A Syntactic Criterion for Injectivity of Authentication Protocols

Cas Cremers

joint work with Sjouke Mauw and Erik de Vink

ccremers@win.tue.nl

ECSS group

Eindhoven Computer Science Security (ECSS) group

Goal:

To study the design and analysis of secure systems from a fundamental point of view

Topics:

- Security protocol analysis
- Multi-party protocols
- Ad-hoc/sensor networks
- Smartcard security
- Attack trees
- Digital Rights Management
- RFID security
- Privacy

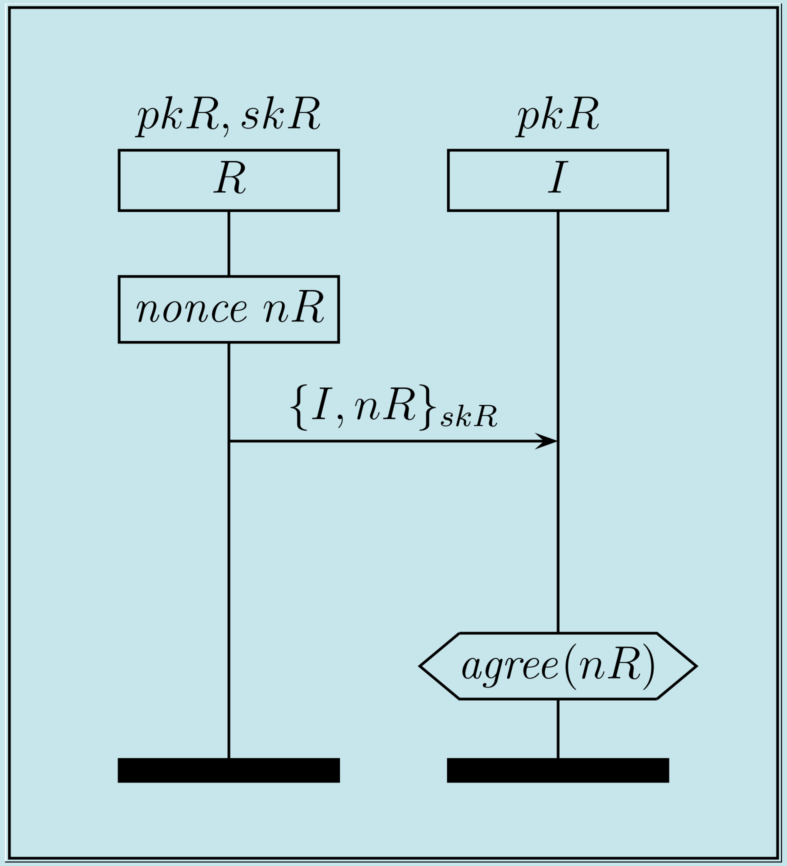
TU/e

Overview

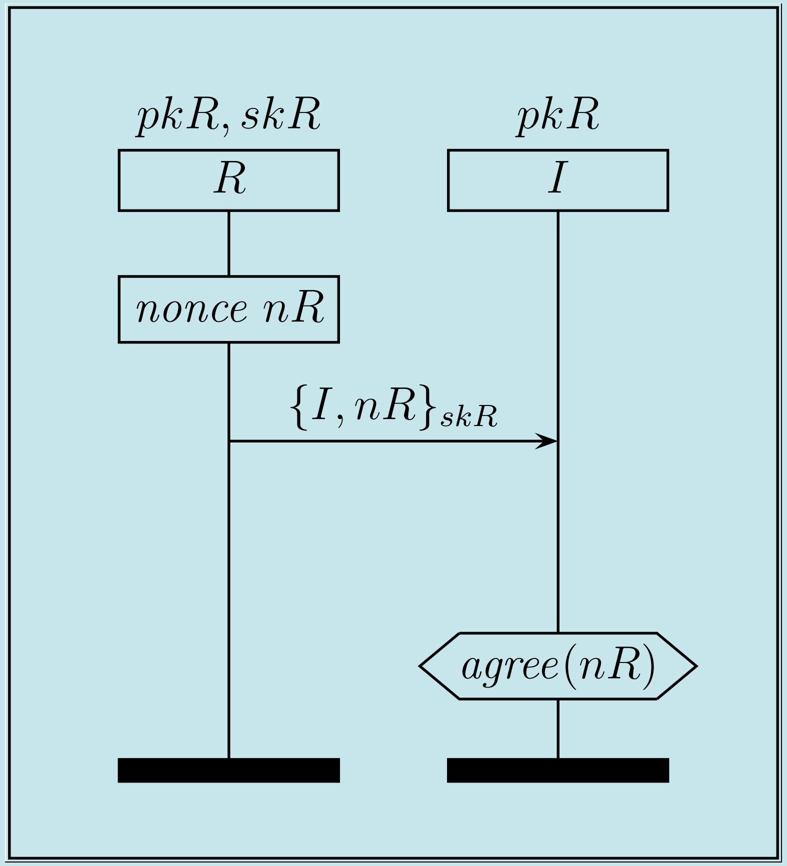
- Motivation
- Problem statement
- Main theorem
- Necessity of preconditions
- Conclusions

