Topics in Concurrency

Lecture 12

Jonathan Hayman

10 February 2015

An attack against the original protocol

Man-in-the-middle attacker E convinces A to start communication with E and uses the messages generated by A to follow the protocol with B, posing as A.

A	E	B

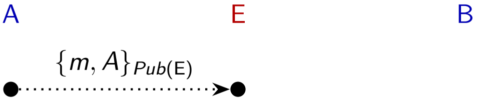
$\mathsf{A} \longrightarrow \mathsf{B} : \ \{m, \mathsf{A}\}_{Pub(B)}$

$\mathsf{B} \longrightarrow \mathsf{A} : \ \{m, n\}_{Pub(A)}$

$\mathsf{A} \longrightarrow \mathsf{B} : \ \{n\}_{Pub(B)}$

An attack against the original protocol

Man-in-the-middle attacker E convinces A to start communication with E and uses the messages generated by A to follow the protocol with B, posing as A.



$\mathrm{A} \longrightarrow \mathrm{B} : \ \{m, \mathrm{A}\}_{Pub(B)}$

$\mathrm{B} \longrightarrow \mathrm{A} : \ \{m, n\}_{Pub(A)}$

$\mathrm{A} \longrightarrow \mathrm{B} : \ \{n\}_{Pub(B)}$

An attack against the original protocol

Man-in-the-middle attacker E convinces A to start communication with E and uses the messages generated by A to follow the protocol with B, posing as A.

$$\mathsf{A} \longrightarrow \mathsf{B} : \ \{m, \mathsf{A}\}_{Pub(B)}$$

$$\mathsf{A} \xrightarrow{\{m, A\}_{Pub(\mathsf{E})}} \mathsf{E} \xrightarrow{\{m, A\}_{Pub(\mathsf{B})}} \mathsf{B}$$

$$\mathsf{B} \longrightarrow \mathsf{A} : \ \{m, n\}_{Pub(A)}$$

$$\mathsf{A} \longrightarrow \mathsf{B} : \ \{n\}_{Pub(B)}$$

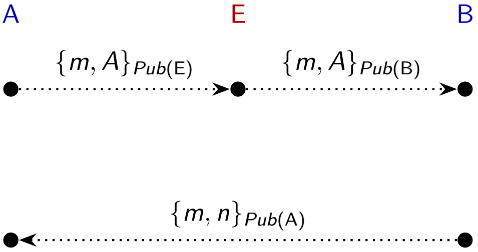
An attack against the original protocol

Man-in-the-middle attacker E *convinces* A *to start communication with* E *and uses the messages generated by* A *to follow the protocol with* B*, posing as* A*.*

$\mathsf{A} \longrightarrow \mathsf{B} : \{m, \mathsf{A}\}_{Pub(B)}$

$\mathsf{B} \longrightarrow \mathsf{A} : \{m, n\}_{Pub(A)}$

$\mathsf{A} \longrightarrow \mathsf{B} : \{n\}_{Pub(B)}$



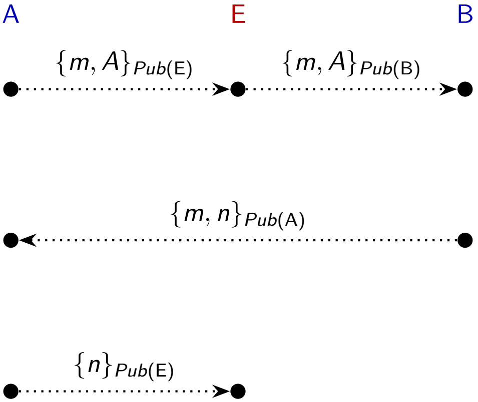
An attack against the original protocol

Man-in-the-middle attacker E *convinces* A *to start communication with* E *and uses the messages generated by* A *to follow the protocol with* B*, posing as* A*.*

