in Nonce\}$,
- $MSG \equiv MSG1 \cup MSG2 \cup MSG3$.

Lowe's Specification using CSP

- Standard communications in the system will be modelled by the channel comm;
- We also want to model the fact that the intruder can fake or intercept messages, and so we introduce extra channels fake and intercept;
- $\mathit{channel} \quad \mathit{comm}, \mathit{fake}, \mathit{intercept} : \mathit{MSG}.$
- This will ensure that the receiver of a faked message is not aware that it is a fake, and that the sender of an intercepted message is not aware that it is intercepted.

Lowe's Specification using CSP

- We introduce two extra channels, defining the external interface of the protocol;
- We represent a request from a user for initiator "a" to connect with responder "b" by the event *user.a.b*;
- We represent the resulting session by the event *session.a.b*;

Lowe's Specification using CSP

- We also add channels to represent the state of the agents:
  - We represent the initiator "a" thinking it is taking part in a run of the protocol with "b" by the event *I_running.a.b*;
  - We represent the responder "b" thinking it is taking part in a run of the protocol with "a" by the event *R_running.a.b*;
  - We represent the initiator committing to the session by the *I_commit.a.b*;
  - We represent the responder committing to the session by *R_commit.a.b*;
- We declare these channels by:
- *channel user, session, I_running, R_running, I_commit, R_.commit : Initiator.Responder.*

Lowe's Specification using CSP

- We will represent a responder with identity "a", who has a single nonce n_a, by the CSP process $INITIATOR(a, na)$.;

Lowe's Specification using CSP

- We will represent a responder with identity "a", who has a single nonce n_a, by the CSP process $INITIATOR(a, na)$.;
- If we want to consider a responder with more than one nonce, then we can compose several such processes, either sequentially or interleaved;

Lowe's Specification using CSP

- We will represent a responder with identity "a", who has a single nonce n_a, by the CSP process $INITIATOR(a, na)$.;
- If we want to consider a responder with more than one nonce, then we can compose several such processes, either sequentially or interleaved;
- Ignoring, for the moment, the possibility of intruder action, the process can be defined by:
    - $INITIATOR(a, n_a) \equiv user.a?b \rightarrow I\_.running.a.b \rightarrow$
    $comm!Msg1.a.b.Encrypt.key(b).n_a, a \rightarrow$
    $comm.Msg2.b.a.Encrypt.key(a)?n'_a.n_b \rightarrow$
    $if \quad n_a = n'_a$
    $then \quad comm!Msg3.a.b.Encrypt.key(b).n_b \rightarrow$
    $I\_commit.a.b \rightarrow session.a.b \rightarrow SKIP$
    $else \quad STOP.$

Lowe's Specification using CSP

- We now introduce the possibility of enemy action by applying a renaming to the above process;

Lowe's Specification using CSP

- We now introduce the possibility of enemy action by applying a renaming to the above process;
- Our renaming should ensure that message ls and message 3s sent by the initiator can be intercepted;

Lowe's Specification using CSP

- We now introduce the possibility of enemy action by applying a renaming to the above process;
- Our renaming should ensure that message ls and message 3s sent by the initiator can be intercepted;
- And message 2s can be faked;

Lowe's Specification using CSP

- We now introduce the possibility of enemy action by applying a renaming to the above process;
- Our renaming should ensure that message ls and message 3s sent by the initiator can be intercepted;
- And message 2s can be faked;
- We define an initiator with identity A and nonce Na by:
  $INITIATOR1 \equiv INITIATOR(A, N_a)$
  $[[comm.Msg1 \leftarrow comm.Msg1, comm.Msg1 \leftarrow intercept.Msg1,$
  $comm.Msg2 \leftarrow comm.Msg2, comm.Msg2 \leftarrow fake.Msg2,$
  $comm.Msg3 \leftarrow comm.Msg3, comm.Msg3 \leftarrow intercept.Msg3]].$

Slowing Down!

Slowing Down!

From this point on we will start to slow down because the content thickens...

