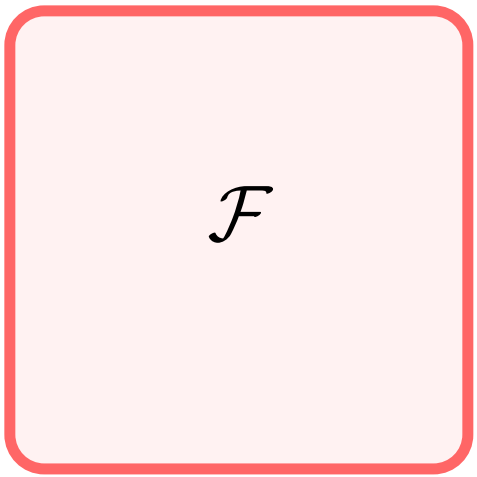
Simulation based security in the applied pi calculus

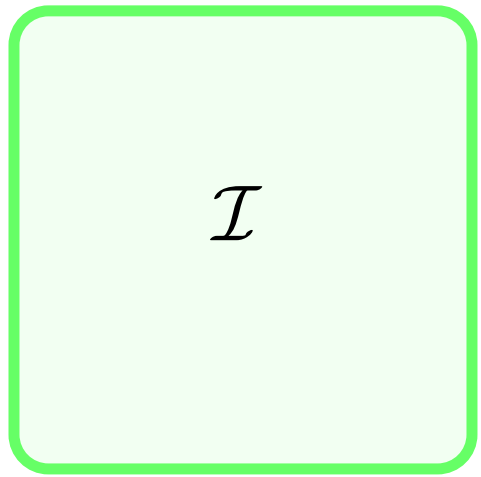
Stéphanie Delaune Steve Kremer Olivier Pereira

Formacrypt, 19/06/2009

Simulation based security

Main idea: composition-refinement framework in an adversarial setting

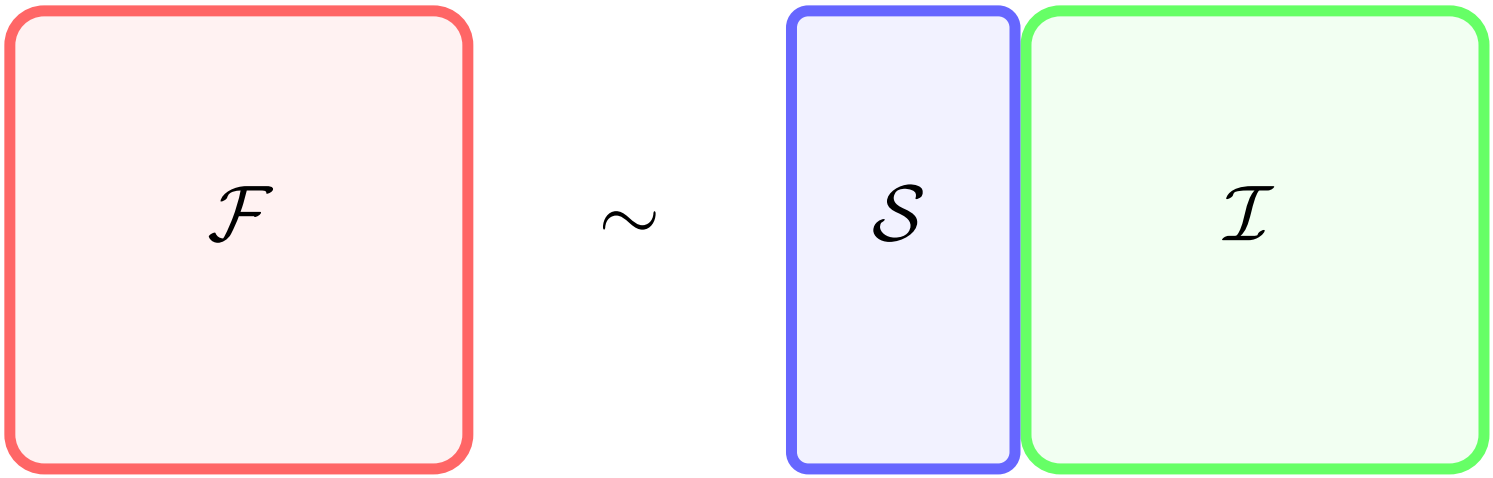




- Canetti's UC framework
- Backes,Pfitzmann, Waidner's reactive simulatability framework
- Küster's IITM
- Canetti, Lynch, Segala et al.'s Task-PIOAs

Simulation based security

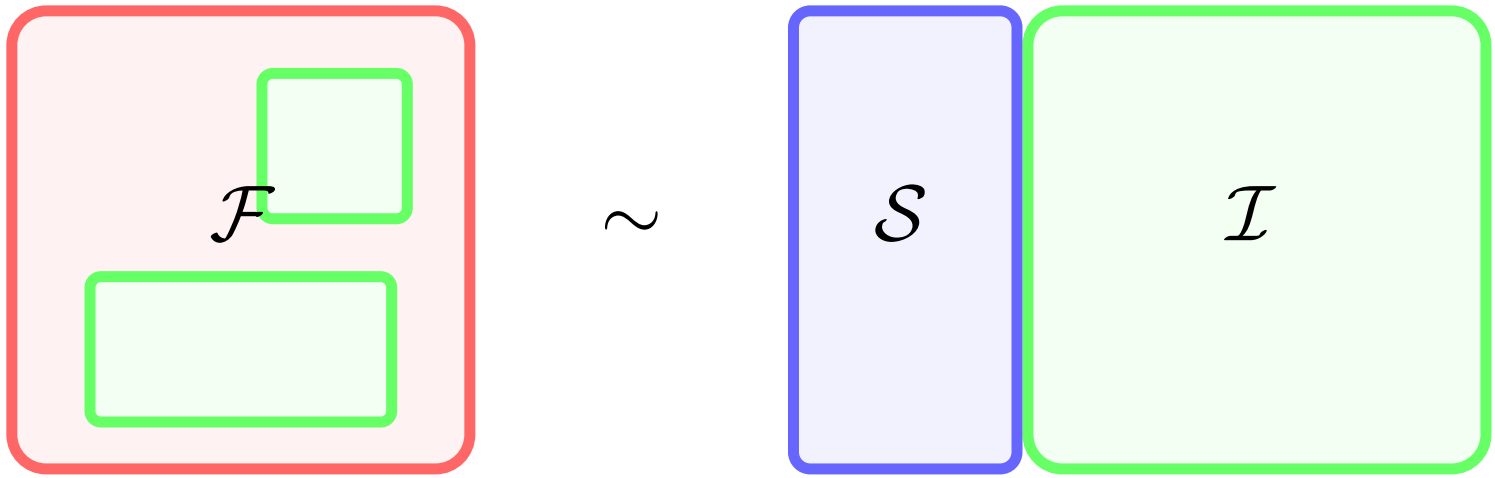
Main idea: composition-refinement framework in an adversarial setting



