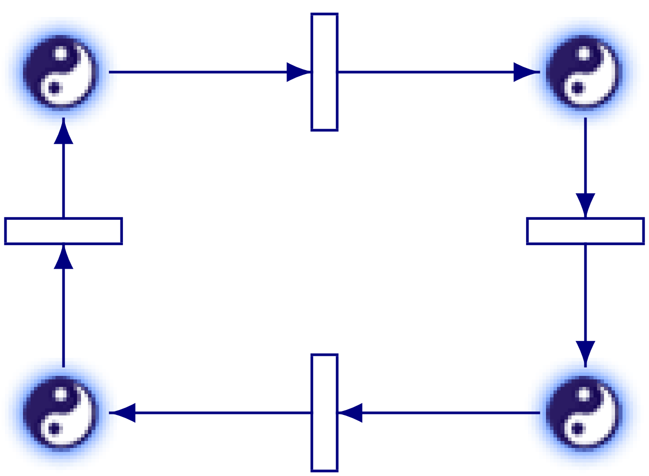
The Tao of PEPA nets

Stephen Gilmore

Joint work with

Jane Hillston, Leïla Kloul and Marina Ribaudo

13th August 2003



Background

Modern enterprise software design assumes a heterogeneous distributed system model where mobile objects are sent from host to host.

Background

Modern enterprise software design assumes a heterogeneous distributed system model where mobile objects are sent from host to host.

Systems are usually structured in this way to separate trusted participants with access rights from untrusted participants without access rights but also to achieve required levels of performance.

Background

Modern enterprise software design assumes a heterogeneous distributed system model where mobile objects are sent from host to host.

Systems are usually structured in this way to separate trusted participants with access rights from untrusted participants without access rights but also to achieve required levels of performance.

Several formalisms exist to analyse security in mobile code systems (Secure Ambients, Spi-calculus) but what about the performance analysis of such systems?

PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston's Performance Evaluation Process Algebra (PEPA).

PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston's Performance Evaluation Process Algebra (PEPA).

$$P ::= \underbrace{(\alpha, r).P}_{\text{prefix}}$$

PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston's Performance Evaluation Process Algebra (PEPA).

$$P ::= \underbrace{(\alpha, r).P}_{\text{prefix}} \mid \underbrace{P + P}_{\text{choice}}$$

PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston's Performance Evaluation Process Algebra (PEPA).

$$P ::= \underbrace{(\alpha, r).P}_{\text{prefix}} \mid \underbrace{P + P}_{\text{choice}} \mid \underbrace{P \underset{L}{\bowtie} P}_{\text{cooperation}}$$

PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston's Performance Evaluation Process Algebra (PEPA).

$$P ::= \underbrace{(\alpha, r).P}_{\text{prefix}} \mid \underbrace{P + P}_{\text{choice}} \mid \underbrace{P \underset{L}{\bowtie} P}_{\text{cooperation}} \mid \underbrace{P/L}_{\text{hiding}}$$

PEPA nets

The PEPA nets language is a high-level modelling formalism for performance analysis of mobile object systems.

A PEPA net is a stochastic Petri net with coloured tokens. The tokens represent mobile objects with state and behaviour.

The tokens are described using a stochastic process algebra, Jane Hillston's Performance Evaluation Process Algebra (PEPA).

$$P ::= \underbrace{(\alpha, r).P}_{\text{prefix}} \mid \underbrace{P + P}_{\text{choice}} \mid \underbrace{P \bowtie_{L} P}_{\text{cooperation}} \mid \underbrace{P/L}_{\text{hiding}} \mid \underbrace{X}_{\text{variable}}$$

Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

The order in which the methods can be applied defines a protocol for a `File` object.

Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

The order in which the methods can be applied defines a protocol for a `File` object.

Objects as tokens

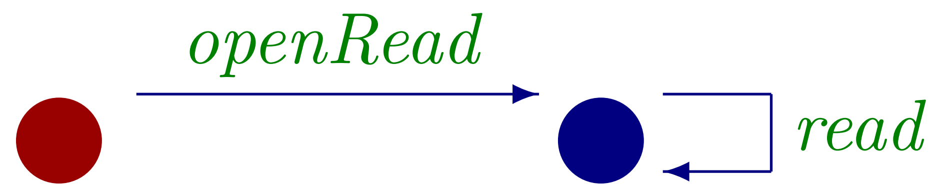
Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

The order in which the methods can be applied defines a protocol for a `File` object.

Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

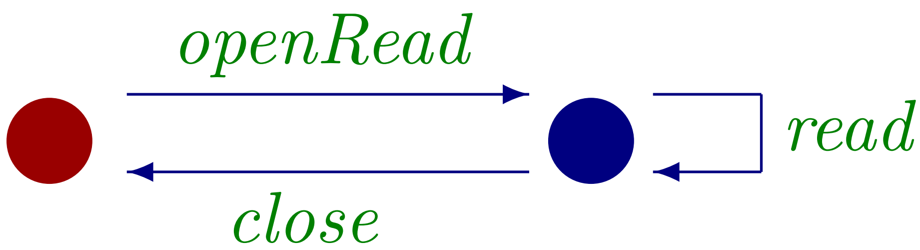
The order in which the methods can be applied defines a protocol for a `File` object.



Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

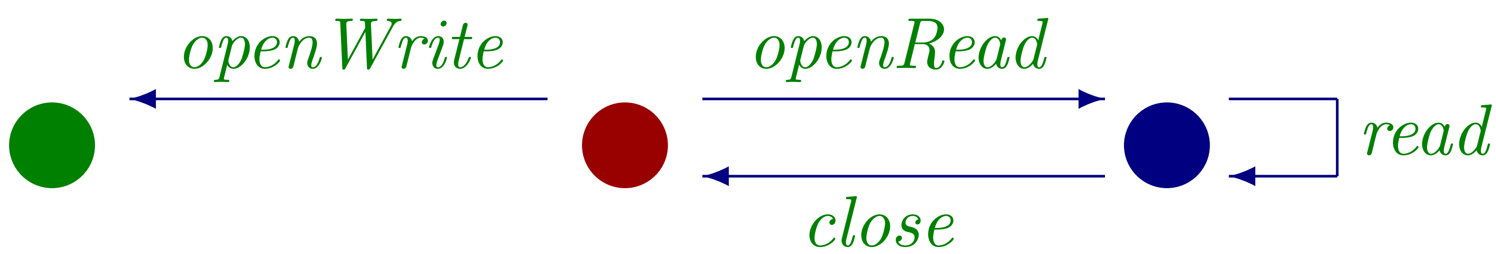
The order in which the methods can be applied defines a protocol for a `File` object.



Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

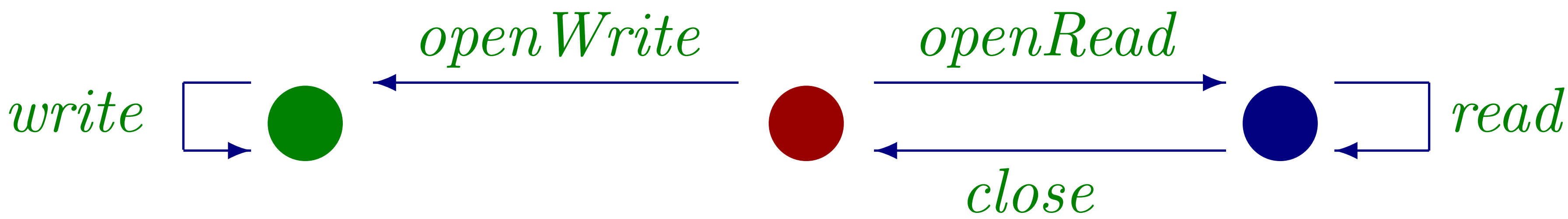
The order in which the methods can be applied defines a protocol for a `File` object.



Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

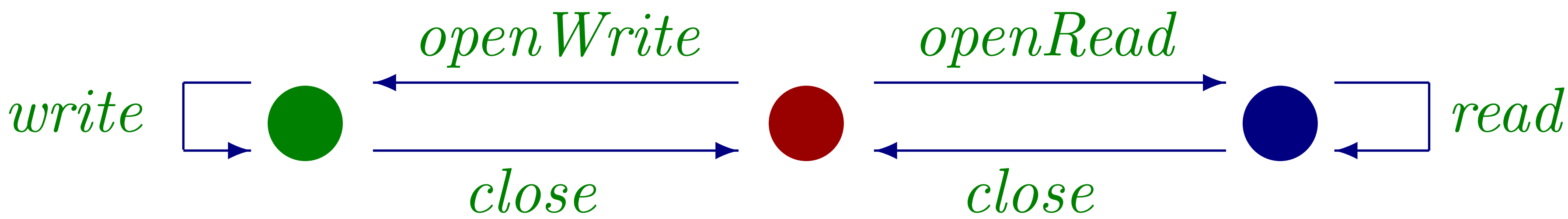
The order in which the methods can be applied defines a protocol for a `File` object.



Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

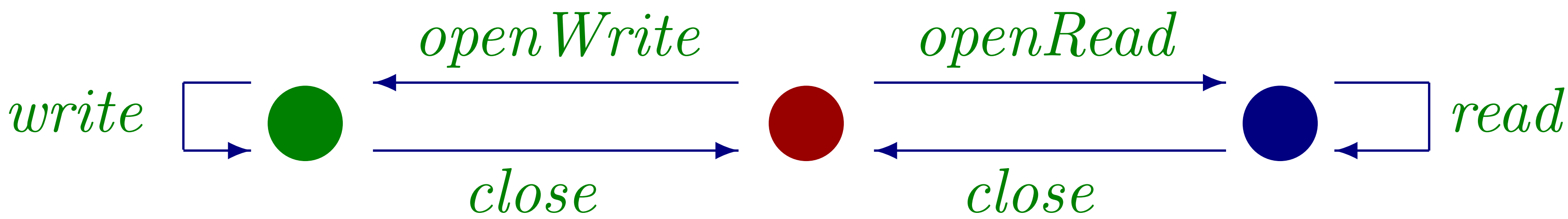
The order in which the methods can be applied defines a protocol for a `File` object.



Objects as tokens

Consider a `File` class with methods `openRead()`, `openWrite()`, `read()`, `write()` and `close()`.

The order in which the methods can be applied defines a protocol for a `File` object.



We can express this as a PEPA component.

$$
\begin{aligned}
File &\stackrel{def}{=} (openRead, r_o).InStream \\
&\quad + (openWrite, r_o).OutStream \\
InStream &\stackrel{def}{=} (read, r_r).InStream + (close, r_c).File \\
OutStream &\stackrel{def}{=} (write, r_w).OutStream + (close, r_c).File
\end{aligned}
$$

Contexts and cells

A PEPA net is made up of PEPA contexts, one at each place in the net.

Contexts and cells

A PEPA net is made up of PEPA contexts, one at each place in the net.

Contexts contain static components and cells, which store tokens.

Contexts and cells

A PEPA net is made up of PEPA contexts, one at each place in the net.

Contexts contain static components and cells, which store tokens.

A typical context might be the following:

$$File[\_] \underset{L}{\bowtie} FileReader$$

where the synchronisation set L in this case is $\vec{\mathcal{A}}(File)$, the complete action type set of the component, ($openRead$, $read$, $close$, ...).

Token movement

Tokens move by participating in firings of the net.

Token movement

Tokens move by participating in firings of the net.

Continuing our example, we introduce an instant message as a type of transmissible file.

$$InstantMessage \stackrel{def}{=} (\mathbf{transmit}, r_t).File$$

Token movement

Tokens move by participating in firings of the net.

Continuing our example, we introduce an instant message as a type of transmissible file.

$$InstantMessage \stackrel{def}{=} (\mathbf{transmit}, r_t).File$$

Part of a PEPA net which models the passage of instant messages is shown below.

$$\boxed{InstantMessage[IM]} \longrightarrow \overset{(\mathbf{transmit}, r_t)}{\boxed{}} \longrightarrow \boxed{File[\_\_\_] \underset{L}{\bowtie} FileReader}$$

Token movement

Tokens move by participating in firings of the net.

Continuing our example, we introduce an instant message as a type of transmissible file.

$$\mathit{InstantMessage} \stackrel{\mathit{def}}{=} (\mathbf{transmit}, r_t).\mathit{File}$$

Part of a PEPA net which models the passage of instant messages is shown below.

$$\boxed{\mathit{InstantMessage}[\_\_\_]} \longrightarrow \overset{(\mathbf{transmit}, r_t)}{\|} \longrightarrow \boxed{\mathit{File}[\mathit{File}] \underset{L}{\bowtie} \mathit{FileReader}}$$

Token movement

Tokens move by participating in firings of the net.

Continuing our example, we introduce an instant message as a type of transmissible file.

$$InstantMessage \stackrel{def}{=} (\mathbf{transmit}, r_t).File$$

Part of a PEPA net which models the passage of instant messages is shown below.

$$\boxed{InstantMessage[\_\_\_]} \xrightarrow{(\mathbf{transmit}, r_t)} \boxed{File[File] \underset{L}{\bowtie} FileReader}$$

An instant message IM can be moved by the **transmit** firing. In moving it changes state to a $File$ derivative, which can be read by the $FileReader$.

Semantics: Enabling Set

An enabling set is a set of (token, place) pairs.

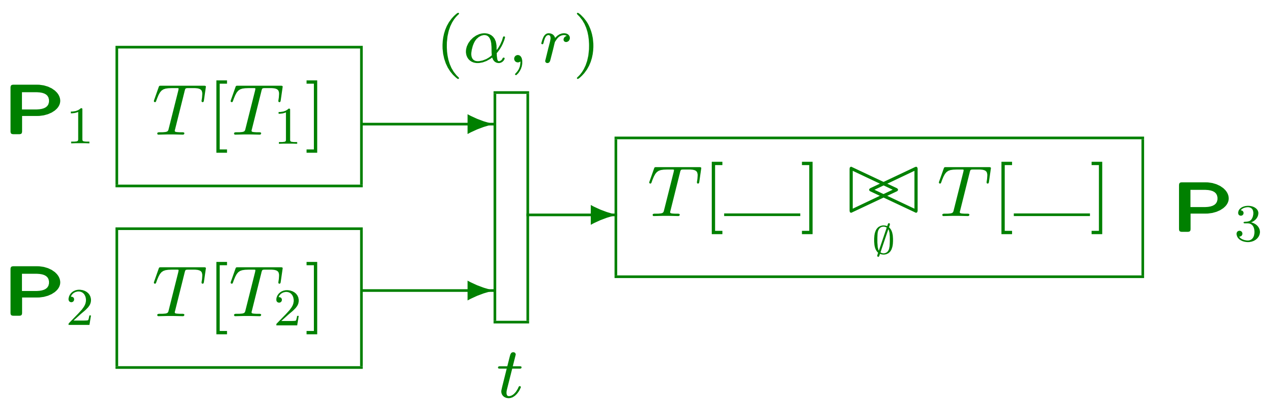
A transition t has an enabling set of firing type α, $\mathrm{ES}(t, \alpha)$, if for each input place $\mathbf{P}_i$ of t there is an element $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$ such that T is a token in the current marking of $\mathbf{P}_i$, which has a one-step α-derivative, T'.

Semantics: Enabling Set

An enabling set is a set of (token, place) pairs.

A transition t has an enabling set of firing type α, $\mathrm{ES}(t, \alpha)$, if for each input place $\mathbf{P}_i$ of t there is an element $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$ such that T is a token in the current marking of $\mathbf{P}_i$, which has a one-step α-derivative, T'.

Example:

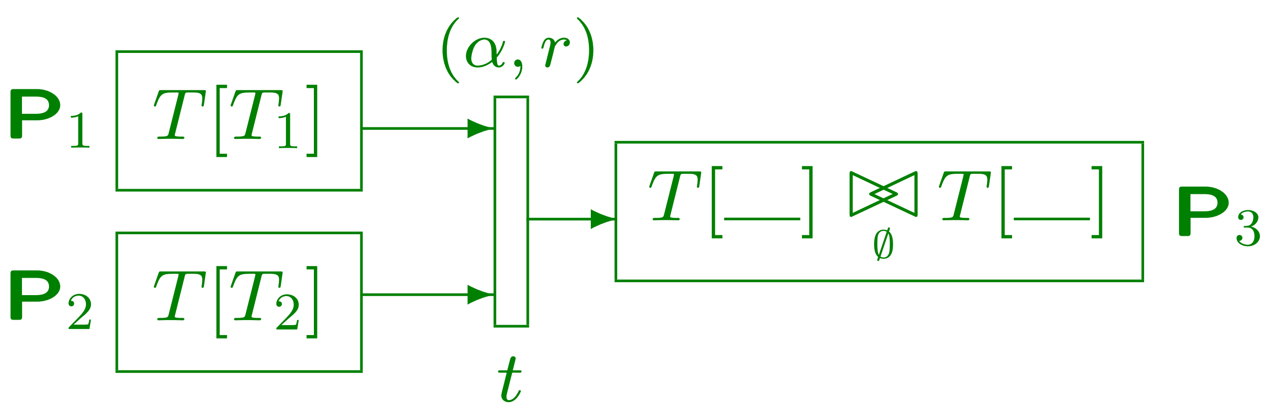


Semantics: Enabling Set

An enabling set is a set of (token, place) pairs.

A transition t has an enabling set of firing type α, $\mathrm{ES}(t, \alpha)$, if for each input place $\mathbf{P}_i$ of t there is an element $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$ such that T is a token in the current marking of $\mathbf{P}_i$, which has a one-step α-derivative, T'.

Example:



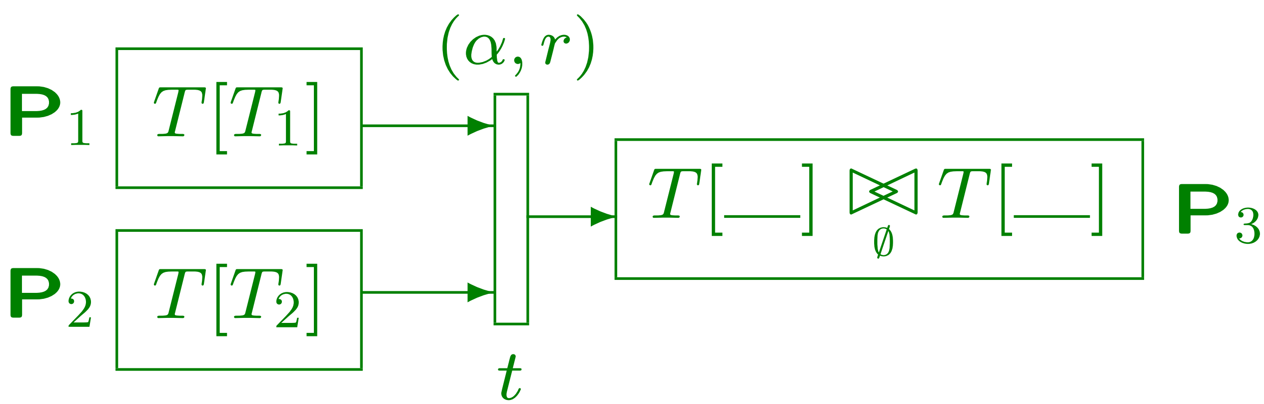
$T_1 \xrightarrow{(\alpha,r)} T_3$,

Semantics: Enabling Set

An enabling set is a set of (token, place) pairs.

A transition t has an enabling set of firing type α, $\mathrm{ES}(t, \alpha)$, if for each input place $\mathbf{P}_i$ of t there is an element $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$ such that T is a token in the current marking of $\mathbf{P}_i$, which has a one-step α-derivative, T'.

Example:



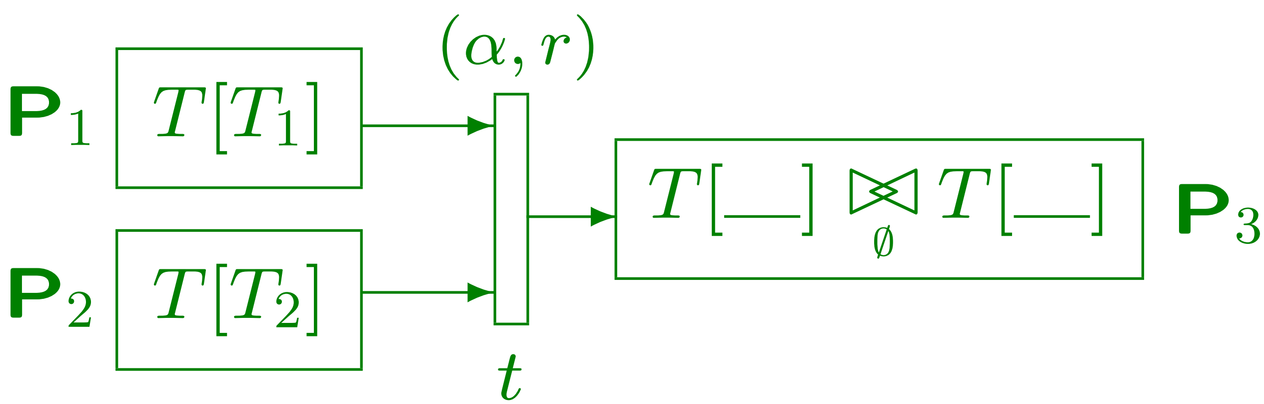
$T_1 \xrightarrow{(\alpha,r)} T_3$, $T_2 \xrightarrow{(\alpha,r)} T_4$,

Semantics: Enabling Set

An enabling set is a set of (token, place) pairs.