Lowe's Intruder

- The intruder should be able to:

Lowe's Intruder

- The intruder should be able to:
  - Overhear and/or intercept any messages being passed in the system;

Lowe's Intruder

- The intruder should be able to:
    - Overhear and/or intercept any messages being passed in the system;
    - Decrypt messages that are encrypted with his own public key, so as to learn new nonces;

Lowe's Intruder

- The intruder should be able to:
  - Overhear and/or intercept any messages being passed in the system;
  - Decrypt messages that are encrypted with his own public key, so as to learn new nonces;
  - Introduce new messages into the system, using nonces he knows;

Lowe's Intruder

- The intruder should be able to:
  - Overhear and/or intercept any messages being passed in the system;
  - Decrypt messages that are encrypted with his own public key, so as to learn new nonces;
  - Introduce new messages into the system, using nonces he knows;
  - Replay any message he has seen (possibly changing plain-text parts), even if he does not understand the contents of the encrypted part;

Lowe's Intruder

- The intruder should be able to:
  - Overhear and/or intercept any messages being passed in the system;
  - Decrypt messages that are encrypted with his own public key, so as to learn new nonces;
  - Introduce new messages into the system, using nonces he knows;
  - Replay any message he has seen (possibly changing plain-text parts), even if he does not understand the contents of the encrypted part;

Lowe's Intruder

- Replay any message he has seen (possibly changing plain-text parts), even if he does not understand the contents of the encrypted part;

Lowe's Intruder

- Replay any message he has seen (possibly changing plain-text parts), even if he does not understand the contents of the encrypted part;
- We assume that the intruder is a user of the computer network;

Lowe's Intruder

- Replay any message he has seen (possibly changing plain-text parts), even if he does not understand the contents of the encrypted part;
- We assume that the intruder is a user of the computer network;
- We will define the most general (i.e. the most non-deterministic) intruder who can act as above;

Lowe's Intruder

- Replay any message he has seen (possibly changing plain-text parts), even if he does not understand the contents of the encrypted part;
- We assume that the intruder is a user of the computer network;
- We will define the most general (i.e. the most non-deterministic) intruder who can act as above;
- We consider an intruder with identity I, with public key K_i, who initially knows a nonce N_i.

Lowe's Intruder Definition in CSP

$I(m1s, m2s, m3s, ns)$

Lowe's Intruder Definition in CSP

$$
\begin{aligned}
&I(m1s, m2s, m3s, ns) \\
&\quad comm.Msg1?a.b.Encrypt.k.n.a' \rightarrow \\
&\quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\}) \\
&\quad else \quad I(m1s \cup \{Encrypt.k.n.a'\}, m2s, m3s, ns)
\end{aligned}
$$

Lowe's Intruder Definition in CSP

$$
\begin{array}{l}
I(m1s, m2s, m3s, ns) \\
\quad comm.Msg1?a.b.Encrypt.k.n.a' \rightarrow \\
\quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\}) \\
\quad else \quad I(m1s \cup \{Encrypt.k.n.a'\}, m2s, m3s, ns) \\
\quad \Box intercept.Msg1?a.b.Encrypt.k.n.a' \rightarrow \\
\quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\}) \\
\quad else \quad I(m1s \cup \{Encrypt.k.n.a'\}, m2s, m3s, ns)
\end{array}
$$

Lowe's Intruder Definition in CSP

$$
\begin{array}{l}
I(m1s, m2s, m3s, ns) \\
\quad comm.Msg1?a.b.Encrypt.k.n.a' \rightarrow \\
\quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\}) \\
\quad else \quad I(m1s \cup \{Encrypt.k.n.a'\}, m2s, m3s, ns) \\
\quad \Box intercept.Msg1?a.b.Encrypt.k.n.a' \rightarrow \\
\quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\}) \\
\quad else \quad I(m1s \cup \{Encrypt.k.n.a'\}, m2s, m3s, ns) \\
\quad \Box comm.Msg2?b.a.Encrypt.k.n.n' \rightarrow \\
\quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n, n'\}) \\
\quad else \quad I(m1s, m2s \cup Encrypt.k, n.n', m3s, ns)
\end{array}
$$

Lowe's Intruder Definition in CSP

$$
\begin{aligned}
& I(m1s, m2s, m3s, ns) \\
& \quad comm.Msg1?a.b.Encrypt.k.n.a' \rightarrow \\
& \quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\}) \\
& \quad else \quad I(m1s \cup \{Encrypt.k.n.a'\}, m2s, m3s, ns) \\
& \quad \Box intercept.Msg1?a.b.Encrypt.k.n.a' \rightarrow \\
& \quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\}) \\
& \quad else \quad I(m1s \cup \{Encrypt.k.n.a'\}, m2s, m3s, ns) \\
& \quad \Box comm.Msg2?b.a.Encrypt.k.n.n' \rightarrow \\
& \quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n, n'\}) \\
& \quad else \quad I(m1s, m2s \cup Encrypt.k, n.n', m3s, ns) \\
& \quad \Box intercept.Msg2?b.a.Encrypt.k.n.n' \rightarrow \\
& \quad if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n, n'\}) \\
& \quad elseI(m1s, m2s \cup Encrypt.k.n.n', m3s, ns)
\end{aligned}
$$