- Canetti's UC framework
- Backes,Pfitzmann, Waidner's reactive simulatability framework
- Küster's IITM
- Canetti, Lynch, Segala et al.'s Task-PIOAs

Simulation based security

Main idea: composition-refinement framework in an adversarial setting



- Canetti's UC framework
- Backes,Pfitzmann, Waidner's reactive simulatability framework
- Küster's IITM
- Canetti, Lynch, Segala et al.'s Task-PIOAs

Motivations

- Applied pi calculus is simpler and allows more rigorous proofs than networks of interactive TM (e.g. definition of poly-time is non-trivial!)
- Concurrent executions *vs* sequential scheduling
- Refinement and composition framework in a Dolev-Yao model

The applied π-calculus

Applied pi-calculus: [Abadi & Fournet, 01]

basic programming language with constructs for concurrency, communication and cryptographic primitives

- based on the π-calculus [Milner *et al.*, 92]
- in some ways similar to the spi-calculus [Abadi & Gordon, 98], but more general w.r.t. cryptography

Advantages:

- naturally models a Dolev-Yao attacker
- allows us to model less classical cryptographic primitives
- both reachability and indistinguishability-based specification of properties
  - Observational equivalence
- automated proofs (not complete, termination not guaranteed) using ProVerif tool [Blanchet]
- powerful proof techniques for hand proofs
- successfully used to analyze a variety of security protocols

The applied π-calculus on an example

Syntax:

- Equational theory: $dec(enc(x, y), y) = x$
- Process:

$$P = \nu s, k.(\text{out}(c_1, enc(s, k)) \mid \text{in}(c_1, y).\text{out}(c_2, dec(y, k))).$$

Special processes: active substitutions $P \mid \{^{M}/_{x}\}$

The applied π-calculus on an example

Syntax:

- Equational theory: $dec(enc(x, y), y) = x$
- Process:

$$P = \nu s, k.(\text{out}(c_1, enc(s, k)) \mid \text{in}(c_1, y).\text{out}(c_2, dec(y, k))).$$

Special processes: active substitutions $P \mid \{^M/_x\}$

Semantics:

- Operational semantics $\rightarrow$: closed by structural equivalence ($\equiv$) and application of evaluation contexts such that

Comm	$\text{out}(a, x).P \mid \text{in}(a, x).Q \rightarrow P \mid Q$
Then	if $M = M$ then P else $Q \rightarrow P$
Else	if $M = N$ then P else $Q \rightarrow Q \quad (M \neq N)$

The applied π-calculus on an example

Syntax:

- Equational theory: $dec(enc(x, y), y) = x$
- Process:

$$P = \nu s, k.(\text{out}(c_1, enc(s, k)) \mid \text{in}(c_1, y).\text{out}(c_2, dec(y, k))).$$

Special processes: active substitutions $P \mid \{^M/_x\}$

Semantics:

- Operational semantics $\rightarrow$: closed by structural equivalence ($\equiv$) and application of evaluation contexts such that

Comm	$\text{out}(a, x).P \mid \text{in}(a, x).Q \rightarrow P \mid Q$
Then	if $M = M$ then P else $Q \rightarrow P$
Else	if $M = N$ then P else $Q \rightarrow Q \quad (M \neq N)$

Example: $P \rightarrow \nu s, k.\text{out}(c_2, s)$

The applied π-calculus on an example (2)

- Labeled operational semantics $\xrightarrow{\alpha}$

  Labelled transitions where α is either $\mathsf{in}(c, M)$, $(\nu c').\mathsf{out}(c, c')$ or $(\nu x.)\mathsf{out}(c, x)$

  Example:

  $$\nu a, \nu k.out(c, enc(a, k)).P \xrightarrow{\nu x.out(c,x)} P \mid \{^{enc(a,k)}/_{x}\}$$

  Allows processes to communicate with an unspecified environment

  Output is done by reference and creates active substitutions

The applied π-calculus on an example (2)

- Labeled operational semantics $\xrightarrow{\alpha}$

  Labelled transitions where α is either $\mathsf{in}(c, M)$, $(\nu c').\mathsf{out}(c, c')$ or $(\nu x.)\mathsf{out}(c, x)$

  Example:

  $$\nu a, \nu k.out(c, enc(a, k)).P \xrightarrow{\nu x.out(c,x)} P \mid \{^{enc(a,k)}/_{x}\}$$

  Allows processes to communicate with an unspecified environment

  Output is done by reference and creates active substitutions

- Frames

  The frame of a process $\phi(A)$ is build from the process restrictions and active substitutions

  Approximation of the process accounting for the static knowledge exposed to the environment

Equivalences and preorders in applied pi

Definition (Observational equivalence and preorder)

Observational preorder ($\preceq$) (resp. equivalence ($\approx$)) is the largest (symmetric for $\approx$) relation on extended processes of same domain such that $A \mathcal{R} B$ implies

1. if $A \Downarrow a$ then $B \Downarrow a$;
2. if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B';
3. $C[A] \mathcal{R} C[B]$ for all closing evaluation contexts $C[\_]$.

