Complexity of automatic verification of cryptographic protocols

Clermont Ferrand
02/02/2017

Vincent Cheval
Equipe Pesto, INRIA, Nancy

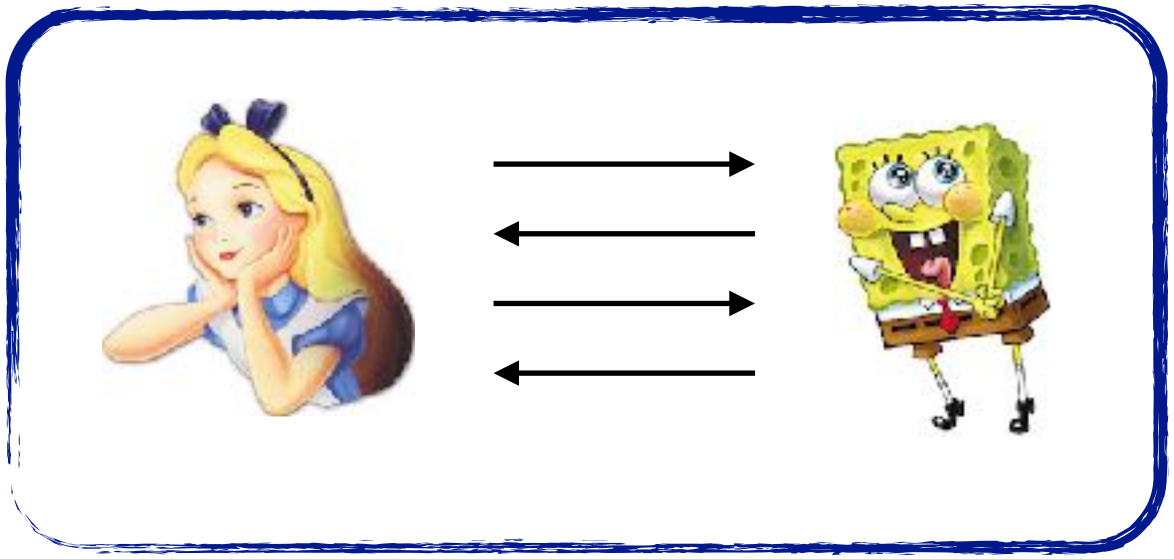
Cryptographic protocols

Communication on **public network**

Cryptographic protocols:

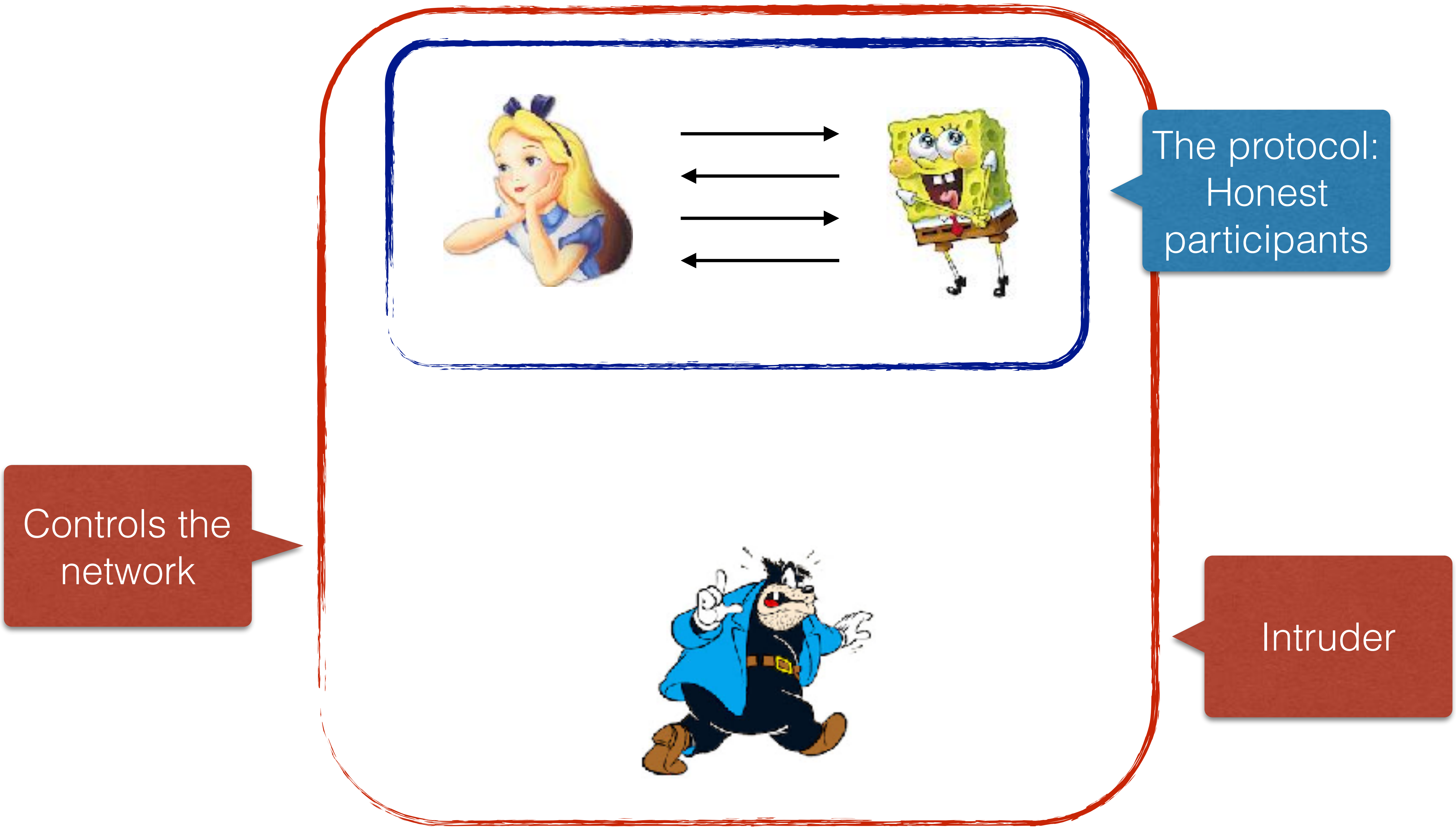
- Concurrent programs designed to secure communications
- Rely on cryptographic primitives