Lowe's Intruder Definition in CSP

$\square comm.Msg3?a.b.Encrypt.k.n \rightarrow$
$if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\})$
$elseI(mls, m2s, rn3s \cup Encrypt.k.n, ns)$

Lowe's Intruder Definition in CSP

$\Box comm.Msg3?a.b.Encrypt.k.n \rightarrow$
$if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\})$
$else I(mls, m2s, rn3s \cup Encrypt.k.n, ns)$
$\Box intercept.Msg3?a.b.Encrypt.k.n \rightarrow$
$if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\})$
$else I(m1s, m2s, m3s \cup Encrypt.k.n, ns)$

Lowe's Intruder Definition in CSP

$\Box comm.Msg3?a.b.Encrypt.k.n \rightarrow$
$if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\})$
$elseI(mls, m2s, rn3s \cup Encrypt.k.n, ns)$
$\Box intercept.Msg3?a.b.Encrypt.k.n \rightarrow$
$if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\})$
$elseI(m1s, m2s, m3s \cup Encrypt.k.n, ns)$
$\Box fake.Msg1?a.b?m : m1s \rightarrow I(m1s, m2s, m3s, ns)$
$\Box fake.Msg2?a.b?m : m2s \rightarrow I(m1s, m2s, m3s, ns)$
$\Box fake.Msg3?a.b?m : m3s \rightarrow I(m1s, m2s, m3s, ns)$

Lowe's Intruder Definition in CSP

$\Box comm.Msg3?a.b.Encrypt.k.n \rightarrow$
$if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\})$
$elseI(mls, m2s, rn3s \cup Encrypt.k.n, ns)$
$\Box intercept.Msg3?a.b.Encrypt.k.n \rightarrow$
$if \quad k = K_i \quad then \quad I(m1s, m2s, m3s, ns \cup \{n\})$
$elseI(m1s, m2s, m3s \cup Encrypt.k.n, ns)$
$\Box fake.Msg1?a.b?m : m1s \rightarrow I(m1s, m2s, m3s, ns)$
$\Box fake.Msg2?a.b?m : m2s \rightarrow I(m1s, m2s, m3s, ns)$
$\Box fake.Msg3?a.b?m : m3s \rightarrow I(m1s, m2s, m3s, ns)$
$\Box fake.Msg1?a.blEncrypt?k?n : ns?a' \rightarrow I(m1s, m2s, m3s, ns)$
$\Box fake.Msg2?b.a!Encrypt?k?n : ns?n' : ns \rightarrow I(m1s, m2s, m3s, ns)$
$\Box fake.Msg3?a.b!Encrypt?k?n : ns \rightarrow I(m1s, m2s, m3s, ns).$

Lowe's Intruder Definition in CSP

- We consider an intruder who initially knows the nonce N_i:
  - $INTRUDER \equiv I(\{\}, \{\}, \{\}, \{N_i\})$.

Lowe's Intruder Definition in CSP

- We consider an intruder who initially knows the nonce N_i:
  - $INTRUDER \equiv I(\{\}, \{\}, \{\}, \{N_i\})$.
- We may now define a system with an intruder:
  - $AGENTS \equiv$
    $INITIATOR1 |[\{| comm, session.A.B |\}]| RESPONDER1$,
    $SYSTEM \equiv AGENTS |[\{| fake, comm, intercept |\}]| INTRUDER$.

Refining CSP with FDR

- FDR takes as two inputs, a specification and an implementation, and test whether the implementation refines the specification;

Refining CSP with FDR

- FDR takes as two inputs, a specification and an implementation, and test whether the implementation refines the specification;
- We are working in the traces model of CSP, so checking for refinement amounts to testing whether each trace of the implementation is also a trace of the specification;

Testing NSPKP with FDR

- To test whether the protocol correctly authenticates the responder, we need to find a specification that allows only those traces where the initiator A commits to a session with B only if B has indeed taken part in the protocol run;

Testing NSPKP with FDR

- To test whether the protocol correctly authenticates the responder, we need to find a specification that allows only those traces where the initiator A commits to a session with B only if B has indeed taken part in the protocol run;
- The initiator committing to a session is represented by an I_commit.A.B event; the responder taking part in a run of the protocol with A is represented by R_running.A.B;

Testing NSPKP with FDR

- To test whether the protocol correctly authenticates the responder, we need to find a specification that allows only those traces where the initiator A commits to a session with B only if B has indeed taken part in the protocol run;
- The initiator committing to a session is represented by an I_commit.A.B event; the responder taking part in a run of the protocol with A is represented by R_running.A.B;
- Thus the authenticity of the responder can be tested using the specification AR:
  - $AR_0 \equiv R\_running.A.B \rightarrow I\_commit.A.B \rightarrow AR_0$,
    $A1 \equiv \{|R\_running.A.B, I\_commit.A.B|\}$,
    $AR \equiv AR_0 ||| RUN(\sum \ A1)$.