Labelled (bi)simulation

Definition (Labelled (bi)similarity)

A relation $\mathcal{R}$ on closed extended processes is a simulation relation if $A \mathcal{R} B$ implies

1. $\phi(A) \approx_s \phi(B)$,
2. if $A \to A'$, then $B \to^* B'$ and $A' \mathcal{R} B'$ for some B',
3. if $A \xrightarrow{\alpha} A'$ and $fv(\alpha) \subseteq \mathrm{dom}(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$, then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A' \mathcal{R} B'$ for some B'.

If $\mathcal{R}$ and $\mathcal{R}^{-1}$ are both simulations $\mathcal{R}$ is a bisimulation. Labelled (bi)similarity ($\preceq_\ell$,$\approx_\ell$), is the largest (bi)simulation.

Equivalence, preorder and (bi)simulation

Observational preorder and labelled bisimulation were not defined in [AF01], but are natural.

In [AF01]: $\approx_\ell = \approx$

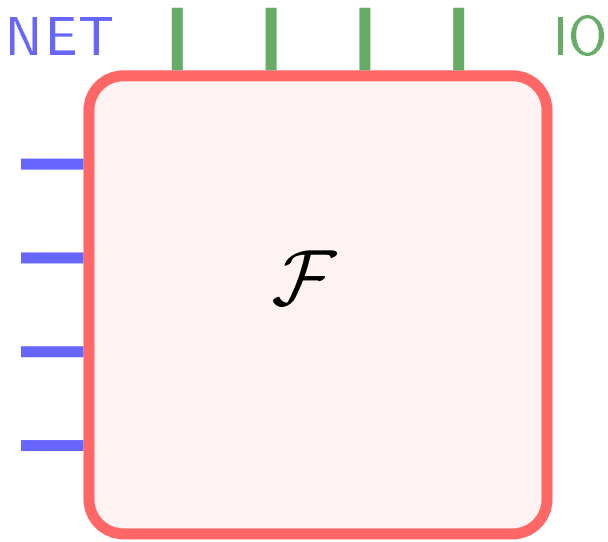
Proposition

If $A \preceq_\ell B$ *then* $C[A] \preceq_\ell C[B]$ *for any closing evaluation context* $C[\_]$.

It follows that $\preceq_\ell \subseteq \preceq$.

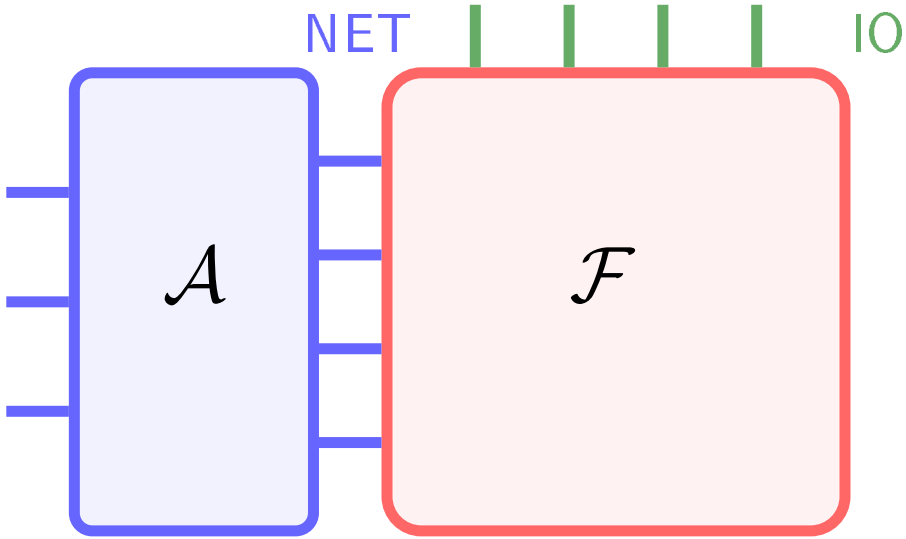
We actually conjecture that $\preceq_\ell = \preceq$ but not needed in this work.

Basic definitions



A functionality $\mathcal{F}$ is a closed plain process with channel names in $\mathsf{IO} \cup \mathsf{NET}$

Basic definitions

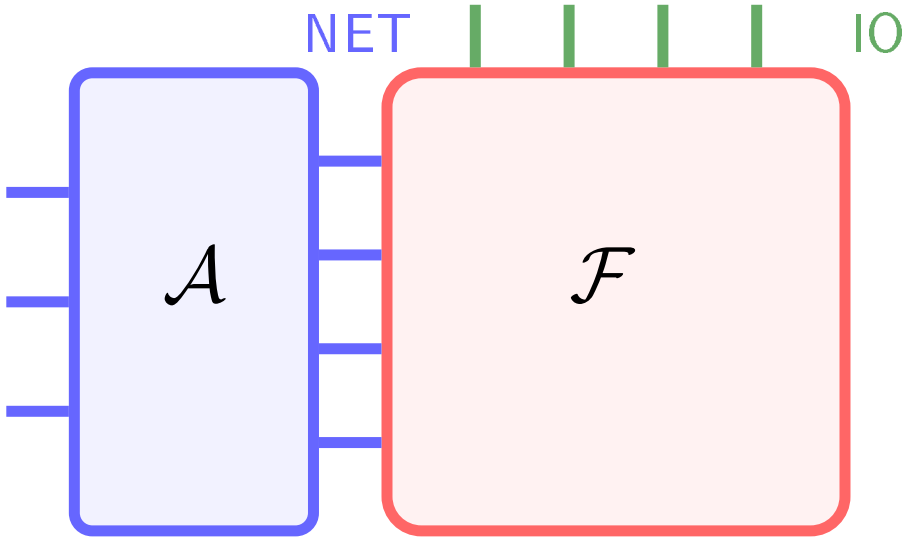


An adversary for $\mathcal{F}$ is an evaluation context $\mathcal{A}[\_]$ of the form:

$\nu\widetilde{net}_1.(A_1 \mid \nu\widetilde{net}_2.(A_2 \mid \ldots \mid \nu\widetilde{net}_k.(A_k \mid \_)\ldots))$

with $\mathsf{fnet}(\mathcal{F}) \subseteq \bigcup_{1\leq i\leq k} \widetilde{net}_i \subseteq \mathsf{NET}$, A_i $(1 \leq i \leq k)$ closed plain processes and $fn(\mathcal{A}[\_]) \cap \mathsf{IO} = \emptyset$.