TU/e

Example: unilateral authentication protocol

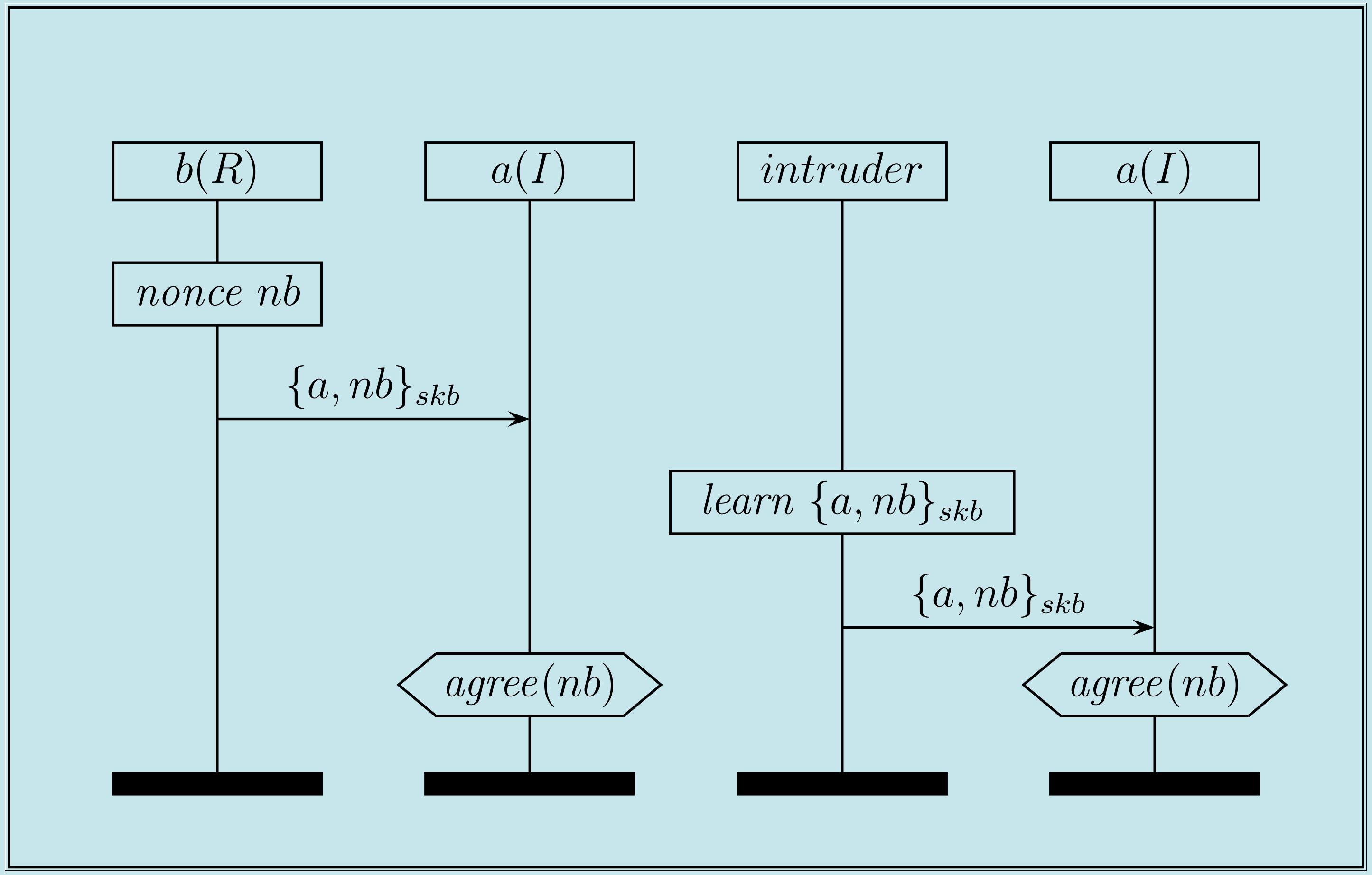


Example: unilateral authentication protocol



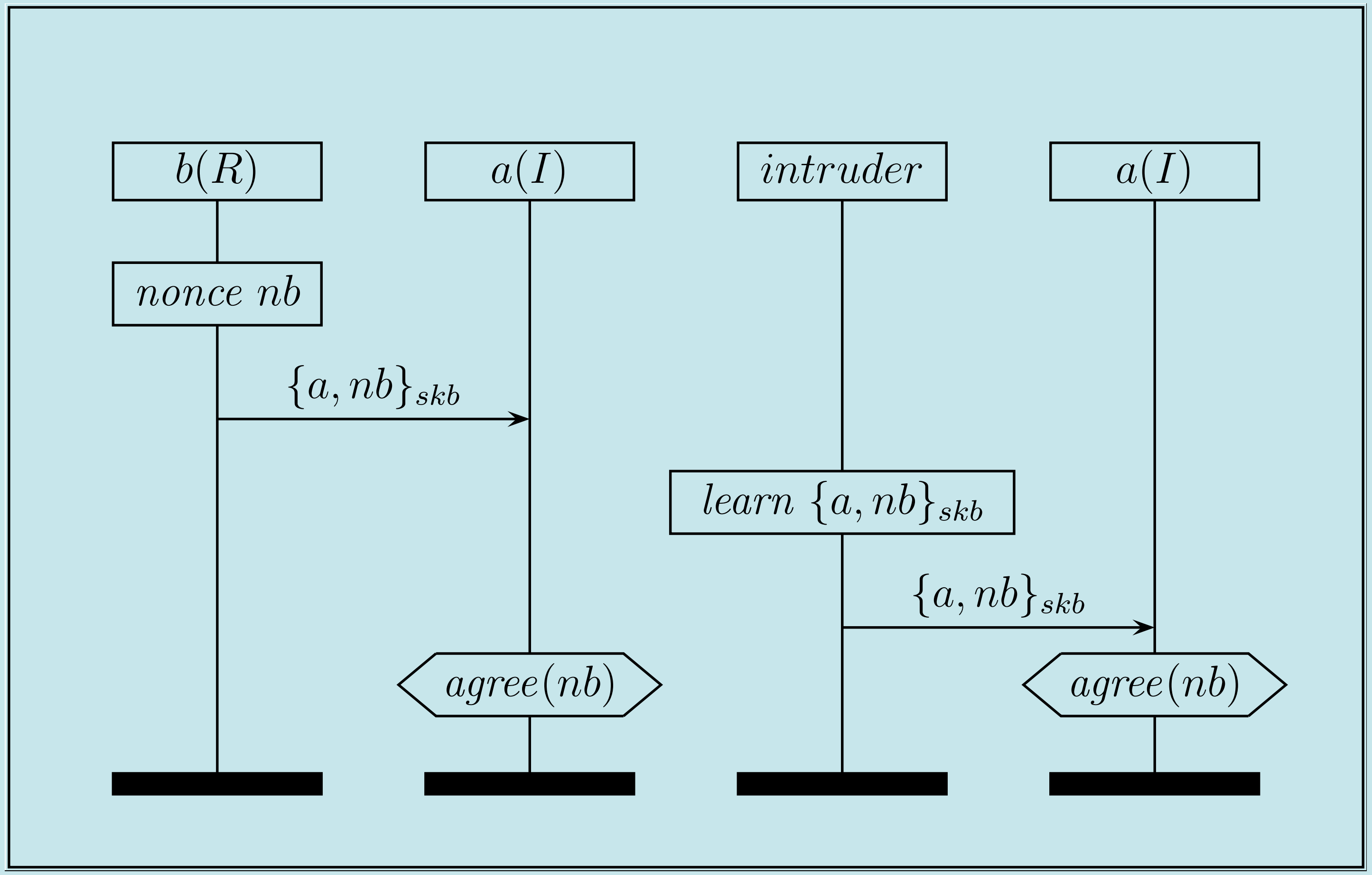
Question: Does this protocol satisfy agreement?

A replay attack



TU/e

A replay attack

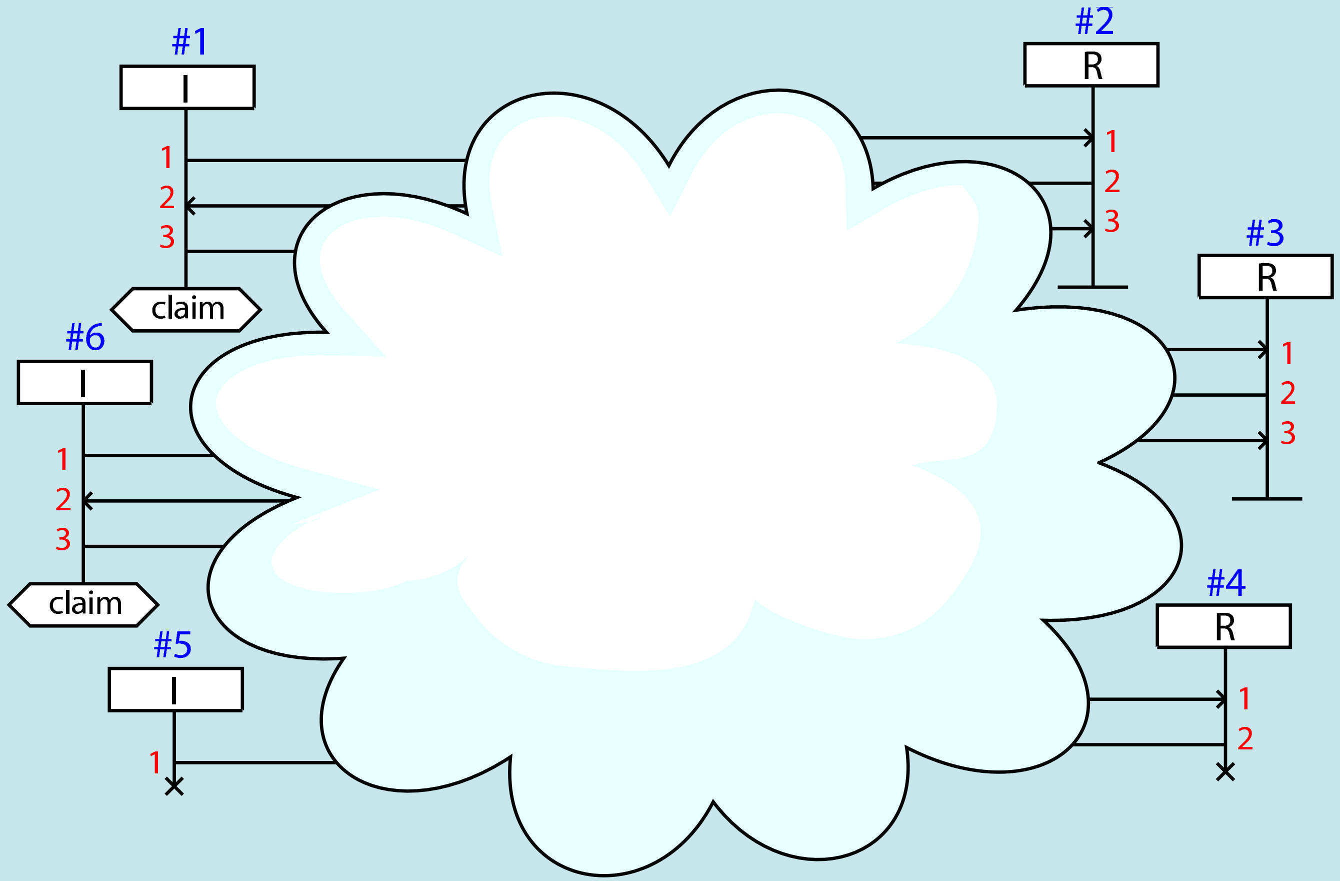


Question: How to fix this protocol?