Testing NSPKP with FDR

- We now consider authentication of the initiator. The protocol should ensure that the responder B commits to a session with initiator A only if A took part in the protocol run;

Testing NSPKP with FDR

- We now consider authentication of the initiator. The protocol should ensure that the responder B commits to a session with initiator A only if A took part in the protocol run;
- Formally, an R_commit.A.B event should occur only if there has been a corresponding I_running.A.B event;

Testing NSPKP with FDR

- We now consider authentication of the initiator. The protocol should ensure that the responder B commits to a session with initiator A only if A took part in the protocol run;
- Formally, an R_commit.A.B event should occur only if there has been a corresponding I_running.A.B event;
- This requirement is captured by the specification AI:
  - $AI_0 \equiv I\_running.A.B \rightarrow R_commit.A.B \rightarrow AI_0$,
    $A_2 \equiv \{|I\_running.A.B, R\_commit.A.B|\}$,
    $AI \equiv AI_0 ||| RUN(\sum \; A2)$.

Testing NSPKP with FDR - the Flaw

- FDR can be used to discover that SYSTEM does not refine AI;

Testing NSPKP with FDR - the Flaw

- FDR can be used to discover that SYSTEM does not refine AI;
- It finds that after the trace:
  - $(user.A.I, I\_running.A.I,$
    $intercept.Msg1.A.I.Encrypt.K_i.N_a.A,$
    $fake.Msgl.A.B.Encrypt.K_b.N_a.A,$
    $intercept.Msg2.B.A.Encrypt.K_a.N_a.N_b,$
    $fake.Msg2.I.A.Encrypt.K_a.N_a.N_b,$
    $intercept.Msg3.A.I.Encrypt.K_i.N_b,$
    $fake.Msg3.A.B.Encrypt.K_b.N_b)$

Lowe's Attack - Translated

1. A $\rightarrow$ C: $\{|N_a, A|\}_{K_c}$
1'. C(A) $\rightarrow$ B: $\{|N_a, A|\}_{K_b}$
2'. B $\rightarrow$ C(A): $\{|N_a, N_b|\}_{K_a}$
2. C $\rightarrow$ A: $\{|N_a, N_b|\}_{K_a}$
3. A $\rightarrow$ C: $\{|N_b|\}_{K_c}$
3'. C $\rightarrow$ B: $\{|N_b|\}_{K_b}$

- Bob believes to be talking to Alice , while he is talking to Charlie;

Lowe's Attack - Translated

1. $\quad$ A $\rightarrow$ C: $\{|N_a, A|\}_{K_c}$
1'. $\quad$ C(A) $\rightarrow$ B: $\{|N_a, A|\}_{K_b}$
2'. $\quad$ B $\rightarrow$ C(A): $\{|N_a, N_b|\}_{K_a}$
2. $\quad$ C $\rightarrow$ A: $\{|N_a, N_b|\}_{K_a}$
3. $\quad$ A $\rightarrow$ C: $\{|N_b|\}_{K_c}$
3'. $\quad$ C $\rightarrow$ B: $\{|N_b|\}_{K_b}$

- Bob believes to be talking to Alice , while he is talking to Charlie;
- Charlie uses Alice as an oracle to answers Bob's challenges;

Lowe's Attack - Translated

1. A $\rightarrow$ C: $\{|N_a, A|\}_{K_c}$
1'. C(A) $\rightarrow$ B: $\{|N_a, A|\}_{K_b}$
2'. B $\rightarrow$ C(A): $\{|N_a, N_b|\}_{K_a}$
2. C $\rightarrow$ A: $\{|N_a, N_b|\}_{K_a}$
3. A $\rightarrow$ C: $\{|N_b|\}_{K_c}$
3'. C $\rightarrow$ B: $\{|N_b|\}_{K_b}$

- Bob believes to be talking to Alice , while he is talking to Charlie;
- Charlie uses Alice as an oracle to answers Bob's challenges;
- Charlie can use Nb to prove to Bob he is Alice.

Discussion

- Would Lowe be able to find this attack by hand?

Discussion

- Would Lowe be able to find this attack by hand?
- What else could be testes using this strategy?

Discussion

- Would Lowe be able to find this attack by hand?
- What else could be testes using this strategy?
- Was it the methodology or the Threat Model?

Questions????

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

UNIVERSIDADE FEDERAL
DE SANTA CATARINA