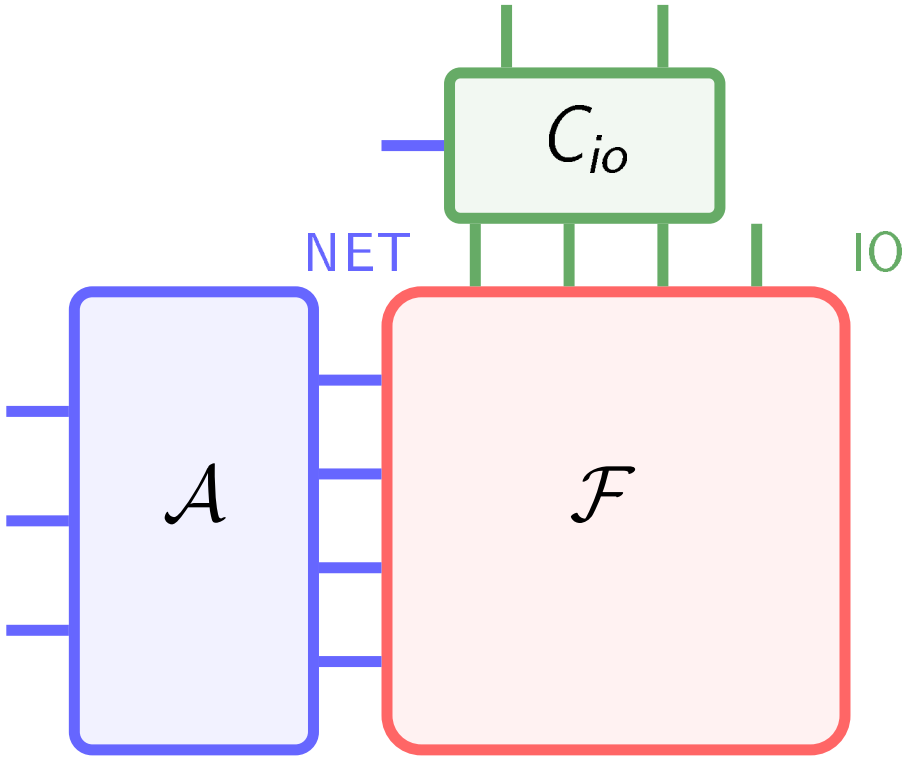
Basic definitions



Remarks:

- If $\mathcal{A}_1[\_]$ is an adversary for $\mathcal{F}$ then $\mathcal{A}_1[\mathcal{F}]$ is a functionality.
- If $\mathcal{A}_2[\_]$ is an adversary for $\mathcal{A}_1[\mathcal{F}]$, then $\mathcal{A}_2[\mathcal{A}_1[\_]]$ is an adversary for $\mathcal{F}$.

Basic definitions



An IO context is an evaluation context $C_{io}[\_]$ of the form

$\nu \widetilde{io}_1.(C_1 \mid \nu \widetilde{io}_2.(C_2 \mid \ldots \mid \nu \widetilde{io}_k.(C_k \mid \_)\ldots))$

with $\bigcup_{1 \leq i \leq k} \widetilde{io}_i \subseteq \mathsf{IO}$, C_i $(1 \leq i \leq k)$ closed plain processes.

Strong simulatability

Definition (Strong simulatability)

$$\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2$$
$$\text{iff}$$
$$\exists \mathcal{S}.\ \mathcal{F}_1 \preceq \mathcal{S}[\mathcal{F}_2] \wedge \mathsf{fnet}(\mathcal{F}_1) = \mathsf{fnet}(\mathcal{S}(\mathcal{F}_2))$$

First idea was to use $\approx$ instead of $\preceq$ but the notion is too strong (the simulator is not "invisible")

Properties of $\leq^{\mathsf{SS}}$

Proposition (Preorder)

- *Rerflexivity:* $\mathcal{F} \leq^{\mathsf{SS}} \mathcal{F}$
- *Transitivity: If* $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2$ *and* $\mathcal{F}_2 \leq^{\mathsf{SS}} \mathcal{F}_3$ *then* $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_3$

Properties of $\leq^{\mathsf{SS}}$

Proposition (Preorder)

- *Rerflexivity:* $\mathcal{F} \leq^{\mathsf{SS}} \mathcal{F}$
- *Transitivity:* *If* $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2$ *and* $\mathcal{F}_2 \leq^{\mathsf{SS}} \mathcal{F}_3$ *then* $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_3$

Proposition (Closure under application of IO contexts)

$$\textit{If } \mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2 \textit{ then } C_{io}[\mathcal{F}_1] \leq^{\mathsf{SS}} C_{io}[\mathcal{F}_2]$$

Properties of $\leq^{\mathsf{SS}}$

Proposition (Preorder)

- *Rerflexivity:* $\mathcal{F} \leq^{\mathsf{SS}} \mathcal{F}$
- *Transitivity:* *If* $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2$ *and* $\mathcal{F}_2 \leq^{\mathsf{SS}} \mathcal{F}_3$ *then* $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_3$

Proposition (Closure under application of IO contexts)

$$\textit{If } \mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2 \textit{ then } C_{io}[\mathcal{F}_1] \leq^{\mathsf{SS}} C_{io}[\mathcal{F}_2]$$

Proposition (Composition of functionalities)

- *If* $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2$ *then* $\mathcal{F}_1 \mid \mathcal{F}_3 \leq^{\mathsf{SS}} \mathcal{F}_2 \mid \mathcal{F}_3$
- *If* $\mathcal{F}_1 \leq^{\mathsf{SS}} \mathcal{F}_2$ *then* $!\mathcal{F}_1 \leq^{\mathsf{SS}} !\mathcal{F}_2$