A transition t has an enabling set of firing type α, $\mathrm{ES}(t, \alpha)$, if for each input place $\mathbf{P}_i$ of t there is an element $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$ such that T is a token in the current marking of $\mathbf{P}_i$, which has a one-step α-derivative, T'.

Example:



$T_1 \xrightarrow{(\alpha,r)} T_3$, $T_2 \xrightarrow{(\alpha,r)} T_4$, $\mathrm{ES}(t, \alpha) = \{ (T_1, \mathbf{P}_1), (T_2, \mathbf{P}_2) \}$

Semantics: Enabling Rule

A transition t enables a firing of type α if there is an enabling set $\mathrm{ES}(t, \alpha)$ such that there is a surjective mapping ϕ from $\mathrm{ES}(t, \alpha)$ to vacant cells in the current markings of output places of t.

Semantics: Enabling Rule

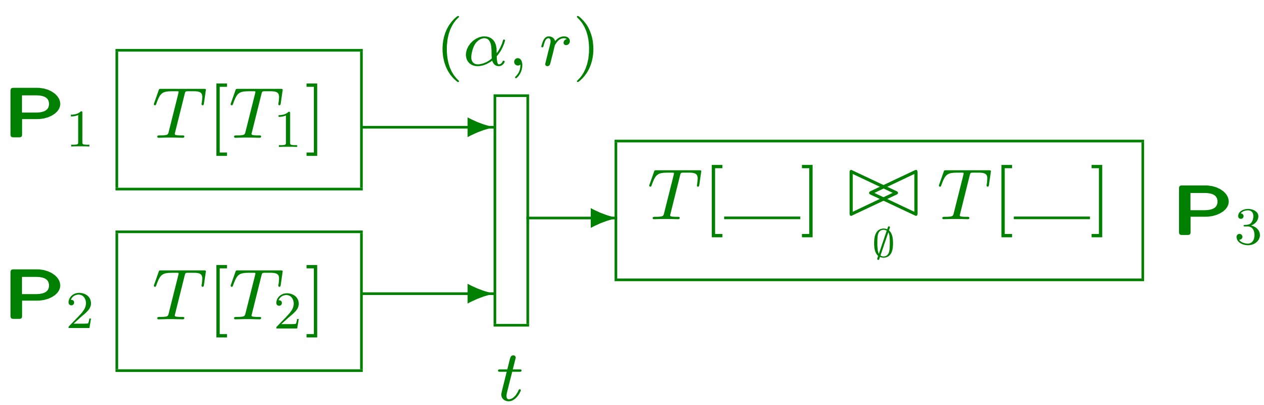
A transition t enables a firing of type α if there is an enabling set $\mathrm{ES}(t, \alpha)$ such that there is a surjective mapping ϕ from $\mathrm{ES}(t, \alpha)$ to vacant cells in the current markings of output places of t.

I.e. for each $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$ there is a distinct empty T-type cell in the current marking of one of the output places of t.

Semantics: Enabling Rule

A transition t enables a firing of type α if there is an enabling set $\mathrm{ES}(t, \alpha)$ such that there is a surjective mapping ϕ from $\mathrm{ES}(t, \alpha)$ to vacant cells in the current markings of output places of t.

I.e. for each $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$ there is a distinct empty T-type cell in the current marking of one of the output places of t.

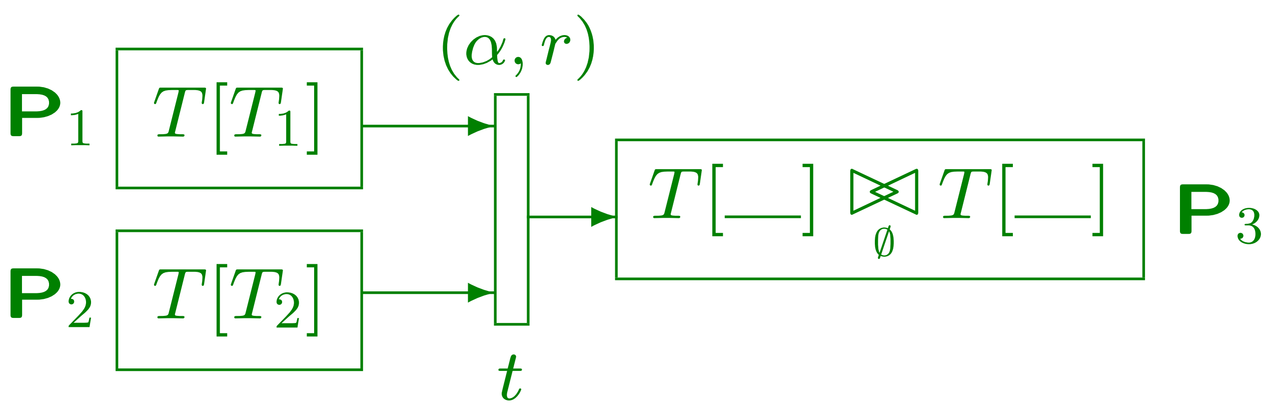


Semantics: Firing Rule

When a transition t fires with type α on the basis of the enabling set $\mathrm{ES}(t, \alpha)$, then for each $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$, $T[T]$ is replaced by $T[\_]$ in the marking of $\mathbf{P}_i$, and the current marking of each output place is updated according to ϕ.

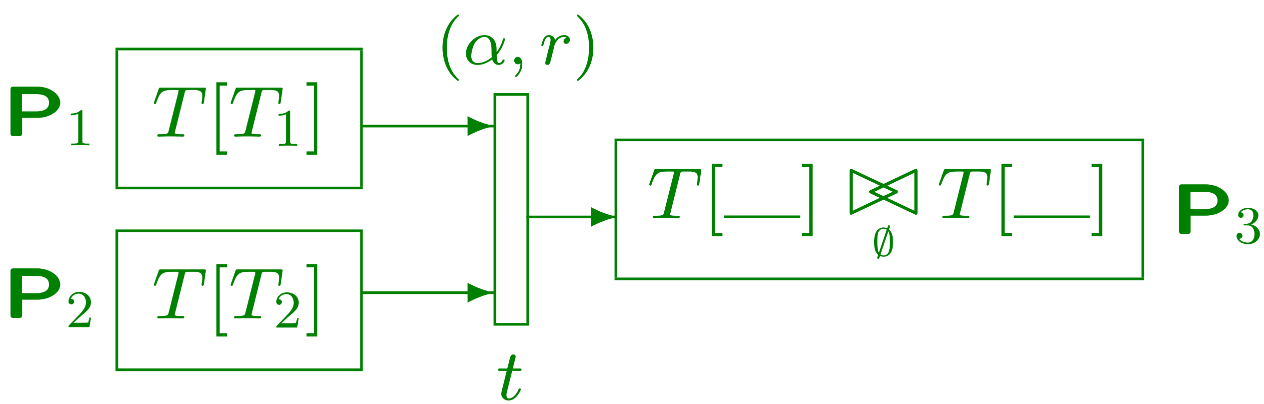
Semantics: Firing Rule

When a transition t fires with type α on the basis of the enabling set $\mathrm{ES}(t, \alpha)$, then for each $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$, $T[T]$ is replaced by $T[\_]$ in the marking of $\mathbf{P}_i$, and the current marking of each output place is updated according to ϕ.



Semantics: Firing Rule

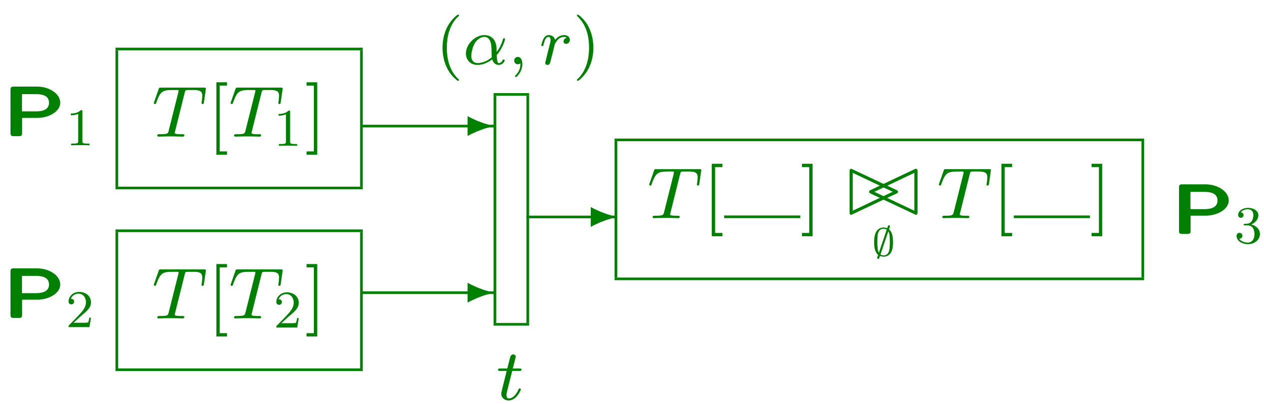
When a transition t fires with type α on the basis of the enabling set $\mathrm{ES}(t, \alpha)$, then for each $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t, \alpha)$, $T[T]$ is replaced by $T[\_]$ in the marking of $\mathbf{P}_i$, and the current marking of each output place is updated according to ϕ.



$T_1 \xrightarrow{(\alpha,r)} T_3$,

Semantics: Firing Rule

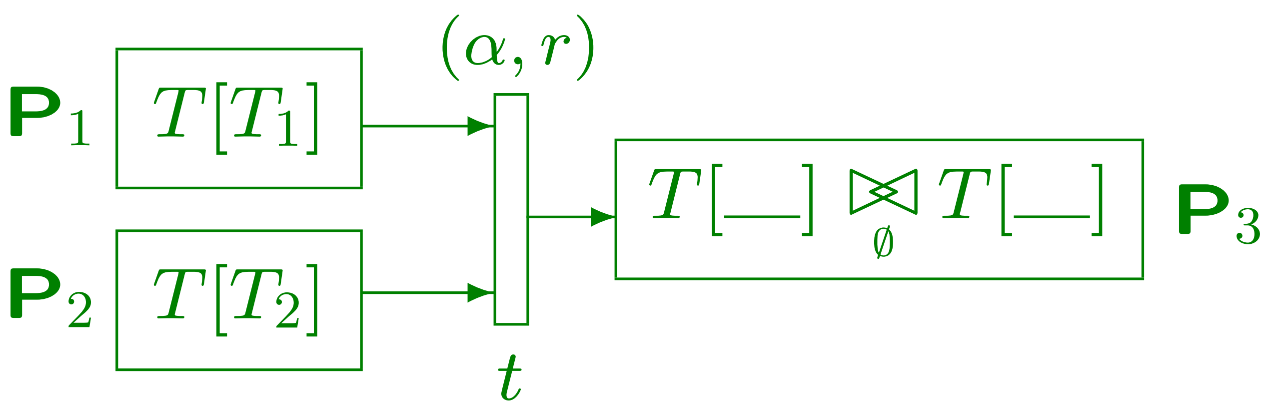
When a transition t fires with type α on the basis of the enabling set $\mathrm{ES}(t,\alpha)$, then for each $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t,\alpha)$, $T[T]$ is replaced by $T[\_]$ in the marking of $\mathbf{P}_i$, and the current marking of each output place is updated according to ϕ.



$T_1 \xrightarrow{(\alpha,r)} T_3$, $T_2 \xrightarrow{(\alpha,r)} T_4$,

Semantics: Firing Rule

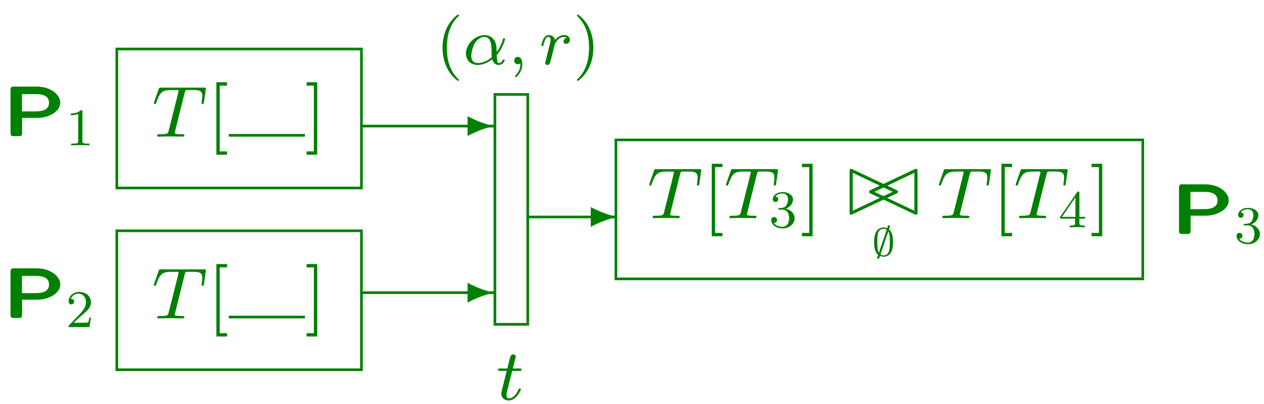
When a transition t fires with type α on the basis of the enabling set $\mathrm{ES}(t,\alpha)$, then for each $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t,\alpha)$, $T[T]$ is replaced by $T[\_]$ in the marking of $\mathbf{P}_i$, and the current marking of each output place is updated according to ϕ.



$$T_1 \xrightarrow{(\alpha,r)} T_3,\ T_2 \xrightarrow{(\alpha,r)} T_4,\ \mathrm{ES}(t,\alpha) = \{\, (T_1, \mathbf{P}_1), (T_2, \mathbf{P}_2) \,\}$$

Semantics: Firing Rule

When a transition t fires with type α on the basis of the enabling set $\mathrm{ES}(t,\alpha)$, then for each $(T, \mathbf{P}_i)$ in $\mathrm{ES}(t,\alpha)$, $T[T]$ is replaced by $T[\_]$ in the marking of $\mathbf{P}_i$, and the current marking of each output place is updated according to ϕ.



$$T_1 \xrightarrow{(\alpha,r)} T_3,\; T_2 \xrightarrow{(\alpha,r)} T_4,\; \mathrm{ES}(t,\alpha) = \{\, (T_1, \mathbf{P}_1), (T_2, \mathbf{P}_2) \,\}$$

Analysing PEPA nets

In the PEPA nets notation we have a modelling language which allows us to express performance models of mobile object systems.

The benefit of making such a model comes from the fact that we can gain insights into the system under study through the analysis of the model.

The state space of the model is typically too large to allow us to consider every state individually.

Many types of analysis (steady-state, transient, passage time) begin by identifying distinguished subsets of the state space.

Analysing PEPA nets

In the PEPA nets notation we have a modelling language which allows us to express performance models of mobile object systems.

The benefit of making such a model comes from the fact that we can gain insights into the system under study through the analysis of the model.

The state space of the model is typically too large to allow us to consider every state individually.

Many types of analysis (steady-state, transient, passage time) begin by identifying distinguished subsets of the state space.

Analysing PEPA nets

In the PEPA nets notation we have a modelling language which allows us to express performance models of mobile object systems.

The benefit of making such a model comes from the fact that we can gain insights into the system under study through the analysis of the model.

The state space of the model is typically too large to allow us to consider every state individually.

Many types of analysis (steady-state, transient, passage time) begin by identifying distinguished subsets of the state space.

Analysing PEPA nets

In the PEPA nets notation we have a modelling language which allows us to express performance models of mobile object systems.

The benefit of making such a model comes from the fact that we can gain insights into the system under study through the analysis of the model.

The state space of the model is typically too large to allow us to consider every state individually.

Many types of analysis (steady-state, transient, passage time) begin by identifying distinguished subsets of the state space.

Analysing PEPA nets

In the PEPA nets notation we have a modelling language which allows us to express performance models of mobile object systems.

The benefit of making such a model comes from the fact that we can gain insights into the system under study through the analysis of the model.

The state space of the model is typically too large to allow us to consider every state individually.

Many types of analysis (steady-state, transient, passage time) begin by identifying distinguished subsets of the state space.

Using logic to specify performance measures

The appropriate logic for PEPA nets is one which can specify performance measures over the places of the net, and has the capability of expressing requirements on tokens in addition to requirements on the transitions and firings of the net.

Using logic to specify performance measures

The appropriate logic for PEPA nets is one which can specify performance measures over the places of the net, and has the capability of expressing requirements on tokens in addition to requirements on the transitions and firings of the net.

We introduce the PML_ν logic by means of a two-level grammar which separates the specification of place formulae and token formulae from the specification of transition and firing activities.

Using logic to specify performance measures

The appropriate logic for PEPA nets is one which can specify performance measures over the places of the net, and has the capability of expressing requirements on tokens in addition to requirements on the transitions and firings of the net.

We introduce the PML_ν logic by means of a two-level grammar which separates the specification of place formulae and token formulae from the specification of transition and firing activities.

Behaviour at the transition and firing level is captured by formulae of a sub-logic, PML_μ.

PML$_\mu$

Based on probabilistic modal logic [Larsen & Skou].

$$\phi \quad ::= \quad \texttt{tt} \quad | \quad \neg\phi \quad | \quad \phi_1 \wedge \phi_2 \quad | \quad \nabla_\alpha \quad | \quad \langle\alpha\rangle_\rho \phi$$

PML$_\mu$

Based on probabilistic modal logic [Larsen & Skou].

$$\phi \quad ::= \quad \texttt{tt} \quad | \quad \neg\phi \quad | \quad \phi_1 \wedge \phi_2 \quad | \quad \nabla_\alpha \quad | \quad \langle\alpha\rangle_\rho\phi$$

$P \models_\mu \texttt{tt}$

PML$_\mu$

Based on probabilistic modal logic [Larsen & Skou].

$$\phi \quad ::= \quad \texttt{tt} \quad | \quad \neg\phi \quad | \quad \phi_1 \wedge \phi_2 \quad | \quad \nabla_\alpha \quad | \quad \langle\alpha\rangle_\rho\phi$$

$P \models_\mu \texttt{tt}$

$P \models_\mu \neg\phi$ iff $P \not\models_\mu \phi$

$\mathbf{PML}_\mu$

Based on probabilistic modal logic [Larsen & Skou].

$$\phi \quad ::= \quad \mathtt{tt} \quad | \quad \neg\phi \quad | \quad \phi_1 \wedge \phi_2 \quad | \quad \nabla_\alpha \quad | \quad \langle\alpha\rangle_\rho \phi$$

$P \models_\mu \mathtt{tt}$

$P \models_\mu \neg\phi$ iff $P \not\models_\mu \phi$

$P \models_\mu \phi_1 \wedge \phi_2$ iff $P \models_\mu \phi_1 \wedge P \models_\mu \phi_2$

PML$_\mu$

Based on probabilistic modal logic [Larsen & Skou].

$$\phi \quad ::= \quad \mathtt{tt} \quad | \quad \neg\phi \quad | \quad \phi_1 \wedge \phi_2 \quad | \quad \nabla_\alpha \quad | \quad \langle\alpha\rangle_\rho\phi$$

$$
\begin{array}{lll}
P \models_\mu \mathtt{tt} & & \\
P \models_\mu \neg\phi & \text{iff} & P \not\models_\mu \phi \\
P \models_\mu \phi_1 \wedge \phi_2 & \text{iff} & P \models_\mu \phi_1 \wedge P \models_\mu \phi_2 \\
P \models_\mu \nabla_\alpha & \text{iff} & P \stackrel{\alpha}{\not\longrightarrow}
\end{array}
$$

PML$_\mu$

Based on probabilistic modal logic [Larsen & Skou].

$$\phi \quad ::= \quad \mathtt{tt} \quad | \quad \neg\phi \quad | \quad \phi_1 \wedge \phi_2 \quad | \quad \nabla_\alpha \quad | \quad \langle\alpha\rangle_\rho\phi$$

$$
\begin{array}{lll}
P \models_\mu \mathtt{tt} & & \\
P \models_\mu \neg\phi & \text{iff} & P \not\models_\mu \phi \\
P \models_\mu \phi_1 \wedge \phi_2 & \text{iff} & P \models_\mu \phi_1 \wedge P \models_\mu \phi_2 \\
P \models_\mu \nabla_\alpha & \text{iff} & P \stackrel{\alpha}{\not\longrightarrow} \\
P \models_\mu \langle\alpha\rangle_\rho\phi & \text{iff} & P \stackrel{(\alpha,\lambda)}{\Longrightarrow} S \text{ for some } \lambda \geq \rho, \\
 & & \text{and for all } P' \in S, P' \models_\mu \phi.
\end{array}
$$

$\mathbf{PML}_\nu$

$$\psi \quad ::= \quad \phi \quad | \quad \neg\psi \quad | \quad \psi_1 \wedge \psi_2 \quad | \quad P_i[\phi] \quad | \quad P_i\#T_i \sim n$$

where $\sim\, = \{=, \neq, <, \leq, >, \geq\}$.