$\mathsf{A} \longrightarrow \mathsf{B} : \{m, \mathsf{A}\}_{Pub(B)}$

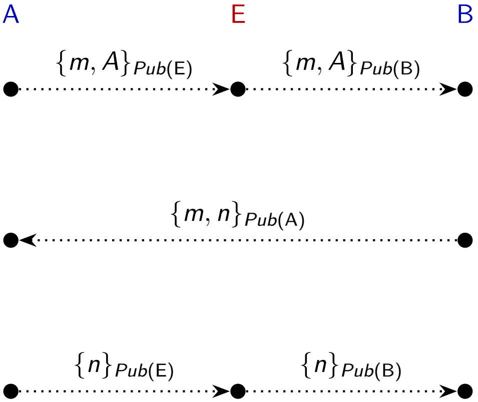
$\mathsf{B} \longrightarrow \mathsf{A} : \{m, n\}_{Pub(A)}$

$\mathsf{A} \longrightarrow \mathsf{B} : \{n\}_{Pub(B)}$



An attack against the original protocol

Man-in-the-middle attacker E convinces A to start communication with E and uses the messages generated by A to follow the protocol with B, posing as A.



$\mathrm{A} \longrightarrow \mathrm{B} : \{m, \mathrm{A}\}_{Pub(B)}$

$\mathrm{B} \longrightarrow \mathrm{A} : \{m, n\}_{Pub(A)}$

$\mathrm{A} \longrightarrow \mathrm{B} : \{n\}_{Pub(B)}$

The fixed protocol

$$\begin{array}{lll} (1) & \mathsf{A} \longrightarrow \mathsf{B}: & \{m, \mathsf{A}\}_{Pub(B)} \\ (2) & \mathsf{B} \longrightarrow \mathsf{A}: & \{m, n, \mathsf{B}\}_{Pub(A)} \\ (3) & \mathsf{A} \longrightarrow \mathsf{B}: & \{n\}_{Pub(B)} \end{array}$$

- Only B can decrypt the message sent in (1)
- A knows that only B can have sent the message in (2)
- B knows that only A can have sent the message in (1)
- the nonces m and n are shared secrets

But these properties are informal and approximate, and we've only described what's *supposed* to happen . . .

SPL

Security Protocol Language

- One of a range of languages and models for analyzing crypto-protocols
- Others include Spi calculus, strand spaces
- Supports reasoning based on events (vs transitions)

- Asynchronous communication
- Messages persist on network
- New-name generation on output
- Input pattern-matches messages on network

Messages as patterns

Messages can contain variables.

$$\psi \qquad n, x \qquad \{m, y, \mathsf{B}\}_{Pub(\mathsf{A})}$$

These are used to perform matching.

Examples:

- match $\{\mathsf{A}, \mathsf{B}\}_{Pub(A)}$ against the pattern ψ
- match $\{m, n, \mathsf{B}\}_{Pub(\mathsf{A})}$ against the pattern $\{m, x, Y\}_{Pub(\mathsf{A})}$
- match $m, (n, \mathsf{A})$ against the pattern n, x where $m \neq n$

Messages as patterns

Messages can contain variables.

$$\psi \qquad n, x \qquad \{m, y, \mathsf{B}\}_{Pub(\mathsf{A})}$$

These are used to perform matching.

Examples:

- match $\{\mathsf{A}, \mathsf{B}\}_{Pub(A)}$ against the pattern ψ $\qquad \psi \mapsto \{\mathsf{A}, \mathsf{B}\}_{Pub(A)}$
- match $\{m, n, \mathsf{B}\}_{Pub(\mathsf{A})}$ against the pattern $\{m, x, Y\}_{Pub(\mathsf{A})}$

- match $m, (n, \mathsf{A})$ against the pattern n, x where $m \neq n$

Messages as patterns

Messages can contain variables.

$$\psi \qquad n, x \qquad \{m, y, \mathsf{B}\}_{Pub(\mathsf{A})}$$

These are used to perform matching.