The simulatability zoo

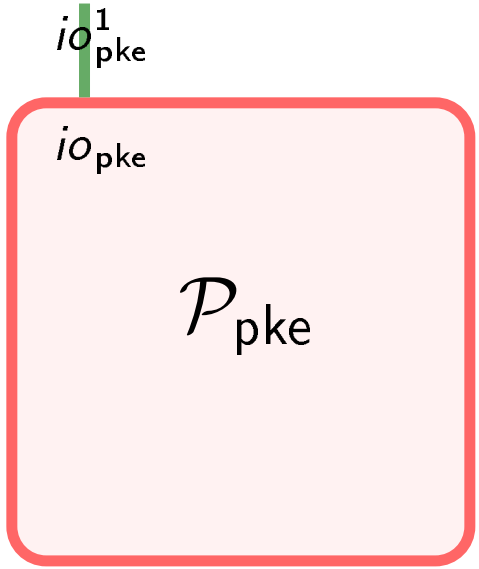
- Strong ($\leq^{\mathsf{SS}}$): $\exists S.\ \mathcal{F}_1 \preceq S[\mathcal{F}_2]$
- Black box ($\leq^{\mathsf{BB}}$): $\exists S.\ \forall A.\ A[\mathcal{F}_1] \preceq A[S[\mathcal{F}_2]]$
- Universally composable ($\leq^{\mathsf{UC}}$): $\forall A.\ \exists S.\ A[\mathcal{F}_1] \preceq S[\mathcal{F}_2]$
- UC with dummy adversary ($\leq^{\mathsf{UCDA}}$): $\exists S.\ D[\mathcal{F}_1] \preceq S[\mathcal{F}_2]$

Theorem

$$\leq^{\mathsf{SS}} = \leq^{\mathsf{BB}} = \leq^{\mathsf{UC}} = \leq^{\mathsf{UCDA}}$$

(Does not hold in all frameworks!)

Applications: Asymmetric encryption functionality

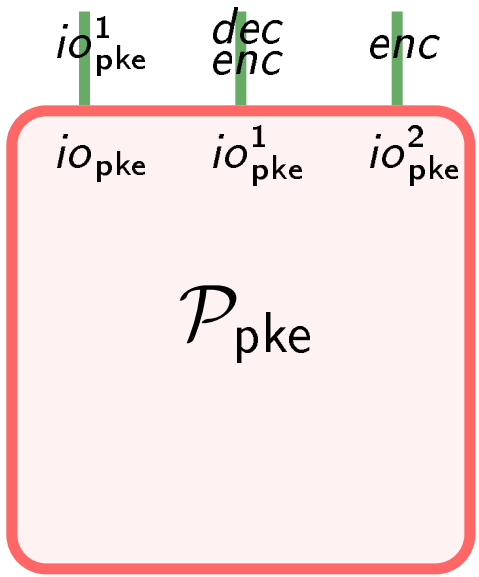


$$\mathcal{P}_{\mathsf{pke}} := \mathsf{in}(io_{\mathsf{pke}}, io^1_{\mathsf{pke}}).\nu sk.\mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{key}, \mathsf{pk}(sk)\rangle).$$
$$(\mathsf{let}\ io^i_{\mathsf{pke}} = io^1_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{P}_{\mathsf{enc}} \mid \mathsf{let}\ io^i_{\mathsf{pke}} = io^2_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{P}_{\mathsf{enc}} \mid !\mathcal{P}_{\mathsf{dec}})$$

$$\mathcal{P}_{\mathsf{enc}} := \mathsf{in}(io^i_{\mathsf{pke}}, \langle = \mathsf{enc}, m\rangle).$$
$$\nu r_2.\ \mathsf{let}\ menc = \mathsf{aenc}(\langle \mathsf{tag}_0, m\rangle, \mathsf{pk}(sk), r_2)\ \mathsf{in}\ \mathsf{out}(io^i_{\mathsf{pke}}, \langle \mathsf{cipher}, menc\rangle)$$

$$\mathcal{P}_{\mathsf{dec}} := \mathsf{in}(io^1_{\mathsf{pke}}, \langle = \mathsf{dec}, m\rangle).$$
$$\mathsf{let}\ \langle = \mathsf{tag}_0, m_1\rangle = \mathsf{adec}(m, sk)\ \mathsf{in}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, m\rangle)$$

Applications: Asymmetric encryption functionality

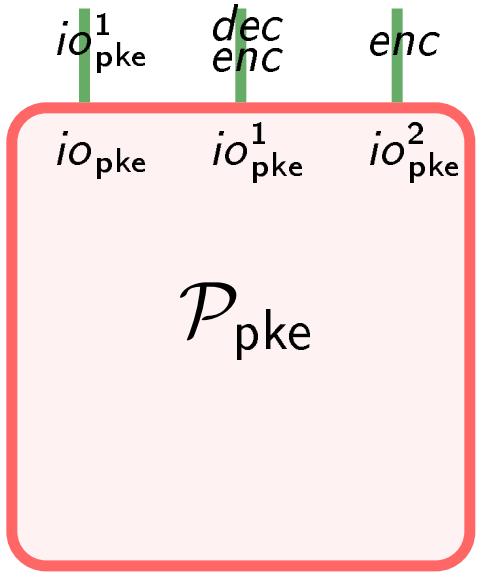


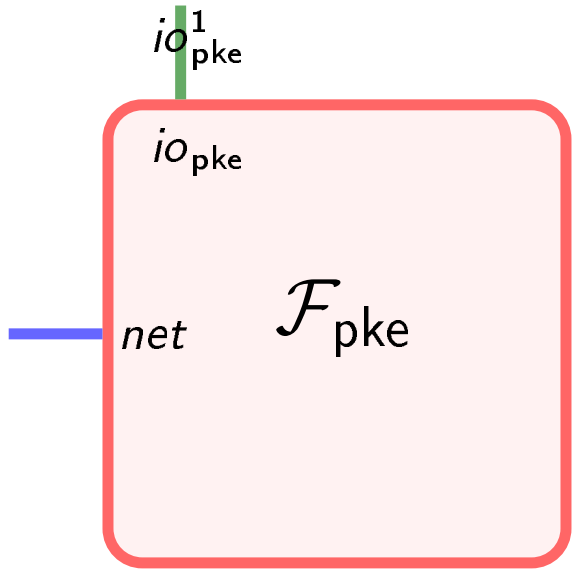
$$\mathcal{P}_{\mathsf{pke}} := \mathsf{in}(io_{\mathsf{pke}}, io^1_{\mathsf{pke}}).\nu sk.\mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{key}, \mathsf{pk}(sk)\rangle).$$
$$(\mathsf{let}\ io^i_{\mathsf{pke}} = io^1_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{P}_{\mathsf{enc}} \mid \mathsf{let}\ io^i_{\mathsf{pke}} = io^2_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{P}_{\mathsf{enc}} \mid !\mathcal{P}_{\mathsf{dec}})$$

