Cross-Tool Semantics for Protocol Security Goals

SSR

December 5, 2016

Gaithersburg, MD

Joshua D. Guttman, John D. Ramsdell, Paul D. Rowe

The MITRE Corporation

{guttman, ramsdell, prowe}@mitre.org

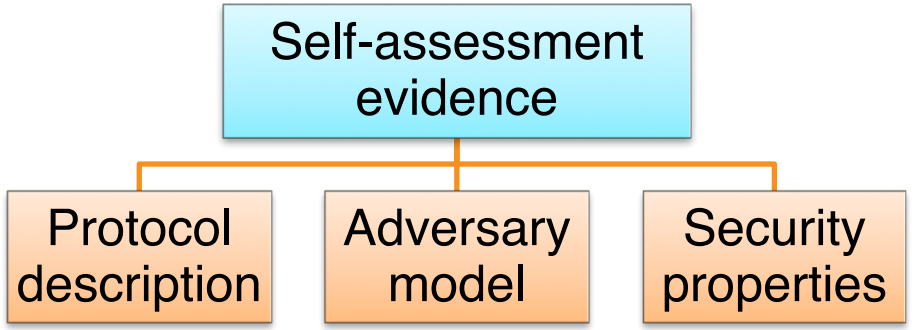
MITRE

Transparency in Security Standardization

- **Public trust in standardized security protocols is based in transparency**
  - Rigorous analysis can help rule out hidden insecurities
- **True transparency requires**
  - Reproducibility of results
  - By multiple, independent parties
  - Using a diversity of methods or tools
- **Sometimes analysis can be opaque**
  - Reliance on expert knowledge
  - Reliance on specific tool set

ISO/IEC 29128

- **Standardized framework for the verification of cryptographic protocols**



- **Highest assurance level (4) requires**
  - Formal, tool-supported, analysis of unbounded sessions

- **Reproducibility calls for tool-independent inputs**
  - We claim our first-order language of security goals is tool-independent

Main Contributions

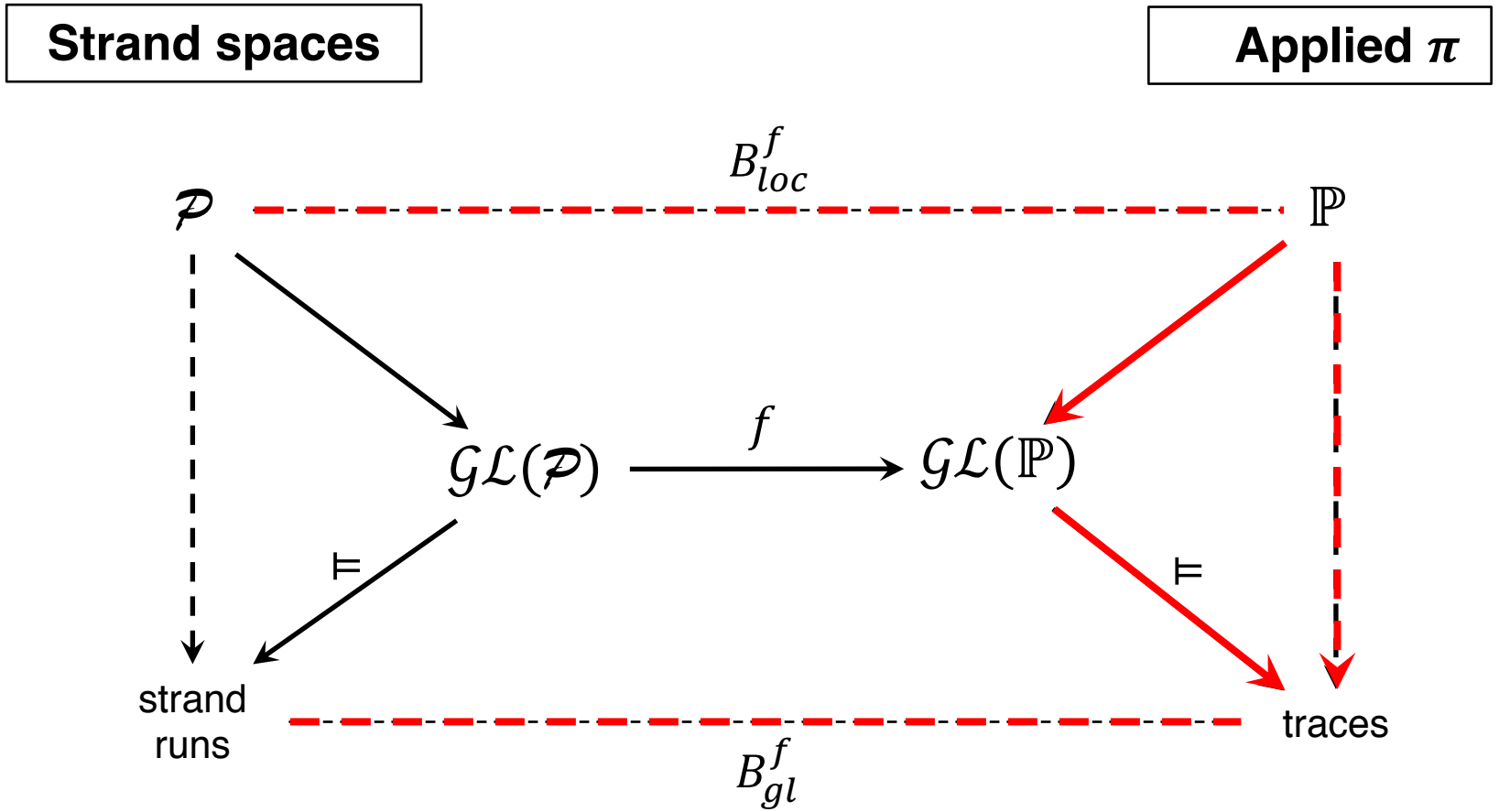
Impact:

- **Improve reproducibility of formal analyses**

Technical:

- **New semantics of first-order language for security goals**
  - Originally designed for strand spaces (CPSA)
  - Adapted for applied π (ProVerif)
- **Proof of compatibility of the two semantics**

Compatible Cross-Tool Semantics



Simple Example Protocol (SEP)

$$\boldsymbol{A} \longrightarrow \boldsymbol{B}: \{\!|\, [\![k]\!]_{\boldsymbol{sk}(\boldsymbol{a})} \,|\!\}^{\boldsymbol{a}}_{\boldsymbol{pk}(\boldsymbol{b})}$$

$$\boldsymbol{B} \longrightarrow \boldsymbol{A}: \{\!|\, \boldsymbol{d} \,|\!\}^{\boldsymbol{s}}_{k}$$

Clients A may not always choose symmetric key k randomly

Servers B always choose data d randomly

Sample Goals

If	A has finished a session with B;	then	B previously transmitted d with matching parameters
and	B's private decryption key $pk(B)^{-1}$ is uncompromised		
and	the session key k is freshly chosen	then	d remains confidential

Goal Language

Protocol-Dependent

- **Role position predicates**
  - `InitStart(n), RespDone(m)`
- **Parameter predicates**
  - `Self(n,a), SessKey(m,k)`

Protocol-Independent

- **Ordering and equality**
  - `Preceq(m,n), Coll(m,n), d = d'`
- **Freshness and secrecy**
  - `Unq(d), UnqAt(m,d), Non(sk(a))`

