Information flow safety in multiparty sessions

Sara Capecchi, Ilaria Castellani and Mariangiola Dezani-Ciancaglini

(TORINO University and INRIA Sophia Antipolis Méditerranée)

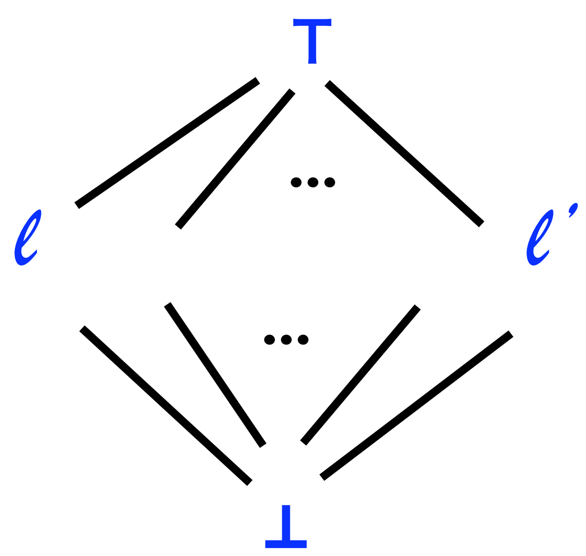
Behavioural Types Workshop Lisbon, 19-21 April 2011

General goal

Information flow control in multiparty sessions where data may have different security levels.

A finite lattice of security levels :

levels assigned to variables and values



Secure information flow: the send or receive of a value v^{ℓ}

can only depend on a receive or test of a value $v_0^{\ell_0}$ with $\ell_0 \leq \ell$

General goal

Information flow control in multiparty sessions, to preserve confidentiality of participant data.

How to prevent / detect information leaks ?

- Typing (prevention): session type system with security
- Security (detection): behavioural property based on observational equivalence / bisimulation

Goal (past)

Information flow control in multiparty sessions,
to preserve confidentiality of participant data.

How to prevent / detect information leaks ?

- Typing (prevention): session type system with security

$\Downarrow$ done in previous work [CCD & Rezk, CONCUR'10]

- Security (detection): behavioural property based on observational equivalence / bisimulation

Goal (present)

Information flow control in multiparty sessions,
to preserve confidentiality of participant data.

How to prevent / detect information leaks ?

- Typing (prevention): session type system with security
- Safety (detection): induced by a monitored semantics
- Security (detection): behavioural property based on observational equivalence / bisimulation

Tracking information leaks

3 ways to prevent / detect information leaks:

typical leak: $s[1]?(2, x^{\top}).s[1]!\langle 2, \mathsf{true}^{\perp}\rangle$

- Typability (prevention): any "syntactic leak" is bad
- Safety (local detection): any "semantic leak" is bad
- Security (global detection): any "global semantic leak", detectable by observing the overall process, is bad

Tracking information leaks

3 ways to prevent / detect information leaks:

$$\nu(a)(a[1](\alpha).\, s[1]?(2, x^{\top}).s[1]!\langle 2, \mathsf{true}^{\perp}\rangle\,)$$

- Typability (prevention): any "syntactic leak" is bad
- Safety (local detection): any "semantic leak" is bad
- Security (global detection): any "global semantic leak", detectable by observing the overall process, is bad

Tracking information leaks

Another typical information leak:

$$s[1]?(2, x^\top).\ \mathsf{if}\ x^\top\ \mathsf{then}\ s[1]!\langle 2, \mathsf{true}^\perp\rangle\ \mathsf{else}\ s[1]!\langle 2, \mathsf{false}^\perp\rangle$$

- Typability (prevention): any "syntactic leak" is bad
- Safety (local detection): any "semantic leak" is bad
- Security (global detection): any "global semantic leak", detectable by observing the overall process, is bad

Tracking information leaks

Another typical information leak:

$$s[1]?(2, x^{\top}).\ \mathsf{if}\ x^{\top}\ \mathsf{then}\ s[1]!\langle 2, \mathsf{true}^{\perp}\rangle\ \mathsf{else}\ s[1]!\langle 2, \mathsf{true}^{\perp}\rangle$$

- Typability (prevention): any "syntactic leak" is bad
- Safety (local detection): any "semantic leak" is bad
- Security (global detection): any "global semantic leak", detectable by observing the overall process, is bad

Relating the three properties

Relationship between the three properties ?

- Typability (prevention): any "syntactic leak" is bad
  $\Downarrow$?
- Safety (local detection): any "semantic leak" is bad
  $\Downarrow$?
- Security (global detection): any "global semantic leak", detectable by observing the overall process, is bad

Relating the three properties

Relationship between the three properties ?