Context

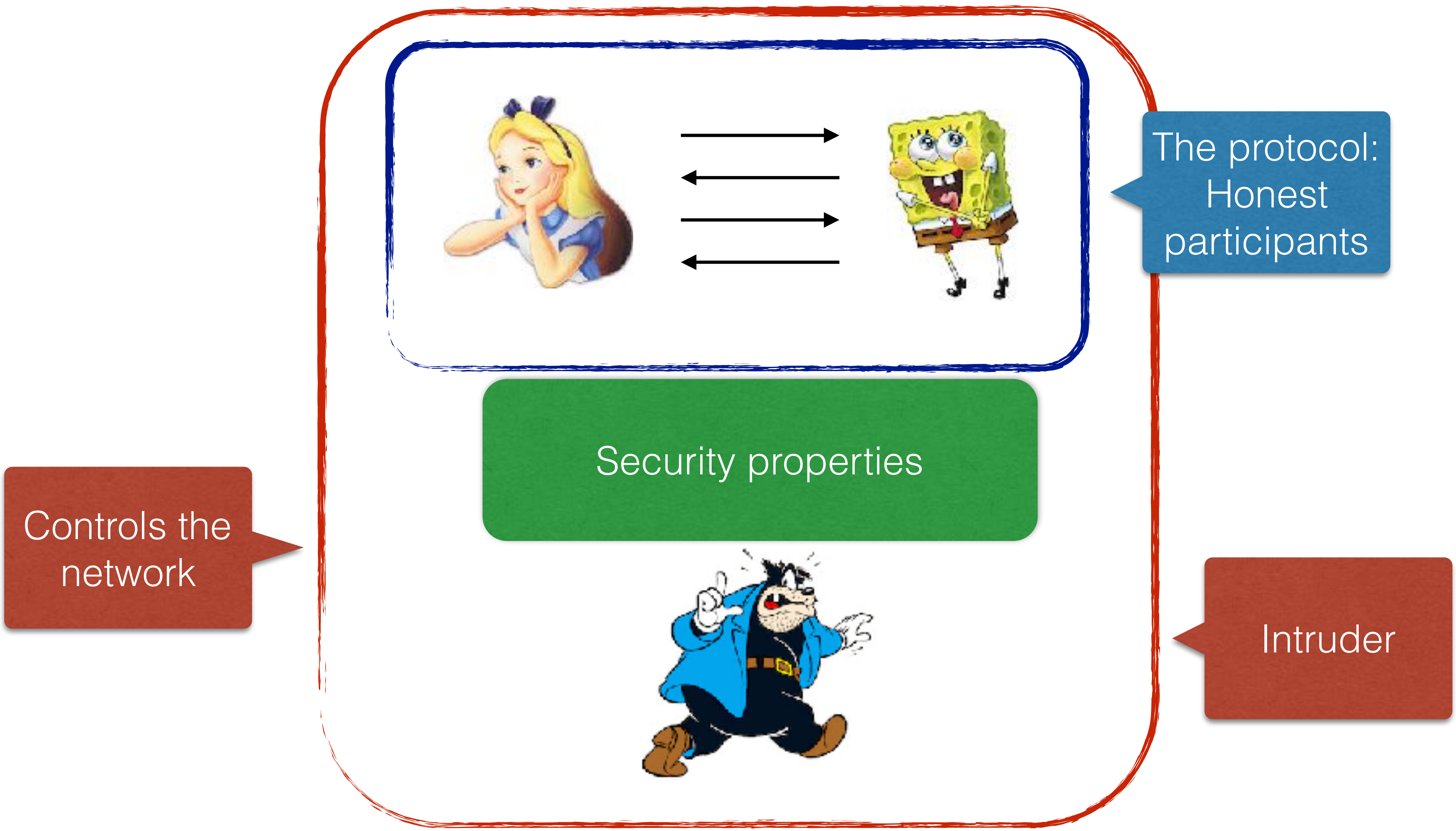


The protocol:
Honest
participants

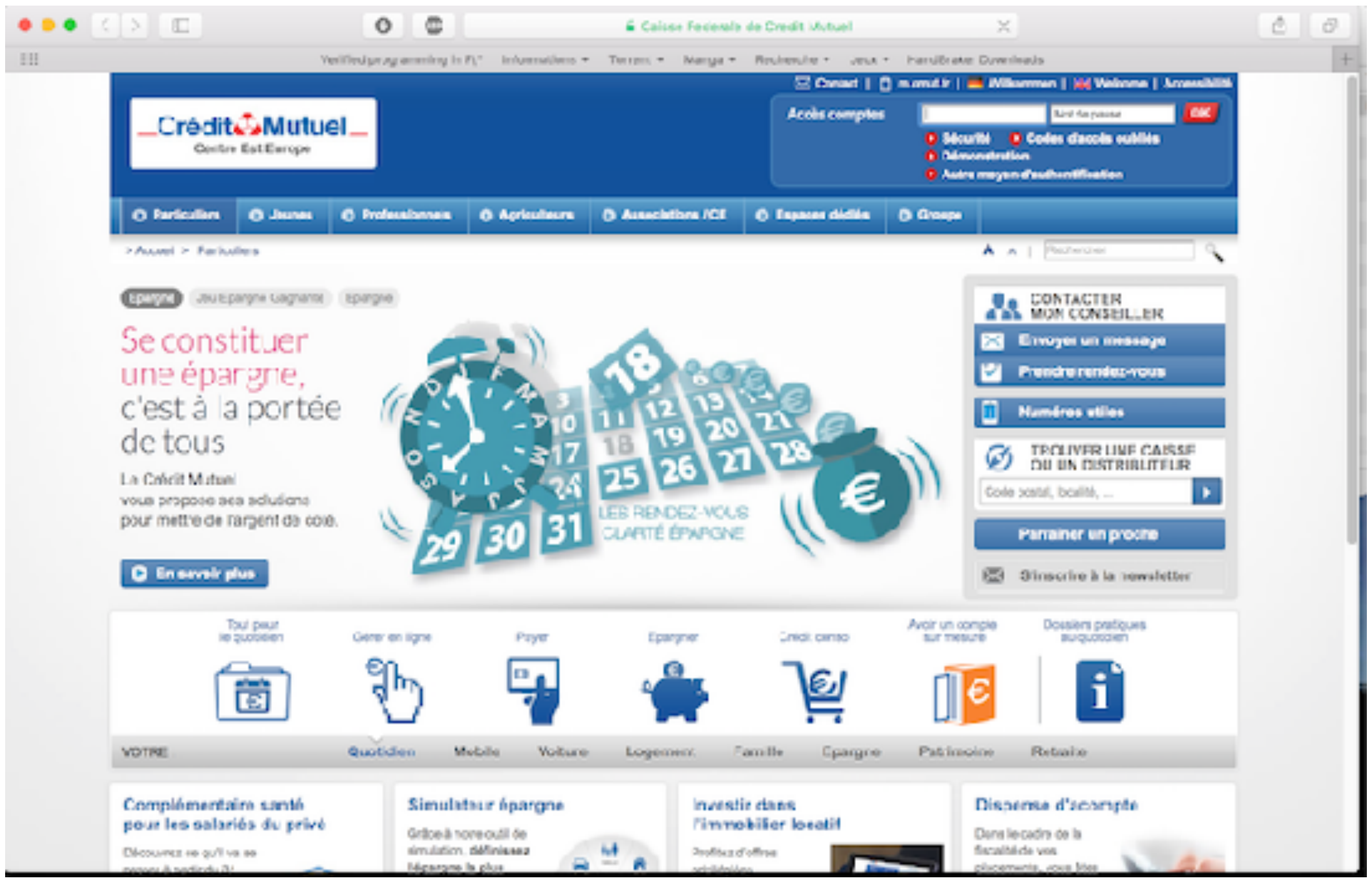
Context


The protocol: Honest participants
Controls the network
Intruder

Context


The protocol:
Honest
participants
Security properties
Controls the
network
Intruder

On the web



Protocol HTTPS

- Password based authentication
- Confidentiality of personal data

Payment with credit card

- Authorization through PIN code
- Wireless payment

- Confidentiality of the transaction
- Authenticity of the bank card

Electronic passport

- RFID chip inside the passport
- Secret key printed on the passport

- Confidentiality of personal data
- Anonymity
- Untraceability

Electronic voting

- Vote from personal computer
- Vote from dedicated machines

- Verifiability of the votes
- Confidentiality of the vote
- No partial results
- One vote per voter
- Anonymity of the voter
- Coercition resistance

Attacks

Designing a secure protocol is hard !

Concrete attacks on:

- authentication used by Google Apps
- unlinkability of french passports
- authentication of credit card (Yes-Card)
- vote privacy on the Helios e-voting system
- anonymity of routing protocols
- …

These attacks are the consequence of a bad design and not of a:

- implementation bug
- weak cryptographic primitives
- usage of magical hacking techniques

Existing models

Cryptographic model

Symbolic model

Existing models

Cryptographic model		Symbolic model
Bitstring	Messages	

Existing models

Cryptographic model		Symbolic model
Bitstring	Messages	
Real algorithms	Cryptographic primitives	

Existing models

Cryptographic model		Symbolic model
Bitstring	Messages	
Real algorithms	Cryptographic primitives	Function symbols

Existing models

Cryptographic model		Symbolic model
Bitstring	Messages	Terms
Real algorithms	Cryptographic primitives	Function symbols

Existing models

Cryptographic model		Symbolic model
Bitstring	Messages	Terms
Real algorithms	Cryptographic primitives	Function symbols
PPT	Attacker	Idealized

Existing models

Cryptographic model		Symbolic model
Bitstring	Messages	Terms
Real algorithms	Cryptographic primitives	Function symbols
PPT	Attacker	Idealized
Difficult and by hand	Proofs	"Easier" and mechanized

Existing models

Cryptographic model		Symbolic model
Bitstring	Messages	Terms
Real algorithms	Cryptographic primitives	Function symbols
PPT	Attacker	Idealized
Difficult and by hand	Proofs	"Easier" and mechanized
Strong and clear	Security guarantees	Unclear

Symbolic models

Symbolic terms

Nonces: $a, b, c, \ldots$ Variables: $x, y, z, \ldots$

Functions symbols: $\mathsf{enc}, \mathsf{dec}, \langle\ \rangle, \oplus, \ldots$

$\mathsf{enc}(x, y)$ $\langle x, y \rangle$ $a \oplus x$

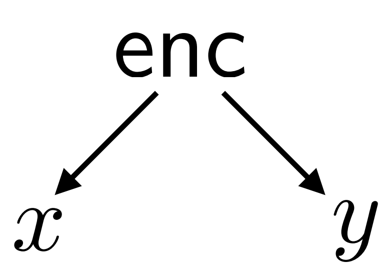
Symbolic models

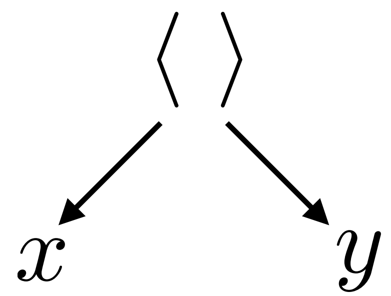
Symbolic terms

Nonces: $a, b, c, \ldots$ Variables: $x, y, z, \ldots$

Functions symbols: $\mathsf{enc}, \mathsf{dec}, \langle\ \rangle, \oplus, \ldots$

$\mathsf{enc}(x, y)$ $\langle x, y \rangle$ $a \oplus x$





Rewrite rules

$\mathsf{dec}(\mathsf{enc}(x, y), y) \rightarrow x$ $\mathsf{proj}_1(\langle x, y \rangle) \rightarrow x$

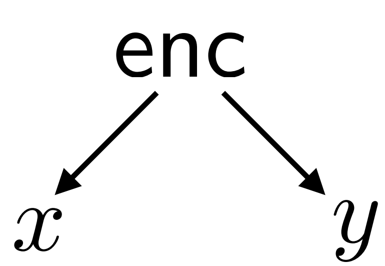
Symbolic models

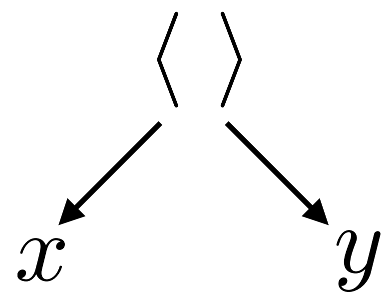
Symbolic terms

Nonces: $a, b, c, \ldots$ Variables: $x, y, z, \ldots$

Functions symbols: $\mathsf{enc}, \mathsf{dec}, \langle\ \rangle, \oplus, \ldots$