Sample Goal Formalized

If A **has finished a session with** B**;**

and B**'s private decryption key** $pk(B)^{-1}$ **is uncompromised**

and the session key k **is freshly chosen**

then B **previously transmitted** d **with matching parameters**

$$\forall n,a,b,k,d.\ \mathtt{InitDone}(n) \wedge \mathtt{Self}(n,a) \wedge \mathtt{Peer}(n,b) \wedge \mathtt{SessKey}(n,k) \wedge \mathtt{Data}(n,d) \wedge \mathtt{Non}(pk(b)^{-1}) \wedge \mathtt{Unq}(k) \Longrightarrow \exists m.\ \mathtt{RespDone}(m) \wedge \mathtt{Self}(m,b) \wedge \mathtt{Peer}(m,a) \wedge \mathtt{SessKey}(n,k) \wedge \mathtt{Data}(m,d) \wedge \mathtt{Precedes}(m,n)$$

Applied π Syntax

P, Q	$=$	0	$\text{in}(c, x) \,.\, P$	$\text{out}^{\ell}(c, u) \,.\, P$	$\text{let } x : \text{s} = v \text{ in } P \text{ else } Q$
		$(P \mid Q)$	$\text{new } n : \text{s} \,.\, P$	$\text{sum } n' : \text{s} \,.\, P$	$!\text{new } tid \,.\, \text{out}(c, tid) \,.\, P$
		$\ell \,.\, P$	$(c, tid \in Ch, x \in \mathcal{X}, n \in \mathcal{N}_s^{\nu}, n' \in \mathcal{N}_s^{0})$		

MITRE

Applied π Protocols

$$\boldsymbol{A} \longrightarrow \boldsymbol{B}: \{\!|\, [\![k]\!]_{sk(a)} \,|\!\}^{a}_{pk(b)}$$

$$\boldsymbol{B} \longrightarrow \boldsymbol{A}: \{\!|\, d \,|\!\}^{s}_{k}$$

Init =
!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **sum** k: skey . **out**$^{\texttt{InitStart}}(tid, \{\!|\, [\![k]\!]_{sk(a)} \,|\!\}^{a}_{pk(b)})$.
in(tid, z) . **let** d : data $= dec^{s}(z, k)$ **in** **`InitDone`** . **0**

Resp =
!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **in**(tid, z) . **let** x: $\top = dec^{a}(z, sk(b))$ **in**
let k: skey $= ver(x, pk(a))$ **in** **`RespStart`** . **new** d : data . **out**$^{\texttt{RespDone}}(tid, \{\!|\, d \,|\!\}^{s}_{k})$. **0**

MITRE

Operational Trace Semantics

$$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$$
$$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{sum}\ k : \mathsf{skey}\ .\ \mathbf{out}^{\texttt{InitStart}}(tid, \{\!|\ [\![k]\!]_{sk(a)}\ |\!\}^{a}_{pk(b)})\ .$$
$$\mathbf{in}(tid, z)\ .\ \mathbf{let}\ d : \mathsf{data} = dec^{s}(z, k)\ \mathbf{in}\ \texttt{InitDone}\ .\ \mathbf{0}$$

$$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$$
$$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \ \mathbf{in}(tid, z)\ .\ \mathbf{let}\ x : \top = dec^{a}(z, sk(b))\ \mathbf{in}$$
$$\mathbf{let}\ k : \mathsf{skey} = ver(x, pk(a))\ \mathbf{in}\ \ \texttt{RespStart}\ .\ \mathbf{new}\ d : \mathsf{data}\ .\ \mathbf{out}^{\texttt{RespDone}}(tid, \{\!|\ d\ |\!\}^{s}_{k})\ .\ \mathbf{0}$$

Operational Trace Semantics

!new tid **.** **out**(c, tid) **.**
sum a : agt **. sum** b : agt **. sum** k: skey **.** $\mathbf{out}^{\texttt{InitStart}}(tid, \{\!|\, [\![k]\!]_{sk(a)} \,|\}^{a}_{pk(b)})$ **.**

in (tid, z) **.** **let** d : data $= dec^{s}(z, k)$ **in** **`InitDone`** **.** $\mathbf{0}$

!new tid **.** **out**(c, tid) **.**
sum a : agt **. sum** b : agt **.** **in** (tid, z) **.** **let** x: $\top = dec^{a}(z, sk(b))$ **in**

let k : skey $= ver(x, pk(a))$ **in** **`RespStart`** **.** **new** d : data **.** $\mathbf{out}^{\texttt{RespDone}}(tid, \{\!|\, d \,|\}^{s}_{k})$ **.** $\mathbf{0}$

Operational Trace Semantics

!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **sum** k: skey . $\mathbf{out}^{\texttt{InitStart}}(tid, \{\!|\, [\![k]\!]_{sk(a)} \,|\}^{a}_{pk(b)})$.

in (tid, z) . **let** d : data $= dec^{s}(z, k)$ **in** **`InitDone`** . **0**

!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **in** (tid, z) . **let** x: $\top = dec^{a}(z, sk(b))$ **in**

let k : skey $= ver(x, pk(a))$ **in** **`RespStart`** . **new** d : data . $\mathbf{out}^{\texttt{RespDone}}(tid, \{\!|\, d \,|\}^{s}_{k})$. **0**

MITRE

Operational Trace Semantics

!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **sum** k: skey . **out**$^{\texttt{InitStart}}(tid, \{\!|\, [\![k]\!]_{sk(a)} \,|\}^{a}_{pk(b)})$.

in (tid, z) . **let** d : data $= dec^{s}(z, k)$ **in** **`InitDone`** . **0**

!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **in** (tid, z) . **let** x: $\top = dec^{a}(z, sk(b))$ **in**

let k : skey $= ver(x, pk(a))$ **in** **`RespStart`** . **new** d : data . **out**$^{\texttt{RespDone}}(tid, \{\!|\, d \,|\}^{s}_{k})$. **0**

(`InitStart`, **out**(tid, m_1)**,** $\mathcal{E}_1$**)** .

MITRE

Operational Trace Semantics

!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **sum** k: skey . $\mathbf{out}^{\texttt{InitStart}}(tid, \{\!|\, [\![k]\!]_{sk(a)} \,|\}^{a}_{pk(b)})$.

in (tid, z) . **let** d : data $= dec^{s}(z, k)$ **in** **`InitDone`** . **0**

!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **in** (tid, z) . **let** x: $\top = dec^{a}(z, sk(b))$ **in**

let k : skey $= ver(x, pk(a))$ **in** **`RespStart`** . **new** d : data . $\mathbf{out}^{\texttt{RespDone}}(tid, \{\!|\, d \,|\}^{s}_{k})$. **0**

(`InitStart`, **out**(tid, m_1)**,** $\mathcal{E}_1$**)** .

MITRE

Operational Trace Semantics

!new tid **.** **out**(c, tid) **.**
sum a : agt **.** **sum** b : agt **.** **sum** k: skey **.** $\mathbf{out}^{\texttt{InitStart}}(tid, \{\!|\, [\![k]\!]_{sk(a)} \,|\}^{a}_{pk(b)})$ **.**

in (tid, z) **.** **let** d : data $= dec^{s}(z, k)$ **in** **`InitDone`** **.** $\mathbf{0}$

!new tid **.** **out**(c, tid) **.**
sum a : agt **.** **sum** b : agt **.** **in** (tid, z) **.** **let** x: $\top = dec^{a}(z, sk(b))$ **in**

let k : skey $= ver(x, pk(a))$ **in** **`RespStart`** **.** **new** d : data **.** $\mathbf{out}^{\texttt{RespDone}}(tid, \{\!|\, d \,|\}^{s}_{k})$ **.** $\mathbf{0}$

(`InitStart`, **out**(tid, m_1)**,** $\mathcal{E}_1$**)** **.** **(**$\perp$**,** **in**(tid, m_2)**,** $\mathcal{E}_2$**)** **.**

MITRE

Operational Trace Semantics

!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **sum** k: skey . **out**$^{\texttt{InitStart}}(tid, \{\!|\, [\![k]\!]_{sk(a)} \,|\}^{a}_{pk(b)})$.

in(tid, z) . **let** d : data $= dec^{s}(z, k)$ **in** **`InitDone`** . **0**

!new tid . **out**(c, tid) .
sum a : agt . **sum** b : agt . **in**(tid, z) . **let** x: $\top = dec^{a}(z, sk(b))$ **in**

let k : skey $= ver(x, pk(a))$ **in** **`RespStart`** . **new** d : data . **out**$^{\texttt{RespDone}}(tid, \{\!|\, d \,|\}^{s}_{k})$. **0**

(`InitStart`, **out**(tid, m_1), $\mathcal{E}_1$**)** . **(**$\perp$, **in**(tid, m_2), $\mathcal{E}_2$**)** .