Fixed protocol should satisfy *injectivity*

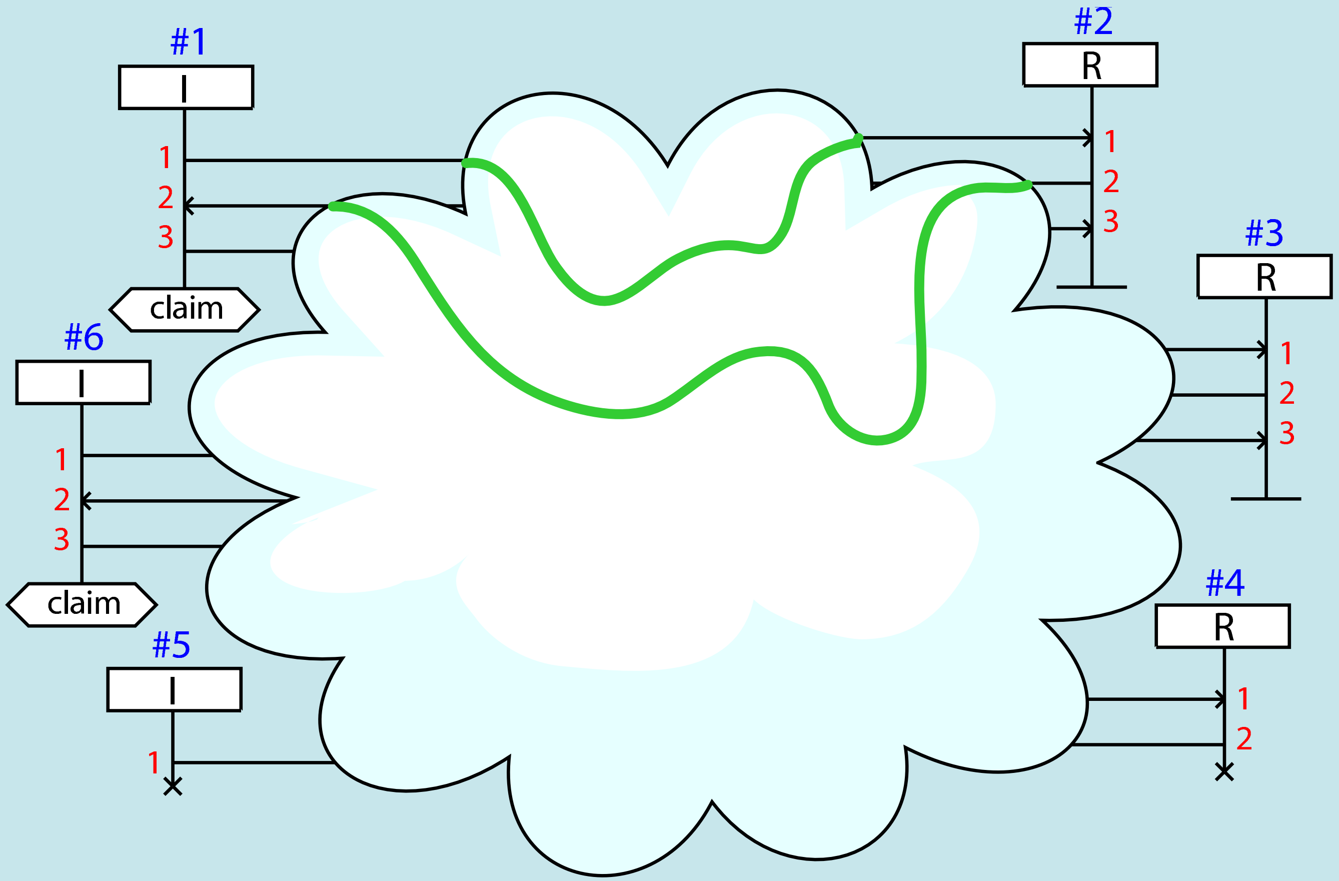
Each instance of an agent executing the authenticating role corresponds to a *unique* instance of its communication partner running the responder role.