- Typability (prevention): any “syntactic leak” is bad
- Safety (local detection): any “semantic leak” is bad

  $\Downarrow$
- Security (global detection): any “global semantic leak”, detectable by observing the overall process, is bad

Multiparty sessions

[Honda, Yoshida, Carbone POPL'08]

Multiparty session: activation of an n-ary service a

$$\bar{a}[n] \mid a[1](\alpha_1).P_1 \mid \cdots \mid a[n](\alpha_n).P_n$$

arity

roles

initiator $\bar{a}[n]$: starts a new session on service a
when there are n suitable participants

Security session calculus

- Security levels ℓ, ℓ', forming a finite lattice $(\mathscr{S}, \leq)$.
- Services a, b, with an *arity* n.
- Sessions s, s' (activations of services). At n-ary session initiation, creation of private name s and channels with role $s[\mathrm{p}]$, $\mathrm{p} \in \{1, \ldots, n\}$.

$$
\begin{array}{llll}
\text{value} & v & ::= & \mathsf{true} \mid \mathsf{false} \mid \ldots \\
\text{expression} & e & ::= & x^{\ell} \mid v^{\ell} \mid \mathsf{not}\ e \mid e\ \mathsf{and}\ e' \mid \ldots \\
\text{identifier} & u & ::= & \zeta \mid a \\
\text{channel} & c & ::= & \alpha \mid s[\mathrm{p}]
\end{array}
$$

Syntax: processes

$$
\begin{array}{lcll}
P & ::= & \bar{u}[n] & n\text{-ary session initiator} \\
 & | & u[\mathrm{p}](\alpha).P & \text{p-th session participant} \\
 & | & c!\langle \Pi, e\rangle.P & \text{value send} \\
 & | & c?(\mathrm{p}, x^{\ell}).P & \text{value recv} \\
 & | & c!^{\ell}\langle\langle \mathrm{q}, c'\rangle\rangle.P & \text{channel send} \\
 & | & c?^{\ell}((\mathrm{p}, \alpha)).P & \text{channel recv} \\
 & | & c\oplus^{\ell}\langle \Pi, \lambda\rangle.P & \text{selection} \\
 & | & c\&^{\ell}(\mathrm{p}, \{\lambda_i : P_i\}_{i\in I}) & \text{branching} \\
 & | & \text{if } e \text{ then } P \text{ else } Q & \text{conditional} \\
 & | & \mathbf{0} \mid P \mid Q \mid (\nu a)P \mid \ldots & \pi\text{-calculus ops}
\end{array}
$$

Runtime syntax: queues

Asynchronous communication: messages transiting in queues

H	$::=$	$H \cup \{s : h\} \mid \emptyset$	**Q**-set
h	$::=$	$m \cdot h \mid \varepsilon$	queue
m	$::=$	$(\mathrm{p}, \Pi, \vartheta)$	message in transit
ϑ	$::=$	$v^\ell \mid s[\mathrm{p}]^\ell \mid \lambda^\ell \mid a^\ell$	message content

Independent message commutation:

$$(\mathrm{p}, \Pi, \vartheta) \cdot (\mathrm{p}', \Pi', \vartheta') \cdot h \equiv (\mathrm{p}', \Pi', \vartheta') \cdot (\mathrm{p}, \Pi, \vartheta) \cdot h$$

$$\text{if } \mathrm{p} \neq \mathrm{p}' \text{ or } \Pi \cap \Pi' = \emptyset$$

Semantics: configurations

In the semantics, **Q**-sets will be the observable part of process behaviour

$\Rightarrow$ need to be separated from the rest of the process.

Configurations $\quad C ::= < P \, , H > \;\mid\; (\nu \tilde{r}) < P \, , H > \;\mid\; C \parallel C$

Reduction semantics:

transitions of the form $< P \, , H > \longrightarrow (\nu \tilde{r}) < P' \, , H' >$

Semantics: computational rules

Session initiation:

$$< a[\alpha_1](P_1). \mid \ldots \mid a[\alpha_n](P_n). \mid \bar{a}[n]\ ,\ \emptyset > \longrightarrow$$

$$(\nu s) < P_1\{s[1]/\alpha_1\} \mid \ldots \mid P_n\{s[n]/\alpha_n\}\ ,\ s : \varepsilon > \qquad \text{[Link]}$$

Value exchange:

$$< s[\mathrm{p}]!\langle \Pi, e \rangle.P\ ,\ s : h > \longrightarrow < P\ ,\ s : h \cdot (\mathrm{p}, \Pi, v^{\ell}) > \qquad (e \downarrow v^{\ell}) \qquad \text{[Send]}$$

$$< s[\mathrm{q}]?(\mathrm{p}, x^{\ell}).P\ ,\ s : (\mathrm{p}, \mathrm{q}, v^{\ell}) \cdot h > \longrightarrow < P\{v^{\ell}/x^{\ell}\}\ ,\ s : h > \qquad \text{[Rec]}$$

Semantics: choice

Selection / branching:

$$< s[\mathrm{p}] \oplus^{\ell} \langle \Pi, \lambda \rangle.P \ , \ s : h > \longrightarrow < P \ , \ s : h \cdot (\mathrm{p}, \Pi, \lambda^{\ell}) > \qquad \text{[Label]}$$

$$< s[\mathrm{q}] \&^{\ell} (\mathrm{p}, \{\lambda_i : P_i\}_{i \in I}) \ , \ s : (\mathrm{p}, \mathrm{q}, \lambda_k^{\ell}) \cdot h > \longrightarrow < P_k \ , \ s : h > \quad (k \in I) \qquad \text{[Branch]}$$