MITRE

Operational Trace Semantics

$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$
$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{sum}\ k : \mathsf{skey}\ .\ \mathbf{out}^{\texttt{InitStart}}(tid, \{|\ [\![k]\!]_{sk(a)}\ |\}^{a}_{pk(b)})\ .$

$\mathbf{in}(tid, z)\ .\ \mathbf{let}\ d : \mathsf{data} = dec^{s}(z, k)\ \mathbf{in}\ \texttt{InitDone}\ .\ \mathbf{0}$

$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$
$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \ \mathbf{in}(tid, z)\ .\ \mathbf{let}\ x : \top = dec^{a}(z, sk(b))\ \mathbf{in}$

$\mathbf{let}\ k : \mathsf{skey} = ver(x, pk(a))\ \mathbf{in}\ \texttt{RespStart}\ .\ \mathbf{new}\ d : \mathsf{data}\ .\ \mathbf{out}^{\texttt{RespDone}}(tid, \{|\ d\ |\}^{s}_{k})\ .\ \mathbf{0}$

$(\texttt{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\ .\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\ .\ (\texttt{RespStart}, \bot, \mathcal{E}_3)\ .$

Operational Trace Semantics

$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$
$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{sum}\ k : \mathsf{skey}\ .\ \mathbf{out}^{\texttt{InitStart}}(tid, \{|\ [\![k]\!]_{sk(a)}\ |\}^{a}_{pk(b)})\ .$

$\mathbf{in}(tid, z)\ .\ \mathbf{let}\ d : \mathsf{data} = dec^{s}(z, k)\ \mathbf{in}\ \texttt{InitDone}\ .\ \mathbf{0}$

$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$
$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \ \mathbf{in}(tid, z)\ .\ \mathbf{let}\ x : \top = dec^{a}(z, sk(b))\ \mathbf{in}$

$\mathbf{let}\ k : \mathsf{skey} = ver(x, pk(a))\ \mathbf{in}\ \ \texttt{RespStart}\ .\ \mathbf{new}\ d : \mathsf{data}\ .\ \mathbf{out}^{\texttt{RespDone}}(tid, \{|\ d\ |\}^{s}_{k})\ .\ \mathbf{0}$

$(\texttt{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\ .\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\ .\ (\texttt{RespStart}, \bot, \mathcal{E}_3)\ .$

Operational Trace Semantics

$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$
$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{sum}\ k : \mathsf{skey}\ .\ \mathbf{out}^{\texttt{InitStart}}(tid, \{|\ [\![k]\!]_{sk(a)}\ |\}^{a}_{pk(b)})\ .$

$\mathbf{in}(tid, z)\ .\ \mathbf{let}\ d : \mathsf{data} = dec^{s}(z, k)\ \mathbf{in}\ \texttt{InitDone}\ .\ \mathbf{0}$

$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$
$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{in}(tid, z)\ .\ \mathbf{let}\ x : \top = dec^{a}(z, sk(b))\ \mathbf{in}$

$\mathbf{let}\ k : \mathsf{skey} = ver(x, pk(a))\ \mathbf{in}\ \texttt{RespStart}\ .\ \mathbf{new}\ d : \mathsf{data}\ .\ \mathbf{out}^{\texttt{RespDone}}(tid, \{|\ d\ |\}^{s}_{k})\ .\ \mathbf{0}$

$(\texttt{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\ .\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\ .\ (\texttt{RespStart}, \bot, \mathcal{E}_3)\ .$

$(\texttt{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4)\ .$

MITRE

Operational Trace Semantics

$$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$$
$$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{sum}\ k : \mathsf{skey}\ .\ \mathbf{out}^{\texttt{InitStart}}(tid, \{|\ [\![k]\!]_{sk(a)}\ |\}^{a}_{pk(b)})\ .$$
$$\mathbf{in}(tid, z)\ .\ \mathbf{let}\ d : \mathsf{data} = dec^{s}(z, k)\ \mathbf{in}\ \texttt{InitDone}.\mathbf{0}$$

$$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$$
$$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{in}(tid, z)\ .\ \mathbf{let}\ x : \top = dec^{a}(z, sk(b))\ \mathbf{in}$$
$$\mathbf{let}\ k : \mathsf{skey} = ver(x, pk(a))\ \mathbf{in}\ \texttt{RespStart}\ .\ \mathbf{new}\ d : \mathsf{data}\ .\ \mathbf{out}^{\texttt{RespDone}}(tid, \{|\ d\ |\}^{s}_{k})\ .\mathbf{0}$$

$$(\texttt{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\ .\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\ .\ (\texttt{RespStart}, \bot, \mathcal{E}_3)\ .$$
$$(\texttt{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4)\ .\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5)\ .$$

MITRE

Operational Trace Semantics

$$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$$
$$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{sum}\ k : \mathsf{skey}\ .\ \mathbf{out}^{\texttt{InitStart}}(tid, \{|\ [\![k]\!]_{sk(a)}\ |\}^{a}_{pk(b)})\ .$$
$$\mathbf{in}(tid, z)\ .\ \mathbf{let}\ d : \mathsf{data} = dec^{s}(z, k)\ \mathbf{in}\ \texttt{InitDone.0}$$

$$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$$
$$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{in}(tid, z)\ .\ \mathbf{let}\ x : \top = dec^{a}(z, sk(b))\ \mathbf{in}$$
$$\mathbf{let}\ k : \mathsf{skey} = ver(x, pk(a))\ \mathbf{in}\ \texttt{RespStart}\ .\ \mathbf{new}\ d : \mathsf{data}\ .\ \mathbf{out}^{\texttt{RespDone}}(tid, \{|\ d\ |\}^{s}_{k})\ .\ \mathbf{0}$$

$$(\texttt{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\ .\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\ .\ (\texttt{RespStart}, \bot, \mathcal{E}_3)\ .$$
$$(\texttt{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4)\ .\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5)\ .$$

MITRE

Operational Trace Semantics

$$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$$
$$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \mathbf{sum}\ k : \mathsf{skey}\ .\ \mathbf{out}^{\texttt{InitStart}}(tid, \{|\ [\![k]\!]_{sk(a)}\ |\}^{a}_{pk(b)})\ .$$
$$\mathbf{in}(tid, z)\ .\ \mathbf{let}\ d : \mathsf{data} = dec^{s}(z, k)\ \mathbf{in}\ \texttt{InitDone}\ .\ \mathbf{0}$$

$$!\mathbf{new}\ tid\ .\ \mathbf{out}(c, tid)\ .$$
$$\mathbf{sum}\ a : \mathsf{agt}\ .\ \mathbf{sum}\ b : \mathsf{agt}\ .\ \ \mathbf{in}(tid, z)\ .\ \mathbf{let}\ x : \top = dec^{a}(z, sk(b))\ \mathbf{in}$$
$$\mathbf{let}\ k : \mathsf{skey} = ver(x, pk(a))\ \mathbf{in}\ \ \texttt{RespStart}\ .\ \mathbf{new}\ d : \mathsf{data}\ .\ \mathbf{out}^{\texttt{RespDone}}(tid, \{|\ d\ |\}^{s}_{k})\ .\ \mathbf{0}$$

$$(\texttt{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\ .\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\ .\ (\texttt{RespStart}, \bot, \mathcal{E}_3)\ .$$
$$(\texttt{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4)\ .\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5)\ .\ (\texttt{InitDone}, \bot, \mathcal{E}_6)$$