$\mathsf{enc}(x, y)$ $\langle x, y \rangle$ $a \oplus x$





Rewrite rules

$\mathsf{dec}(\mathsf{enc}(x, y), y) \rightarrow x$ $\mathsf{proj}_1(\langle x, y \rangle) \rightarrow x$

Example: $\mathsf{dec}(\mathsf{enc}(\langle m_1, m_2 \rangle, k), k) \rightarrow \langle m_1, m_2 \rangle$

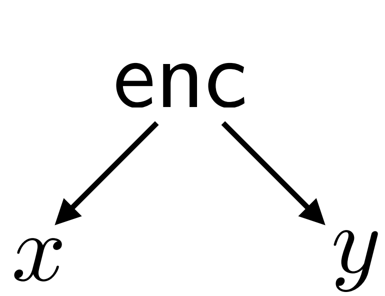
Symbolic models

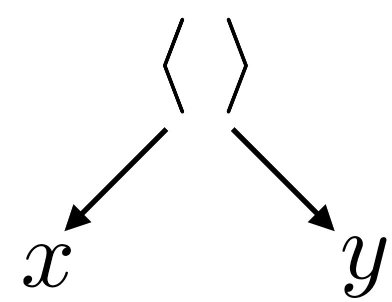
Symbolic terms

Nonces: $a, b, c, \ldots$ Variables: $x, y, z, \ldots$

Functions symbols: enc, dec, $\langle\ \rangle, \oplus, \ldots$

$\mathsf{enc}(x, y)$ $\langle x, y \rangle$ $a \oplus x$





Subterm convergent

Rewrite rules

$\mathsf{dec}(\mathsf{enc}(x, y), y) \to x$ $\mathsf{proj}_1(\langle x, y \rangle) \to x$

Example: $\mathsf{dec}(\mathsf{enc}(\langle m_1, m_2 \rangle, k), k) \to \langle m_1, m_2 \rangle$

Monadic convergent

$\mathsf{unblind}(\mathsf{sign}(\mathsf{blind}(x, y), z), y) \to \mathsf{sign}(x, z)$

Symbolic models

Example : Electronic passport

Knows k_e | Knows k_e

Passeport

n_T →

← $\mathsf{enc}(\langle n_R, n_T, k_R \rangle, k_e)$

$\mathsf{enc}(\langle n_T, n_R, k_T \rangle, k_e)$ →

Reader

Symbolic models

Example : Electronic passport

Freshly generated by Passport

Knows k_e

Knows k_e

Passeport

n_T

$\mathsf{enc}(\langle n_R, n_T, k_R \rangle, k_e)$

$\mathsf{enc}(\langle n_T, n_R, k_T \rangle, k_e)$

Reader

Symbolic models

Example : Electronic passport

Freshly generated by Passport

Knows k_e

Knows k_e

Passeport

n_T →

← $\mathsf{enc}(\langle n_R, n_T, k_R \rangle, k_e)$

$\mathsf{enc}(\langle n_T, n_R, k_T \rangle, k_e)$ →

Reader

Reader expects a nonce and returns it

Symbolic models

Example : Electronic passport

Freshly generated by Passport

Knows k_e

Passeport

n_T

enc$(\langle n_R, n_T, k_R \rangle, k_e)$

enc$(\langle n_T, n_R, k_T \rangle, k_e)$

Knows k_e

Reader

Reader expects a nonce and returns it

Freshly generated by Reader

Symbolic models

Example : Electronic passport

Knows k_e — Knows k_e

Passeport

n_T →

← $\mathsf{enc}(\langle n_R, n_T, k_R \rangle, k_e)$

$\mathsf{enc}(\langle n_T, n_R, k_T \rangle, k_e)$ →

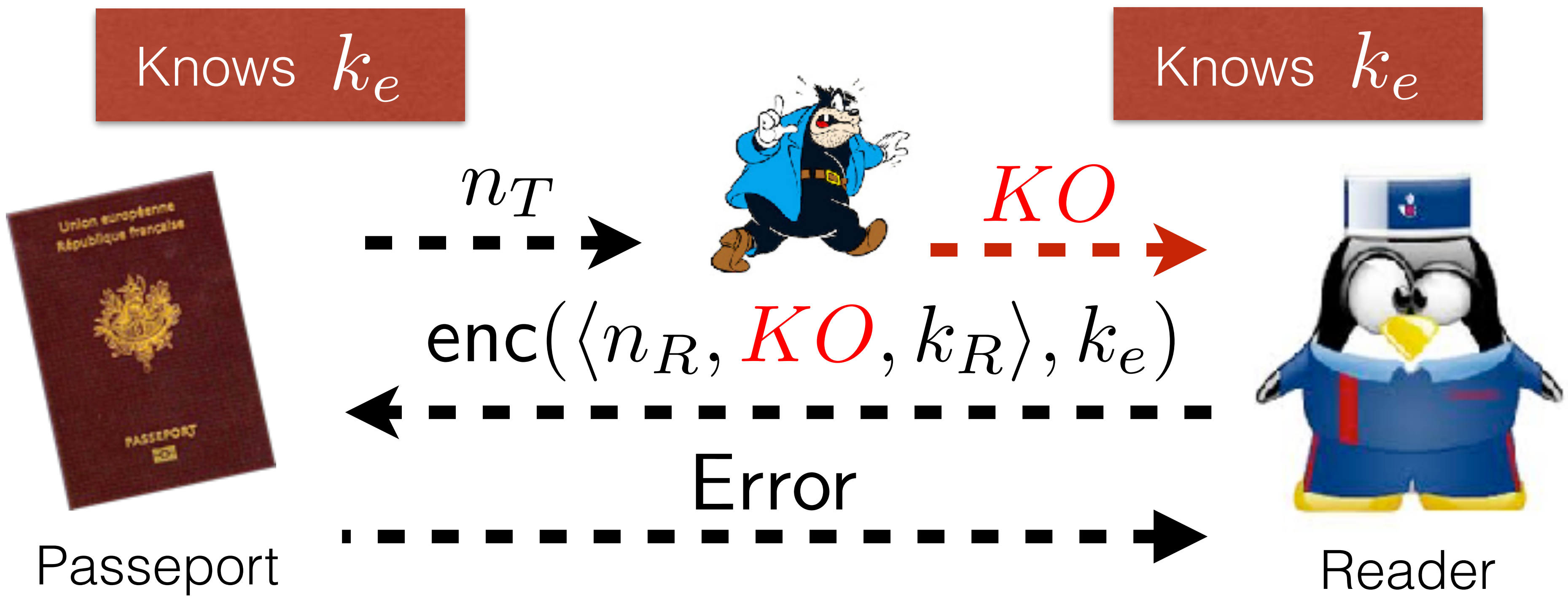
Reader

Knowledge of intruder: nonces he can generate +

1: n_T
2: $\mathsf{enc}(\langle n_R, n_T, k_R \rangle, k_e)$
3: $\mathsf{enc}(\langle n_T, n_R, k_T \rangle, k_e)$

Symbolic models

Another trace of the Electronic passport



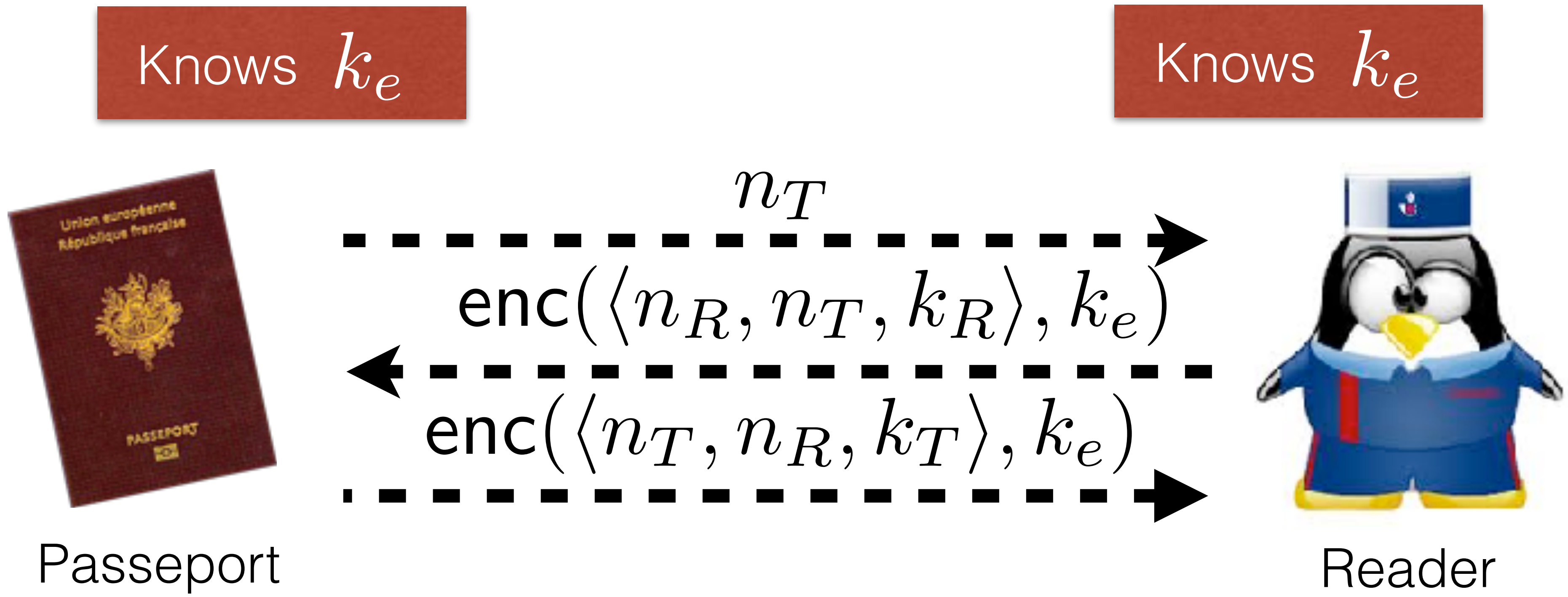
Knowledge of intruder: nonces he can generate (e.g. KO) +