Non-injective authentication

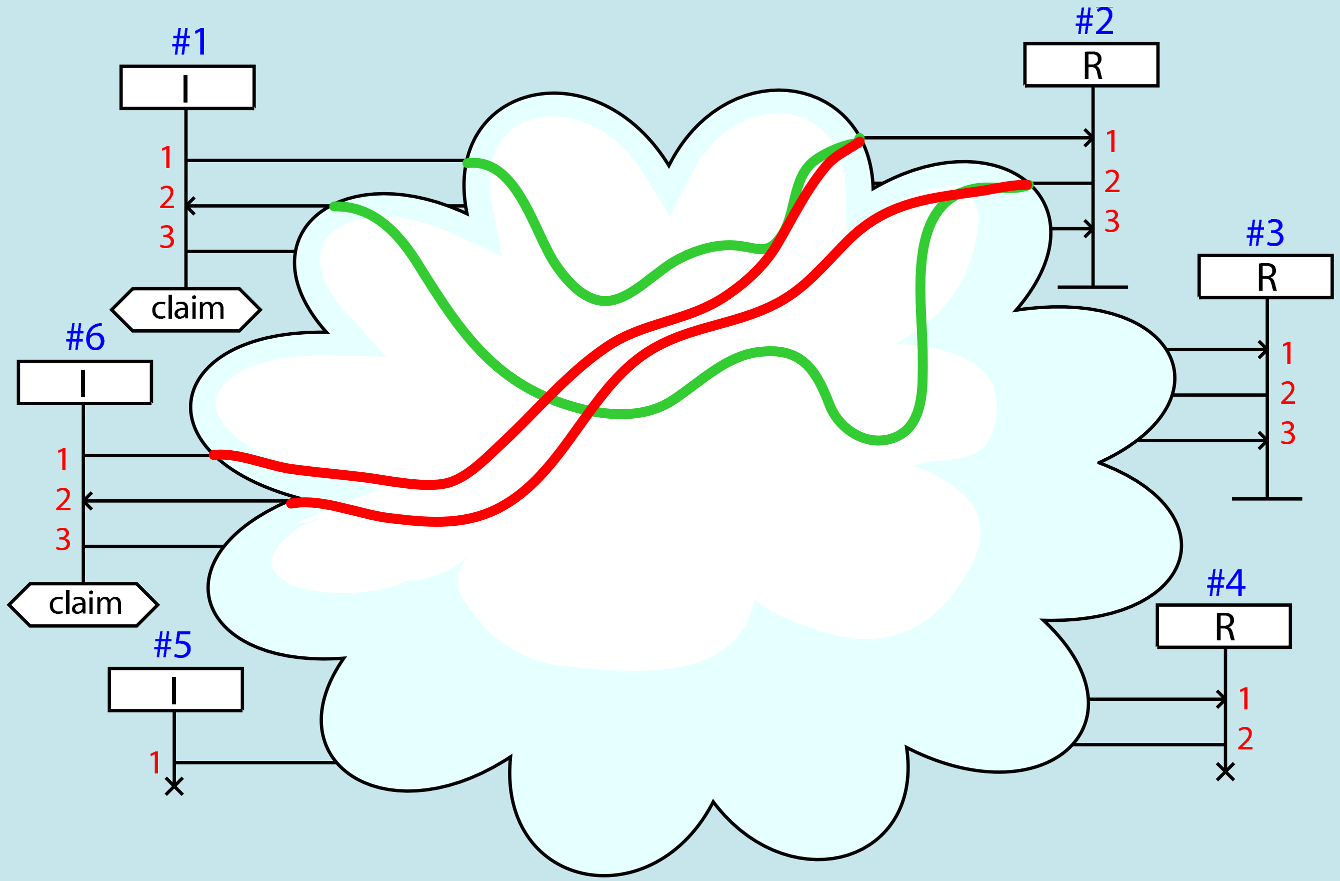


TU/e

Non-injective authentication

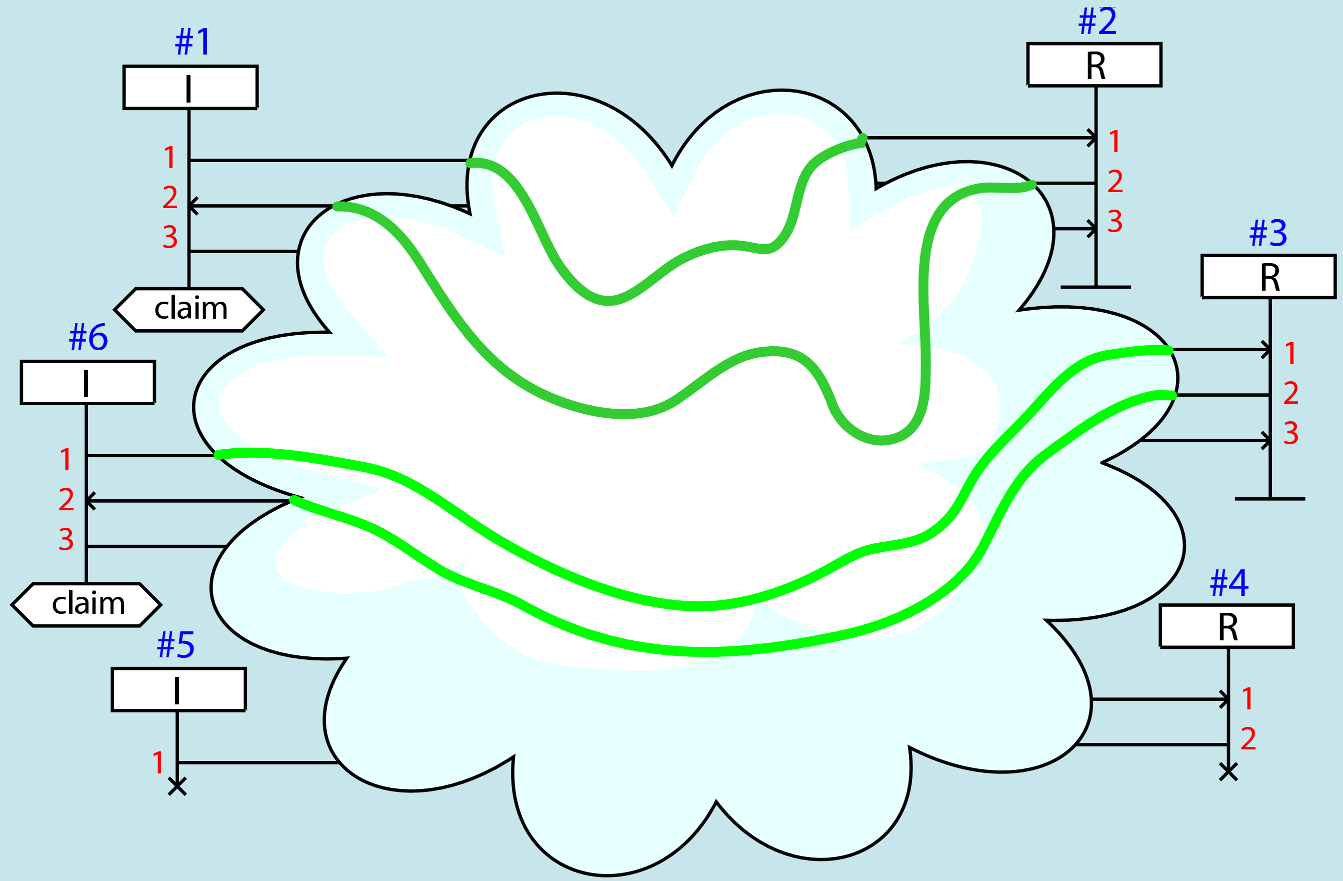


Injective authentication

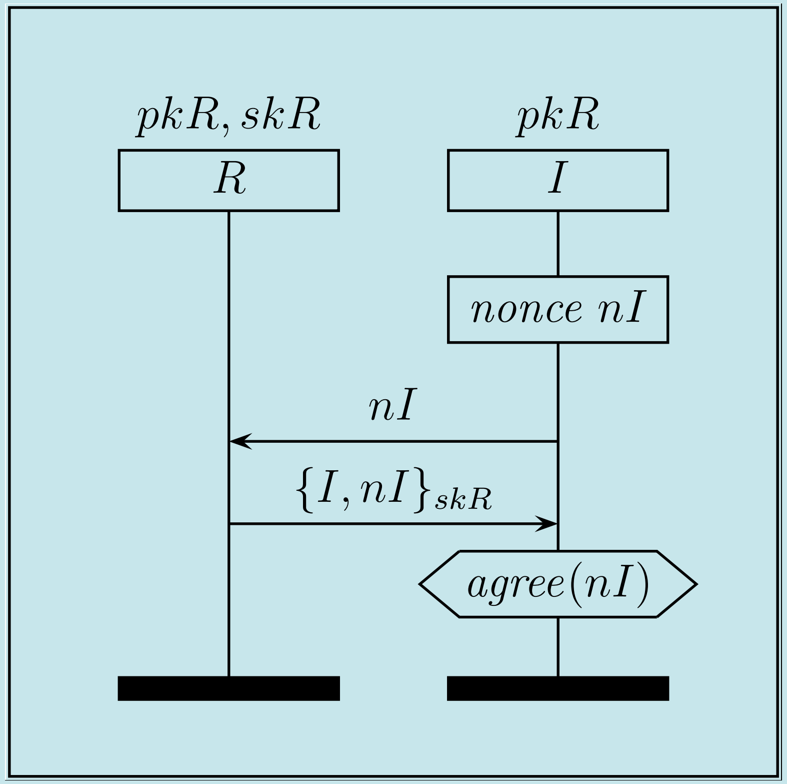


TU/e

Injective authentication

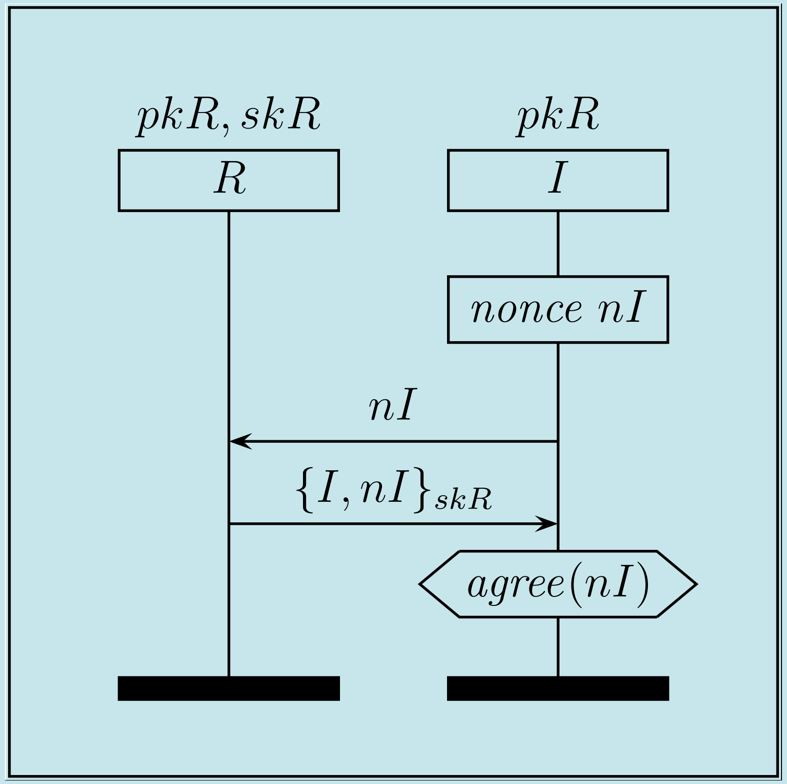


Fixing the injectivity problem



TU/e

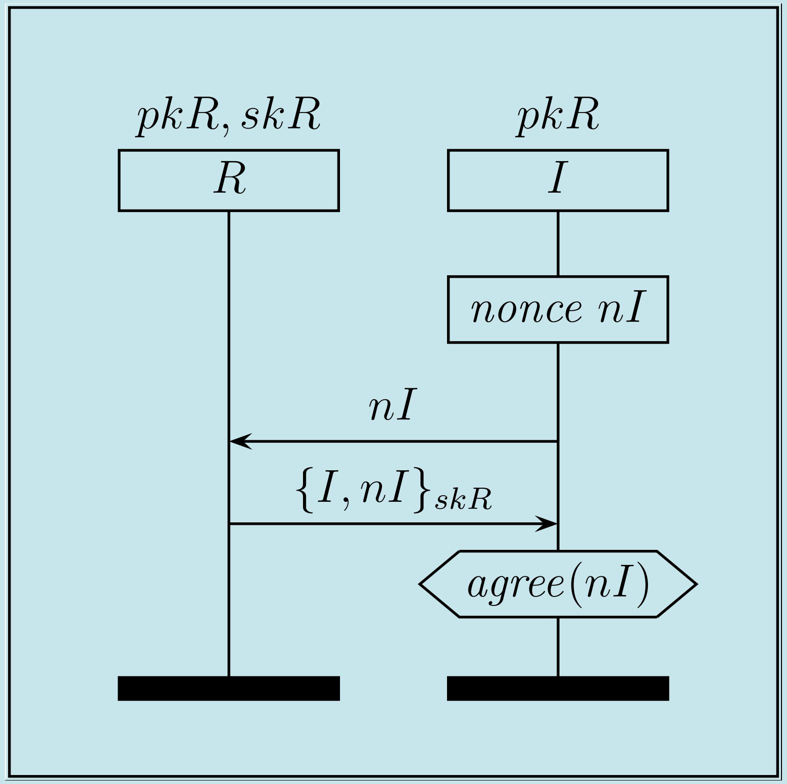
Fixing the injectivity problem



Question: What's the general idea behind this fix?