$$\mathcal{P}_{\mathsf{enc}} := \mathsf{in}(io^i_{\mathsf{pke}}, \langle = \mathsf{enc}, m\rangle).$$
$$\nu r_2.\ \mathsf{let}\ menc = \mathsf{aenc}(\langle \mathsf{tag}_0, m\rangle, \mathsf{pk}(sk), r_2)\ \mathsf{in}\ \mathsf{out}(io^i_{\mathsf{pke}}, \langle \mathsf{cipher}, menc\rangle)$$

$$\mathcal{P}_{\mathsf{dec}} := \mathsf{in}(io^1_{\mathsf{pke}}, \langle = \mathsf{dec}, m\rangle).$$
$$\mathsf{let}\ \langle = \mathsf{tag}_0, m_1\rangle = \mathsf{adec}(m, sk)\ \mathsf{in}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, m\rangle)$$

Applications: Asymmetric encryption functionality



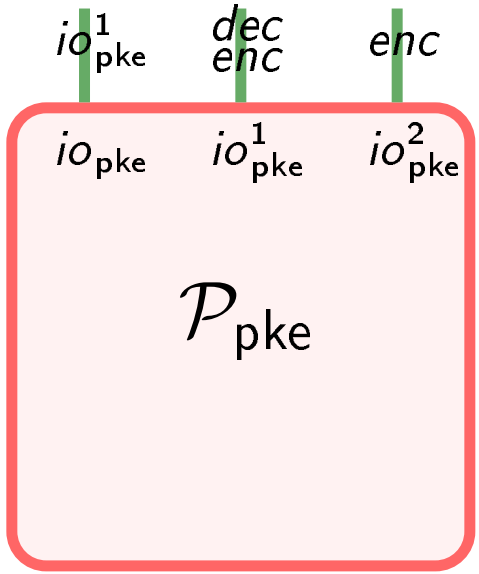


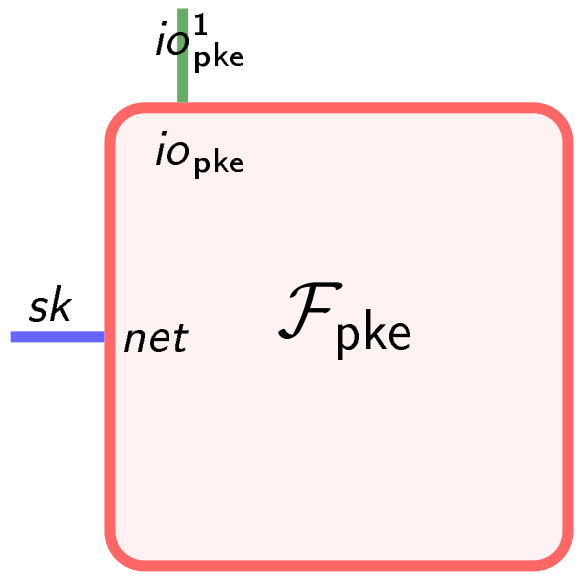
$$\mathcal{F}_{\mathsf{pke}} := \mathsf{in}(io_{\mathsf{pke}}, io^1_{\mathsf{pke}}).\mathsf{out}(net, \mathsf{init}).\mathsf{in}(net, \langle = \mathsf{algo}, sk, tag\rangle).\mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{key}, \mathsf{pk}(sk)\rangle).$$
$$\nu ssk.\ (\mathsf{let}\ io^i_{\mathsf{pke}} = io^1_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{F}_{\mathsf{enc}} \mid \mathsf{let}\ io^i_{\mathsf{pke}} = io^2_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{F}_{\mathsf{enc}} \mid !\mathcal{F}_{\mathsf{dec}})$$

$$\mathcal{F}_{\mathsf{enc}} := \mathsf{in}(io^i_{\mathsf{pke}}, \langle = \mathsf{enc}, m\rangle).\nu r_1.\nu r_2.$$
$$\mathsf{let}\ alea = \mathsf{aenc}(m, \mathsf{pk}(ssk), r_1)\ \mathsf{in}\ \mathsf{let}\ menc = \mathsf{aenc}(\langle tag, alea\rangle, \mathsf{pk}(sk), r_2)\ \mathsf{in}$$
$$\mathsf{out}(io^i_{\mathsf{pke}}, \langle \mathsf{cipher}, menc\rangle)$$

$$\mathcal{F}_{\mathsf{dec}} := \mathsf{in}(io^1_{\mathsf{pke}}, \langle = \mathsf{dec}, m\rangle).\ \mathsf{let}\ \langle = tag, m_1\rangle = \mathsf{adec}(m, sk)\ \mathsf{in}$$
$$\mathsf{if}\ \mathsf{testdec}(m_1, ssk) = \mathsf{ok}\ \mathsf{then}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, \mathsf{adec}(m_1, ssk)\rangle)$$
$$\mathsf{else}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, m_1\rangle)$$