1: n_T
2: enc($\langle n_R, KO, k_R \rangle, k_e$)
3: Error

Applied pi calculus

0	Nil
$P + Q$	Non deterministic choice
$P \mid Q$	Parallel
$\mathsf{if}\ u = v\ \mathsf{then}\ P\ \mathsf{else}\ Q$	Test
$\mathsf{in}(c, x).P$	Input
$\mathsf{out}(c, u).P$	Output
$\nu k.P$	Name restriction
$!P$	Replication

Applied pi calculus



$$
\begin{aligned}
P(k_e) = \quad & \nu n_T.\mathsf{out}(c, n_T).\mathsf{in}(c, x). \\
& \mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then} \\
& \quad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T \rangle) \\
& \mathsf{else}\ \mathsf{out}(c, \mathsf{Error})
\end{aligned}
$$

Main process: $!\nu k_e !P(k_e) \mid R(k_e)$

Trace

A trace = one execution of the process

$$P(k_e) = \quad \nu n_T.\mathsf{out}(c, n_T).\mathsf{in}(c, x).$$
$$\mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then}$$
$$\nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T \rangle)$$
$$\mathsf{else}\ \mathsf{out}(c, \mathsf{Error})$$

Initial configuration: $(\emptyset, \nu k_e.P(k_e), id)$

Set of private names

The process

Substitution representing the knowledge of the attacker

Trace

$$\begin{aligned} P(k_e) = \quad & \nu n_T.\mathsf{out}(c, n_T).\mathsf{in}(c, x). \\ & \mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then} \\ & \quad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T \rangle) \\ & \mathsf{else}\ \mathsf{out}(c, \mathsf{Error}) \end{aligned}$$

$$(\emptyset, \nu k_e.P(k_e), id)$$

w_1

Trace

$$
\begin{aligned}
P(k_e) = \quad & \nu n_T.\mathsf{out}(c, n_T).\mathsf{in}(c, x). \\
& \mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then} \\
& \quad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T \rangle) \\
& \mathsf{else}\ \mathsf{out}(c, \mathsf{Error})
\end{aligned}
$$

$$(\emptyset, \nu k_e.P(k_e), id) \quad \longrightarrow \quad (\{k_e\}, P(k_e), id)$$

w_1

Trace

$$
\begin{aligned}
P_1 = \quad & \mathsf{out}(c, n_T).\mathsf{in}(c, x). \\
& \mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then} \\
& \quad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T \rangle) \\
& \mathsf{else}\ \mathsf{out}(c, \mathsf{Error})
\end{aligned}
$$

$$
\begin{aligned}
(\emptyset, \nu k_e.P(k_e), id) \quad & \longrightarrow \quad (\{k_e\}, P(k_e), id) \\
& \longrightarrow \quad (\{k_e, n_T\}, P_1, id)
\end{aligned}
$$

w_1

Trace

$$P_2 = \quad \mathsf{in}(c, x). \\ \mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then} \\ \quad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T\rangle) \\ \mathsf{else}\ \mathsf{out}(c, \mathsf{Error})$$

$$(\emptyset, \nu k_e.P(k_e), id) \longrightarrow (\{k_e\}, P(k_e), id)$$

$$\longrightarrow (\{k_e, n_T\}, P_1, id)$$

$$\xrightarrow{out(c, w_1)} (\{k_e, n_T\}, P_2, \{{}^{n_T}/_{w_1}\})$$

Trace

$$
\begin{aligned}
P_2 = \quad & \mathsf{in}(c, x). \\
& \mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then} \\
& \quad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T \rangle) \\
& \mathsf{else}\ \mathsf{out}(c, \mathsf{Error})
\end{aligned}
$$

$$
\begin{aligned}
(\emptyset, \nu k_e.P(k_e), id) \quad & \longrightarrow \quad (\{k_e\}, P(k_e), id) \\
& \longrightarrow \quad (\{k_e, n_T\}, P_1, id) \\
& \xrightarrow{out(c, w_1)} \quad (\{k_e, n_T\}, P_2, \{^{n_T}/_{w_1}\}) \\
& \xrightarrow{in(c, M)}
\end{aligned}
$$

Trace

$$P_2 = \ \mathsf{in}(c, x).$$
$$\quad \mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then}$$
$$\qquad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T\rangle)$$
$$\quad \mathsf{else}\ \mathsf{out}(c, \mathsf{Error})$$

$$(\emptyset, \nu k_e.P(k_e), id) \longrightarrow (\{k_e\}, P(k_e), id)$$
$$\longrightarrow (\{k_e, n_T\}, \color{red}{P_1}, id)$$
$$\xrightarrow{out(c, w_1)} (\{k_e, n_T\}, \color{blue}{P_2}, \{{}^{n_T}/_{w_1}\})$$
$$\xrightarrow{in(c, M)}$$

Cannot contain private names

Trace

$$P_2 = \begin{array}{l} \mathsf{in}(c, x). \\ \mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then} \\ \quad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T\rangle) \\ \mathsf{else}\ \mathsf{out}(c, \mathsf{Error}) \end{array}$$

$$(\emptyset, \nu k_e.P(k_e), id) \longrightarrow (\{k_e\}, P(k_e), id)$$

$$\longrightarrow (\{k_e, n_T\}, P_1, id)$$

$$\xrightarrow{out(c, w_1)} (\{k_e, n_T\}, P_2, \{^{n_T}/_{w_1}\})$$

$$\xrightarrow{in(c, M)}$$

Cannot contain private names

Can contain variables from the frame, i.e. w_1

Trace

$$P_2 = \begin{array}{l} \mathsf{in}(c, x). \\ \mathsf{if}\ \mathsf{proj}_3(\mathsf{dec}(x, k_e)) = n_T\ \mathsf{then} \\ \quad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T\rangle) \\ \mathsf{else}\ \mathsf{out}(c, \mathsf{Error}) \end{array}$$

$$(\emptyset, \nu k_e.P(k_e), id) \longrightarrow (\{k_e\}, P(k_e), id)$$

$$\longrightarrow (\{k_e, n_T\}, P_1, id)$$

$$\xrightarrow{out(c, w_1)} (\{k_e, n_T\}, P_2, \{^{n_T}/_{w_1}\})$$

$$\xrightarrow{in(c, M)}$$

Cannot contain private names

Can contain variables from the frame, i.e. w_1

Ex: $M = \langle n_I, w_1 \rangle$

Trace

$$P_2 = \begin{array}{l} \mathsf{in}(c, x). \\ \mathsf{if\ proj}_3(\mathsf{dec}(x, k_e)) = n_T \mathsf{\ then} \\ \quad \nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(x, k_e)), k_T\rangle) \\ \mathsf{else\ out}(c, \mathsf{Error}) \end{array}$$

$$(\emptyset, \nu k_e.P(k_e), id) \longrightarrow (\{k_e\}, P(k_e), id)$$

$$\longrightarrow (\{k_e, n_T\}, P_1, id)$$

$$\xrightarrow{out(c, w_1)} (\{k_e, n_T\}, P_2, \{^{n_T}/_{w_1}\})$$

$$\xrightarrow{in(c, M)}$$

Cannot contain private names

Can contain variables from the frame, i.e. w_1

Ex: $M = \langle n_I, w_1 \rangle$ $\quad x \to \langle n_I, n_T \rangle$

Trace

$$P_3 = \quad \text{if } \mathsf{proj}_3(\mathsf{dec}(\langle n_I, n_T\rangle, k_e)) = n_T \text{ then}$$
$$\nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(\langle n_I, n_T\rangle, k_e)), k_T\rangle)$$
$$\text{else } \mathsf{out}(c, \mathsf{Error})$$

$$(\emptyset, \nu k_e.P(k_e), id) \longrightarrow (\{k_e\}, P(k_e), id)$$
$$\longrightarrow (\{k_e, n_T\}, P_1, id)$$
$$\xrightarrow{out(c, w_1)} (\{k_e, n_T\}, P_2, \{^{n_T}/_{w_1}\})$$
$$\xrightarrow{in(c, M)} (\{k_e, n_T\}, P_3, \{^{n_T}/_{w_1}\})$$