PML$_\nu$

$$\psi \quad ::= \quad \phi \quad | \quad \neg\psi \quad | \quad \psi_1 \wedge \psi_2 \quad | \quad P_i[\phi] \quad | \quad P_i \# T_i \sim n$$

where $\sim = \{=, \neq, <, \leq, >, \geq\}$.

$M \models_\nu \phi$ iff $M \models_\mu \phi$

PML$_\nu$

$$\psi \quad ::= \quad \phi \quad | \quad \neg\psi \quad | \quad \psi_1 \wedge \psi_2 \quad | \quad P_i[\phi] \quad | \quad P_i \# T_i \sim n$$

where $\sim = \{=, \neq, <, \leq, >, \geq\}$.

$$M \models_\nu \phi \qquad \text{iff} \quad M \models_\mu \phi$$

$$M \models_\nu \neg\psi \qquad \text{iff} \quad M \not\models_\nu \psi$$

PML$_\nu$

$$\psi \quad ::= \quad \phi \quad | \quad \neg\psi \quad | \quad \psi_1 \wedge \psi_2 \quad | \quad P_i[\phi] \quad | \quad P_i \# T_i \sim n$$

where $\sim \, = \{=, \neq, <, \leq, >, \geq\}$.

$$M \models_\nu \phi \quad \text{iff} \quad M \models_\mu \phi$$

$$M \models_\nu \neg\psi \quad \text{iff} \quad M \not\models_\nu \psi$$

$$M \models_\nu \psi_1 \wedge \psi_2 \quad \text{iff} \quad M \models_\nu \psi_1 \wedge M \models_\nu \psi_2$$

PML$_\nu$

$$\psi \quad ::= \quad \phi \quad | \quad \neg\psi \quad | \quad \psi_1 \wedge \psi_2 \quad | \quad P_i[\phi] \quad | \quad P_i\#T_i \sim n$$

where $\sim \; = \{=, \neq, <, \leq, >, \geq\}$.

$M \models_\nu \phi$	iff	$M \models_\mu \phi$
$M \models_\nu \neg\psi$	iff	$M \not\models_\nu \psi$
$M \models_\nu \psi_1 \wedge \psi_2$	iff	$M \models_\nu \psi_1 \wedge M \models_\nu \psi_2$
$M \models_\nu P_i[\phi]$	iff	$M_i \models_\mu \phi$

PML$_\nu$

$$\psi \quad ::= \quad \phi \quad | \quad \neg\psi \quad | \quad \psi_1 \wedge \psi_2 \quad | \quad P_i[\phi] \quad | \quad P_i\#T_i \sim n$$

where $\sim\, = \{=, \neq, <, \leq, >, \geq\}$.

$$\begin{array}{lll}
M \models_\nu \phi & \text{iff} & M \models_\mu \phi \\
M \models_\nu \neg\psi & \text{iff} & M \not\models_\nu \psi \\
M \models_\nu \psi_1 \wedge \psi_2 & \text{iff} & M \models_\nu \psi_1 \wedge M \models_\nu \psi_2 \\
M \models_\nu P_i[\phi] & \text{iff} & M_i \models_\mu \phi \\
M \models_\nu P_i\#T_i \sim n & \text{iff} & \mathrm{tokens}(M_i, T_i) \sim n.
\end{array}$$

$\mathbf{PML}_\nu$

$$\psi \quad ::= \quad \phi \quad | \quad \neg\psi \quad | \quad \psi_1 \wedge \psi_2 \quad | \quad P_i[\phi] \quad | \quad P_i\#T_i \sim n$$

where $\sim\ = \{=, \neq, <, \leq, >, \geq\}$.

$$\begin{array}{lll}
M \models_\nu \phi & \text{iff} & M \models_\mu \phi \\
M \models_\nu \neg\psi & \text{iff} & M \not\models_\nu \psi \\
M \models_\nu \psi_1 \wedge \psi_2 & \text{iff} & M \models_\nu \psi_1 \wedge M \models_\nu \psi_2 \\
M \models_\nu P_i[\phi] & \text{iff} & M_i \models_\mu \phi \\
M \models_\nu P_i\#T_i \sim n & \text{iff} & \mathrm{tokens}(M_i, T_i) \sim n.
\end{array}$$

$$\mathrm{tokens}(P, T_i) = \mathrm{tokens}(T[\_], T_i) = 0,$$
$$\mathrm{tokens}(T[T_i], T_i) = 1, \qquad \mathrm{tokens}(T[T_j], T_i) = 0 \text{ if } T_j \neq T_i$$
$$\mathrm{tokens}(P \underset{L}{\bowtie} Q, T_i) = \mathrm{tokens}(P, T_i) + \mathrm{tokens}(Q, T_i)$$

Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.

Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.

Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.

Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.

Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.

Example: Secure Web service

We provide a model of a secure Web service.

Web service requests are sent in encrypted form between the client and the service.

A gatekeeper process runs on the machine at the firewall.

Messages are decrypted and either forwarded on to the server or bounced back to the client.

Inside the firewall messages are exchanged as cleartext but outside the firewall communication is always encrypted.

Tokens

$$
\begin{aligned}
SoapMessage \quad &\stackrel{def}{=} \quad (send_{clr}, r_{sc}).SentClearMessage \\
&+ \quad (encrypt, r_e).EncryptedMsg \\
&+ \quad (parse, r_p).DOMtree
\end{aligned}
$$

Tokens

$$
\begin{aligned}
SoapMessage &\stackrel{def}{=} (send_{clr}, r_{sc}).SentClearMessage \\
&+ (encrypt, r_e).EncryptedMsg \\
&+ (parse, r_p).DOMtree \\
SentClearMessage &\stackrel{def}{=} (\mathbf{copyClear}, \top).SoapMessage
\end{aligned}
$$

Tokens

$$
\begin{aligned}
SoapMessage &\stackrel{def}{=} (send_{clr}, r_{sc}).SentClearMessage \\
&+ (encrypt, r_e).EncryptedMsg \\
&+ (parse, r_p).DOMtree
\end{aligned}
$$

$$
SentClearMessage \stackrel{def}{=} (\mathbf{copyClear}, \top).SoapMessage
$$

$$
\begin{aligned}
EncryptedMsg &\stackrel{def}{=} (decrypt, r_d).SoapMessage \\
&+ (send_{enc}, r_{se}).SentEncMessage
\end{aligned}
$$

Tokens

$$
\begin{aligned}
SoapMessage &\stackrel{def}{=} (send_{clr}, r_{sc}).SentClearMessage \\
&+ (encrypt, r_e).EncryptedMsg \\
&+ (parse, r_p).DOMtree \\
SentClearMessage &\stackrel{def}{=} (\mathbf{copyClear}, \top).SoapMessage \\
EncryptedMsg &\stackrel{def}{=} (decrypt, r_d).SoapMessage \\
&+ (send_{enc}, r_{se}).SentEncMessage \\
SentEncMessage &\stackrel{def}{=} (\mathbf{copyEncrypted}, \top).EncryptedMsg
\end{aligned}
$$

Tokens

$$
\begin{array}{rcl}
SoapMessage & \stackrel{def}{=} & (send_{clr}, r_{sc}).SentClearMessage \\
 & + & (encrypt, r_e).EncryptedMsg \\
 & + & (parse, r_p).DOMtree \\
 & & \\
SentClearMessage & \stackrel{def}{=} & (\mathbf{copyClear}, \top).SoapMessage \\
 & & \\
EncryptedMsg & \stackrel{def}{=} & (decrypt, r_d).SoapMessage \\
 & + & (send_{enc}, r_{se}).SentEncMessage \\
 & & \\
SentEncMessage & \stackrel{def}{=} & (\mathbf{copyEncrypted}, \top).EncryptedMsg \\
 & & \\
DOMtree & \stackrel{def}{=} & (read, r_r).DOMtree \\
 & + & (modify, r_m).DOMtree \\
 & + & (export, r_x).SoapMessage
\end{array}
$$

Static components

$$
\begin{aligned}
User &\stackrel{def}{=} \\
&\quad (encrypt, \top).(send_{enc}, \top).User \\
&+ (decrypt, \top).(parse, \top).(read, \top).(modify, \top).(export, \top).User
\end{aligned}
$$

Static components

$$
\begin{aligned}
&User \stackrel{def}{=} \\
&\quad (encrypt, \top).(send_{enc}, \top).User \\
&+ (decrypt, \top).(parse, \top).(read, \top).(modify, \top).(export, \top).User
\end{aligned}
$$

$$
\begin{aligned}
&GateKeeper \stackrel{def}{=} \\
&\quad (decrypt, \top).(send_{clr}, \top).GateKeeper \\
&+ (decrypt, \top).(encrypt, \top).(send_{enc}, \top).GateKeeper \\
&+ (encrypt, \top).(send_{enc}, \top).GateKeeper
\end{aligned}
$$

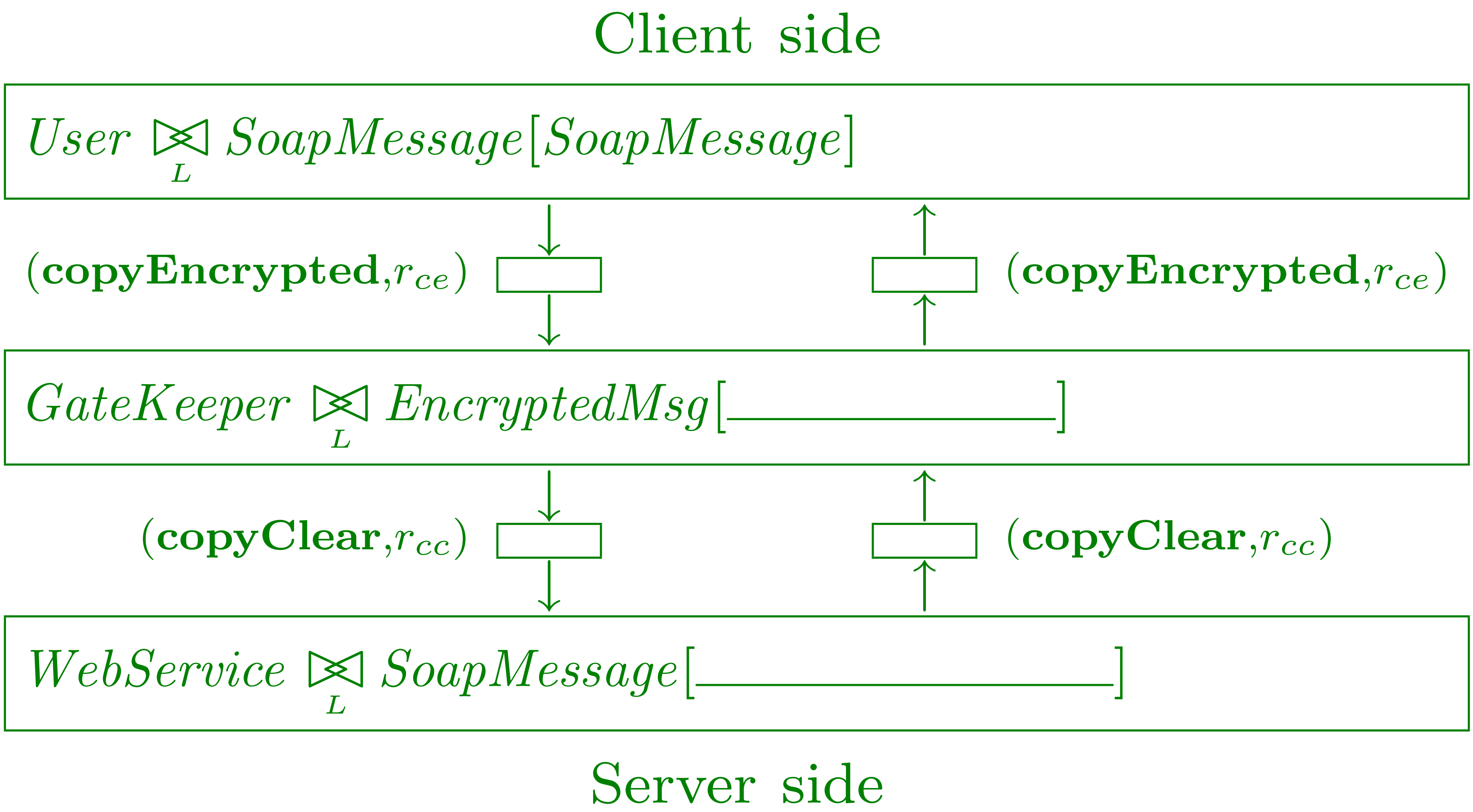
Static components

$$
\begin{aligned}
&User \stackrel{def}{=} \\
&\quad (encrypt, \top).(send_{enc}, \top).User \\
&+ (decrypt, \top).(parse, \top).(read, \top).(modify, \top).(export, \top).User
\end{aligned}
$$

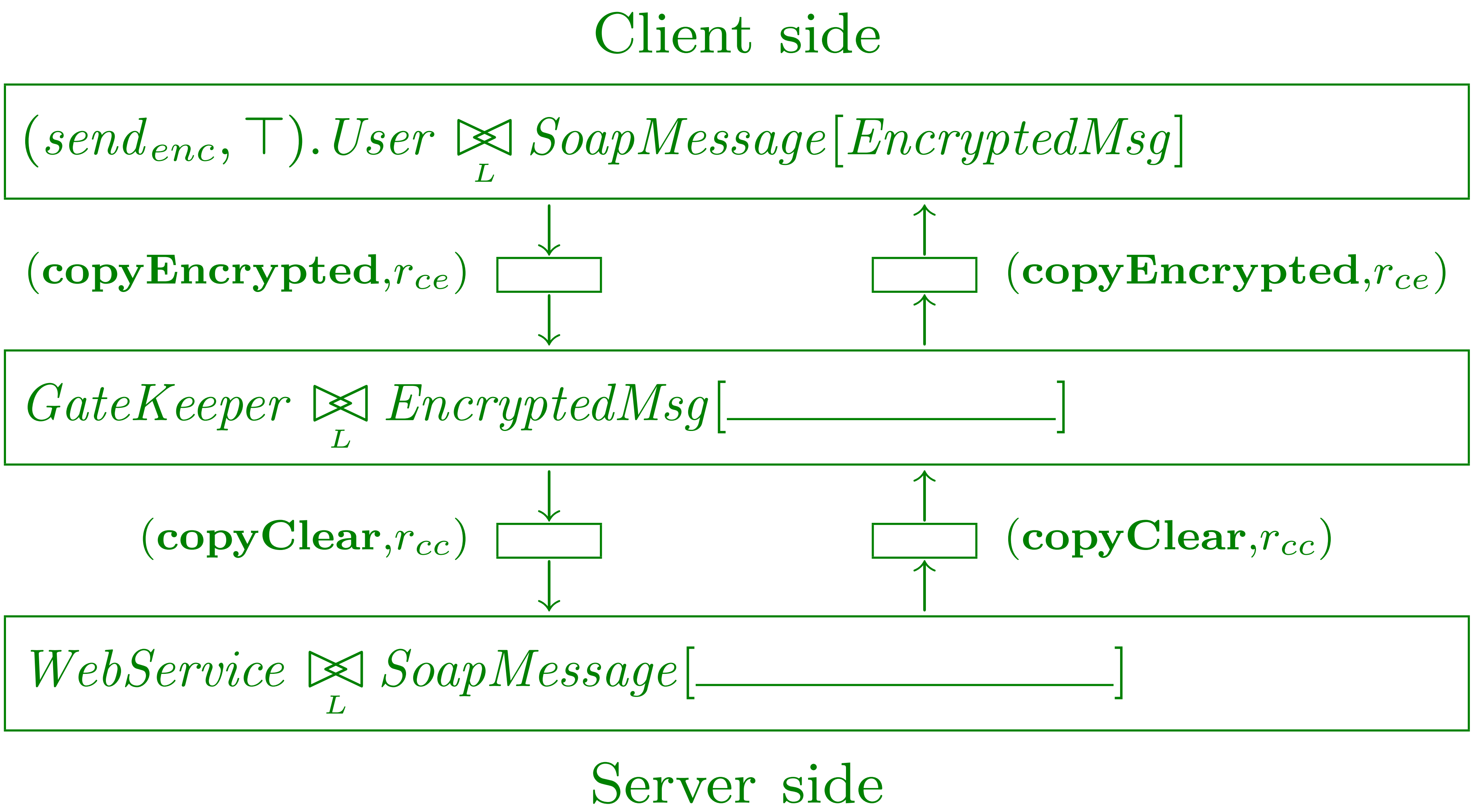
$$
\begin{aligned}
&GateKeeper \stackrel{def}{=} \\
&\quad (decrypt, \top).(send_{clr}, \top).GateKeeper \\
&+ (decrypt, \top).(encrypt, \top).(send_{enc}, \top).GateKeeper \\
&+ (encrypt, \top).(send_{enc}, \top).GateKeeper
\end{aligned}
$$

$$
\begin{aligned}
&WebService \stackrel{def}{=} (parse, \top).(read, \top). \\
&\qquad (modify, \top).(export, \top).(send_{clr}, \top).WebService
\end{aligned}
$$

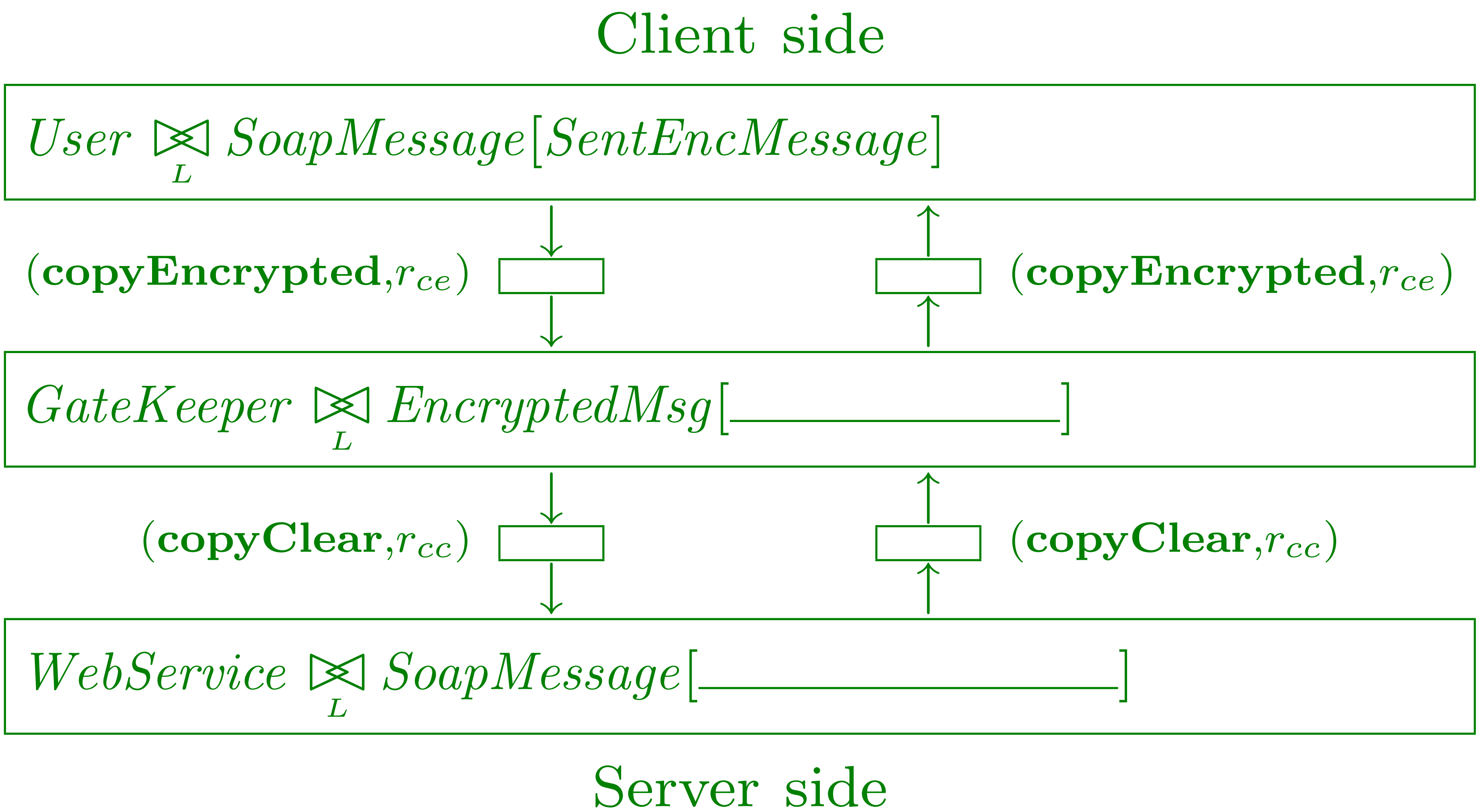
PEPA net


Client side
$User \bowtie_{L} SoapMessage[SoapMessage]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_{L} EncryptedMsg[\_\_\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[\_\_\_]$
Server side