MITRE

Security Goal Semantics

$$(\mathbf{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\,.\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\,.\ (\mathbf{RespStart}, \bot, \mathcal{E}_3)\,.$$
$$(\mathbf{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4)\,.\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5)\,.\ (\mathbf{InitDone}, \bot, \mathcal{E}_6)$$

$$\forall\, \mathbf{n,a,b,k,d.}\ \mathbf{InitDone(n)} \wedge \mathbf{Self(n,a)} \wedge \mathbf{Peer(n,b)} \wedge \mathbf{SessKey(n,k)} \wedge \mathbf{Data(n,d)} \wedge \mathbf{Non(pk(b)^{-1})} \wedge \mathbf{Unq(k)} \implies \exists\, \mathbf{m.}\ \mathbf{RespDone(m)} \wedge \mathbf{Self(m,b)} \wedge \mathbf{Peer(m,a)} \wedge \mathbf{SessKey(m,k)} \wedge \mathbf{Data(m,d)} \wedge \mathbf{Preceq(m,n)}$$

Security Goal Semantics

$$(\mathbf{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\,.\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\,.\ (\mathbf{RespStart}, \bot, \mathcal{E}_3)\,.$$
$$(\mathbf{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4)\,.\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5)\,.\ (\mathbf{InitDone}, \bot, \mathcal{E}_6)$$

$$\forall\, \mathbf{n,a,b,k,d.}\ \mathbf{InitDone(n)} \wedge \mathbf{Self(n,a)} \wedge \mathbf{Peer(n,b)} \wedge \mathbf{SessKey(n,k)} \wedge \mathbf{Data(n,d)} \wedge \mathbf{Non(pk(b)^{-1})} \wedge \mathbf{Unq(k)} \Longrightarrow \exists\, \mathbf{m.}\ \mathbf{RespDone(m)} \wedge \mathbf{Self(m,b)} \wedge \mathbf{Peer(m,a)} \wedge \mathbf{SessKey(m,k)} \wedge \mathbf{Data(m,d)} \wedge \mathbf{Preceq(m,n)}$$

Security Goal Semantics

$$(\mathbf{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1).\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2).\ (\mathbf{RespStart}, \bot, \mathcal{E}_3).$$
$$(\mathbf{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4).\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5).\ (\mathbf{InitDone}, \bot, \mathcal{E}_6)$$

$$\forall n,a,b,k,d.\ \mathbf{InitDone}(n) \wedge \mathbf{Self}(n,a) \wedge \mathbf{Peer}(n,b) \wedge \mathbf{SessKey}(n,k) \wedge \mathbf{Data}(n,d) \wedge \mathbf{Non}(\mathbf{pk}(b)^{-1}) \wedge \mathbf{Unq}(k) \Longrightarrow \exists m.\ \mathbf{RespDone}(m) \wedge \mathbf{Self}(m,b) \wedge \mathbf{Peer}(m,a) \wedge \mathbf{SessKey}(m,k) \wedge \mathbf{Data}(m,d) \wedge \mathbf{Preceq}(m,n)$$

Security Goal Semantics

$$(\mathbf{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\,.\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\,.\ (\mathbf{RespStart}, \bot, \mathcal{E}_3)\,.$$
$$(\mathbf{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4)\,.\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5)\,.\ (\mathbf{InitDone}, \bot, \mathcal{E}_6)$$

$$\forall\, n,a,b,k,d.\ \mathbf{InitDone}(n) \wedge \mathbf{Self}(n,a) \wedge \mathbf{Peer}(n,b) \wedge \mathbf{SessKey}(n,k) \wedge \mathbf{Data}(n,d) \wedge \mathbf{Non}(\mathbf{pk}(b)^{-1}) \wedge \mathbf{Unq}(k) \Longrightarrow \exists\, m.\ \mathbf{RespDone}(m) \wedge \mathbf{Self}(m,b) \wedge \mathbf{Peer}(m,a) \wedge \mathbf{SessKey}(m,k) \wedge \mathbf{Data}(m,d) \wedge \mathbf{Preceq}(m,n)$$

Security Goal Semantics

$$(\mathbf{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1).\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2).\ (\mathbf{RespStart}, \bot, \mathcal{E}_3).$$
$$(\mathbf{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4).\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5).\ (\mathbf{InitDone}, \bot, \mathcal{E}_6)$$

$$\forall n,a,b,k,d.\ \mathbf{InitDone}(n) \wedge \mathbf{Self}(n,a) \wedge \mathbf{Peer}(n,b) \wedge \mathbf{SessKey}(n,k) \wedge \mathbf{Data}(n,d) \wedge \mathbf{Non}(\mathbf{pk}(b)^{-1}) \wedge \mathbf{Unq}(k) \Longrightarrow \exists m.\ \mathbf{RespDone}(m) \wedge \mathbf{Self}(m,b) \wedge \mathbf{Peer}(m,a) \wedge \mathbf{SessKey}(m,k) \wedge \mathbf{Data}(m,d) \wedge \mathbf{Preceq}(m,n)$$

Security Goal Semantics

$$(\mathbf{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\ .\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\ .\ (\mathbf{RespStart}, \bot, \mathcal{E}_3)\ .$$
$$(\mathbf{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4)\ .\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5)\ .\ (\mathbf{InitDone}, \bot, \mathcal{E}_6)$$

$$\forall\, \mathbf{n,a,b,k,d.}\ \mathbf{InitDone(n)} \wedge \mathbf{Self(n,a)} \wedge \mathbf{Peer(n,b)} \wedge \mathbf{SessKey(n,k)} \wedge \mathbf{Data(n,d)} \wedge \mathbf{Non(pk(b)^{-1})} \wedge \mathbf{Unq(k)} \Longrightarrow \exists\, \mathbf{m.}\ \mathbf{RespDone(m)} \wedge \mathbf{Self(m,b)} \wedge \mathbf{Peer(m,a)} \wedge \mathbf{SessKey(m,k)} \wedge \mathbf{Data(m,d)} \wedge \mathbf{Preceq(m,n)}$$

Security Goal Semantics

$$(\mathbf{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)\,.\ (\bot, \mathbf{in}(tid, m_2), \mathcal{E}_2)\,.\ (\mathbf{RespStart}, \bot, \mathcal{E}_3)\,.$$
$$(\mathbf{RespDone}, \mathbf{out}(tid, m_4), \mathcal{E}_4)\,.\ (\bot, \mathbf{in}(tid, m_5), \mathcal{E}_5)\,.\ (\mathbf{InitDone}, \bot, \mathcal{E}_6)$$

$$\forall n,a,b,k,d.\ \mathbf{InitDone}(n) \wedge \mathbf{Self}(n,a) \wedge \mathbf{Peer}(n,b) \wedge \mathbf{SessKey}(n,k) \wedge \mathbf{Data}(n,d) \wedge \mathbf{Non}(\mathbf{pk}(b)^{-1}) \wedge \mathbf{Unq}(k) \Longrightarrow \exists m.\ \mathbf{RespDone}(m) \wedge \mathbf{Self}(m,b) \wedge \mathbf{Peer}(m,a) \wedge \mathbf{SessKey}(m,k) \wedge \mathbf{Data}(m,d) \wedge \mathbf{Preceq}(m,n)$$