Applications: Asymmetric encryption functionality



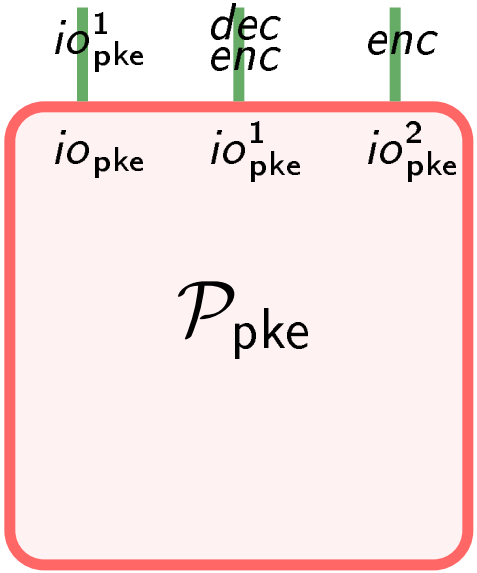


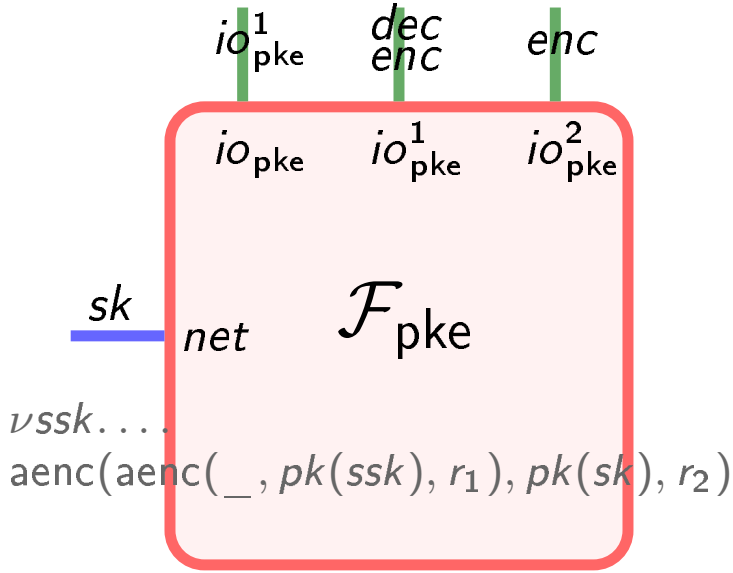
$$\mathcal{F}_{\mathsf{pke}} := \mathsf{in}(io_{\mathsf{pke}}, io^1_{\mathsf{pke}}).\mathsf{out}(net, \mathsf{init}).\mathsf{in}(net, \langle = \mathsf{algo}, sk, tag\rangle).\mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{key}, \mathsf{pk}(sk)\rangle).$$
$$\nu ssk.\ (\mathsf{let}\ io^i_{\mathsf{pke}} = io^1_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{F}_{\mathsf{enc}} \mid \mathsf{let}\ io^i_{\mathsf{pke}} = io^2_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{F}_{\mathsf{enc}} \mid !\mathcal{F}_{\mathsf{dec}})$$

$$\mathcal{F}_{\mathsf{enc}} := \mathsf{in}(io^i_{\mathsf{pke}}, \langle = \mathsf{enc}, m\rangle).\nu r_1.\nu r_2.$$
$$\mathsf{let}\ alea = \mathsf{aenc}(m, \mathsf{pk}(ssk), r_1)\ \mathsf{in}\ \mathsf{let}\ menc = \mathsf{aenc}(\langle tag, alea\rangle, \mathsf{pk}(sk), r_2)\ \mathsf{in}$$
$$\mathsf{out}(io^i_{\mathsf{pke}}, \langle \mathsf{cipher}, menc\rangle)$$

$$\mathcal{F}_{\mathsf{dec}} := \mathsf{in}(io^1_{\mathsf{pke}}, \langle = \mathsf{dec}, m\rangle).\ \mathsf{let}\ \langle = tag, m_1\rangle = \mathsf{adec}(m, sk)\ \mathsf{in}$$
$$\mathsf{if}\ \mathsf{testdec}(m_1, ssk) = \mathsf{ok}\ \mathsf{then}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, \mathsf{adec}(m_1, ssk)\rangle)$$
$$\mathsf{else}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, m_1\rangle)$$