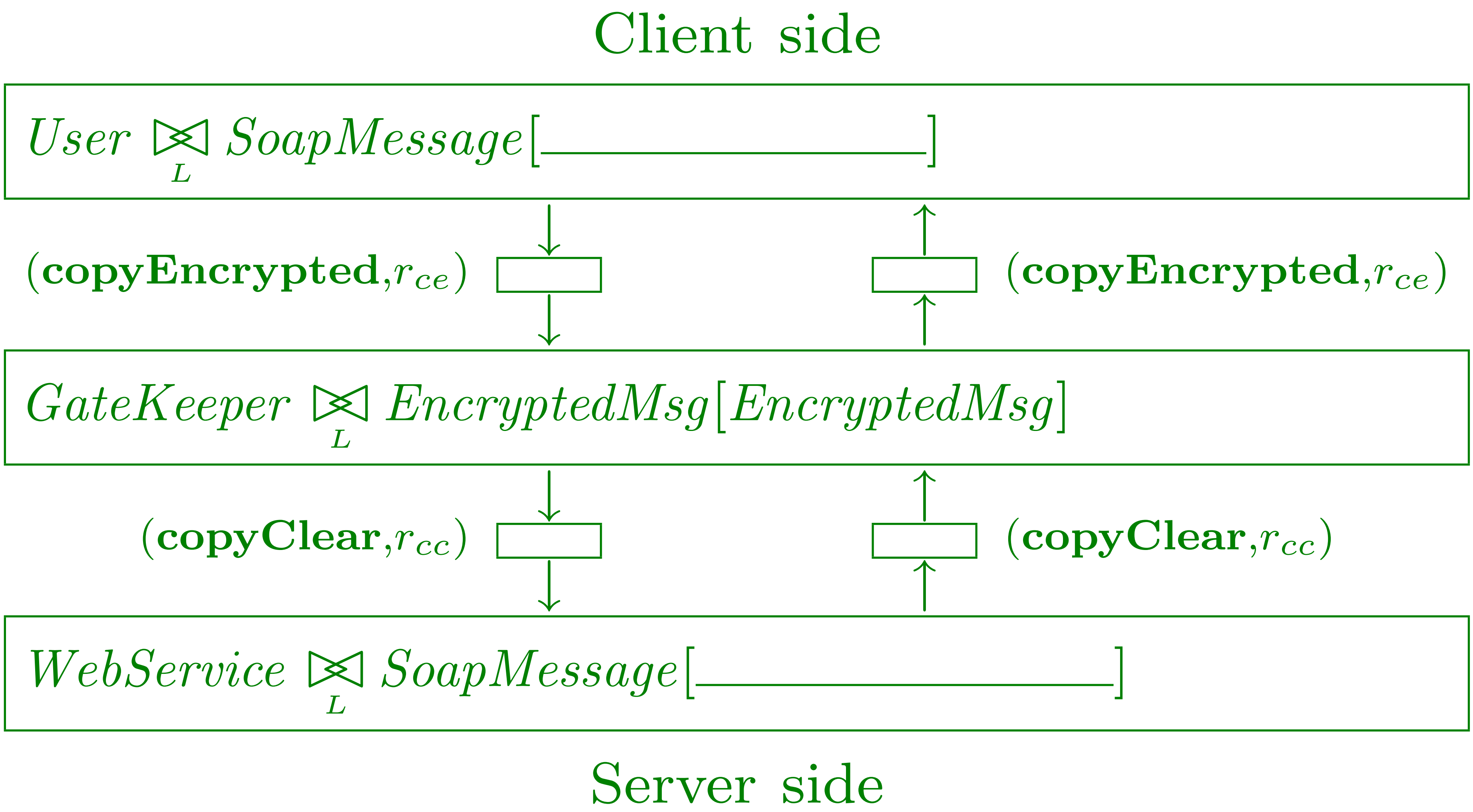
PEPA net


Client side
$(send_{enc}, \top).User \bowtie_{L} SoapMessage[EncryptedMsg]$
(**copyEncrypted**,r_{ce})
(**copyEncrypted**,r_{ce})
$GateKeeper \bowtie_{L} EncryptedMsg[\_\_\_]$
(**copyClear**,r_{cc})
(**copyClear**,r_{cc})
$WebService \bowtie_{L} SoapMessage[\_\_\_]$
Server side

PEPA net

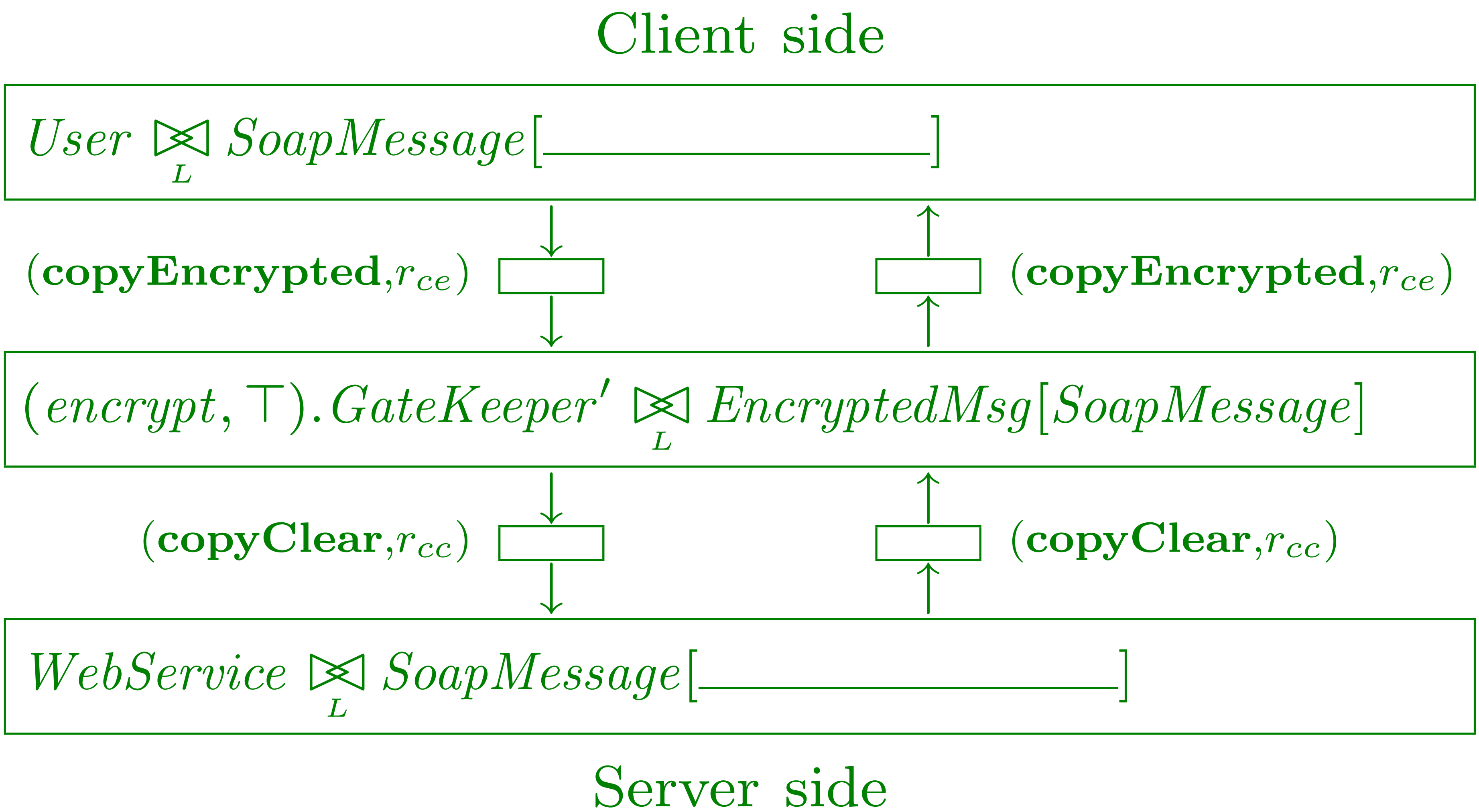

Client side
$User \bowtie_{L} SoapMessage[SentEncMessage]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_{L} EncryptedMsg[\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[\_]$
Server side

PEPA net

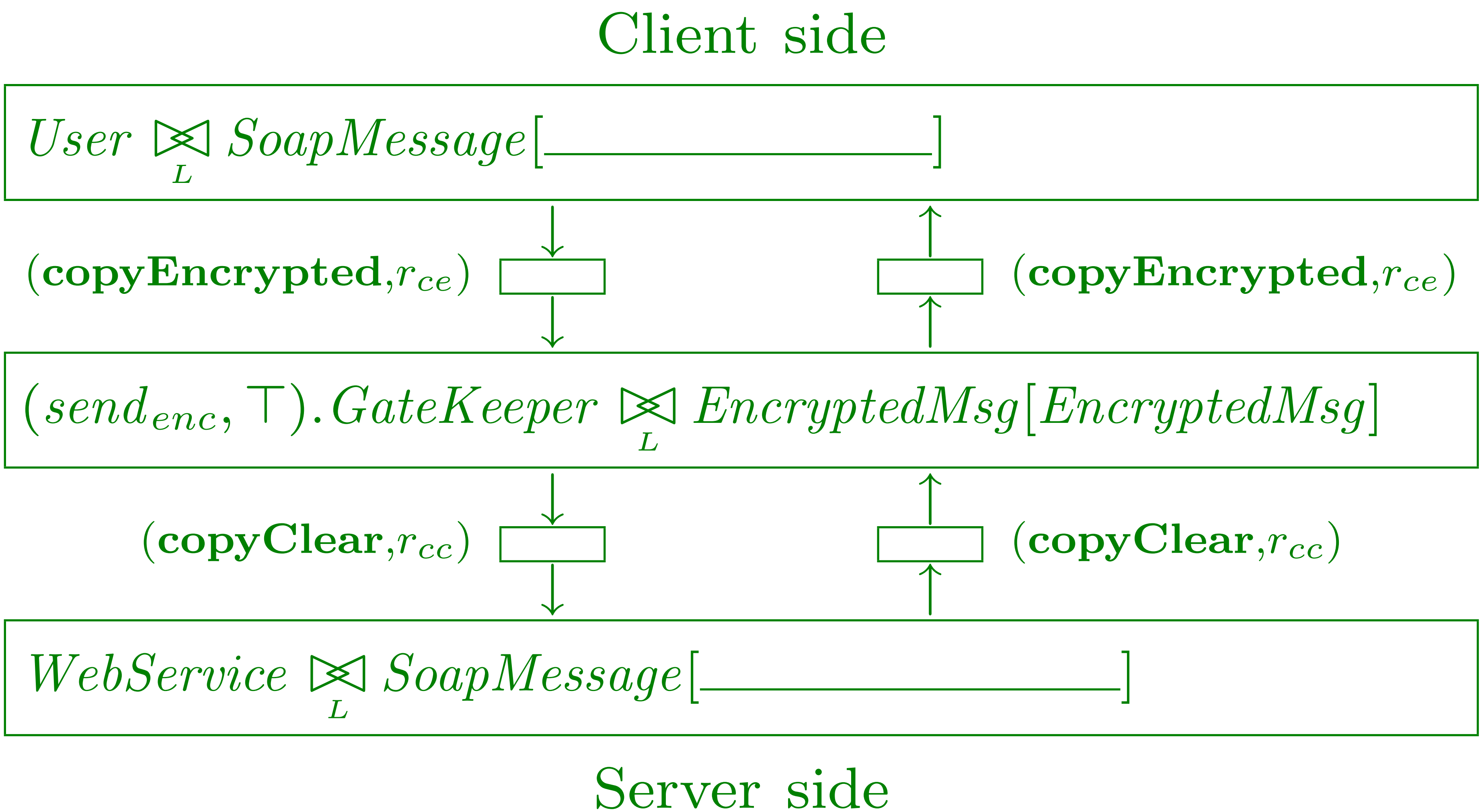


Client side
$User \underset{L}{\bowtie} SoapMessage[\_\_\_]$
(**copyEncrypted**,r_{ce})
(**copyEncrypted**,r_{ce})
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[EncryptedMsg]$
(**copyClear**,r_{cc})
(**copyClear**,r_{cc})
$WebService \underset{L}{\bowtie} SoapMessage[\_\_\_]$
Server side

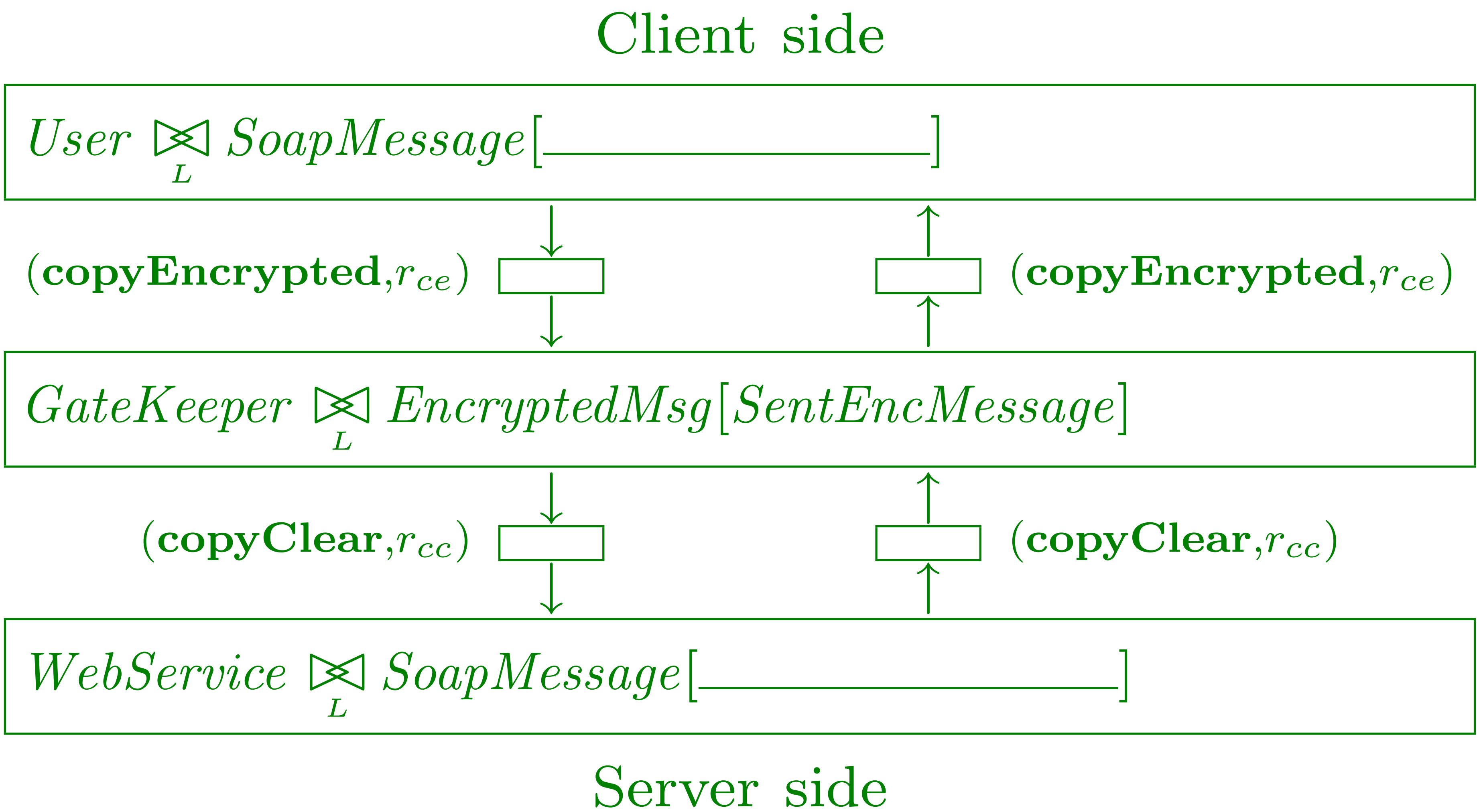
PEPA net


Client side
$User \bowtie_L SoapMessage[\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$(encrypt, \top).GateKeeper' \bowtie_L EncryptedMsg[SoapMessage]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_L SoapMessage[\_\_\_]$
Server side

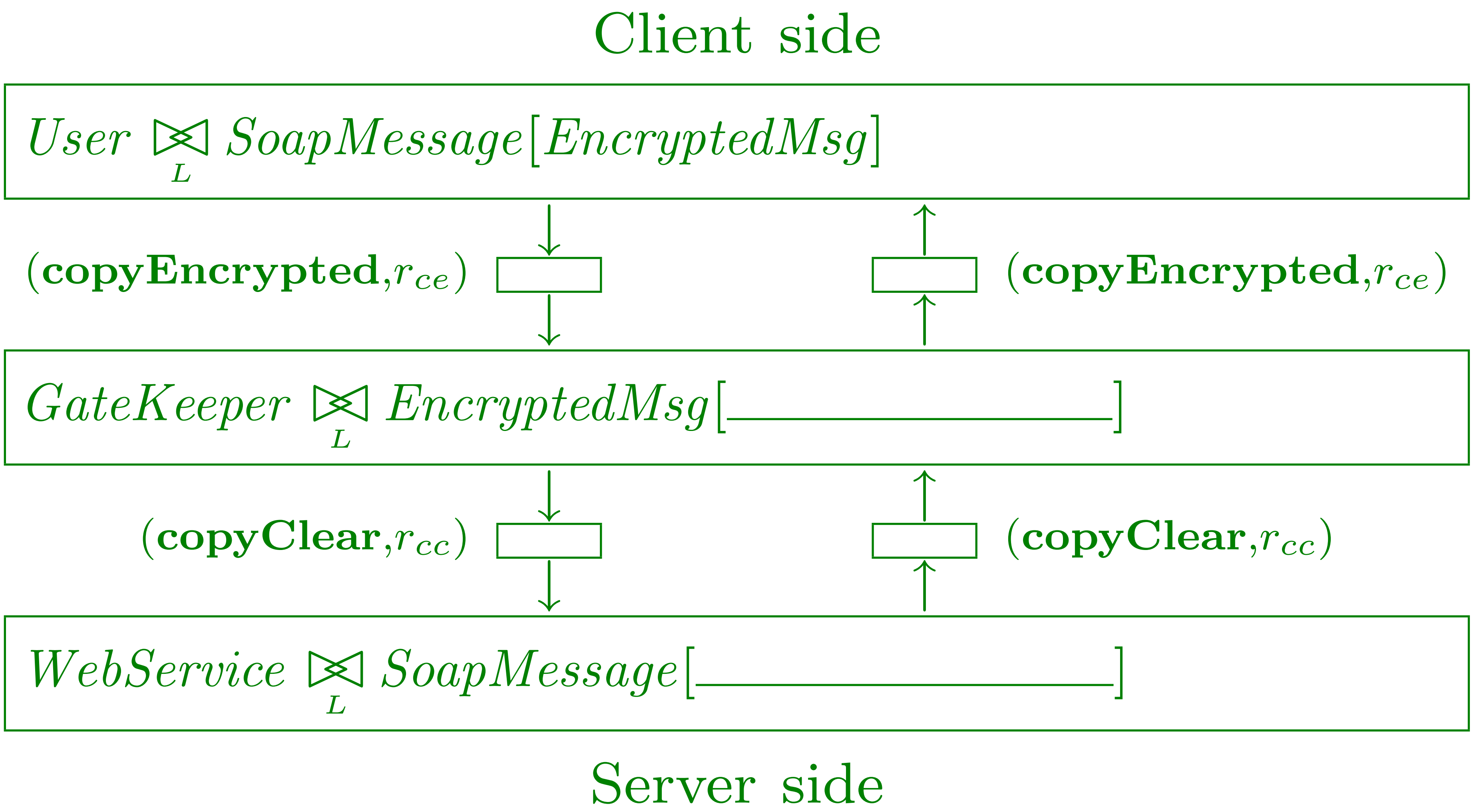
PEPA net


Client side
$User \bowtie_{L} SoapMessage[\underline{\qquad\qquad\qquad}]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$(send_{enc}, \top).GateKeeper \bowtie_{L} EncryptedMsg[EncryptedMsg]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[\underline{\qquad\qquad\qquad}]$
Server side