MITRE

Security Goal Semantics

$$(\texttt{InitStart}, \texttt{out}(tid, m_1), \mathcal{E}_1)\,.\ (\bot, \texttt{in}(tid, m_2), \mathcal{E}_2)\,.\ (\texttt{RespStart}, \bot, \mathcal{E}_3)\,.$$
$$(\texttt{RespDone}, \texttt{out}(tid, m_4), \mathcal{E}_4)\,.\ (\bot, \texttt{in}(tid, m_5), \mathcal{E}_5)\,.\ (\texttt{InitDone}, \bot, \mathcal{E}_6)$$

A protocol achieves a goal Γ iff
all admitted traces satisfy Γ

$$\forall\, \texttt{n,a,b,k,d.}\ \texttt{InitDone(n)} \wedge \texttt{Self(n,a)} \wedge \texttt{Peer(n,b)} \wedge \texttt{SessKey(n,k)} \wedge \texttt{Data(n,d)} \wedge \texttt{Non}(\texttt{pk(b)}^{-1}) \wedge \texttt{Unq(k)} \implies \exists\, \texttt{m.}\ \texttt{RespDone(m)} \wedge \texttt{Self(m,b)} \wedge \texttt{Peer(m,a)} \wedge \texttt{SessKey(m,k)} \wedge \texttt{Data(m,d)} \wedge \texttt{Preceq(m,n)}$$

MITRE

Origination

- **An entity originates a value if it transmits it without having received it**
  - An entity must know or create a value to originate it
- **Method for expressing freshness and secrecy of values**
  - Characterizes the effects of randomness and secrecy

Unique origination:
A randomly chosen value can only originate at most once.

Non-origination:
Keys kept secret never originate.

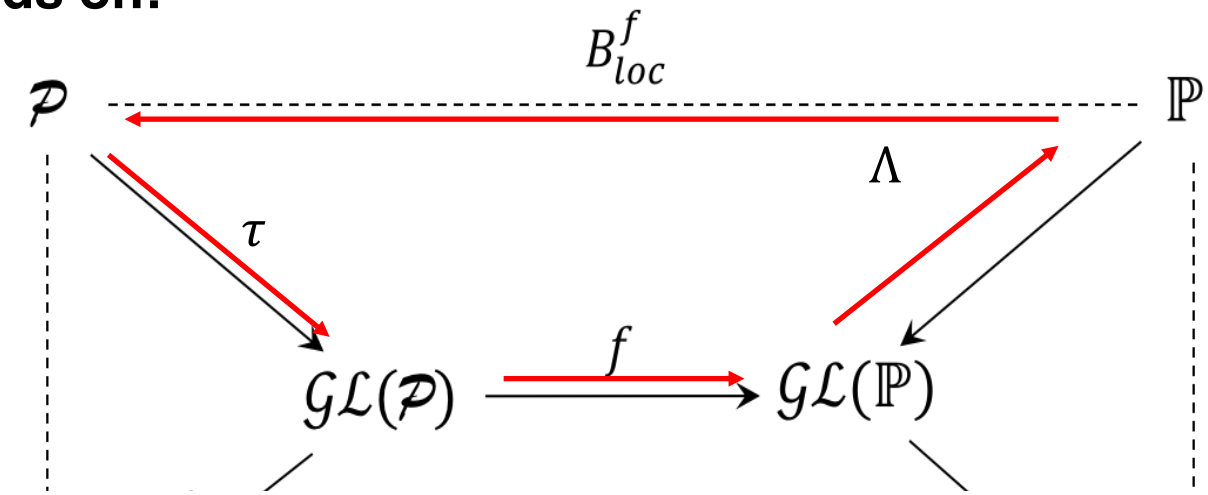
- **Traces allow us to identify points of origination for values**

Notable Details

- **!new** tid **. out**(c, tid) **.** P
  - Replication is bound to channel restriction
  - Enables sensible semantics for `Coll(m,n)`
- **Diverse trace elements**
  - *Labels*: semantics for role position predicates
  - *Msg events*: semantics for origination
- **Labels occur in two ways**
  - **out**$^{\ell}(c, u)$ **.** P and ℓ **.** P
  - Required for compatibility with strand space semantics
- **sum** n' **:** s **.** P
  - Acts as infinite choice operator $P + Q$
  - Allows multiple origination of values
  - Finite choice may suffice (e.g. bounded agent set)

Compatible Protocols

- **Independent replication of results depends on:**
  - Analyzing compatible protocols
    - Roles must create same traces
  - Compatible semantics for predicates
    - $\Lambda\left(f\left(\tau(\rho_i@j)\right)\right) = \rho_i@j$



- **We provide**
  - Prototype compiler from strands to applied π
  - Semantic criterion for compatibility

Compilation

Main Idea:

$+m \longmapsto \mathbf{out}^{\ell}(tid, m)$

$-m \longmapsto \mathbf{in}(tid, z).\ [\ldots]\ .\ell$

Sequence of let bindings to parse z as m

Compilation

Main Idea:

$+m \longmapsto \textbf{out}^{\ell}(tid, m)$

$-m \longmapsto \textbf{in}(tid, z).\ [\ldots]\ .\ell$

Sequence of let bindings to parse z as m

Resp $\quad$ **!new** tid . **out**(c, tid) .

Compilation

Main Idea:

$+m \longmapsto \textbf{out}^{\ell}(tid, m)$

$-m \longmapsto \textbf{in}(tid, z).\ [\ldots]\ .\ \ell$

Sequence of let bindings to parse z as m

Resp

!new tid **. out**(c, tid) **.**

$\{|\ [\![k]\!]_{sk(a)}\ |\}^{a}_{pk(b)}$

in (tid, z **) . let** x: $\top = dec^{a}(z, sk(b))$ **in**
let k: skey $= ver(x, pk(a))$ **in** **`RespStart`** **.**

Compilation

Main Idea:

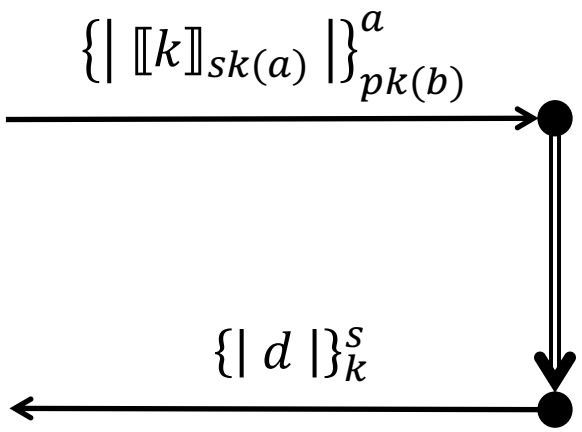
$+m \longmapsto \textbf{out}^{\ell}(tid, m)$

$-m \longmapsto \textbf{in}(tid, z).\ [\ldots]\ .\ \ell$

Sequence of let bindings to parse z as m

Resp

!new tid . **out**(c, tid) .



in (tid, z) . **let** x: $\top = dec^{a}(z, sk(b))$ **in**
let k: skey $= ver(x, pk(a))$ **in** **`RespStart`** .

out`RespDone` $(tid, \{|\ d\ |\}^{s}_{k})$. $\mathbf{0}$

MITRE

Compilation

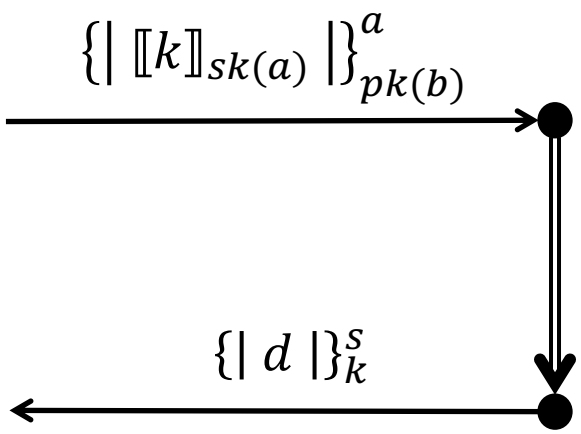
Main Idea:

$+m \longmapsto \textbf{out}^{\ell}(tid, m)$

$-m \longmapsto \textbf{in}(tid, z).\ [\ldots]\ .\ \ell$

Sequence of let bindings to parse z as m

Resp



!new tid **. out**(c, tid) **.**

sum a : agt **. sum** b : agt **.**

in (tid, z) **. let** x: $\top = dec^{a}(z, sk(b))$ **in**
let k: skey $= ver(x, pk(a))$ **in** `RespStart` **.**

out`RespDone` $(tid, \{|\ d\ |\}^{s}_{k})$ **. 0**

MITRE

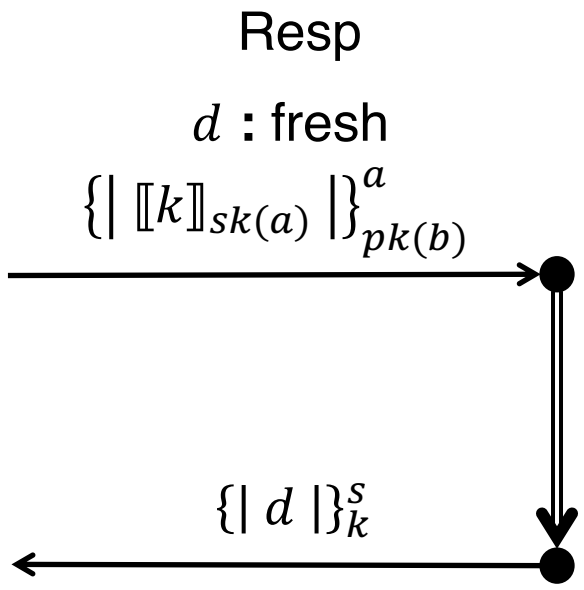
Compilation

Main Idea:

$+m \longmapsto \mathbf{out}^{\ell}(tid, m)$

$-m \longmapsto \mathbf{in}(tid, z).\ [\ldots]\ .\ \ell$

Sequence of let bindings to parse z as m



!new tid . **out**(c, tid) .

sum a : agt . **sum** b : agt .

in (tid, z) . **let** x: $\top = dec^{a}(z, sk(b))$ **in**
let k: skey $= ver(x, pk(a))$ **in** **`RespStart`** .

new d : data .

out$^{\texttt{RespDone}}(tid, \{|\ d\ |\}^{s}_{k})$. **0**

MITRE

Compatibility Criterion: Local Bisimulation

$B_1^{\Lambda}(\iota; \mathcal{S}, P, \mathcal{E})$ iff ι and $(\mathcal{S}, P, \mathcal{E})$ have

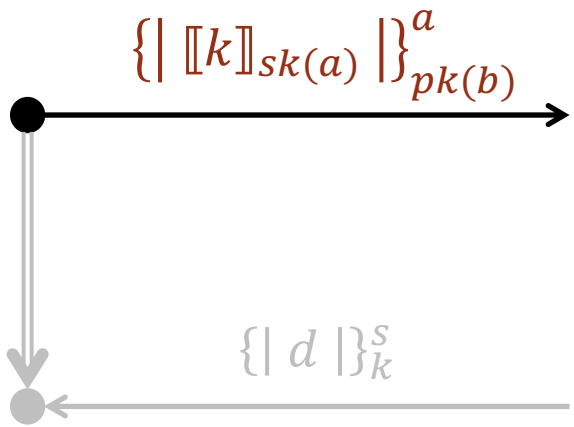
- compatible pasts
- compatible possible next steps

Compatibility Criterion: Local Bisimulation

$B_1^{\Lambda}(\iota; \mathcal{S}, P, \mathcal{E})$ iff ι and $(\mathcal{S}, P, \mathcal{E})$ have

- compatible pasts
- compatible possible next steps

$\boldsymbol{\iota} =$ (init, 1, $\sigma = \{\text{a} \mapsto \boldsymbol{a}, b \mapsto \boldsymbol{b}, s \mapsto \boldsymbol{s}\}$)



$\mathcal{S} =$ $(\texttt{InitStart}, \texttt{out}(tid, m_1), \mathcal{E}_1)$.

$P =$ **in** (tid, z) . **let** d : data $= dec^{s}(z, k)$ **in**
`InitDone.0`

$\mathcal{E} =$ $\mathcal{E}_1 = \{\text{a} \mapsto \boldsymbol{a}, b \mapsto \boldsymbol{b}, s \mapsto \boldsymbol{s}, tid \mapsto \boldsymbol{tid_1}\}$

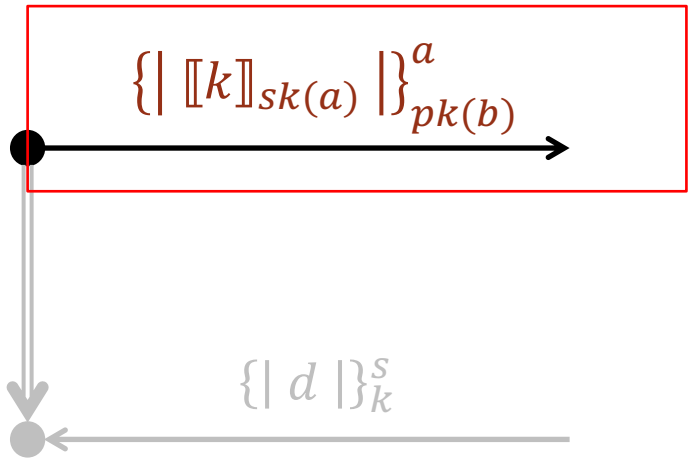
MITRE

Compatibility Criterion: Local Bisimulation

$B_1^{\Lambda}(\iota; \mathcal{S}, P, \mathcal{E})$ iff ι and $(\mathcal{S}, P, \mathcal{E})$ have

- compatible pasts
- compatible possible next steps

$\iota =$ (init, 1, $\sigma = \{\mathrm{a} \mapsto \boldsymbol{a}, b \mapsto \boldsymbol{b}, s \mapsto \boldsymbol{s}\}$)



$\mathcal{S} =$ **(InitStart, out**(tid, m_1)**,** $\mathcal{E}_1$**) .**

$P =$ **in (**tid, z**) . let** d : data $= dec^{s}(z, k)$ **in**
InitDone . 0

$\mathcal{E} =$ $\boldsymbol{\mathcal{E}_1} = \{\mathrm{a} \mapsto \boldsymbol{a}, b \mapsto \boldsymbol{b}, s \mapsto \boldsymbol{s}, tid \mapsto \boldsymbol{tid_1}\}$

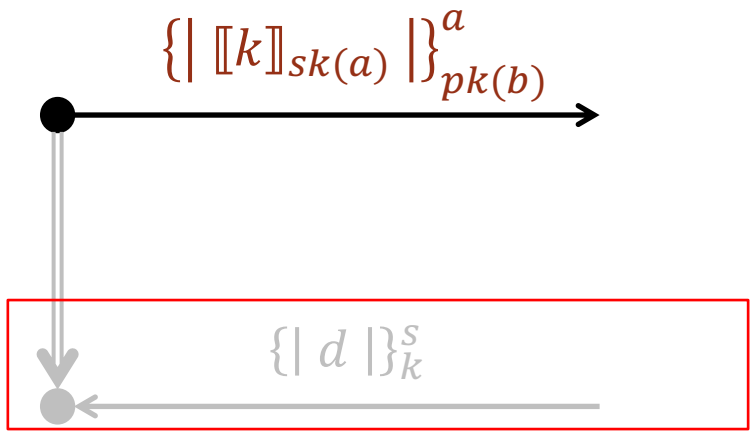
MITRE

Compatibility Criterion: Local Bisimulation

$B_1^{\Lambda}(\iota; \mathcal{S}, P, \mathcal{E})$ iff ι and $(\mathcal{S}, P, \mathcal{E})$ have