Cannot contain private names

Can contain variables from the frame, i.e. w_1

Ex: $M = \langle n_I, w_1\rangle$ $\quad x \to \langle n_I, n_T\rangle$

Trace

$$P_3 = \quad \text{if } \mathsf{proj}_3(\mathsf{dec}(\langle n_I, n_T\rangle, k_e)) = n_T \text{ then}$$
$$\nu k_T.\mathsf{out}(c, \langle n_T, \mathsf{proj}_1(\mathsf{dec}(\langle n_I, n_T\rangle, k_e)), k_T\rangle)$$
$$\text{else } \mathsf{out}(c, \mathsf{Error})$$

$$(\emptyset, \nu k_e.P(k_e), id) \xrightarrow{\quad} (\{k_e\}, P(k_e), id)$$
$$\xrightarrow{\quad} (\{k_e, n_T\}, P_1, id)$$
$$\xrightarrow{out(c, w_1)} (\{k_e, n_T\}, P_2, \{^{n_T}/_{w_1}\})$$
$$\xrightarrow{in(c, M)} (\{k_e, n_T\}, P_3, \{^{n_T}/_{w_1}\})$$

The sequence of label $\mathsf{out}(c, w_1).\mathsf{in}(c, M)$ with the set $\{k_e, n_T\}$ and the frame $\{^{n_T}/_{w_1}\}$ represents a trace.

Internal communication

Communication can happen internally
on private channels

$$(\{d\}, \mathsf{in}(d, x).P \mid \mathsf{out}(d, a).Q, \Phi)$$

$$\longrightarrow (\{d\}, P\{^a/_x\} \mid Q, \Phi)$$

Security properties

- What we verify well : Confidentiality, authenticity

Trace properties

Tools : Avispa, ProVerif, Scyther, CSP/FdR…

- What we don't verify well : Anonymity, privacy, traceability, properties for electronic voting

Equivalence properties

Tools : ProVerif, AkiSs, SPEC, APTE

Trace equivalence

Untraceability of electronic passport

Situation 1

Situation 2

Trace equivalence

Untraceability of electronic passport

Situation 1

Situation 2

Trace equivalence

Untraceability of electronic passport

Situation 1

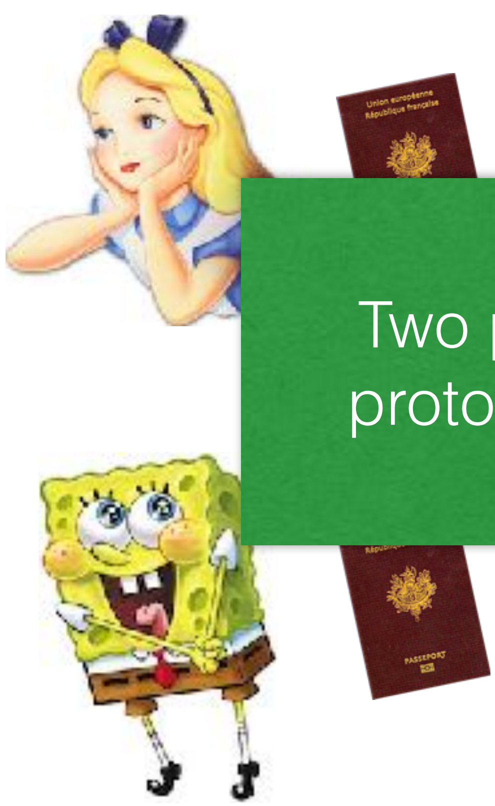
Situation 2

Two protocols are in equivalence if for all traces of one of the protocol, we can find an equivalent trace in the other protocol

Trace equivalence

Untraceability of electronic passport

Situation 1



Situation 2

Two protocols are in equivalence if for all traces of one of the protocol, we can find an equivalent trace in the other protocol

$M_1, M_2, \ldots, M_k$

Knowledges of the attacker obtain in two traces with similar actions

$N_1, N_2, \ldots, N_k$

Decision procedure

How to automatically decide equivalence ?

- General problem: Undecidable
- Usual restrictions: Bounded number of sessions, stronger notion of equivalence, simple algebraic properties for the cryptographic properties,…
- Technics used : Saturation of Horn Clauses, Constraint solving, Equivalence of constraint systems

Decision procedure

How difficult is it to automatically decide equivalence ?

- P : Decidable by a deterministic Turing machine in polynomial time
- EXP : Decidable by a deterministic Turing machine in exponential time
- NP : Decidable by a non-deterministic Turing machine in polynomial time
- NEXP : Decidable by a non-deterministic Turing machine in exponential time
- PSPACE : Decidable by a deterministic Turing machine in polynomial space

- $\Sigma_0 = P$ and $\Sigma_i = NP^{\Sigma_{i-1}}$
- $\Sigma_1 = NP$

Complexity

Applied Pi Calculus	Static equivalence	Trace equivalence	Observational equivalence	Diff equivalence
Positive, finite, subterm convergent	P complete [AC'04]	Decidable	?	CoNP complete
Finite, subterm convergent	P complete [AC'04]	Decidable	?	?
Finite, monadic convergent	P hard	?	?	?

Complexity Results

Applied Pi Calculus	Static equivalence	Trace equivalence	Observational equivalence	Diff equivalence
Positive, finite, subterm convergent	P complete [AC'04]	Decidable [CK'17]	coNEXP hard	CoNP complete
Finite, subterm convergent	P complete [AC'04]	Decidable [CK'17] coNEXP hard	coNEXP hard	?
Finite, monadic convergent	P hard	coNEXP hard	coNEXP hard	?

Pure Pi Calculus	Static equivalence	Trace equivalence	Observational equivalence
Positive, finite	LOGSPACE	$co\Sigma_2$ complete	PSPACE easy $co\Sigma_4$ hard
Finite	LOGSPACE	$co\Sigma_2$ complete	PSPACE easy $co\Sigma_4$ hard

Complexity Results

Applied Pi Calculus	Static equivalence	Trace equivalence	Observational equivalence	Diff equivalence
Positive, finite, subterm convergent	PTIME complete [AC'04]	Decidable [CK'17] coNEXP easy ?	coNEXP hard coNEXP complete ?	CoNP complete
Finite, subterm convergent	PTIME complete [AC'04]	Decidable [CK'17] coNEXP hard coNEXP complete ?	coNEXP hard coNEXP complete ?	coNP complete ?
Finite, monadic convergent	PTIME hard	coNEXP hard	coNEXP hard	?

Pure Pi Calculus	Static equivalence	Trace equivalence	Observational equivalence
Positive, finite	LOGSPACE	coΣ_2 complete	PSPACE easy coΣ_4 hard
Finite	LOGSPACE	coΣ_2 complete	PSPACE easy coΣ_4 hard

Trace equivalence in pi-calculus

Reduction from QSAT_2

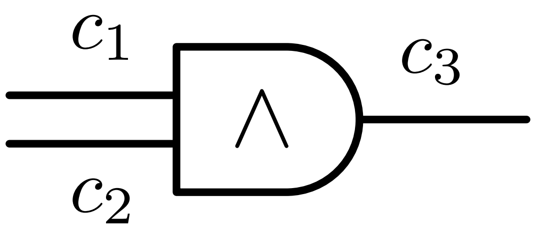
No else branche, no cryptographic primitive

$$A \not\approx_{tr} B \text{ iff } \exists \vec{x}\, \forall \vec{y},\ \varphi(\vec{x}, \vec{y}) = 1$$

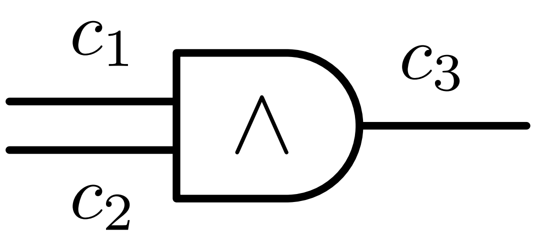
$\exists \vec{x} = \exists x_1 \exists x_2 \ldots \exists x_n$

$\forall \vec{y} = \forall y_1 \forall y_2 \ldots \forall y_m$

Boolean formula in pi-calculus

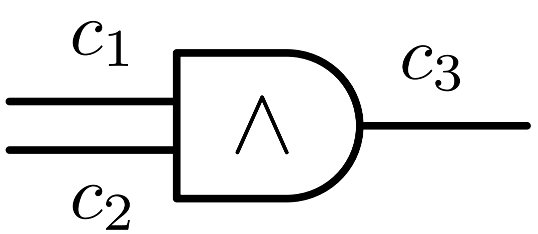


Boolean formula in pi-calculus



$$
\begin{array}{l}
\mathsf{in}(c_1, x).\mathsf{in}(c_2, y).(\\
\quad\quad \mathsf{if}\ x = 1\ \mathsf{then}\ \mathsf{if}\ y = 1\ \mathsf{then}\ \mathsf{out}(c_3, 1) \\
\quad \mid \mathsf{if}\ x = 1\ \mathsf{then}\ \mathsf{if}\ y = 0\ \mathsf{then}\ \mathsf{out}(c_3, 0) \\
\quad \mid \mathsf{if}\ x = 0\ \mathsf{then}\ \mathsf{if}\ y = 0\ \mathsf{then}\ \mathsf{out}(c_3, 0) \\
\quad \mid \mathsf{if}\ x = 0\ \mathsf{then}\ \mathsf{if}\ y = 0\ \mathsf{then}\ \mathsf{out}(c_3, 0) \\
)
\end{array}
$$