Examples:

- match $\{\mathsf{A}, \mathsf{B}\}_{Pub(A)}$ against the pattern ψ $\qquad \psi \mapsto \{\mathsf{A}, \mathsf{B}\}_{Pub(A)}$
- match $\{m, n, \mathsf{B}\}_{Pub(\mathsf{A})}$ against the pattern $\{m, x, Y\}_{Pub(\mathsf{A})}$
  $x \mapsto n, Y \mapsto \mathsf{B}$
- match $m, (n, \mathsf{A})$ against the pattern n, x where $m \neq n$

Messages as patterns

Messages can contain variables.

$$\psi \qquad n, x \qquad \{m, y, \mathsf{B}\}_{Pub(\mathsf{A})}$$

These are used to perform matching.

Examples:

- match $\{\mathsf{A}, \mathsf{B}\}_{Pub(A)}$ against the pattern ψ $\qquad \psi \mapsto \{\mathsf{A}, \mathsf{B}\}_{Pub(A)}$
- match $\{m, n, \mathsf{B}\}_{Pub(\mathsf{A})}$ against the pattern $\{m, x, Y\}_{Pub(\mathsf{A})}$
  $x \mapsto n, Y \mapsto \mathsf{B}$
- match $m, (n, \mathsf{A})$ against the pattern n, x where $m \neq n$ $\qquad$ no match

The NSL protocol in SPL

The initiator initiator of the protocol is parameterized by the identity of the initiator and their intended participant:

$$\begin{array}{rcl} \mathit{Init}(\mathrm{A},\mathrm{B}) & \equiv & \mathrm{out}\,\mathrm{new}\,x\ \ \{x,\mathrm{A}\}_{\mathit{Pub}(\mathrm{B})}. \\ & & \mathrm{in}\ \{x,y,B\}_{\mathit{Pub}(A)}. \\ & & \mathrm{out}\ \{y\}_{\mathit{Pub}(B)} \end{array}$$

The responder:

$$\begin{array}{rcl} \mathit{Resp}(\mathrm{B}) & \equiv & \mathrm{in}\ \{x,Z\}_{\mathit{Pub}(\mathrm{B})}. \\ & & \mathrm{out}\,\mathrm{new}\,y\ \ \{x,y,\mathrm{B}\}_{\mathit{Pub}(Z)}. \\ & & \mathrm{in}\ \{y\}_{\mathit{Pub}(\mathrm{B})} \end{array}$$

Dolev-Yao assumptions

We can program various forms of attacker process. Viewing messages as persisting once output to the network, they output new messages built from existing ones.

$$
\begin{array}{rcl}
\mathit{Spy}_1 & \equiv & \mathsf{in}\ \psi_1.\mathsf{in}\ \psi_2.\,\mathsf{out}\,(\psi_1, \psi_2) \\
\mathit{Spy}_2 & \equiv & \mathsf{in}\ (\psi_1, \psi_2).\,\mathsf{out}\,\psi_1.\,\mathsf{out}\,\psi_2 \\
\mathit{Spy}_3 & \equiv & \mathsf{in}\ X.\mathsf{in}\ \psi.\,\mathsf{out}\,\{\psi\}_{\mathit{Pub}(X)} \\
\mathit{Spy}_4 & \equiv & \mathsf{in}\ \mathit{Priv}(X).\mathsf{in}\ \{\psi\}_{\mathit{Pub}(X)}.\,\mathsf{out}\,\psi
\end{array}
$$

$$
\mathit{Spy} \equiv \|_{i \in \{1,2,3,4\}}\ \mathit{Spy}_i
$$

The NSL system [p91]

We reason about concurrent runs of the protocol in parallel with ω-copies of the attacker.

$$\begin{array}{rcl} P_{spy} & \equiv & !Spy \\ P_{init} & \equiv & \displaystyle\coprod_{\mathrm{A},\mathrm{B} \in \mathbf{Agents}} !Init(\mathrm{A}, \mathrm{B}) \\ P_{resp} & \equiv & \displaystyle\coprod_{\mathrm{A} \in \mathbf{Agents}} !Resp(\mathrm{A}) \end{array}$$