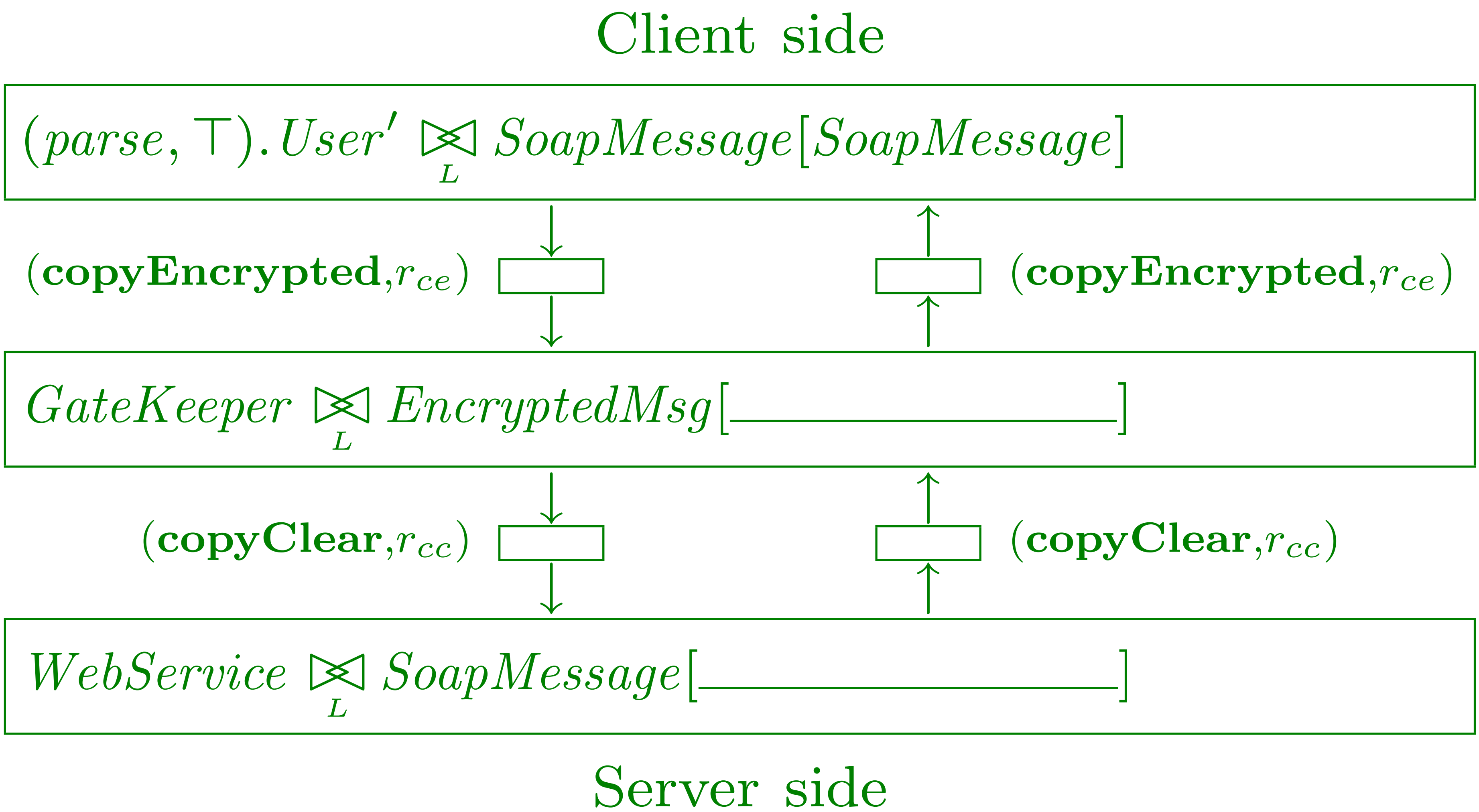
PEPA net


Client side
$User \bowtie_{L} SoapMessage[\_\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_{L} EncryptedMsg[SentEncMessage]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[\_\_\_\_]$
Server side

PEPA net

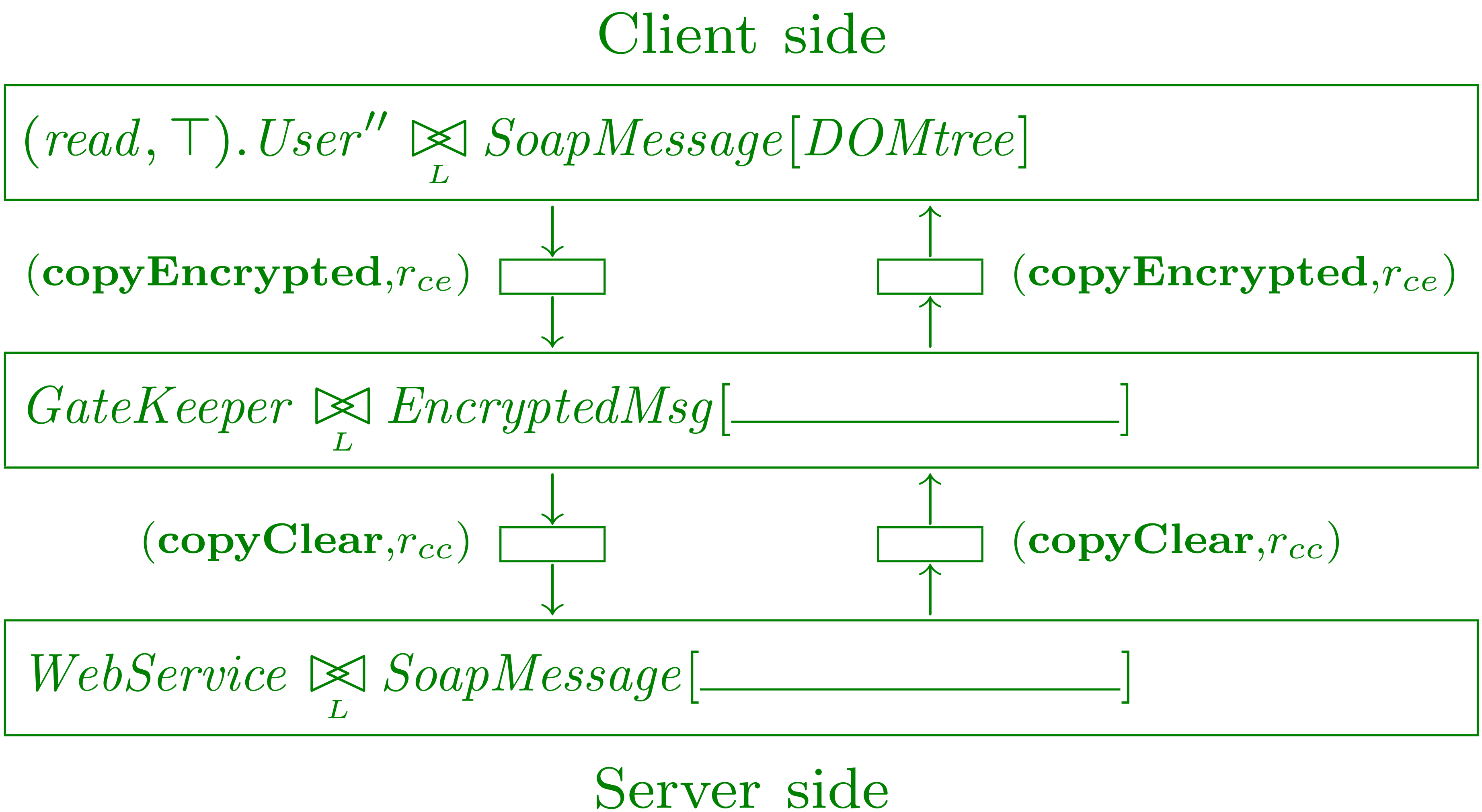

Client side
$User \bowtie_{L} SoapMessage[EncryptedMsg]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_{L} EncryptedMsg[\_\_\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[\_\_\_]$
Server side

PEPA net

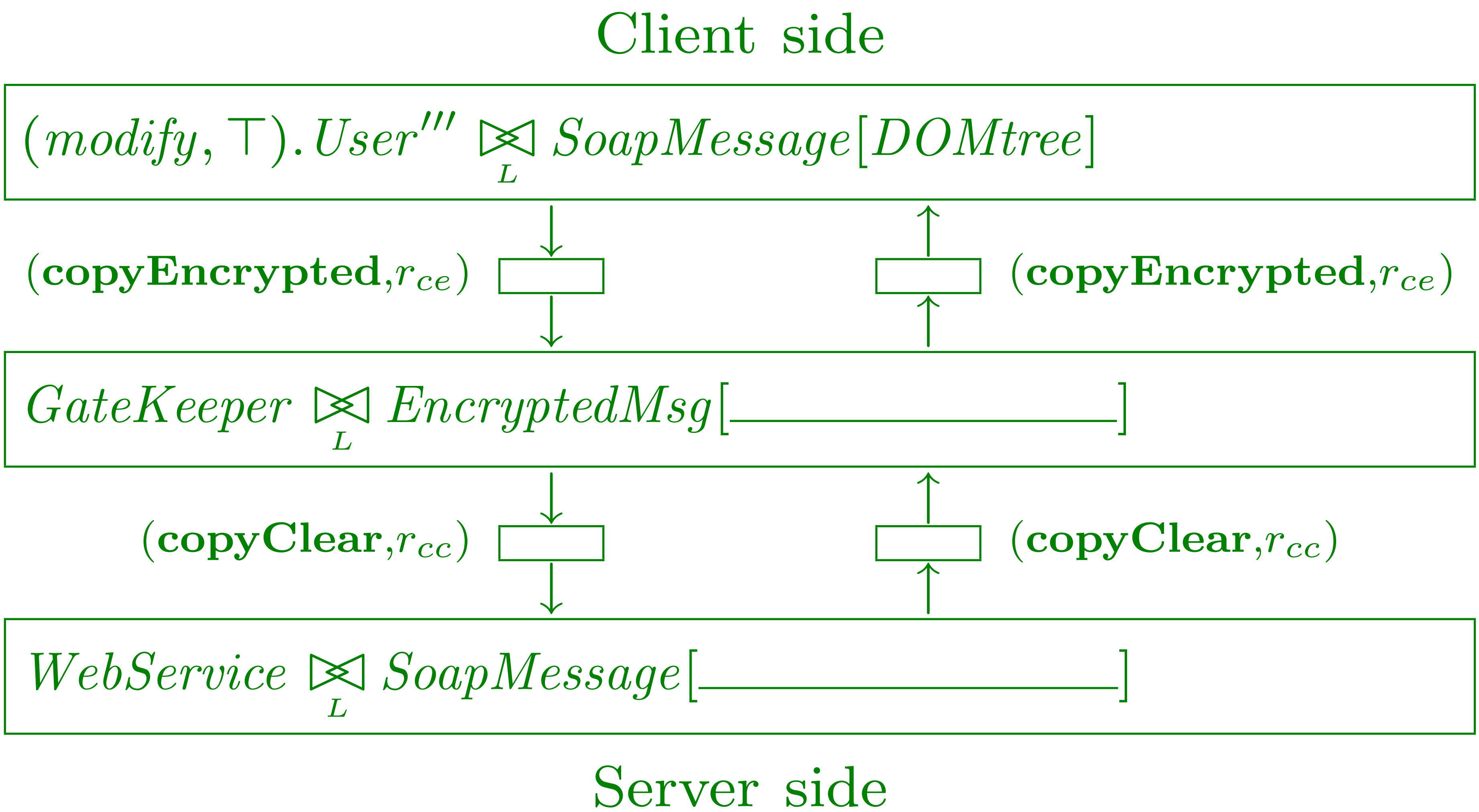


Client side
$(parse, \top).User' \bowtie_{L} SoapMessage[SoapMessage]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_{L} EncryptedMsg[\_\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[\_\_]$
Server side

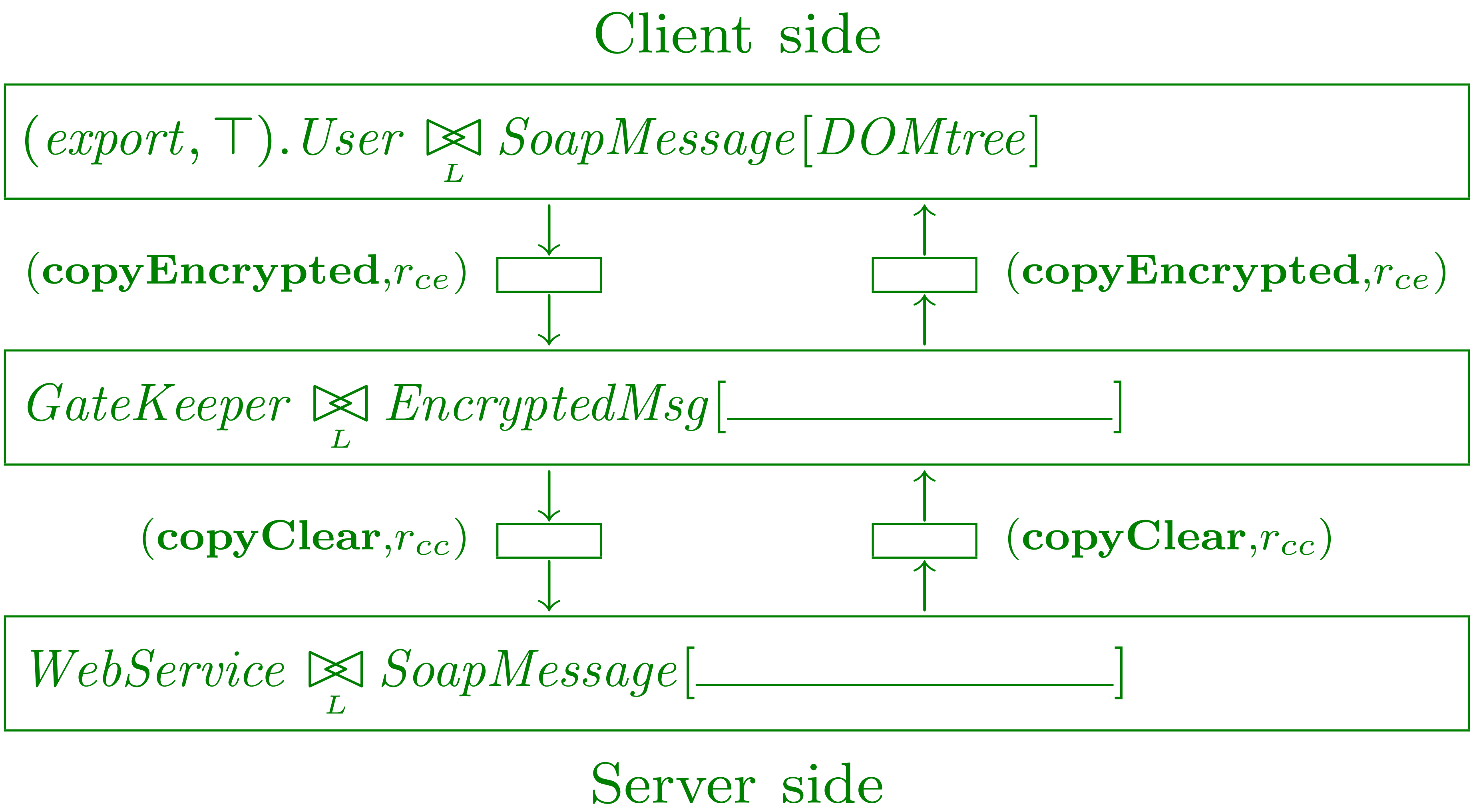
PEPA net


Client side
$(read, \top).User'' \bowtie_L SoapMessage[DOMtree]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_L EncryptedMsg[\_\_\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_L SoapMessage[\_\_\_]$
Server side

PEPA net

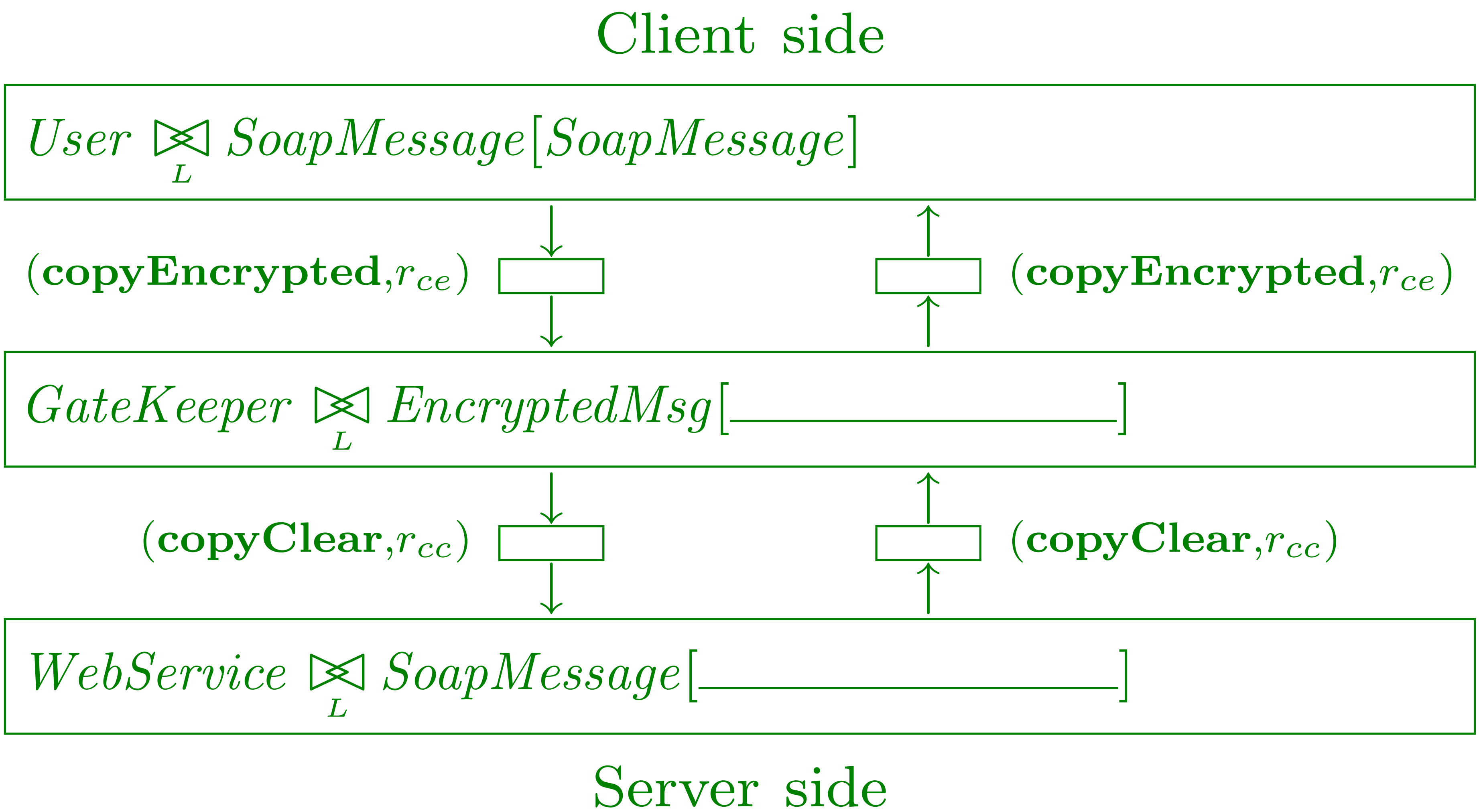

Client side
$(modify, \top).User''' \underset{L}{\bowtie} SoapMessage[DOMtree]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[\_\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \underset{L}{\bowtie} SoapMessage[\_\_]$
Server side

PEPA net

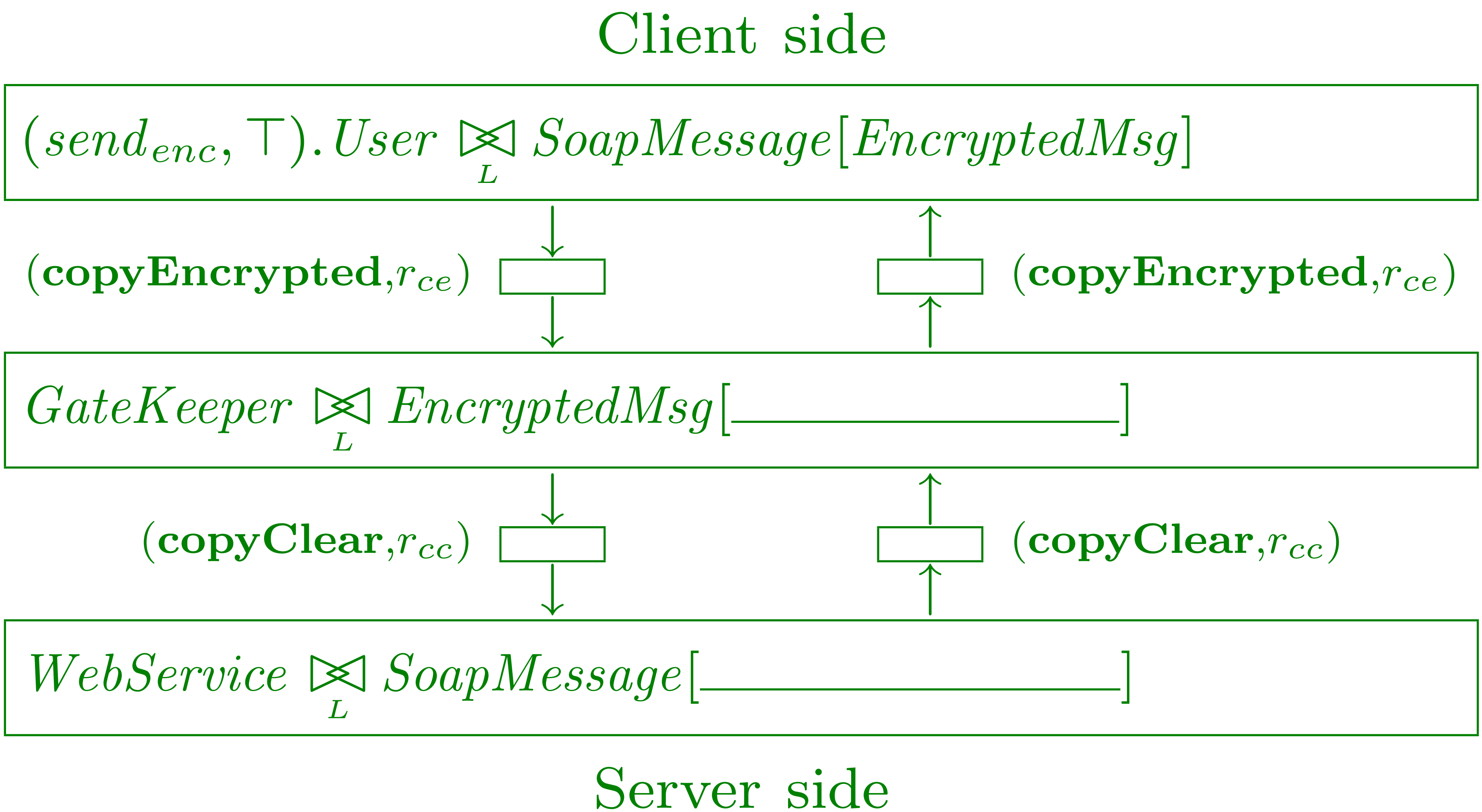


Client side
$(export, \top).User \underset{L}{\bowtie} SoapMessage[DOMtree]$
(**copyEncrypted**,r_{ce})
(**copyEncrypted**,r_{ce})
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[\_\_\_]$
(**copyClear**,r_{cc})
(**copyClear**,r_{cc})
$WebService \underset{L}{\bowtie} SoapMessage[\_\_\_]$
Server side