TU/e

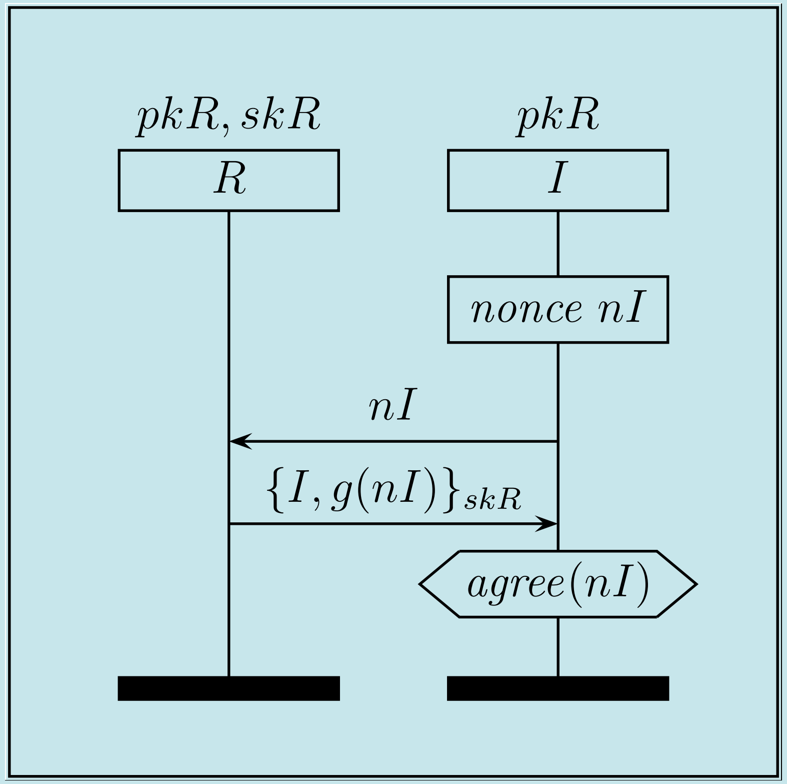
Fixing the injectivity problem



Question: What's the general idea behind this fix?
Answer 1: By letting I control the nonce.
Answer 2: By introducing a challenge-response mechanism from I via R back to I. (add a loop)

Doesn't a nonce suffice?

Adding nonces does not trivially lead to injectivity.

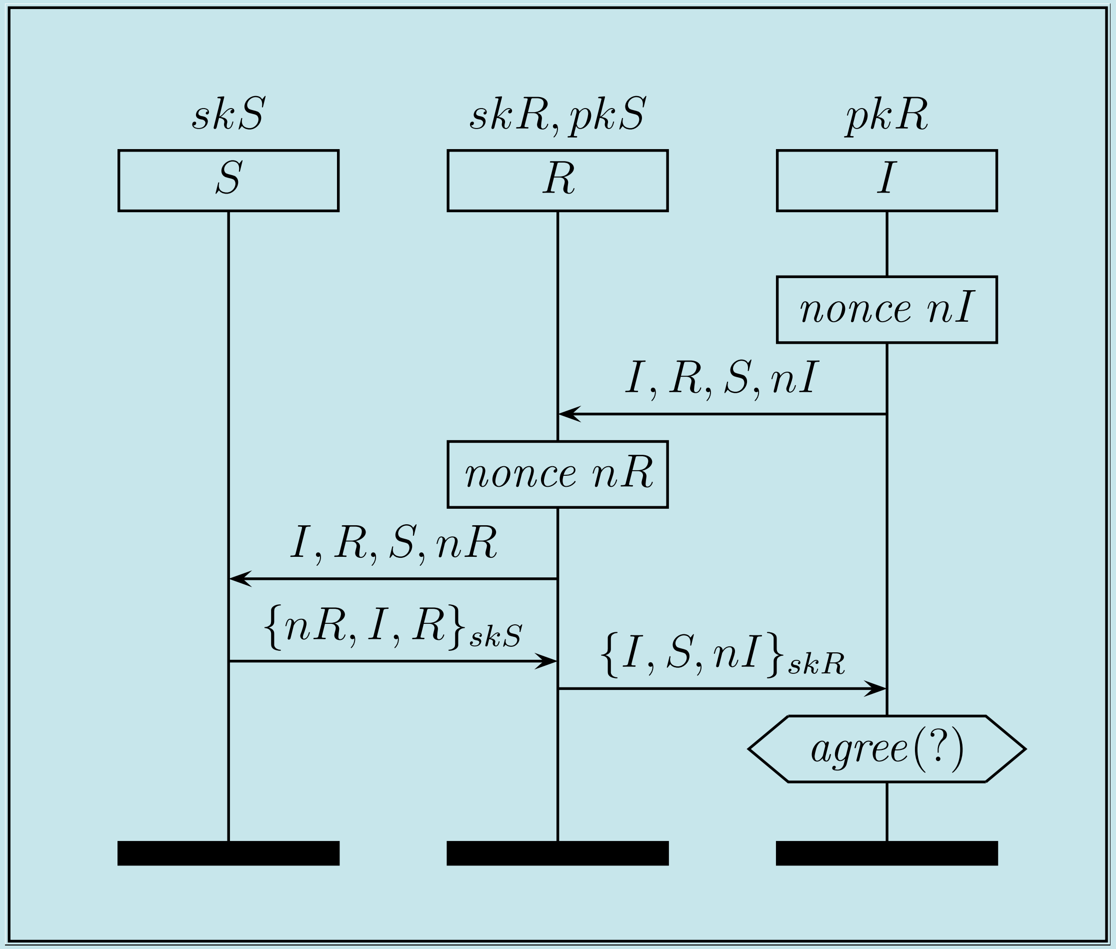


Here, injectivity depends on the properties of the function g.

TU/e

Agreement over what?

Sometimes roles have no shared value to determine injectivity from (I and S?)



TU/e

Authentication

Agreement

Upon successfully finishing a protocol session, parties agree on the values of (common) variables.
(G. Lowe)

Synchronization

Upon successfully finishing a protocol session, all messages have been executed in intended order, with intended contents.
(Similar to Intensional Specifications, A.W. Roscoe)

Synchronization is strictly stronger than agreement, but the differences are subtle.

Both available in injective ($i\text{-}synch$, $i\text{-}agree$) and non-injective ($ni\text{-}synch$) variants.

Claim: well-designed protocols satisfy both properties.

TU/e

Problem statement

Find a *generic* and *easy* way to validate injectivity for synchronizing protocols.

Generic:
As few assumptions on the security model as possible.

Easy:
Statically decidable.

Models for Security Protocols

We require that the following two properties hold:

Intruder Model:

Intruder must have the ability to duplicate messages

- Satisfied by the standard Dolev-Yao model.
- No need to encrypt/decrypt.

Agent/Execution Model:

TU/e

Models for Security Protocols

We require that the following two properties hold:

Intruder Model:

Intruder must have the ability to duplicate messages

- Satisfied by the standard Dolev-Yao model.
- No need to encrypt/decrypt.

Agent/Execution Model:

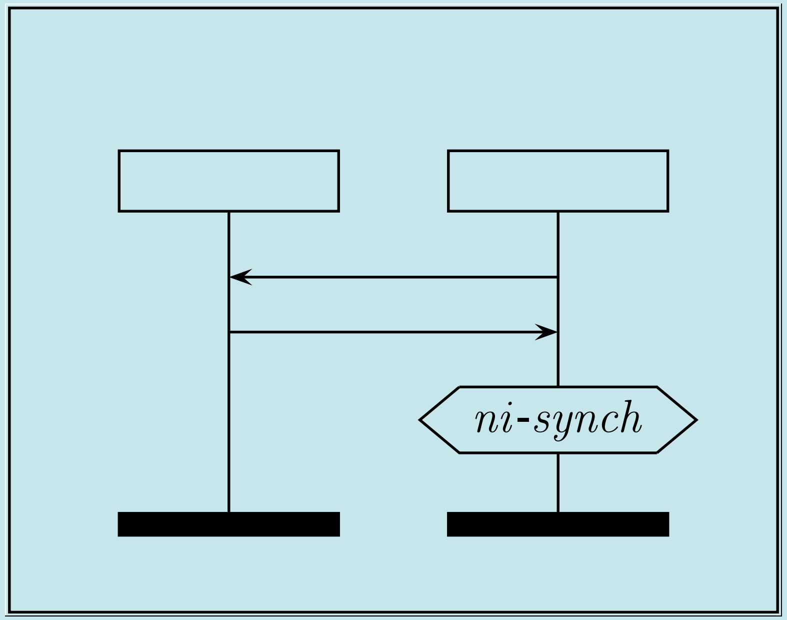
Role instances must be independent: can be executed in any order

- Satisfied by Strand Spaces, Operational Semantics.
- No shared memory. (buffers/time)

The $LOOP$ property

After the start of the authenticating role, but before it ends, each involved role must have a read action and a send action.