Messages from one run of the protocol can be used by the attacker against another run of the protocol.

$$NSL \equiv \coprod_{i \in \{resp, init, spy\}} P_i$$

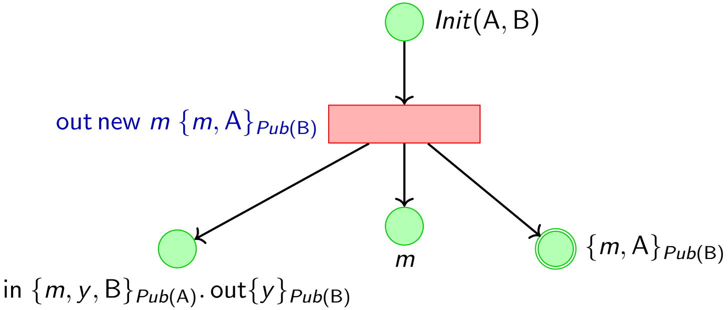
Net semantics of SPL

- Details won't be given, but a semantics along the lines of the basic net semantics for CCS can be given for the language used to represent processes
- Nets formed with events representing the possible behaviour of processes
- Three forms of condition: control, state and name.

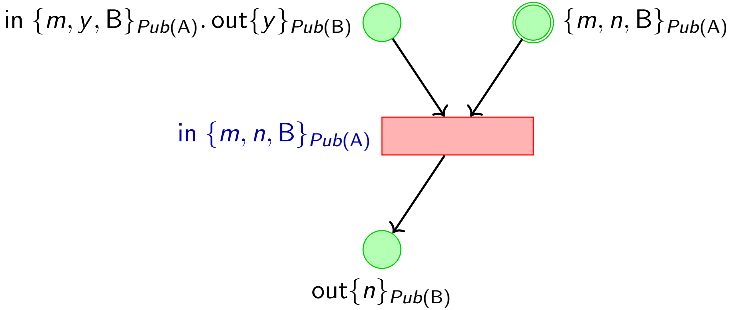
The events of NSL [p100]: Initiator events

(Omitting tags!)

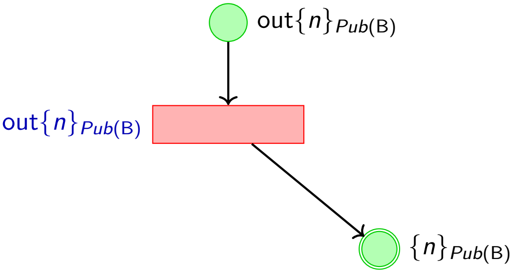
$\textbf{Out}(\mathit{Init}(\mathsf{A},\mathsf{B}); m)$