Applications: Asymmetric encryption functionality



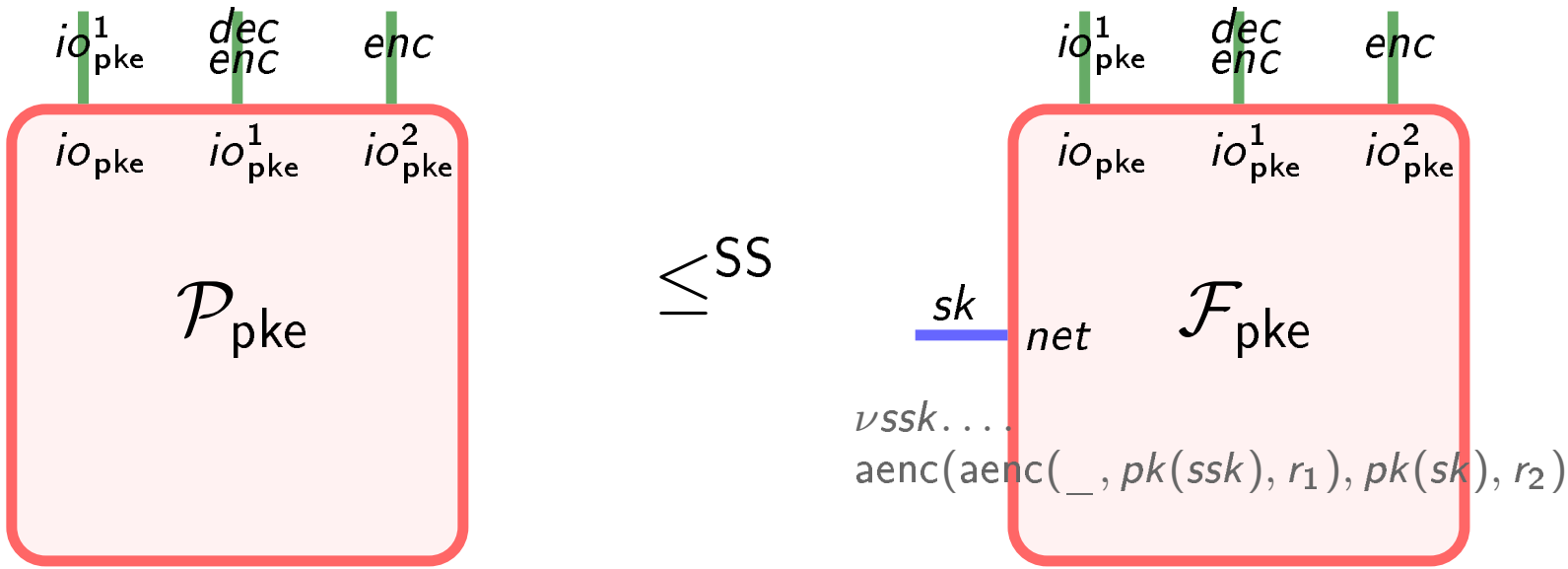


$$\mathcal{F}_{\mathsf{pke}} := \mathsf{in}(io_{\mathsf{pke}}, io^1_{\mathsf{pke}}).\mathsf{out}(net, \mathsf{init}).\mathsf{in}(net, \langle = \mathsf{algo}, sk, tag\rangle).\mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{key}, \mathsf{pk}(sk)\rangle).$$
$$\nu ssk.\ (\mathsf{let}\ io^i_{\mathsf{pke}} = io^1_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{F}_{\mathsf{enc}}\ \mid\ \mathsf{let}\ io^i_{\mathsf{pke}} = io^2_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{F}_{\mathsf{enc}}\ \mid !\mathcal{F}_{\mathsf{dec}})$$

$$\mathcal{F}_{\mathsf{enc}} := \mathsf{in}(io^i_{\mathsf{pke}}, \langle = \mathsf{enc}, m\rangle).\nu r_1.\nu r_2.$$
$$\mathsf{let}\ alea = \mathsf{aenc}(m, \mathsf{pk}(ssk), r_1)\ \mathsf{in}\ \mathsf{let}\ menc = \mathsf{aenc}(\langle tag, alea\rangle, \mathsf{pk}(sk), r_2)\ \mathsf{in}$$
$$\mathsf{out}(io^i_{\mathsf{pke}}, \langle \mathsf{cipher}, menc\rangle)$$

$$\mathcal{F}_{\mathsf{dec}} := \mathsf{in}(io^1_{\mathsf{pke}}, \langle = \mathsf{dec}, m\rangle).\ \mathsf{let}\ \langle = tag, m_1\rangle = \mathsf{adec}(m, sk)\ \mathsf{in}$$
$$\mathsf{if}\ \mathsf{testdec}(m_1, ssk) = \mathsf{ok}\ \mathsf{then}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, \mathsf{adec}(m_1, ssk)\rangle)$$
$$\mathsf{else}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, m_1\rangle)$$

Applications: Asymmetric encryption functionality



$$\mathcal{F}_{\mathsf{pke}} := \mathsf{in}(io_{\mathsf{pke}}, io^1_{\mathsf{pke}}).\mathsf{out}(net, \mathsf{init}).\mathsf{in}(net, \langle = \mathsf{algo}, sk, tag\rangle).\mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{key}, \mathsf{pk}(sk)\rangle).$$
$$\nu ssk.\ (\mathsf{let}\ io^i_{\mathsf{pke}} = io^1_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{F}_{\mathsf{enc}}\ \mid\ \mathsf{let}\ io^i_{\mathsf{pke}} = io^2_{\mathsf{pke}}\ \mathsf{in}\ !\mathcal{F}_{\mathsf{enc}}\ \mid !\mathcal{F}_{\mathsf{dec}})$$

$$\mathcal{F}_{\mathsf{enc}} := \mathsf{in}(io^i_{\mathsf{pke}}, \langle = \mathsf{enc}, m\rangle).\nu r_1.\nu r_2.$$
$$\mathsf{let}\ alea = \mathsf{aenc}(m, \mathsf{pk}(ssk), r_1)\ \mathsf{in}\ \mathsf{let}\ menc = \mathsf{aenc}(\langle tag, alea\rangle, \mathsf{pk}(sk), r_2)\ \mathsf{in}$$
$$\mathsf{out}(io^i_{\mathsf{pke}}, \langle \mathsf{cipher}, menc\rangle)$$

$$\mathcal{F}_{\mathsf{dec}} := \mathsf{in}(io^1_{\mathsf{pke}}, \langle = \mathsf{dec}, m\rangle).\ \mathsf{let}\ \langle = tag, m_1\rangle = \mathsf{adec}(m, sk)\ \mathsf{in}$$
$$\mathsf{if}\ \mathsf{testdec}(m_1, ssk) = \mathsf{ok}\ \mathsf{then}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, \mathsf{adec}(m_1, ssk)\rangle)$$
$$\mathsf{else}\ \mathsf{out}(io^1_{\mathsf{pke}}, \langle \mathsf{plain}, m_1\rangle)$$

Applications: Asymmetric encryption with joint state

We have that

$$\mathcal{P}_1 \leq^{\mathsf{SS}} \mathcal{P}_2 \Longrightarrow !\mathcal{P}_1 \leq^{\mathsf{SS}} !\mathcal{P}_2$$

However: if $\mathcal{P}_1$ uses $\mathcal{F}_{\mathsf{pke}}$ different instances (and different keys) are used in $!\mathcal{P}_1$

We show a joint state result:

$$\begin{array}{ccc} \mathcal{P}_{\mathsf{js}}[\mathcal{F}_{\mathsf{pke}}] & \leq^{\mathsf{SS}} & !\underline{\mathcal{F}_{\mathsf{pke}}} \\ \mathsf{SS}\ \vee| & & \mathsf{SS}\ \vee| \\ \mathcal{P}_{\mathsf{js}}[\mathcal{P}_{\mathsf{pke}}] & \not\leq^{\mathsf{SS}} & !\underline{\mathcal{P}_{\mathsf{pke}}} \end{array}$$

Ingredients: use of session tags + simulator chooses same secret key for each instance

Applications: Mutual authentication functionality

- Definition of an ideal mutual authentication functionality based on a trusted host
- Distributed implementation based on the NSL protocol
- Use joint state theorem to go "from one session to many"

Conclusion

- Nice framework for hierarchical reasoning in a Dolev-Yao like model
- Applications: asymmetric encryption, joint state, mutual authentication
- Ideal functionalities look appealing for complicated properties, e.g. electronic voting

S. Delaune, S. Kremer and O. Pereira. Simulation based security in the applied pi calculus. Research Report 2009/267, Cryptology ePrint Archive, 2009.