(As prescribed by the partial order on the protocol)

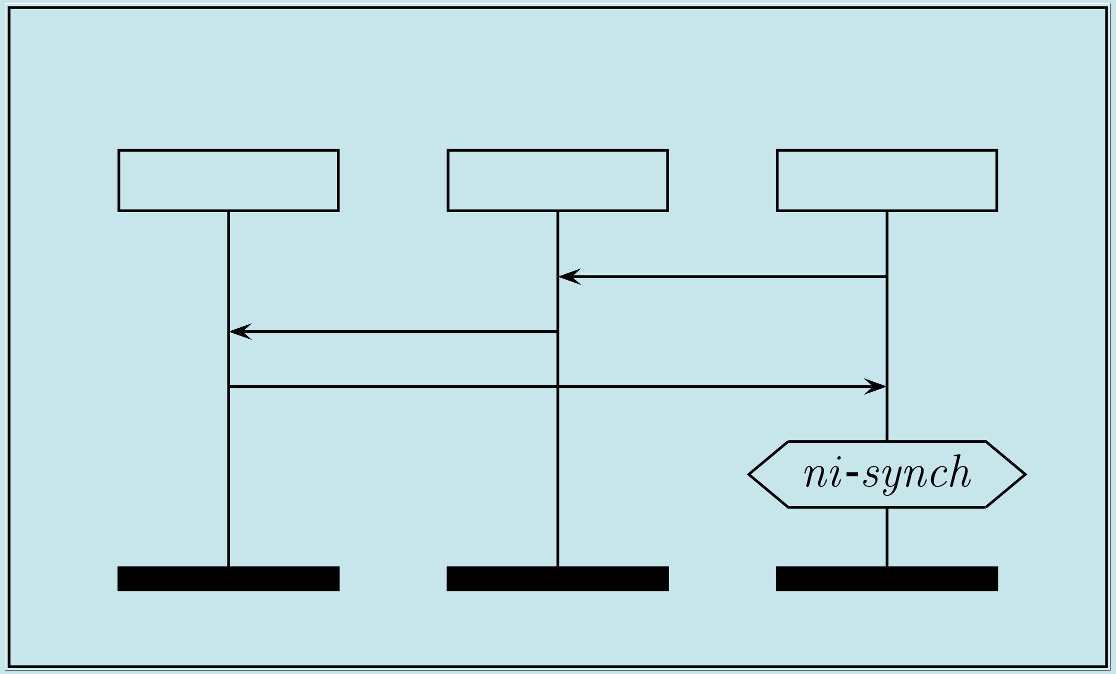


This protocol satisfies $LOOP$

The $LOOP$ property

After the start of the authenticating role, but before it ends, each involved role must have a read action and a send action.

(As prescribed by the partial order on the protocol)



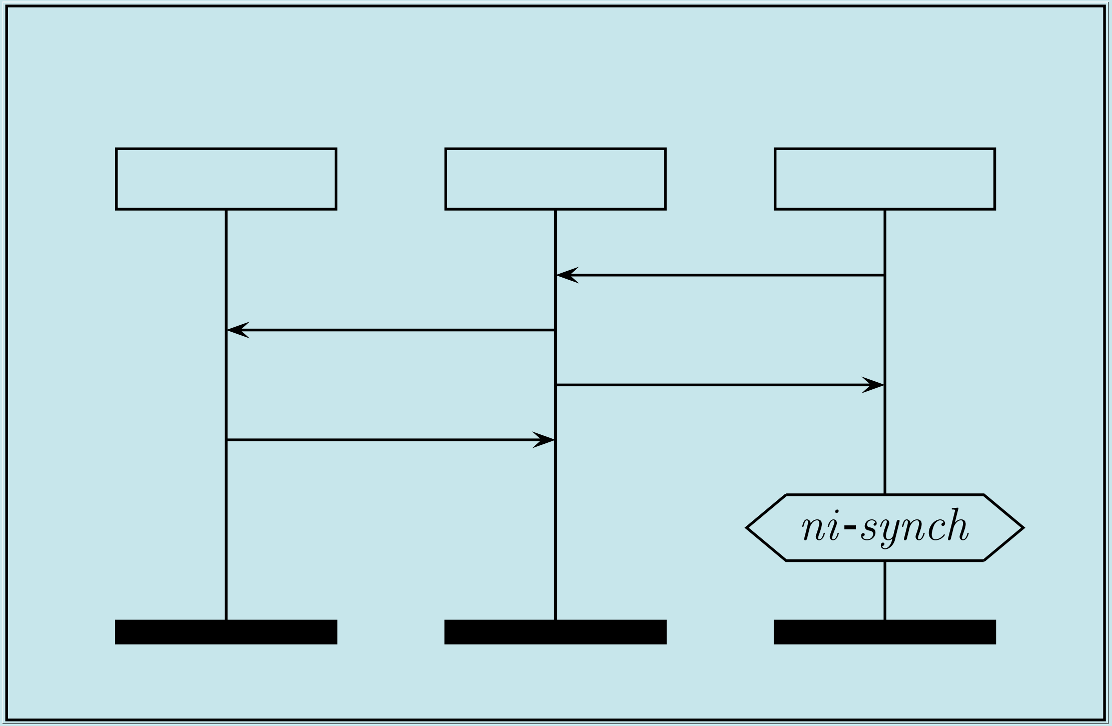
This protocol satisfies $LOOP$

TU/e

The $LOOP$ property

After the start of the authenticating role, but before it ends, each involved role must have a read action and a send action.

(As prescribed by the partial order on the protocol)



This protocol does *not* satisfy $LOOP$

TU/e

Main theorem

Preconditions:

- duplicating intruder
- independent role instances

$$ni\text{-}synch \wedge LOOP \Rightarrow i\text{-}synch$$

So, for synchronizing protocols, injectivity follows from the $LOOP$ property.

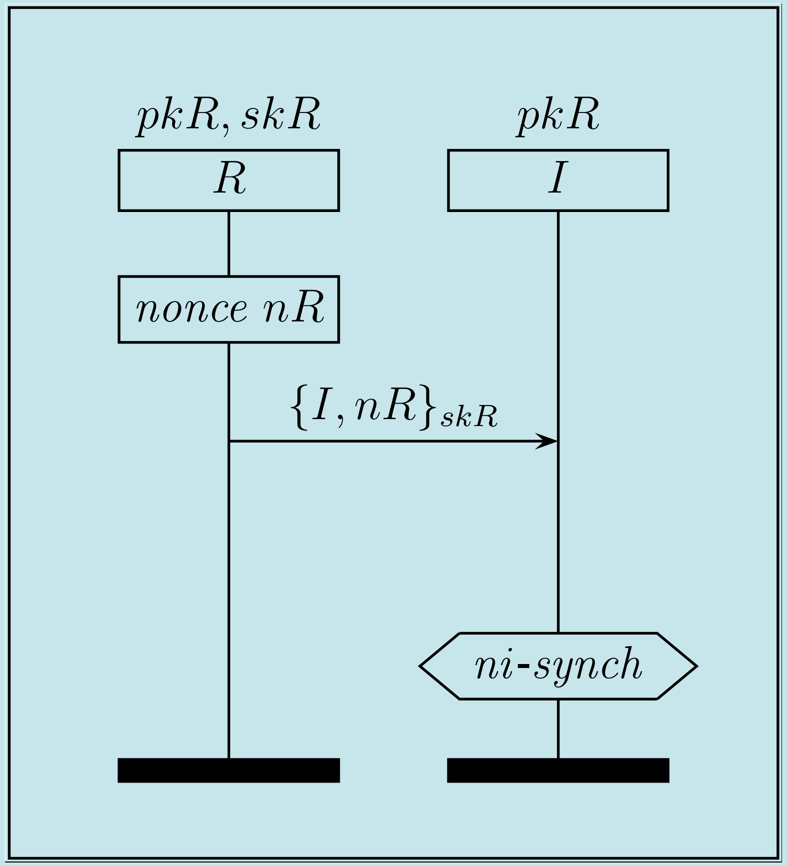
No reference is made to the data model (operators, etc.) or the contents of the messages (e.g. nonces)

TU/e

Do we need a loop?