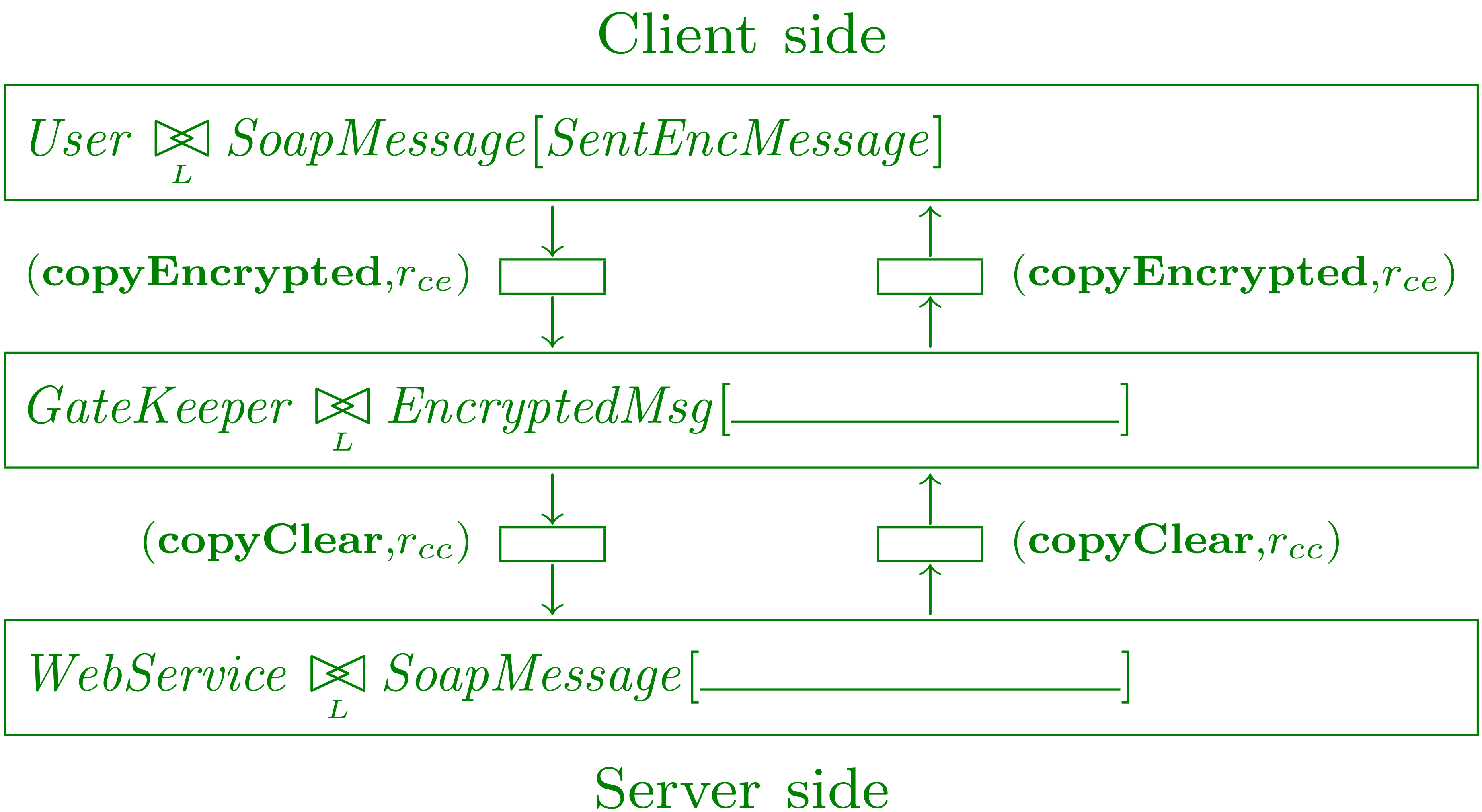
PEPA net


Client side
$User \underset{L}{\bowtie} SoapMessage[SoapMessage]$
$(\textbf{copyEncrypted}, r_{ce})$
$(\textbf{copyEncrypted}, r_{ce})$
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[\_\_]$
$(\textbf{copyClear}, r_{cc})$
$(\textbf{copyClear}, r_{cc})$
$WebService \underset{L}{\bowtie} SoapMessage[\_\_]$
Server side

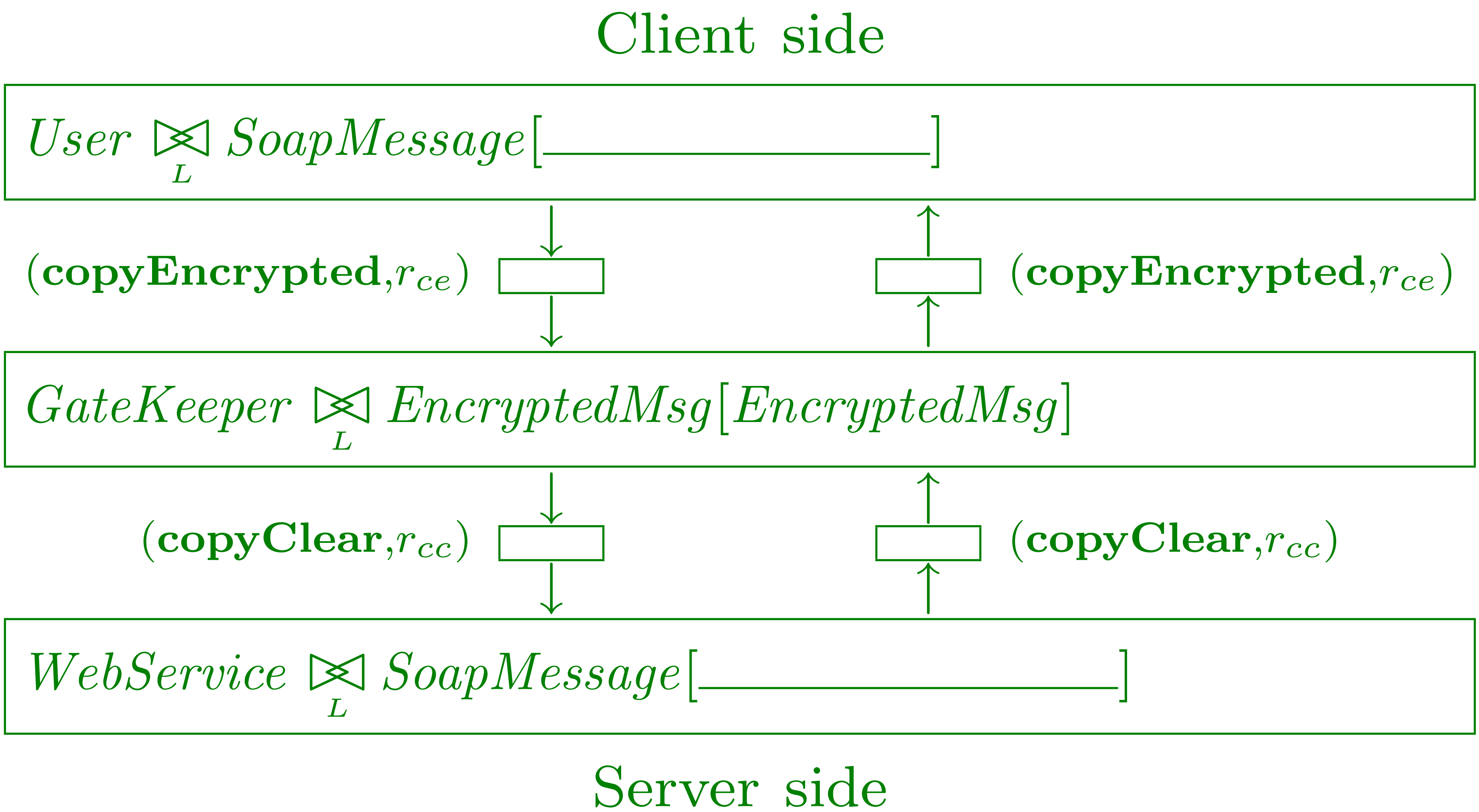
PEPA net


Client side
$(send_{enc}, \top).User \bowtie_L SoapMessage[EncryptedMsg]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_L EncryptedMsg[\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_L SoapMessage[\_]$
Server side

PEPA net

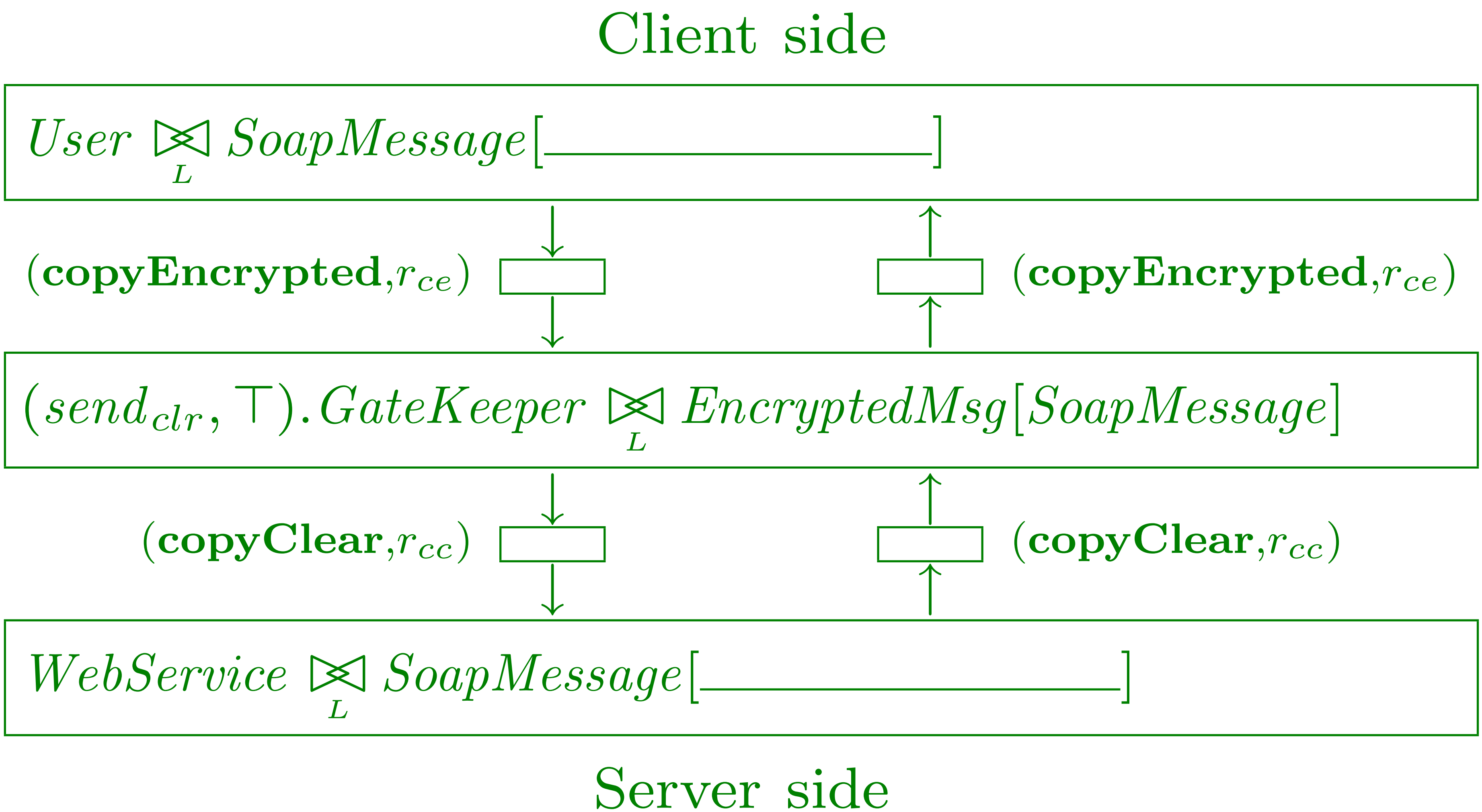

Client side
$User \underset{L}{\bowtie} SoapMessage[SentEncMessage]$
(**copyEncrypted**,r_{ce})
(**copyEncrypted**,r_{ce})
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[\_\_\_]$
(**copyClear**,r_{cc})
(**copyClear**,r_{cc})
$WebService \underset{L}{\bowtie} SoapMessage[\_\_\_]$
Server side

PEPA net

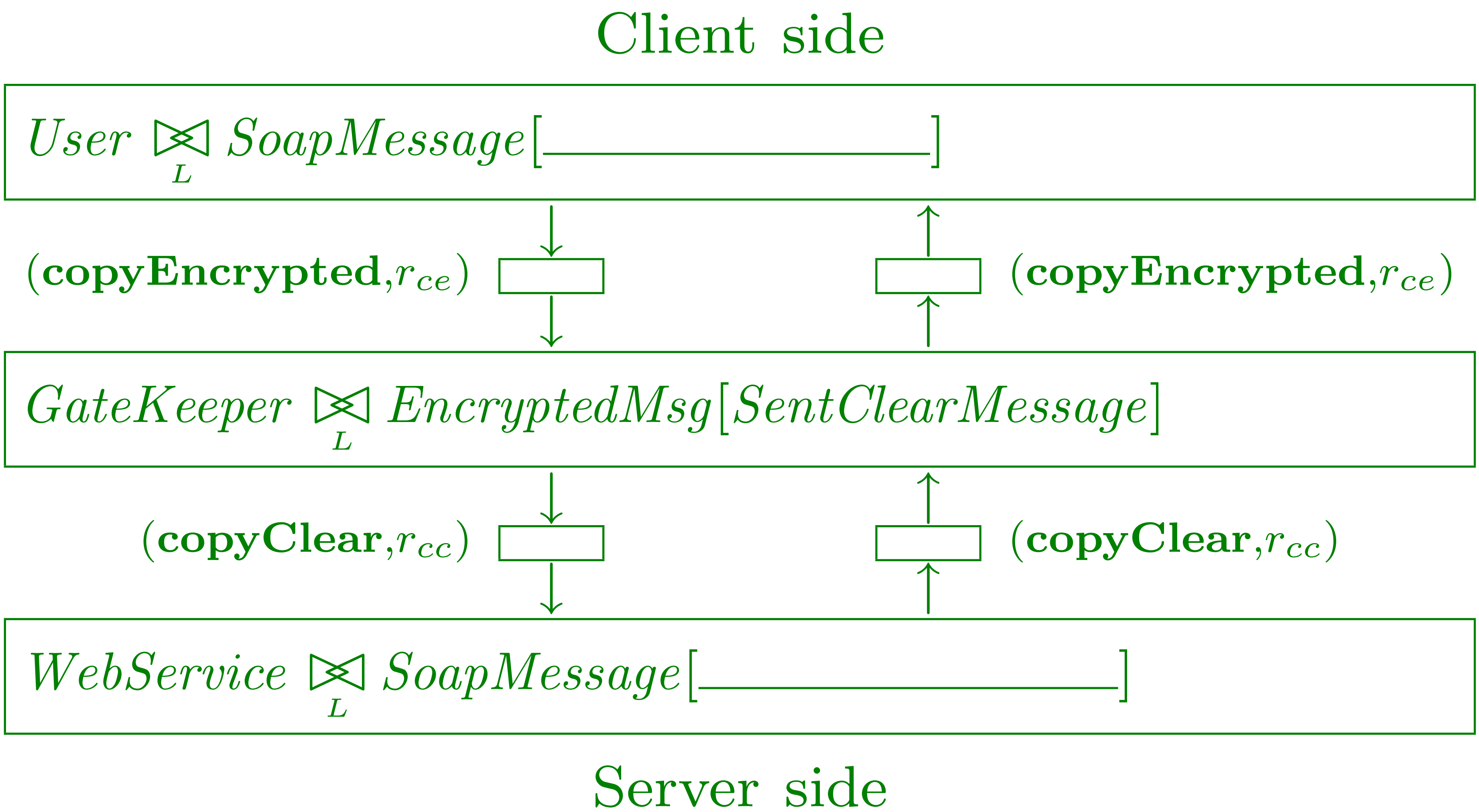


Client side
$User \bowtie_{L} SoapMessage[\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_{L} EncryptedMsg[EncryptedMsg]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[\_\_\_]$
Server side

PEPA net

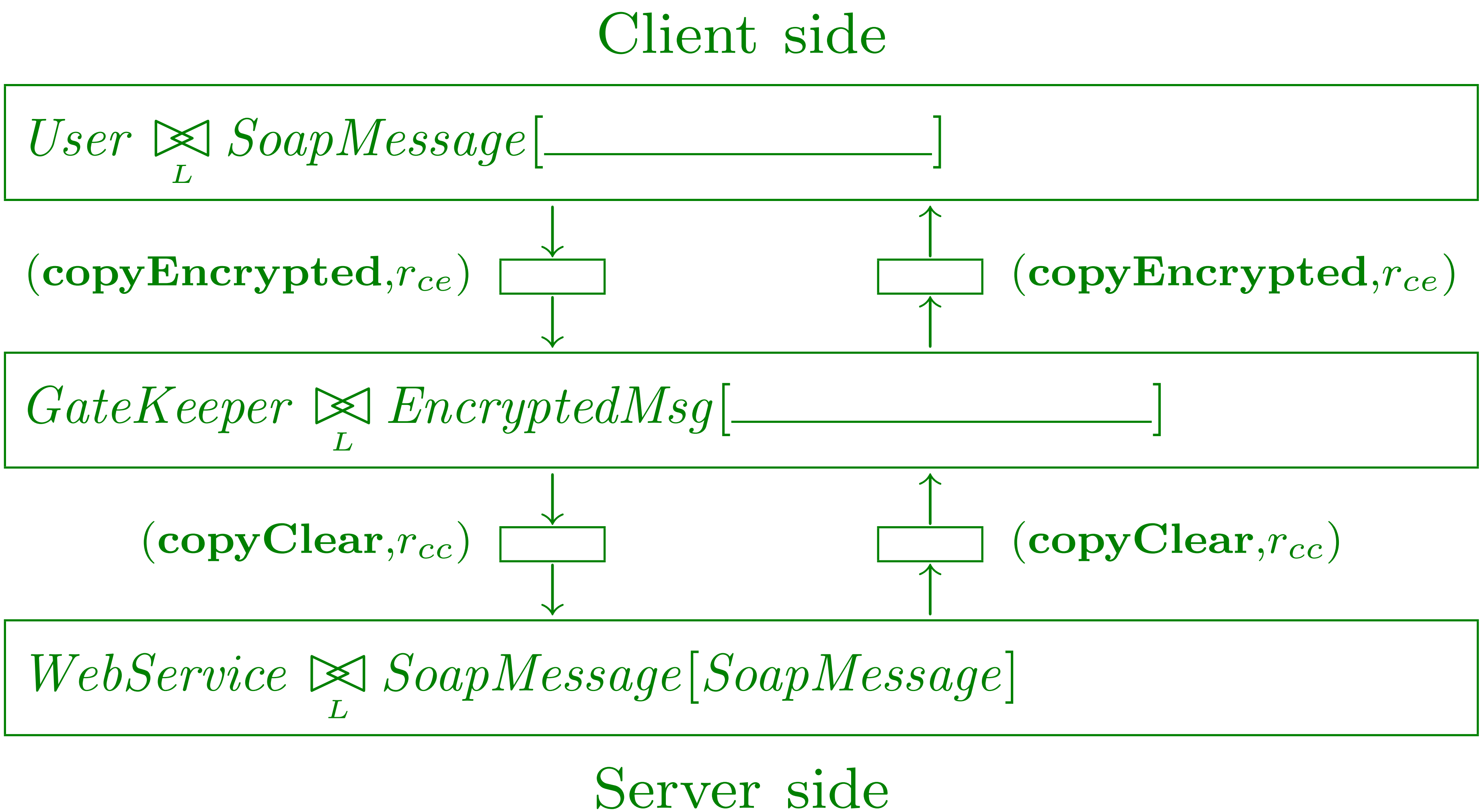

Client side
$User \underset{L}{\bowtie} SoapMessage[\_\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$(send_{clr}, \top).GateKeeper \underset{L}{\bowtie} EncryptedMsg[SoapMessage]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \underset{L}{\bowtie} SoapMessage[\_\_\_\_]$
Server side