In(in $\{m, y, \mathsf{B}\}_{Pub(\mathsf{A})}$. out$\{y\}_{Pub(\mathsf{B})}$

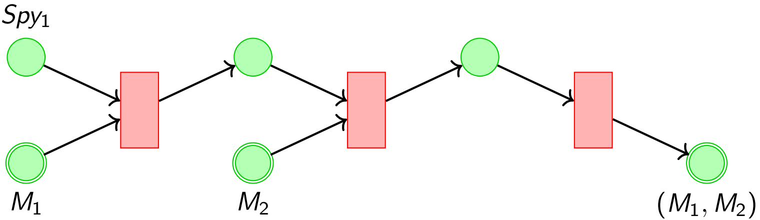


$\mathbf{Out}(\mathrm{out}\{n\}_{Pub(\mathrm{B})})$

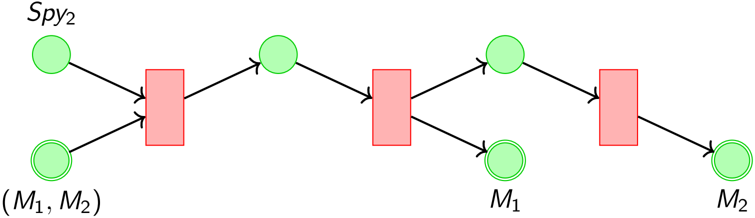


The events of NSL [p101]: Attacker events

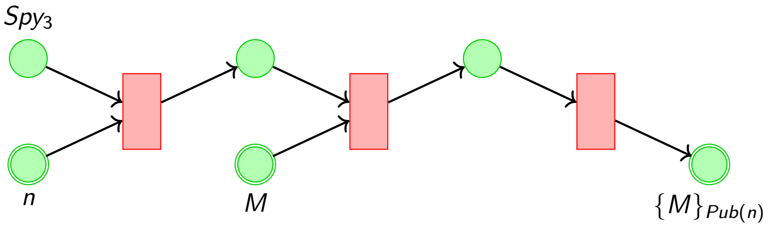
$$Spy_1 \equiv \mathsf{in}\ \psi_1.\mathsf{in}\ \psi_2.\,\mathsf{out}\,(\psi_1, \psi_2)$$



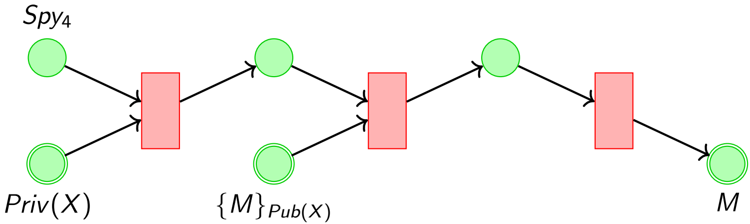
$$Spy_2 \equiv \mathsf{in}\ (\psi_1, \psi_2).\ \mathsf{out}\, \psi_1.\ \mathsf{out}\, \psi_2$$



$$Spy_3 \equiv \mathsf{in}\ X.\mathsf{in}\ \psi.\ \mathsf{out}\ \{\psi\}_{Pub(X)}$$



$$Spy_4 \equiv \mathsf{in}\ Priv(X).\mathsf{in}\ \{\psi\}_{Pub(X)}.\,\mathsf{out}\,\psi$$



Secrecy of responder's nonce [p104]

Theorem

Consider a run

$$\langle NSL, s_0, t_0\rangle \xrightarrow{e_1} \cdots \xrightarrow{e_r} \langle p_r, s_r, t_r\rangle \xrightarrow{e_{r+1}} \cdots .$$

Suppose there is e_r with

$$act(e_r) = resp : \mathrm{B}_0 : j_0 : \mathrm{out\,new}\ n_0\ \{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)}$$

where j_0 is an index. If $Priv(\mathrm{A}_0) \not\sqsubset t_0$ and $Priv(\mathrm{B}_0) \not\sqsubset t_0$ then at all stages $n_0 \notin t_l$.

Prove a stronger invariant: For any stage l

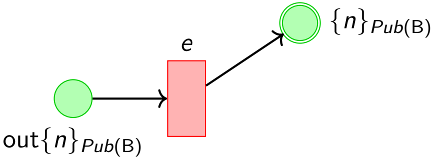
> for all messages $M \in t_l$, if $n_0 \sqsubset M$ then either $\{m_0, n_0, \mathsf{B}_0\}_{Pub(\mathsf{A}_0)} \sqsubset M$ or $\{n_0\}_{Pub(\mathsf{B}_0)} \sqsubset M$.

Prove a stronger invariant: For any stage l

> for all messages $M \in t_l$, if $n_0 \sqsubset M$ then either $\{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)} \sqsubset M$ or $\{n_0\}_{Pub(\mathrm{B}_0)} \sqsubset M$.