- compatible pasts
- compatible possible next steps

$\boldsymbol{\iota} =$ (init, 1, $\sigma = \{\text{a} \mapsto \boldsymbol{a}, b \mapsto \boldsymbol{b}, s \mapsto \boldsymbol{s}\}$)



$\mathcal{S} =$ **(InitStart, out**(tid, m_1)**,** $\mathcal{E}_1$**).**

$P =$ **in (**tid, z**) . let** d : data $= dec^{s}(z, k)$ **in**
InitDone . 0

$\mathcal{E} =$ $\boldsymbol{\mathcal{E}_1} = \{\text{a} \mapsto \boldsymbol{a}, b \mapsto \boldsymbol{b}, s \mapsto \boldsymbol{s}, tid \mapsto \boldsymbol{tid_1}\}$

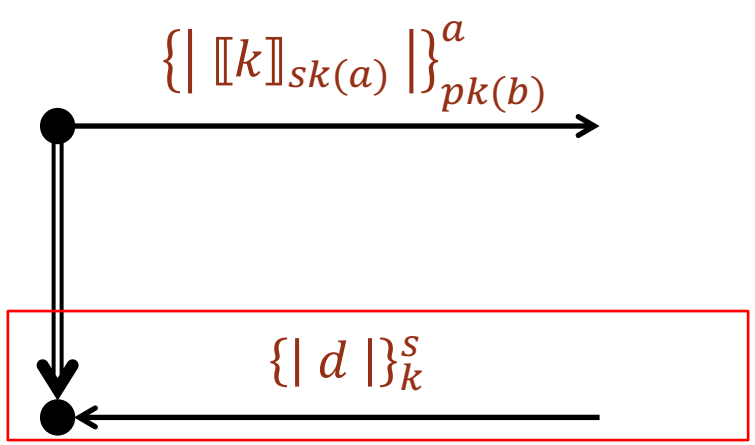
MITRE

Compatibility Criterion: Local Bisimulation

$B_1^{\Lambda}(\iota; \mathcal{S}, P, \mathcal{E})$ iff ι and $(\mathcal{S}, P, \mathcal{E})$ have

- compatible pasts
- compatible possible next steps

$\iota =$ (init, 1, $\sigma = \{a \mapsto \boldsymbol{a}, b \mapsto \boldsymbol{b}, s \mapsto \boldsymbol{s}\}$)



$\mathcal{S} =$ **(InitStart, out**(tid, m_1)**,** $\mathcal{E}_1$**) .**
($\perp$**, in**(tid, m_5)**,** $\mathcal{E}_5$**) . (InitDone,** $\perp$**,** $\mathcal{E}_6$**)**

$P =$ **in (**tid, z**) . let** d : data $= dec^s(z, k)$ **in**
InitDone . 0

$\mathcal{E} =$ $\mathcal{E}_1 = \{a \mapsto \boldsymbol{a}, b \mapsto \boldsymbol{b}, s \mapsto \boldsymbol{s}, tid \mapsto \boldsymbol{tid_1}\}$

MITRE

Global Bisimulation

Init

Resp

Global Bisimulation

Global Bisimulation

$(\mathbf{InitStart}, \mathbf{out}(tid, m_1), \mathcal{E}_1)$.

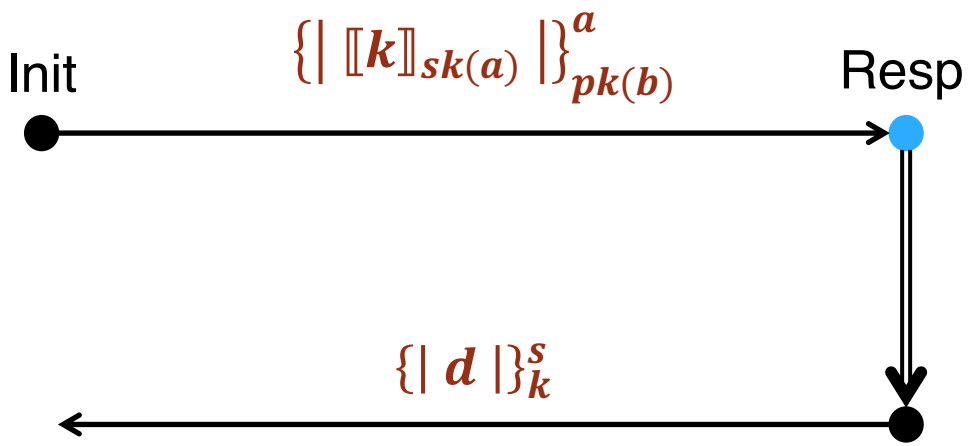
Global Bisimulation

$(\texttt{InitStart}, \textbf{out}(tid, m_1), \mathcal{E}_1)\,.\ (\bot, \textbf{in}(tid, m_2), \mathcal{E}_2)\,.\ (\texttt{RespStart}, \bot, \mathcal{E}_3)\,.$

Global Bisimulation

$(\texttt{InitStart}, \textbf{out}(tid, m_1), \mathcal{E}_1)\,.\ (\bot, \textbf{in}(tid, m_2), \mathcal{E}_2)\,.\ (\texttt{RespStart}, \bot, \mathcal{E}_3)\,.$

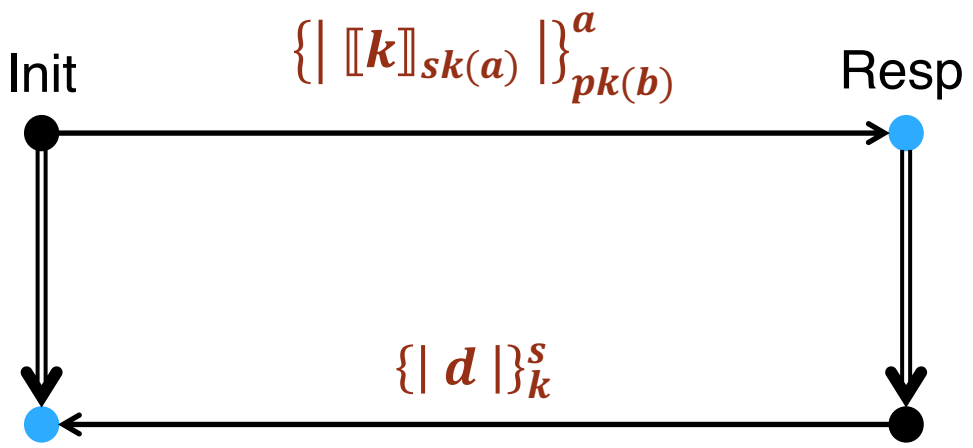
Global Bisimulation



$(\texttt{InitStart}, \textbf{out}(tid, m_1), \mathcal{E}_1)$. $(\bot, \textbf{in}(tid, m_2), \mathcal{E}_2)$. $(\texttt{RespStart}, \bot, \mathcal{E}_3)$.

$(\texttt{RespDone}, \textbf{out}(tid, m_4), \mathcal{E}_4)$.