Given a duplicating intruder and independent role instances:

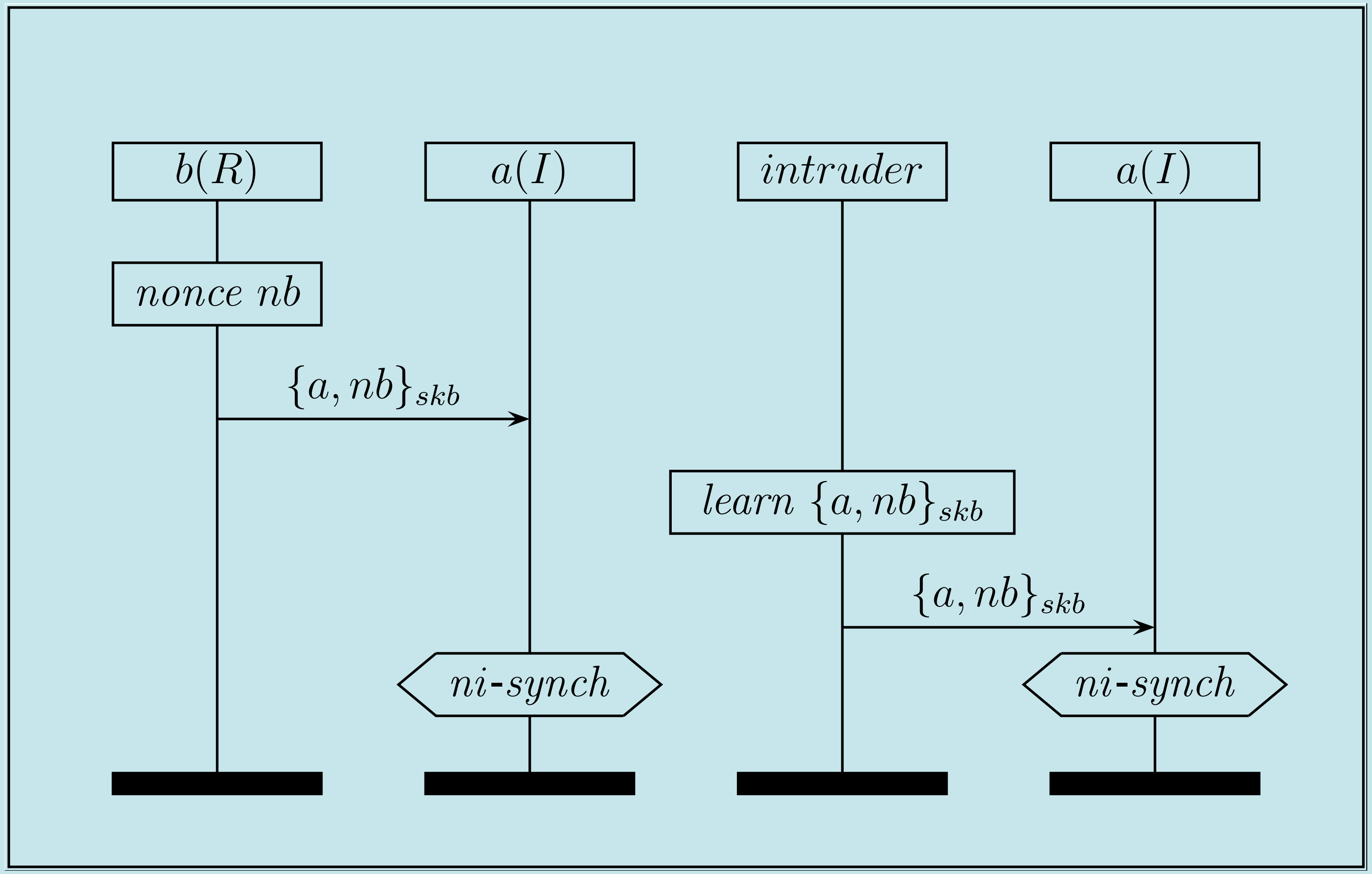
$$\textit{ni-synch} \Rightarrow \textit{i-synch}?$$



Do we need a loop?

Given a duplicating intruder and independent role instances:

$$\textit{ni-synch} \Rightarrow \textit{i-synch?}$$

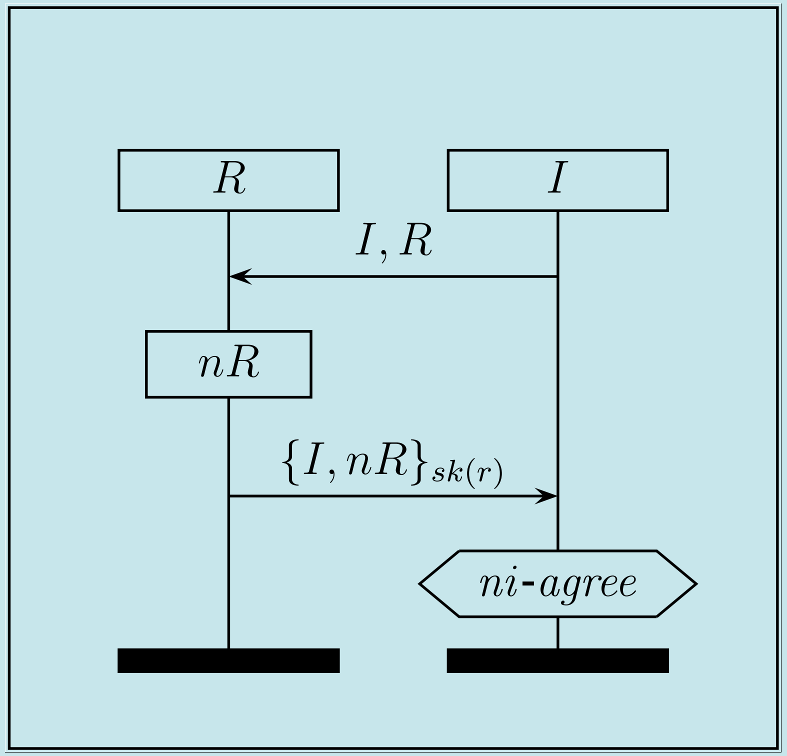


TU/e

Do we need Synchronization?

Given a duplicating intruder and independent role instances:

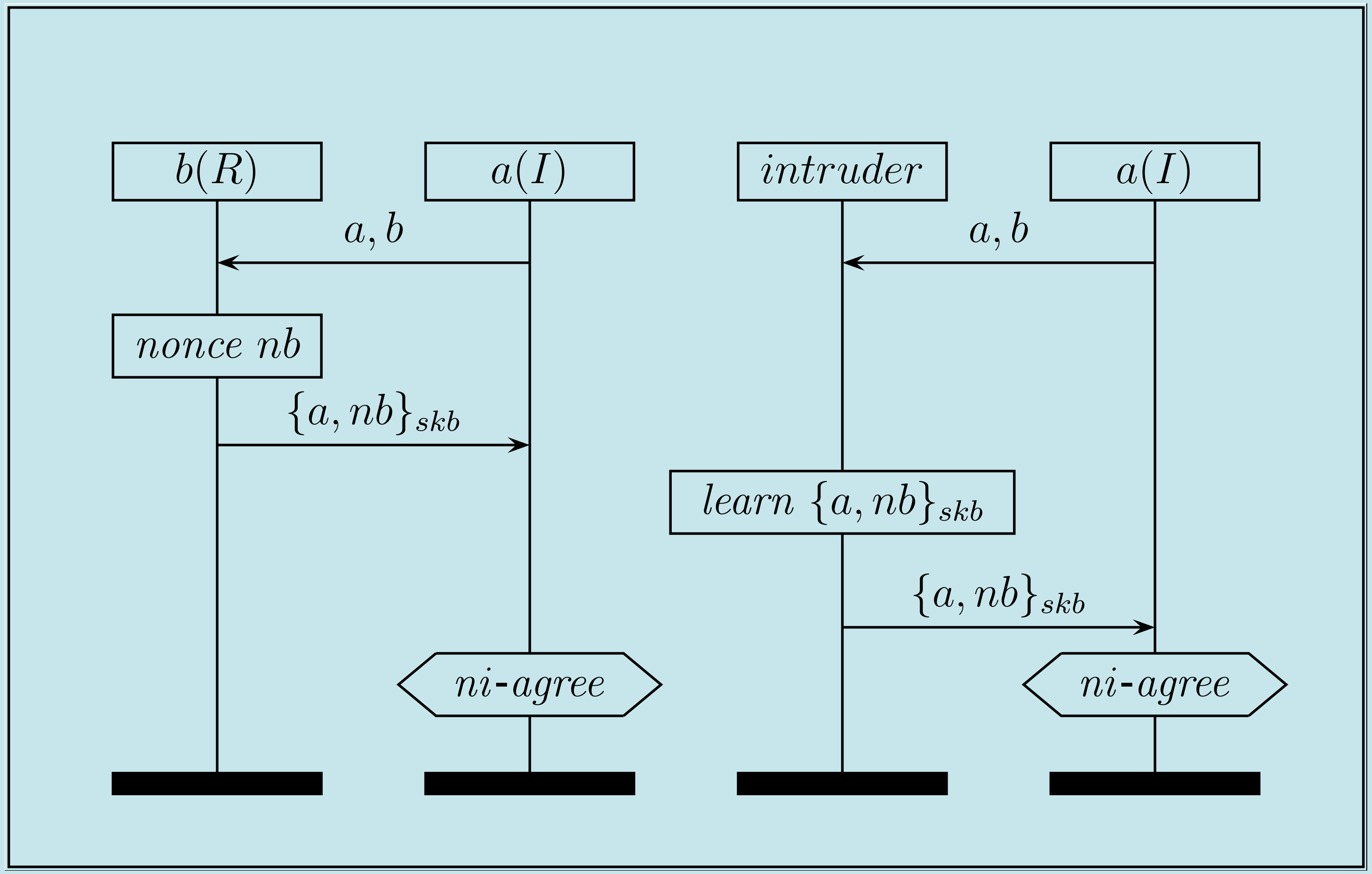
$$ni\text{-}agree \wedge LOOP \Rightarrow i\text{-}agree?$$



Do we need Synchronization?