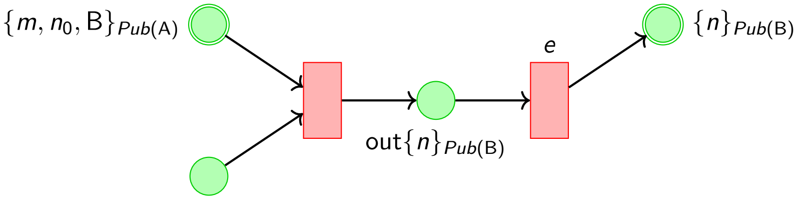
- We have $Fresh(e_r, n)$ and therefore, by freshness, the initial configuration satisfies the invariant
- Suppose for contradiction that there is a configuration that violates the invariant. By well-foundedness, there is an earliest such configuration
- Consider the event e that causes the violation: $\exists M \in e^{\bullet}$ satisfying $n_0 \sqsubset M$ but neither $\{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)} \sqsubset M$ nor $\{n_0\}_{Pub(\mathrm{B}_0)}$
- e must be the earliest event with such a postcondition
- Consider the possible forms of e in NSL: cannot be indexed input

Case: $e = init : (A, B) : i : \mathbf{Out}(\mathrm{out}\{n\}_{Pub(B)})$ for some index i and pair of agents A, B.



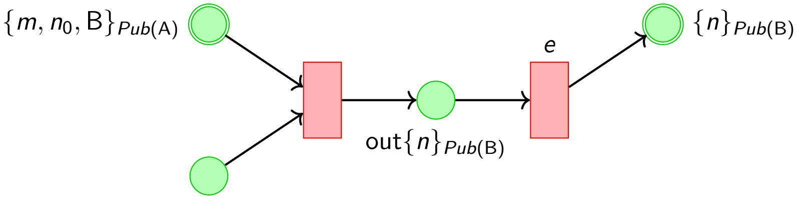
Event violates invariant, so $n = n_0$ and $B \neq B_0$

Case: $e = init : (A, B) : i : \mathbf{Out}(\text{out}\{n\}_{Pub(B)})$ for some index i and pair of agents A, B.



By control precedence, there is an earlier event in the run that marks its pre-control condition which must be of the form shown.

Case: $e = init : (A, B) : i : \mathbf{Out}(\text{out}\{n\}_{Pub(B)})$ for some index i and pair of agents A, B.



By output-input precedence, there is an earlier event that marks the condition $\{m, n_0, B\}_{Pub(A)}$. Since $B \neq B_0$, this also violates the invariant, contradicting e being the earliest event in the run to do so.

Authentication for the responder

Theorem

Consider a run

$$\langle NSL, s_0, t_0\rangle \xrightarrow{e_1} \cdots \xrightarrow{e_r} \langle p_r, s_r, t_r\rangle \xrightarrow{e_{r+1}} \cdots .$$

If it contains events b_1, b_2 and b_3 with

$$\begin{aligned}
act(b_1) &= resp : \mathrm{B}_0 : i : \mathrm{in}\ \{m_0, \mathrm{A}_0\}_{Pub(\mathrm{B}_0)} \\
act(b_2) &= resp : \mathrm{B}_0 : i : \mathrm{out\,new}\ n_0\ \ \{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)} \\
act(b_3) &= resp : \mathrm{B}_0 : i : \mathrm{in}\ \{n_0\}_{Pub(\mathrm{B}_0)}
\end{aligned}$$

and $Priv(\mathrm{A}_0) \not\sqsubset t_0$ then the run contains events a_1, a_2, a_3 with $a_3 \longrightarrow b_3$ where, for some index j

$$\begin{aligned}
act(a_1) &= init : (\mathrm{A}_0, \mathrm{B}_0) : j : \mathrm{out\,new}\ m_0\ \{m_0, \mathrm{A}_0\}_{Pub(\mathrm{B}_0)} \\
act(a_2) &= init : (\mathrm{A}_0, \mathrm{B}_0) : j : \mathrm{in}\ \{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)} \\
act(a_3) &= init : (\mathrm{A}_0, \mathrm{B}_0) : j : \mathrm{out}\{n_0\}_{Pub(\mathrm{B}_0)}
\end{aligned}$$