PEPA net

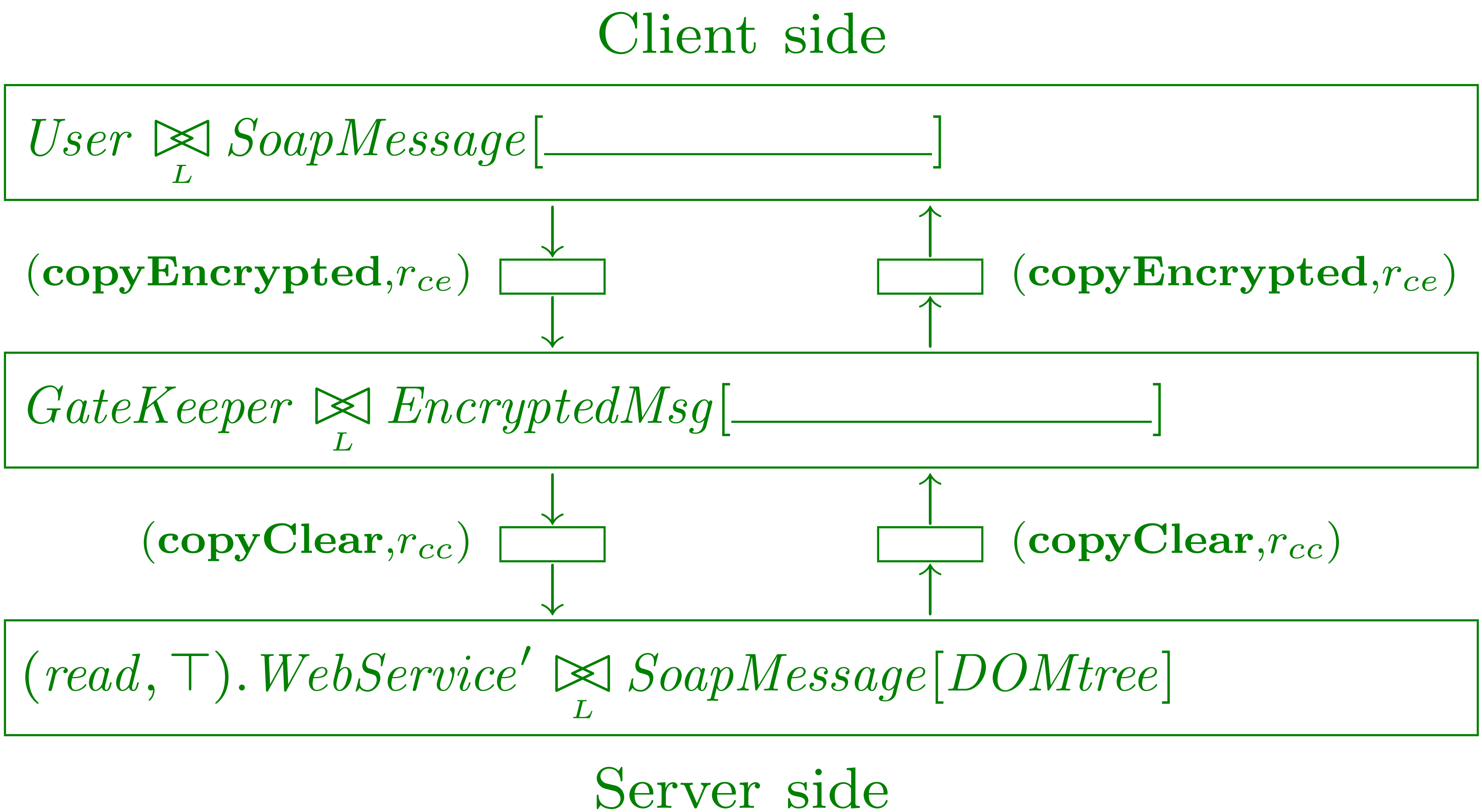


Client side
$User \underset{L}{\bowtie} SoapMessage[\_\_\_\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[SentClearMessage]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \underset{L}{\bowtie} SoapMessage[\_\_\_\_\_\_]$
Server side

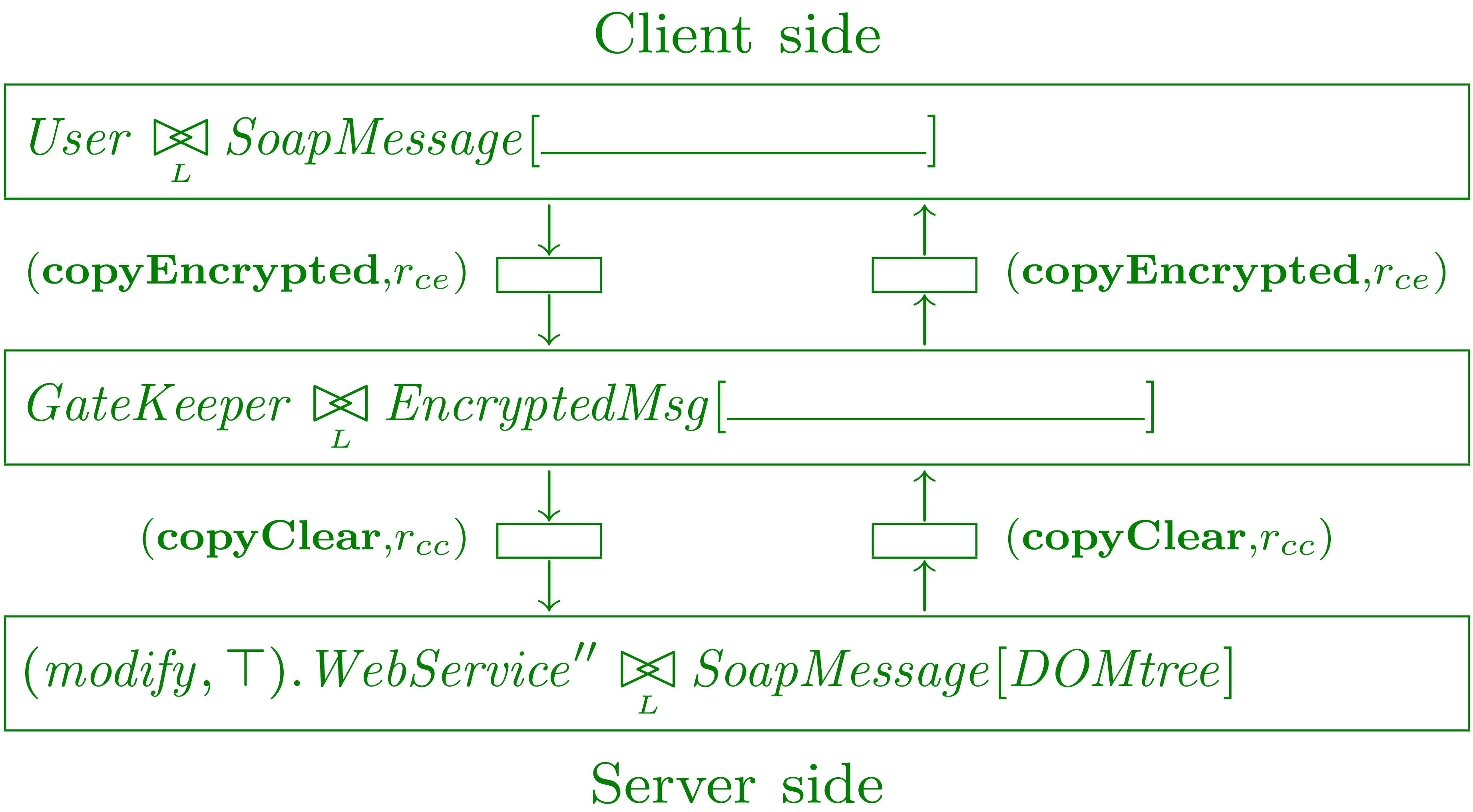
PEPA net


Client side
$User \bowtie_{L} SoapMessage[\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_{L} EncryptedMsg[\_\_\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[SoapMessage]$
Server side

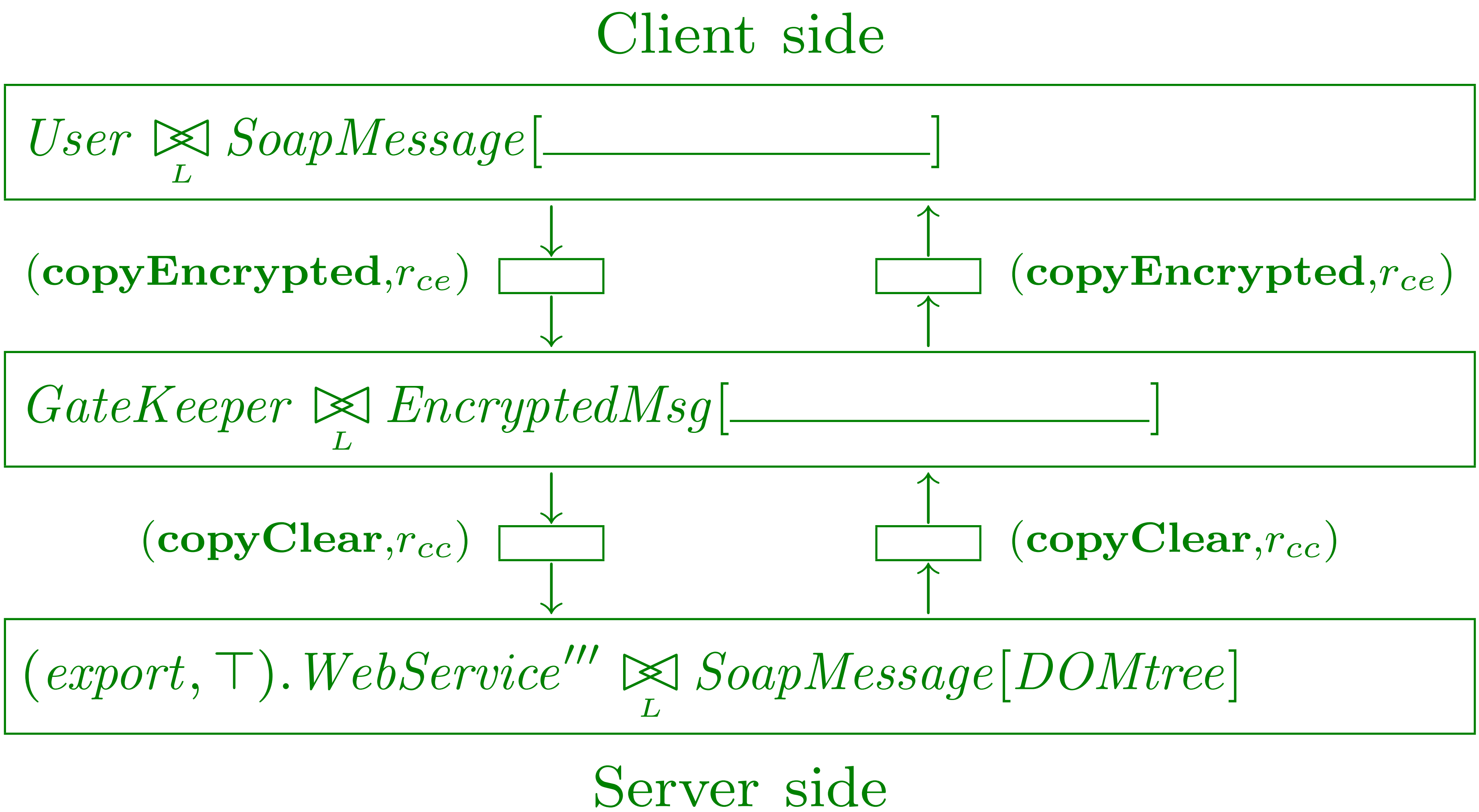
PEPA net


Client side
$User \bowtie_L SoapMessage[\_\_\_]$
(**copyEncrypted**,r_{ce})
(**copyEncrypted**,r_{ce})
$GateKeeper \bowtie_L EncryptedMsg[\_\_\_]$
(**copyClear**,r_{cc})
(**copyClear**,r_{cc})
$(read, \top).WebService' \bowtie_L SoapMessage[DOMtree]$
Server side

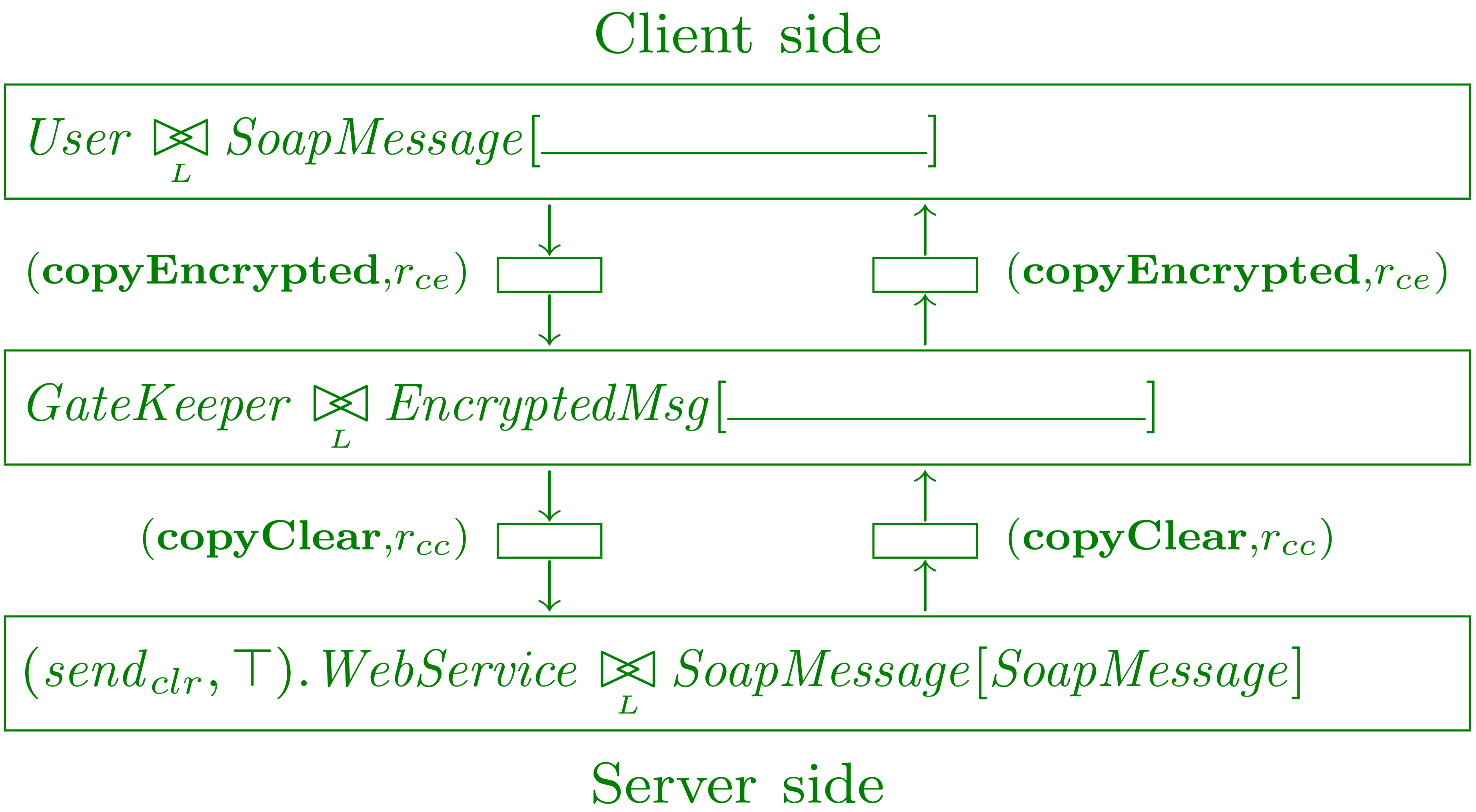
PEPA net


Client side
$User \underset{L}{\bowtie} SoapMessage[\_\_\_\_\_\_\_\_]$
$(\textbf{copyEncrypted}, r_{ce})$
$(\textbf{copyEncrypted}, r_{ce})$
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[\_\_\_\_\_\_\_\_]$
$(\textbf{copyClear}, r_{cc})$
$(\textbf{copyClear}, r_{cc})$
$(modify, \top).WebService'' \underset{L}{\bowtie} SoapMessage[DOMtree]$
Server side

PEPA net

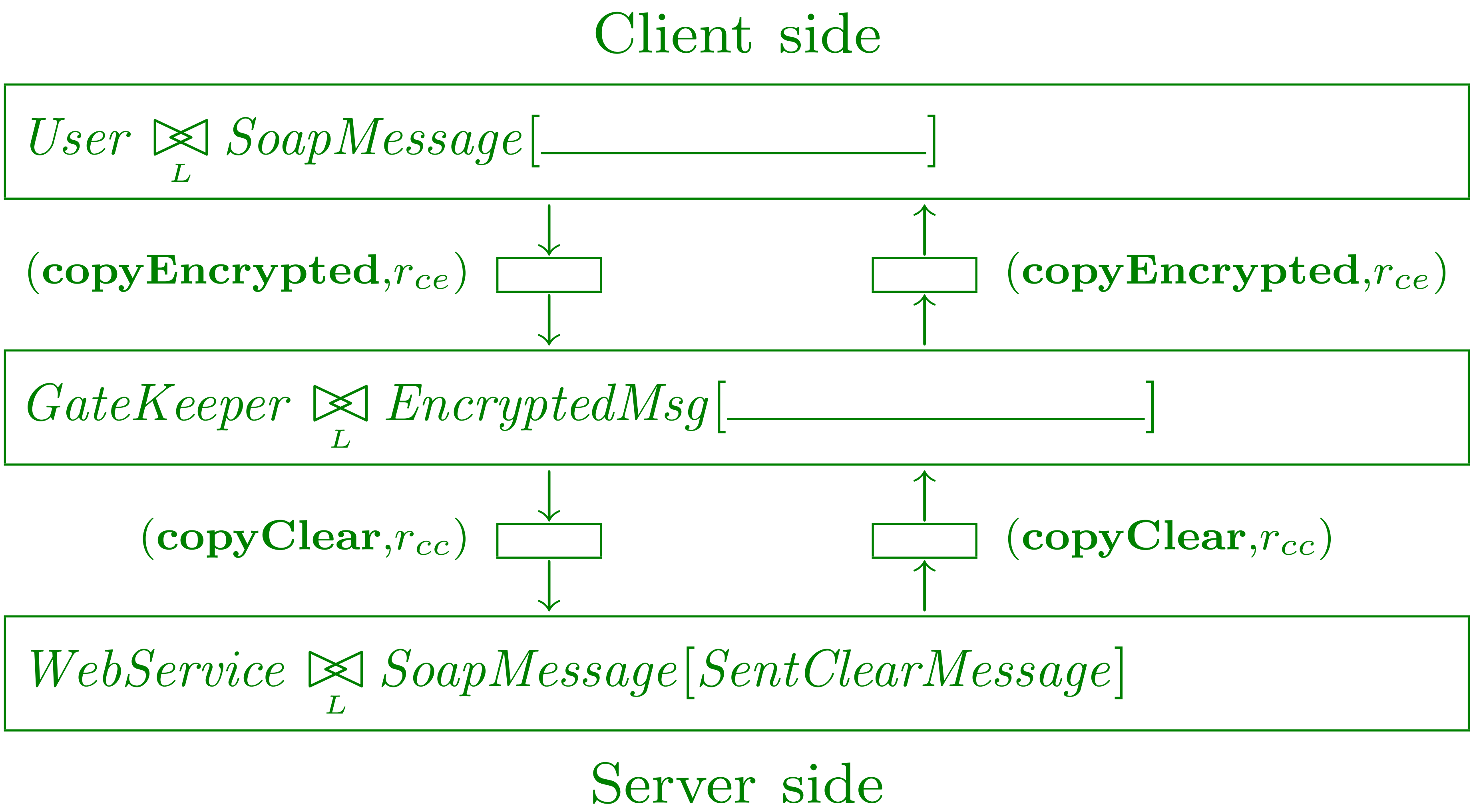

Client side
$User \underset{L}{\bowtie} SoapMessage[\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[\_\_\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$(export, \top).WebService''' \underset{L}{\bowtie} SoapMessage[DOMtree]$
Server side