Given a duplicating intruder and independent role instances:

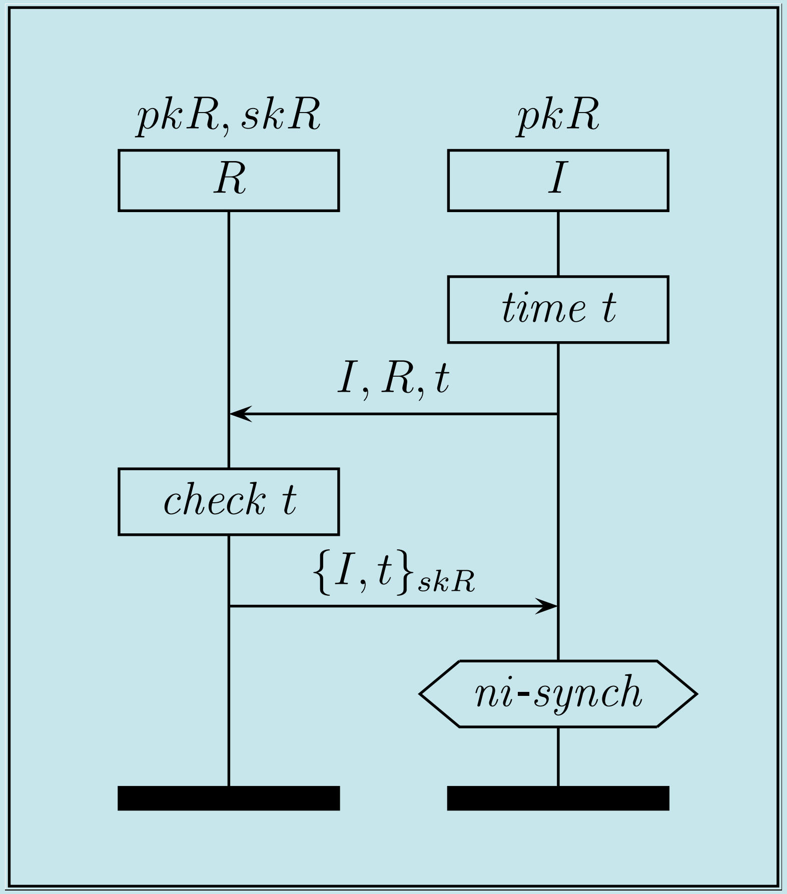
$$ni\text{-}agree \wedge LOOP \Rightarrow i\text{-}agree?$$



Do we need independent role instances?

Given a duplicating intruder:

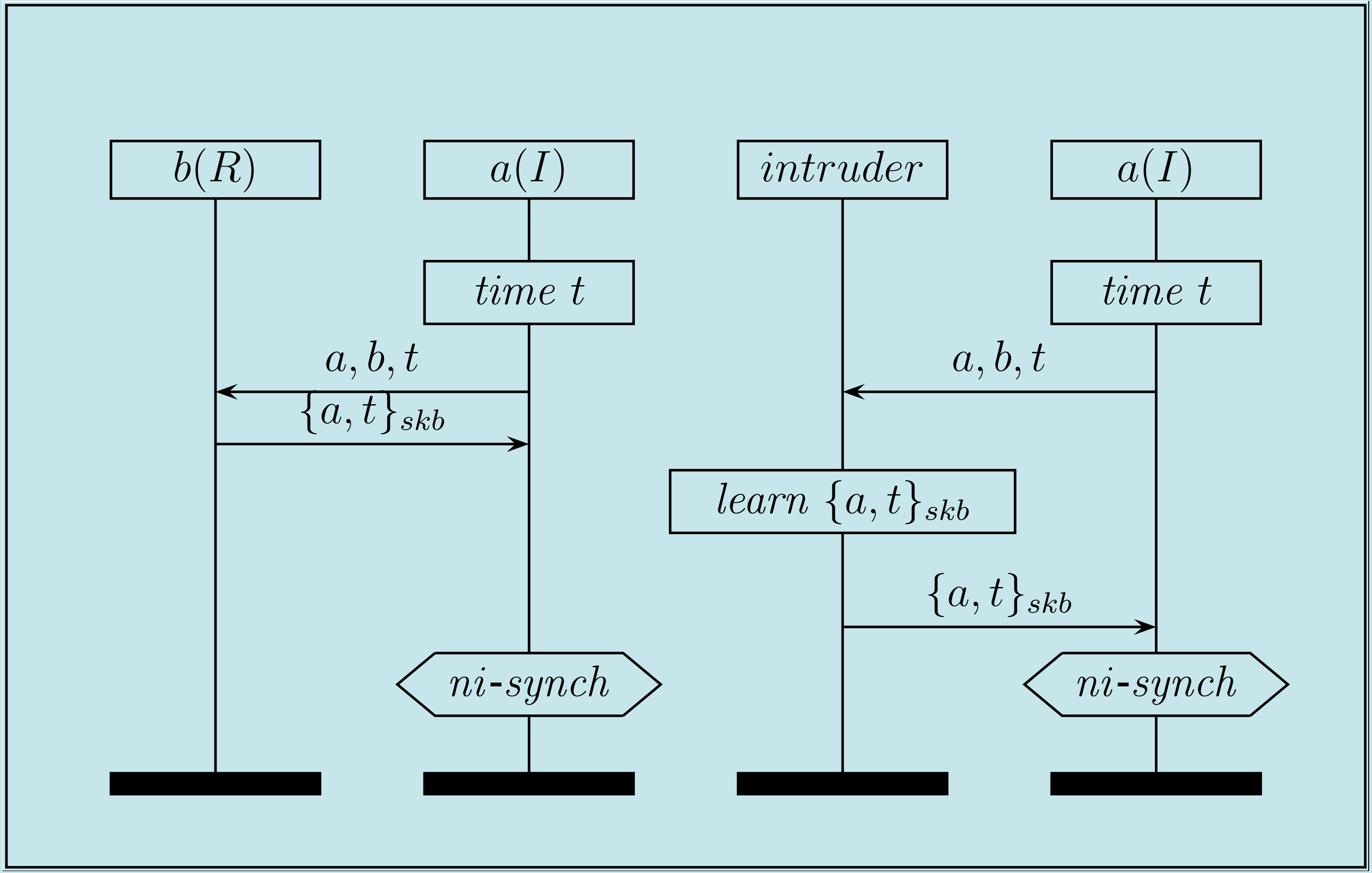
$$ni\text{-}synch \wedge LOOP \Rightarrow i\text{-}synch?$$



Do we need independent role instances?

Given a duplicating intruder:

$$\mathit{ni\text{-}synch} \wedge \mathit{LOOP} \Rightarrow \mathit{i\text{-}synch}?$$



Conclusions

Given a duplicating intruder and independent role instances:

$$\mathit{ni\text{-}synch} \wedge \mathit{LOOP} \Rightarrow \mathit{i\text{-}synch}$$

- LOOP-property can be checked easily.
- Generic: Sufficient condition for large class of security protocol semantics.
- LOOP plus agreement not sufficient to imply injective agreement.
  Extra structure in synchronization is helpful.
- Generalizes easily to multi-party protocols with multiple claims.

TU/e

Future Work

- Challenge: Is there a similar condition for agreement?
  - statically checkable
  - generic
- Use in model checker/theorem prover.
- Analyze other security properties for statically decidable subproperties.

Thanks for your attention

Any Questions?

E-mail: `ccremers@win.tue.nl`

How to verify injectivity?

model-checking approach

Counting: $\sharp(\text{I-runs}) \leq \sharp(\text{corresponding R-runs})$

other approaches (logics, term rewriting)

- Strand spaces: solicited authentication tests (Guttman, Theyer 2002)
- π-calculus: injective correspondence (Gordon, Jeffrey 2002)
- Logic: e-commerce protocol logic (Adi, Debbabi, Mejri 2003)
- Further: Ad-hoc reasoning, informal reasoning, or simply not.

The $LOOP$ property

For all $e \prec_p claim$, such that $role(e) \neq role(claim)$ there exist e' and e'' such that

$$\begin{aligned}
&e' \prec_p e'' \prec_p claim \land \\
&role(e') = role(claim) \land \\
&role(e'') = role(e)
\end{aligned}$$

This property can be easily verified on the syntactic description of the protocol.