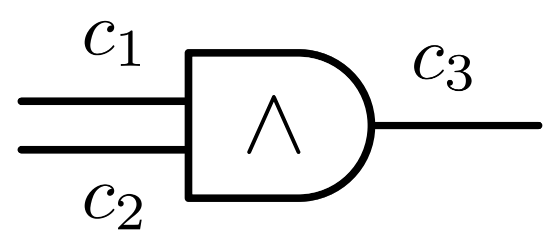
Boolean formula in pi-calculus



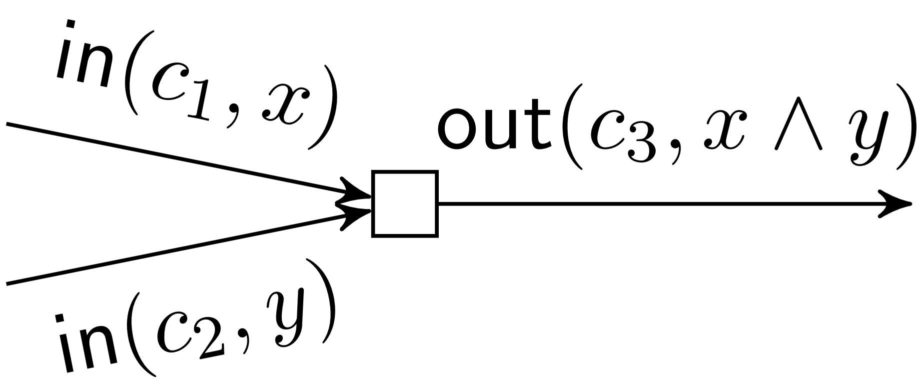
$$
\begin{aligned}
&\mathsf{in}(c_1, x).\mathsf{in}(c_2, y).(\\
&\quad\ \ \mathsf{if}\ x = 1\ \mathsf{then}\ \mathsf{if}\ y = 1\ \mathsf{then}\ \mathsf{out}(c_3, 1) \\
&\quad \mid \mathsf{if}\ x = 1\ \mathsf{then}\ \mathsf{if}\ y = 0\ \mathsf{then}\ \mathsf{out}(c_3, 0) \\
&\quad \mid \mathsf{if}\ x = 0\ \mathsf{then}\ \mathsf{if}\ y = 0\ \mathsf{then}\ \mathsf{out}(c_3, 0) \\
&\quad \mid \mathsf{if}\ x = 0\ \mathsf{then}\ \mathsf{if}\ y = 0\ \mathsf{then}\ \mathsf{out}(c_3, 0) \\
&)
\end{aligned}
$$

Enforces that inputs are booleans !

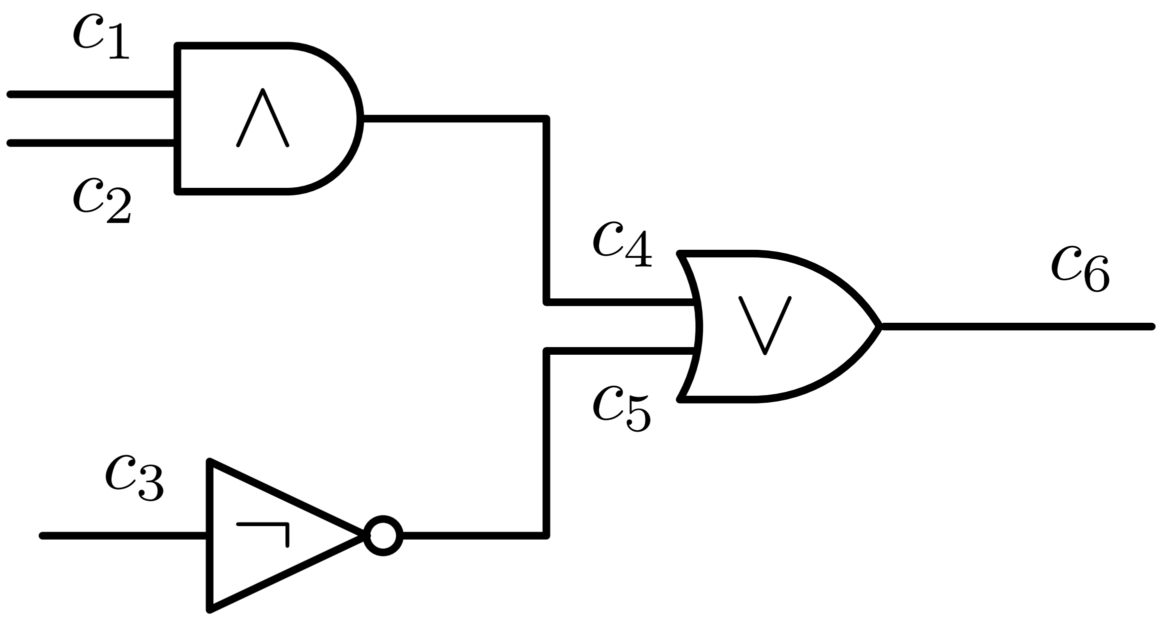
Boolean formula in pi-calculus



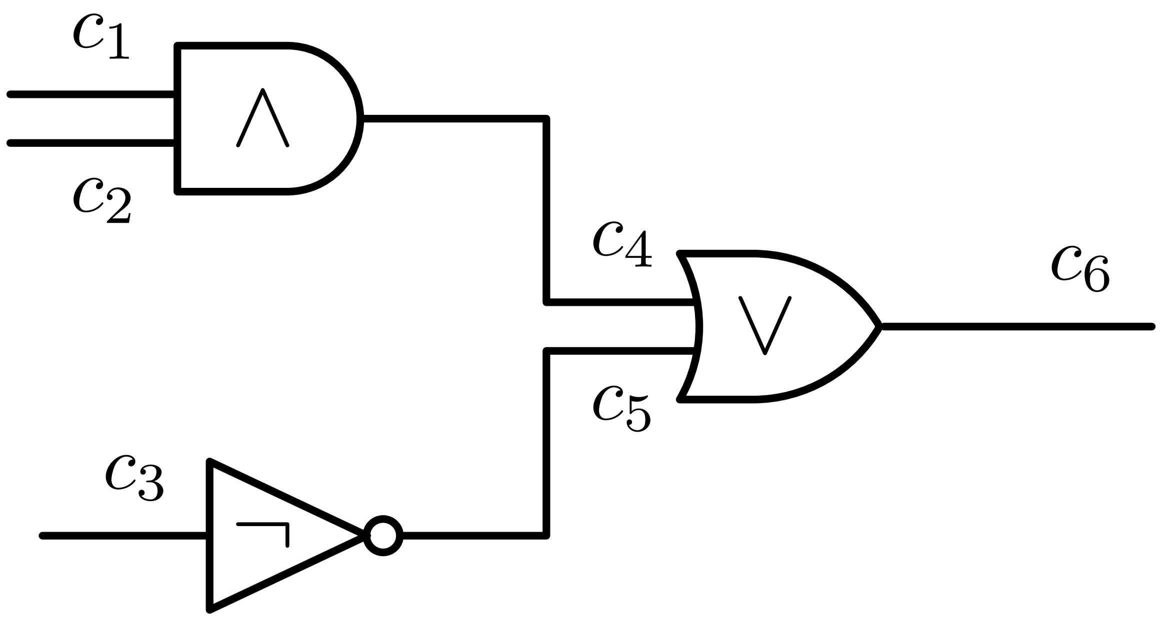
$$
\begin{aligned}
&\mathsf{in}(c_1, x).\mathsf{in}(c_2, y).(\\
&\quad\;\; \mathsf{if}\ x = 1\ \mathsf{then}\ \mathsf{if}\ y = 1\ \mathsf{then}\ \mathsf{out}(c_3, 1) \\
&\quad \mid \mathsf{if}\ x = 1\ \mathsf{then}\ \mathsf{if}\ y = 0\ \mathsf{then}\ \mathsf{out}(c_3, 0) \\
&\quad \mid \mathsf{if}\ x = 0\ \mathsf{then}\ \mathsf{if}\ y = 0\ \mathsf{then}\ \mathsf{out}(c_3, 0) \\
&\quad \mid \mathsf{if}\ x = 0\ \mathsf{then}\ \mathsf{if}\ y = 0\ \mathsf{then}\ \mathsf{out}(c_3, 0) \\
&)
\end{aligned}
$$