Authentication: proof

b_1 b_2 b_3

Draw $e \longrightarrow e'$ if e precedes e' in the run

Authentication: proof

$$b_1 \longrightarrow b_2 \longrightarrow b_3$$

Control precedence

Authentication: proof

$$b_1 \longrightarrow b_2 \longrightarrow b_3$$

The invariant

$$Q(p, s, t) \iff \forall M \in t : n_0 \sqsubset M \implies \{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)} \sqsubset M$$

- must be violated in the configuration immediately before b_3
- must hold in the configuration immediately after and all configurations before b_2, by freshness

Authentication: proof

$$b_1 \longrightarrow b_2 \longrightarrow b_3$$

$$e$$

The invariant

$$Q(p, s, t) \iff \forall M \in t : n_0 \sqsubset M \implies \{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)} \sqsubset M$$

- must be violated in the configuration immediately before b_3
- must hold in the configuration immediately after and all configurations before b_2, by freshness
- so there exists an earliest event e that breaks the invariant

Authentication: proof

$$\begin{tikzcd}
b_1 \arrow[r] & b_2 \arrow[r] \arrow[dr] & b_3 \\
 & & e \arrow[u]
\end{tikzcd}$$

The invariant

$$Q(p,s,t) \iff \forall M \in t : n_0 \sqsubset M \implies \{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)} \sqsubset M$$

- must be violated in the configuration immediately before b_3
- must hold in the configuration immediately after and all configurations before b_2, by freshness
- so there exists an earliest event e that breaks the invariant

Authentication: proof

$$
\begin{tikzcd}
b_1 \arrow[r] & b_2 \arrow[r] \arrow[dr] & b_3 \\
 & & a'_3 \arrow[u]
\end{tikzcd}
$$

The only kind of event that can break the invariant

$$Q(p,s,t) \iff \forall M \in t : n_0 \sqsubset M \implies \{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)} \sqsubset M$$

is an initiator event

$$act(a'_3) = init : (\mathrm{A}, \mathrm{B}_0) : j : \mathrm{out}\{n_0\}_{Pub(\mathrm{B}_0)}$$

using secrecy of $Priv(\mathrm{A}_0)$

Authentication: proof

$$\begin{tikzcd}
b_1 \arrow[r] & b_2 \arrow[r] \arrow[dr] & b_3 \\
a'_1 \arrow[r] & a'_2 \arrow[r] & a'_3 \arrow[u]
\end{tikzcd}$$

Control precedence

Authentication: proof

$$\begin{array}{ccccc} b_1 & \longrightarrow & b_2 & \longrightarrow & b_3 \\ & & \downarrow & & \uparrow \\ a'_1 & \longrightarrow & a'_2 & \longrightarrow & a'_3 \end{array}$$

$$Q(p, s, t) \iff \forall M \in t : n_0 \sqsubset M \implies \{m_0, n_0, \mathsf{B}_0\}_{Priv(\mathsf{A}_0)} \sqsubset M$$

Q holds immediately before a'_2, so $A = A_0$ and $m = m_0$

Authentication: proof

$$
\begin{tikzcd}
b_1 \arrow[r] & b_2 \arrow[r] \arrow[d] & b_3 \\
a'_1 \arrow[u] \arrow[r] & a'_2 \arrow[r] & a'_3 \arrow[u]
\end{tikzcd}
$$

Taking $a_1 = a'_1$, $a_2 = a'_2$ and $a_3 = a'_3$ we have

$$
\begin{aligned}
act(a_1) &= init : \mathrm{A}_0 : i : \text{out new } m_0 \; \{m_0, \mathrm{A}_0\}_{Pub(\mathrm{B}_0)} \\
act(a_2) &= init : \mathrm{A}_0 : i : \text{in } \{m_0, n_0, \mathrm{B}_0\}_{Pub(\mathrm{A}_0)} \\
act(a_3) &= init : \mathrm{A}_0 : i : \text{out}\{n_0\}_{Pub(\mathrm{B}_0)}
\end{aligned}
$$