PEPA net

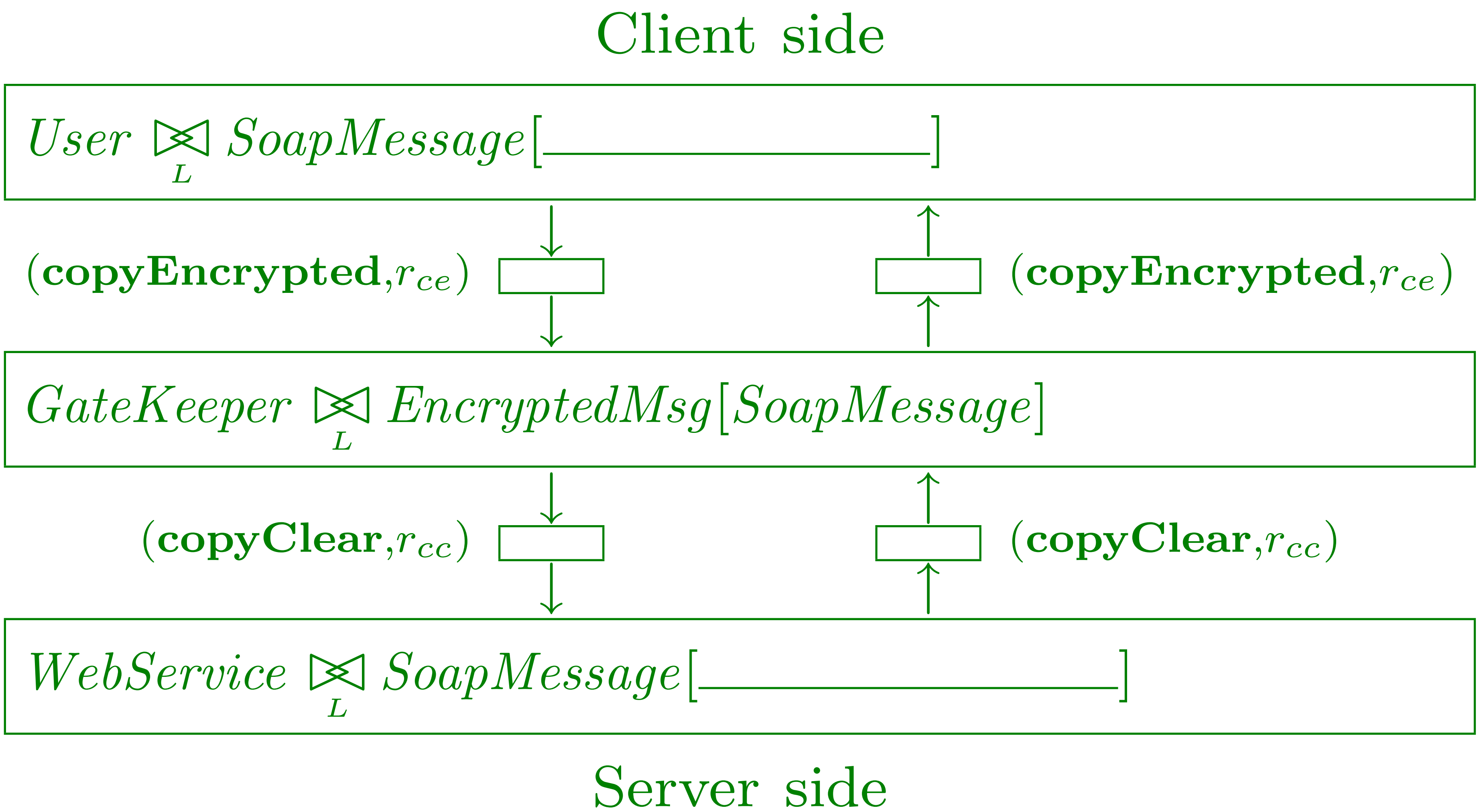


Client side
$User \bowtie_L SoapMessage[\_\_]$
(**copyEncrypted**,r_{ce})
(**copyEncrypted**,r_{ce})
$GateKeeper \bowtie_L EncryptedMsg[\_\_]$
(**copyClear**,r_{cc})
(**copyClear**,r_{cc})
$(send_{clr}, \top).WebService \bowtie_L SoapMessage[SoapMessage]$
Server side

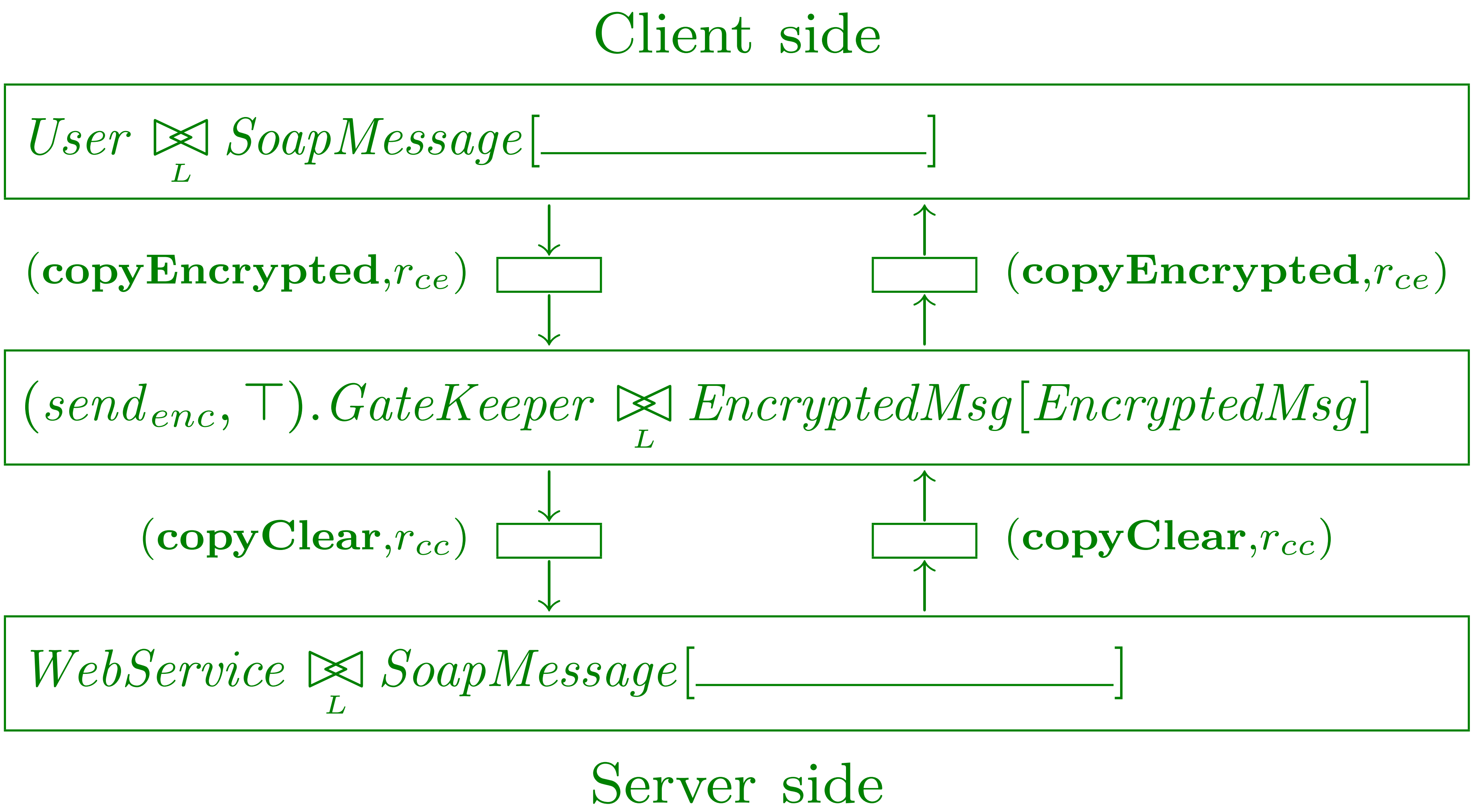
PEPA net


Client side
$User \underset{L}{\bowtie} SoapMessage[\_\_\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[\_\_\_\_\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \underset{L}{\bowtie} SoapMessage[SentClearMessage]$
Server side

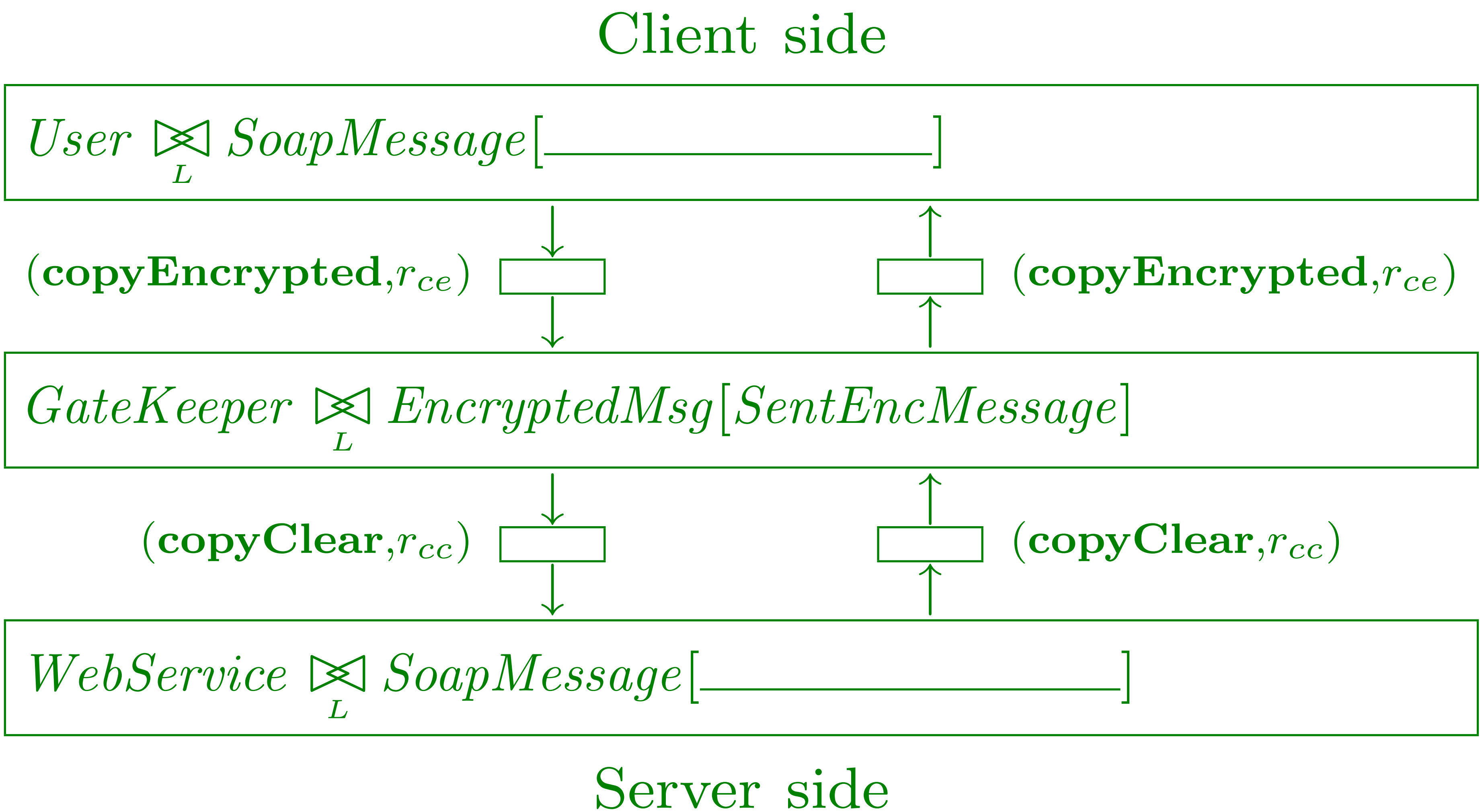
PEPA net


Client side
$User \bowtie_{L} SoapMessage[\_\_\_]$
$(\textbf{copyEncrypted}, r_{ce})$
$(\textbf{copyEncrypted}, r_{ce})$
$GateKeeper \bowtie_{L} EncryptedMsg[SoapMessage]$
$(\textbf{copyClear}, r_{cc})$
$(\textbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[\_\_\_]$
Server side

PEPA net

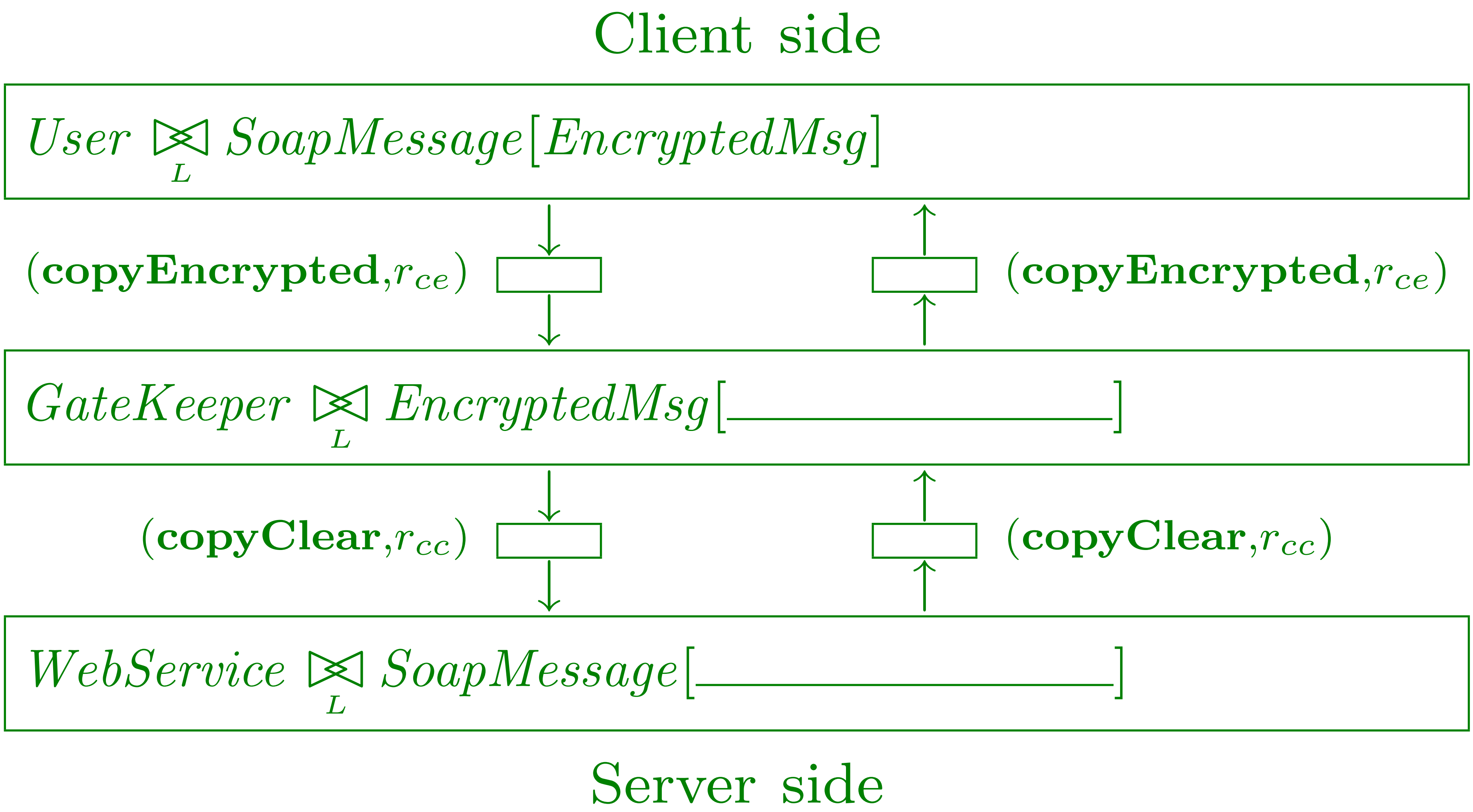

Client side
$User \bowtie_{L} SoapMessage[\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$(send_{enc}, \top).GateKeeper \bowtie_{L} EncryptedMsg[EncryptedMsg]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \bowtie_{L} SoapMessage[\_\_\_]$
Server side

PEPA net



Client side
$User \underset{L}{\bowtie} SoapMessage[\_\_\_]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[SentEncMessage]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \underset{L}{\bowtie} SoapMessage[\_\_\_]$
Server side

PEPA net


Client side
$User \underset{L}{\bowtie} SoapMessage[EncryptedMsg]$
$(\mathbf{copyEncrypted}, r_{ce})$
$(\mathbf{copyEncrypted}, r_{ce})$
$GateKeeper \underset{L}{\bowtie} EncryptedMsg[\_\_\_]$
$(\mathbf{copyClear}, r_{cc})$
$(\mathbf{copyClear}, r_{cc})$
$WebService \underset{L}{\bowtie} SoapMessage[\_\_\_]$
Server side

Expressing performance measures using PML$_\nu$

Probability that the user has an unread reply:

$$Client[\Delta_{decrypt} \vee \Delta_{parse}]$$

Expressing performance measures using PML$_\nu$

Probability that the user has an unread reply:

$$Client[\Delta_{decrypt} \vee \Delta_{parse}]$$

Probability that the client has just sent a request:

$$Client\#SentEncMessage = 1$$

Expressing performance measures using PML$_\nu$

Probability that the user has an unread reply:

$$Client[\Delta_{decrypt} \vee \Delta_{parse}]$$

Probability that the client has just sent a request:

$$Client\#SentEncMessage = 1$$

Service time distribution at the server side:

Start when $Server\#SoapMessage = 1$

Stop when $Firewall\#SentEncMessage = 1$

Solving PEPA nets

Practical performance analysis methods must provide automated support for deriving numerical results from a high-level specification.

Solving PEPA nets

Practical performance analysis methods must provide automated support for deriving numerical results from a high-level specification.

Usually the high-level model is used to derive a Continuous-Time Markov Chain (CTMC) for performance analysis.

Solving PEPA nets

Practical performance analysis methods must provide automated support for deriving numerical results from a high-level specification.

Usually the high-level model is used to derive a Continuous-Time Markov Chain (CTMC) for performance analysis.

We can derive a CTMC directly from a PEPA net using the PEPA Workbench for PEPA nets.

Solving PEPA nets

Practical performance analysis methods must provide automated support for deriving numerical results from a high-level specification.

Usually the high-level model is used to derive a Continuous-Time Markov Chain (CTMC) for performance analysis.

We can derive a CTMC directly from a PEPA net using the PEPA Workbench for PEPA nets.

An alternative is to compile a PEPA net to an equivalent PEPA model and then use one of the PEPA tools.

Compiling PEPA nets to PEPA

The PEPA net compiler compiles a PEPA net to a PEPA model. Activities are renamed to enforce the PEPA net idiom that components at different places cannot synchronise on transitions.

The given net and the generated PEPA model produce isomorphic CTMCs (but via different labelled transition systems).

The renaming of activities is systematic so that it is possible to recover the transition system of the PEPA net from the transition system of the PEPA model.

Compiling PEPA nets to PEPA

The PEPA net compiler compiles a PEPA net to a PEPA model. Activities are renamed to enforce the PEPA net idiom that components at different places cannot synchronise on transitions.

The given net and the generated PEPA model produce isomorphic CTMCs (but via different labelled transition systems).

The renaming of activities is systematic so that it is possible to recover the transition system of the PEPA net from the transition system of the PEPA model.

Compiling PEPA nets to PEPA

The PEPA net compiler compiles a PEPA net to a PEPA model. Activities are renamed to enforce the PEPA net idiom that components at different places cannot synchronise on transitions.

The given net and the generated PEPA model produce isomorphic CTMCs (but via different labelled transition systems).

The renaming of activities is systematic so that it is possible to recover the transition system of the PEPA net from the transition system of the PEPA model.

Compiling PEPA nets to PEPA

The PEPA net compiler compiles a PEPA net to a PEPA model. Activities are renamed to enforce the PEPA net idiom that components at different places cannot synchronise on transitions.

The given net and the generated PEPA model produce isomorphic CTMCs (but via different labelled transition systems).

The renaming of activities is systematic so that it is possible to recover the transition system of the PEPA net from the transition system of the PEPA model.

Solving larger PEPA nets

The motivation for compiling PEPA nets to PEPA models is to use the range of tools available for PEPA.

Solving larger PEPA nets

The motivation for compiling PEPA nets to PEPA models is to use the range of tools available for PEPA.

We solved the secure web service model using

- the PEPA net compiler;
- Jeremy Bradley's Imperial PEPA compiler; and
- Will Knottenbelt's DNAmaca Petri net analyser.

Solving larger PEPA nets

The motivation for compiling PEPA nets to PEPA models is to use the range of tools available for PEPA.

We solved the secure web service model using

- the PEPA net compiler;
- Jeremy Bradley's Imperial PEPA compiler; and
- Will Knottenbelt's DNAmaca Petri net analyser.

Alternatives: Möbius, PRISM.

Conclusions

PEPA nets are a high-level modelling language addressing the performance aspects of the design of modern software systems.

Unlike a Petri net, tokens are programmable components, allowing direct modelling of stateful objects.

Evaluation contexts at the places of the net allow the modeller to represent different areas of computation.

Tools exist which support the PEPA nets language.

Future work

It is possible that the PEPA nets language could be extended, necessitating extensions to the existing tool support.

One possibility would be to add a type system which ensures a consistent interface for tokens.

It is possible that the PML$_\nu$ logic should be extended or revised.

Undertaking real-world examples and case studies is a good way to drive this process.

end of slide show