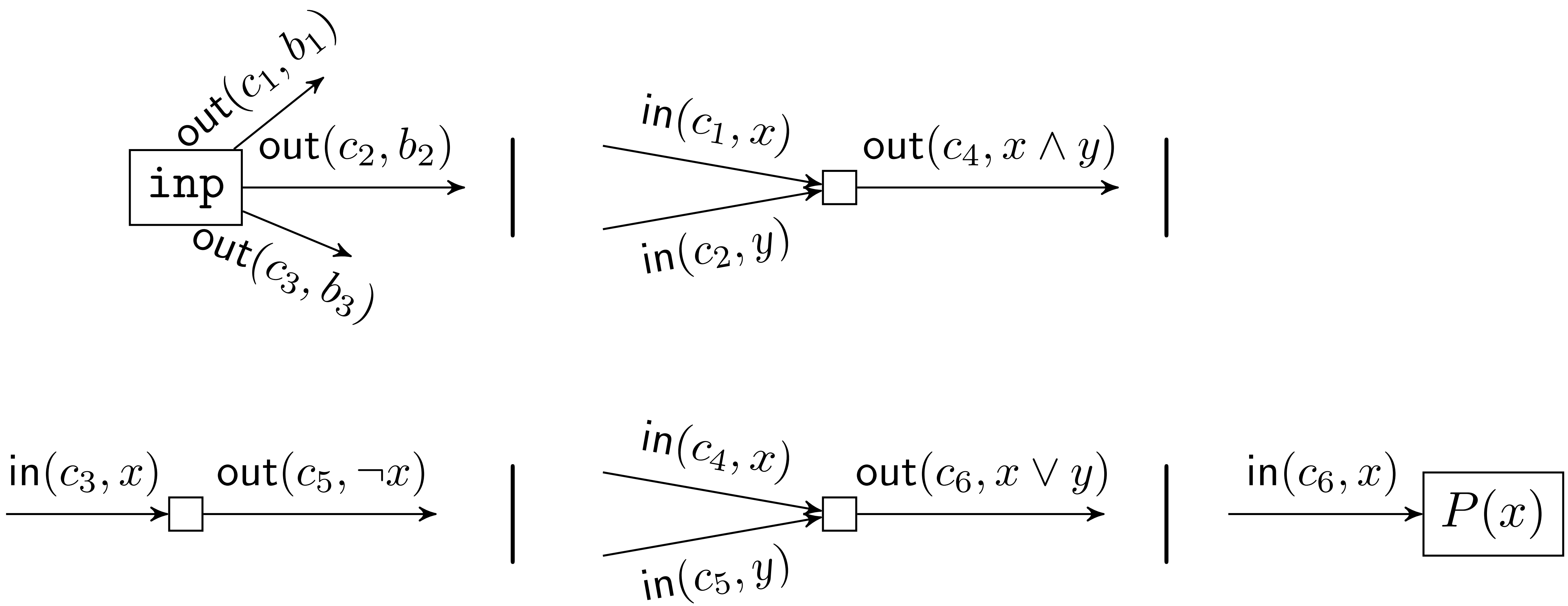


Boolean formula in pi-calculus

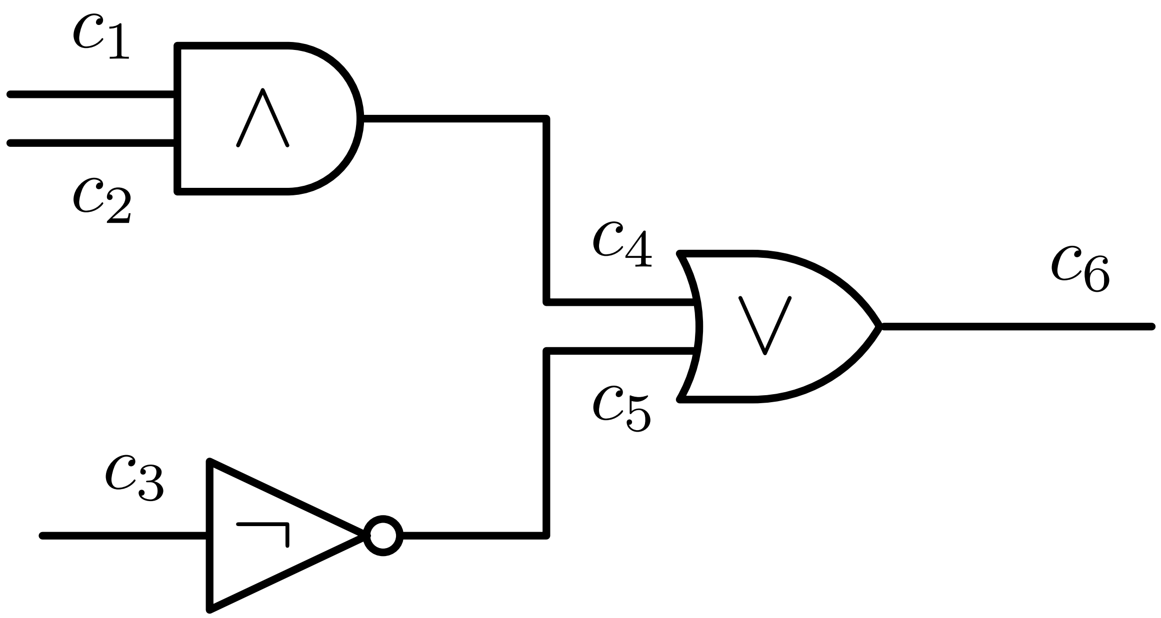


Boolean formula in pi-calculus

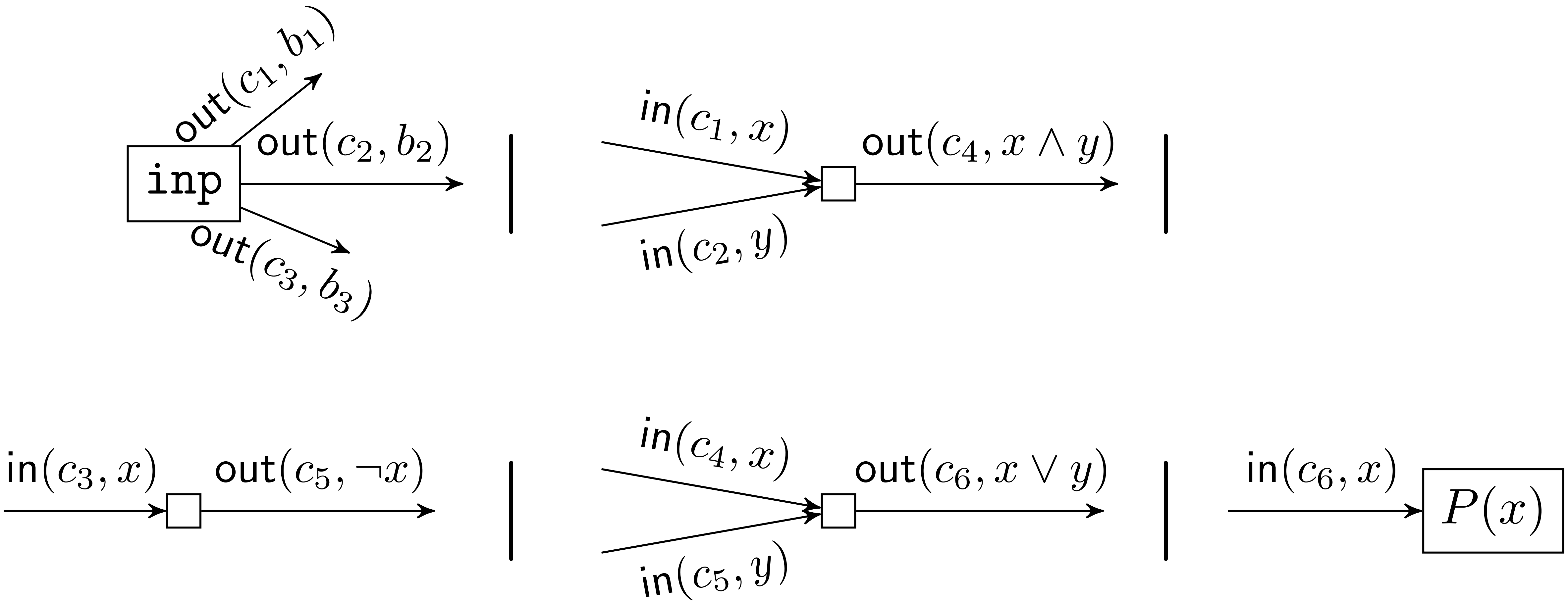




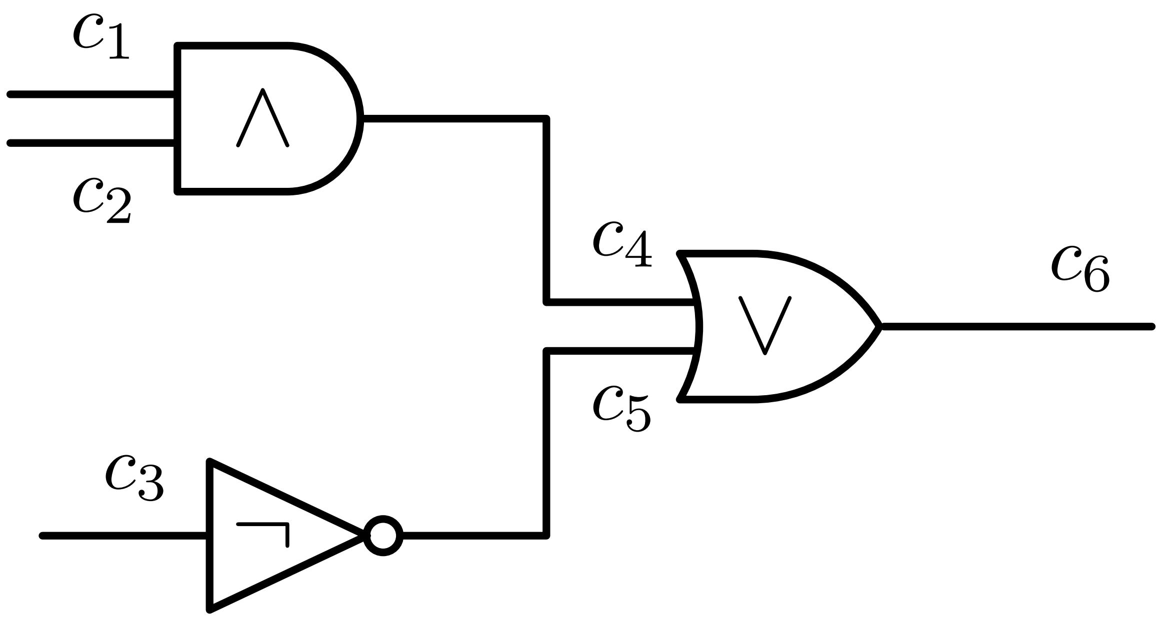
Boolean formula in pi-calculus



Enforces that inputs are booleans !