Global Bisimulation

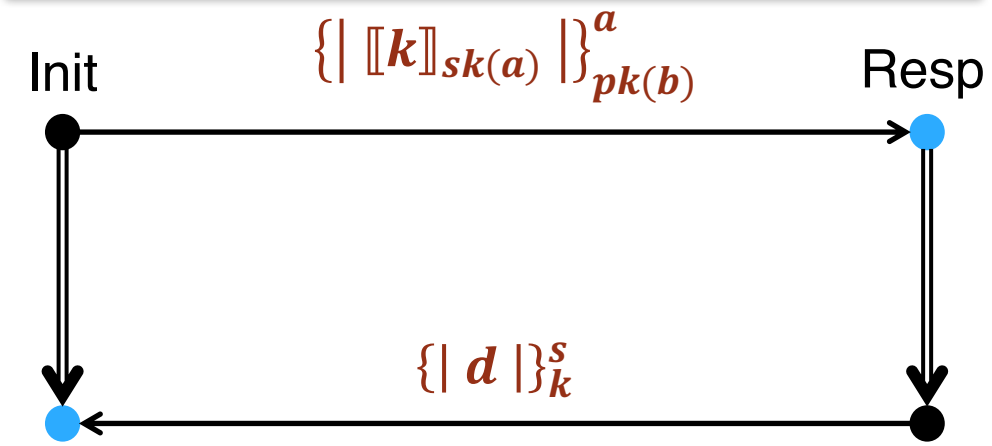


$(\texttt{InitStart}, \textbf{out}(tid, m_1), \mathcal{E}_1)$. $(\bot, \textbf{in}(tid, m_2), \mathcal{E}_2)$. $(\texttt{RespStart}, \bot, \mathcal{E}_3)$.
$(\texttt{RespDone}, \textbf{out}(tid, m_4), \mathcal{E}_4)$. $(\bot, \textbf{in}(tid, m_5), \mathcal{E}_5)$. $(\texttt{InitDone}, \bot, \mathcal{E}_6)$

Global Bisimulation

Theorem:

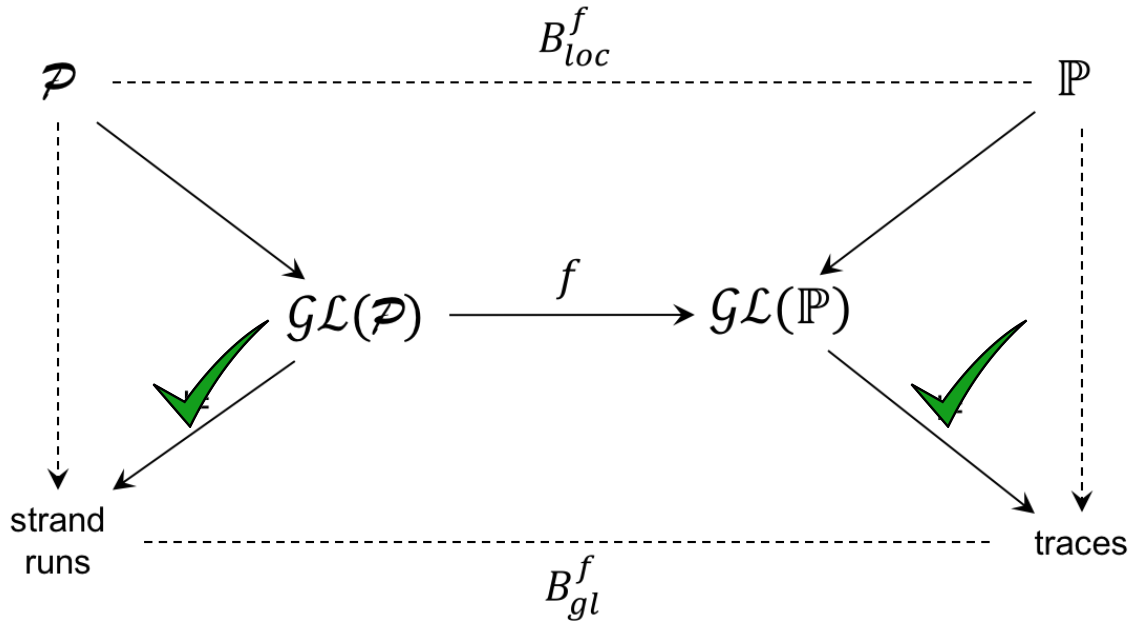
Compatible protocols are globally bisimilar



$(\texttt{InitStart}, \textbf{out}(tid, m_1), \mathcal{E}_1)$. $(\bot, \textbf{in}(tid, m_2), \mathcal{E}_2)$. $(\texttt{RespStart}, \bot, \mathcal{E}_3)$.
$(\texttt{RespDone}, \textbf{out}(tid, m_4), \mathcal{E}_4)$. $(\bot, \textbf{in}(tid, m_5), \mathcal{E}_5)$. $(\texttt{InitDone}, \bot, \mathcal{E}_6)$

Goal Preservation Theorem

Suppose $\mathcal{P}$ and $\mathbb{P}$ are globally bisimilar protocols.



If $\mathcal{P}$ achieves Γ
then $\mathbb{P}$ achieves $f(\Gamma)$

MITRE

Converse is Not Always True

Strand spaces

Partially ordered executions

Applied π

Totally ordered traces

Trichotomy

$\forall$`m,n.Preceq(m,n)` $\vee$ `Preceq(n,m)`

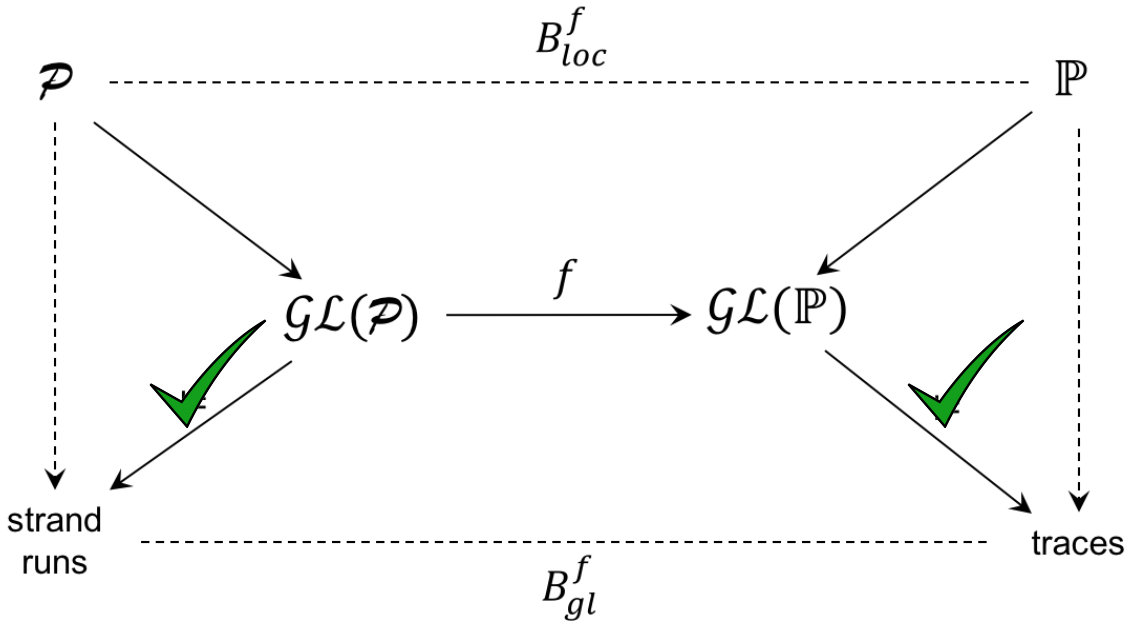
Restricted goals:

`Preceq` **does not occur, or**

$\vee$ **does not occur**

Conjecture: Partial Converse

Suppose $\mathcal{P}$ and $\mathbb{P}$ are globally bisimilar protocols, and Γ is a restricted goal



If $\mathbb{P}$ achieves $f(\Gamma)$
Then $\mathcal{P}$ achieves Γ

MITRE

Summary

- **Improve reproducibility of formal analyses**
- **Demonstrate tool-independence of security goal language**
  - Provide goal semantics for applied π
  - Prove compatibility with goal semantics for strand spaces