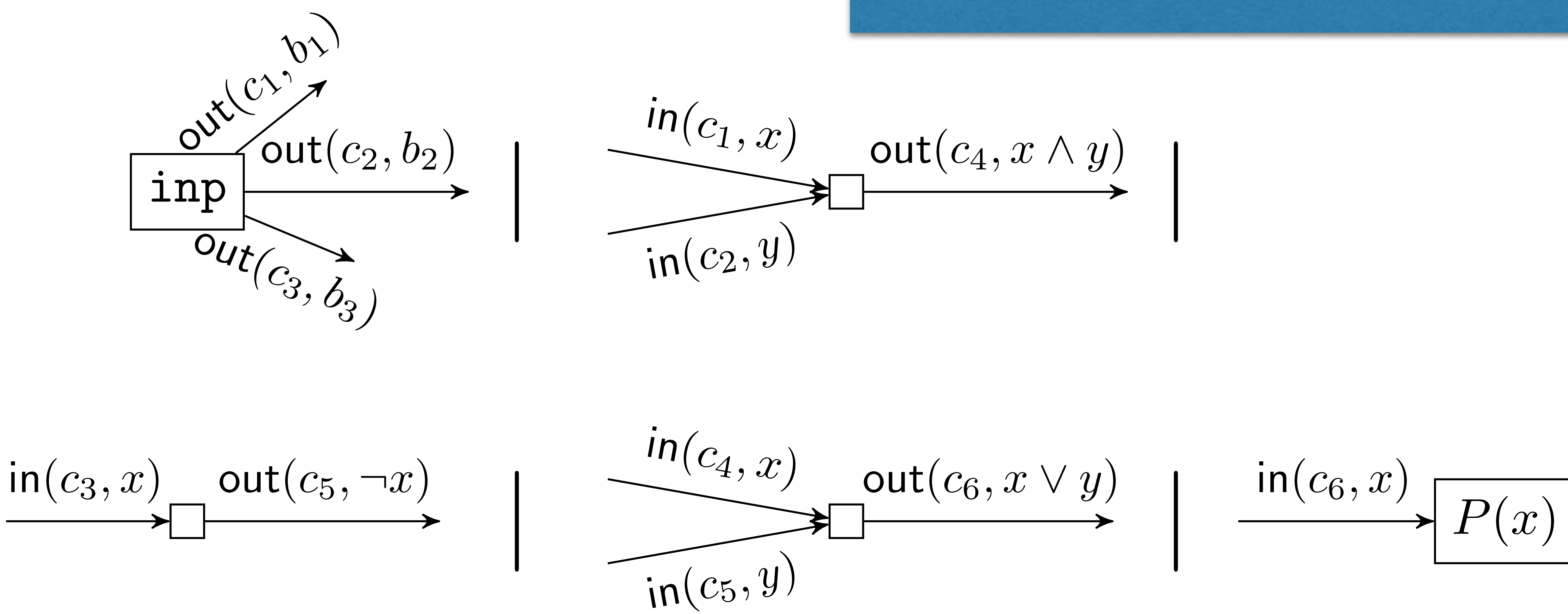


Boolean formula in pi-calculus



Enforces that inputs are booleans !

$$x \leftarrow \varphi(b_1, b_2, b_3).P(x)$$



Universal quantification

$$A \not\approx_{tr} B \text{ iff } \exists \vec{x}\ \forall \vec{y},\ \varphi(\vec{x}, \vec{y}) = 1$$

How to express universality in pi-calculus ?

Universal quantification

$$A \not\approx_{tr} B \text{ iff } \exists \vec{x}\ \forall \vec{y},\ \varphi(\vec{x}, \vec{y}) = 1$$

How to express universality in pi-calculus ?

$$\texttt{Guess}(y).P \stackrel{\text{def}}{=} \nu d.\,((\mathsf{out}(d, 0) + \mathsf{out}(d, 1) \ \mid\ \mathsf{in}(d, y).P)$$

Universal quantification

$$A \not\approx_{tr} B \text{ iff } \exists \vec{x}\ \forall \vec{y},\ \varphi(\vec{x}, \vec{y}) = 1$$

How to express universality in pi-calculus ?

$$\texttt{Guess}(y).P \stackrel{\text{def}}{=} \nu d. ((\mathsf{out}(d, 0) + \mathsf{out}(d, 1) \mid \mathsf{in}(d, y).P)$$

Can reduce in either $P\{^0/_y\}$ or $P\{^1/_y\}$

Reduction

$$A \not\approx_{tr} B \text{ iff } \exists \vec{x}\ \forall \vec{y},\ \varphi(\vec{x}, \vec{y}) = 1$$

$$A \stackrel{\text{def}}{=} \mathsf{in}(c, \vec{x}).\ \mathit{test} \leftarrow \bigwedge \vec{x}.$$
$$\mathtt{Guess}(\vec{y}).$$
$$(v \leftarrow \varphi(\vec{x}, \vec{y}).\ \mathsf{out}(c, v)\ +\ \mathsf{out}(c, 1))$$

$$B \stackrel{\text{def}}{=} \mathsf{in}(c, \vec{x}).\ \mathit{test} \leftarrow \bigwedge \vec{x}.$$
$$\mathsf{out}(c, 0) + \mathsf{out}(c, 1)$$

Reduction

$$A \not\approx_{tr} B \text{ iff } \exists \vec{x}\ \forall \vec{y},\ \varphi(\vec{x}, \vec{y}) = 1$$

$$A \stackrel{\text{def}}{=} \mathsf{in}(c, \vec{x}).\ test \leftarrow \bigwedge \vec{x}.\ \mathtt{Guess}(\vec{y}).\ (v \leftarrow \varphi(\vec{x}, \vec{y}).\ \mathsf{out}(c, v)\ +\ \mathsf{out}(c, 1))$$

$$B \stackrel{\text{def}}{=} \mathsf{in}(c, \vec{x}).\ test \leftarrow \bigwedge \vec{x}.\ \mathsf{out}(c, 0) + \mathsf{out}(c, 1)$$

The processes can only output if the inputs are booleans

Reduction

$$A \not\approx_{tr} B \text{ iff } \exists \vec{x}\ \forall \vec{y},\ \varphi(\vec{x}, \vec{y}) = 1$$

$$A \stackrel{\text{def}}{=} \mathsf{in}(c, \vec{x}).\ test \leftarrow \bigwedge \vec{x}.$$
$$\mathtt{Guess}(\vec{y}).$$
$$(v \leftarrow \varphi(\vec{x}, \vec{y}).\ \mathsf{out}(c, v)\ +\ \mathsf{out}(c, 1))$$

$$B \stackrel{\text{def}}{=} \mathsf{in}(c, \vec{x}).\ test \leftarrow \bigwedge \vec{x}.$$
$$\mathsf{out}(c, 0) + \mathsf{out}(c, 1)$$

The processes can only output if the inputs are booleans

Whatever the boolean input, the process B can always output 0 or 1

Reduction

$$A \not\approx_{tr} B \text{ iff } \exists \vec{x}\ \forall \vec{y},\ \varphi(\vec{x}, \vec{y}) = 1$$

$$A \stackrel{\text{def}}{=} \mathsf{in}(c, \vec{x}).\ test \leftarrow \bigwedge \vec{x}.$$
$$\mathtt{Guess}(\vec{y}).$$
$$(v \leftarrow \varphi(\vec{x}, \vec{y}).\ \mathsf{out}(c, v)\ +\ \mathsf{out}(c, 1))$$

$$B \stackrel{\text{def}}{=} \mathsf{in}(c, \vec{x}).\ test \leftarrow \bigwedge \vec{x}.$$
$$\mathsf{out}(c, 0) + \mathsf{out}(c, 1)$$

The processes can only output if the inputs are booleans

Whatever the boolean input, the process B can always output 0 or 1

If $\forall \vec{y}.\varphi(\vec{x}_0, \vec{y}) = 1$ holds then A can never output 0 when $\vec{x}_0$ is input