Online medical service

$$
\begin{array}{lcll}
\mathtt{I} & = & \bar{a}[2] & \\
\mathtt{U} & = & a[1](\alpha_1). & \text{if } \mathit{simple\text{-}info}^{\perp} \\
 & & & \text{then } \alpha_1 \oplus^{\perp} \langle 2, \mathbf{sv1}\rangle.\alpha_1!\langle 2, \mathsf{que}^{\perp}\rangle.\alpha_1?(1, \mathit{ans}^{\perp}).\mathbf{0} \\
 & & & \text{else } \alpha_1 \oplus^{\perp} \langle 2, \mathbf{sv2}\rangle.\alpha_1!\langle 2, \mathsf{pwd}^{\top}\rangle.\alpha_1?(2, \mathit{form}^{\top}). \\
 & & & \qquad \text{if } \mathit{gooduse}(\mathit{form}^{\top}) \\
 & & & \qquad \text{then } \alpha_1!\langle 2, \mathsf{que}^{\top}\rangle.\alpha_1?(2, \mathit{ans}^{\top}).\mathbf{0} \\
 & & & \qquad \text{else } \alpha_1!\langle 2, \mathsf{que}^{\perp}\rangle.\alpha_1?(2, \mathit{ans}^{\perp}).\mathbf{0} \\
\mathtt{S} & = & a[2](\alpha_2). & \alpha_2 \ \&^{\perp}(1, \{\mathbf{sv1} : \alpha_2?(1, \mathit{que}^{\perp}).\alpha_2!\langle 1, \mathsf{ans}^{\perp}\rangle.\mathbf{0}, \\
 & & & \quad \mathbf{sv2} : \alpha_2?(1, \mathit{pwd}^{\top}).\alpha_2!\langle 1, \mathsf{form}^{\top}\rangle.\alpha_2?(1, \mathit{que}^{\top}).\alpha_2!\langle 1, \mathsf{ans}^{\top}\rangle.\mathbf{0}\})
\end{array}
$$

Online medical service (ctd)

User may accidentally leak data (sending in clear a secret question):

$$\begin{array}{lll} \mathtt{U} & = & \ldots \quad \mathsf{if}\ \mathit{gooduse}\,(\mathit{form}^{\top}) \\ & & \qquad\qquad \mathsf{then}\ \ldots \\ & & \qquad\qquad \mathsf{else}\ \alpha_1!\langle 2, \mathsf{que}^{\perp}\rangle.\alpha_1?(2, \mathit{ans}^{\perp}).\mathbf{0} \end{array}$$

Safety = early detection: monitored execution blocks before the leak.

Security = late detection: bisimulation game fails after the leak.

Monitored semantics

Monitored processes (where $\mu \in \mathscr{S}$):

$$M ::= P^{\rceil\mu} \mid M \mid M \mid (\nu\tilde{r})M \mid \mathsf{def}\ D\ \mathsf{in}\ M$$

Monitored transitions

$$< M\ ,\ H > \multimap\!\!\!\longrightarrow (\nu\tilde{s}) < M'\ ,\ H' >$$

Error predicate

$$< M\ ,\ H > \dagger$$

New structural rules:

$$(P_1 \mid P_2)^{\rceil\mu} \equiv P_1^{\rceil\mu} \mid P_2^{\rceil\mu} \qquad C\dagger \ \wedge\ C \equiv C' \implies C'\dagger$$

Monitored semantics rules

Conditional:

$$\text{if } e \text{ then } P \text{ else } Q^{\rceil\mu} \multimap P^{\rceil\mu\sqcup\ell} \qquad \text{if } e\downarrow \text{true}^{\ell}$$

$$\text{if } e \text{ then } P \text{ else } Q^{\rceil\mu} \multimap Q^{\rceil\mu\sqcup\ell} \qquad \text{if } e\downarrow \text{false}^{\ell}$$

Value input:

$$\text{if } \mu \leq \ell \quad \text{then } < s[\mathrm{q}]?(\mathrm{p},x^{\ell}).P^{\rceil\mu}\ ,\ s:(\mathrm{p},\mathrm{q},v^{\ell})\cdot h > \multimap < P\{v/x\}^{\rceil\ell}\ ,\ s:h >$$

$$\text{else } < s[\mathrm{q}]?(\mathrm{p},x^{\ell}).P^{\rceil\mu}\ ,\ s:(\mathrm{p},\mathrm{q},v^{\ell})\cdot h > \dagger$$

Monitored semantics rules (ctd)

Session initiation:

$$a[1](\alpha_1).P_1^{\rceil \mu_1} \mid \ldots \mid a[n](\alpha_n).P_n^{\rceil \mu_n} \mid \bar{a}[n]^{\rceil \mu_{n+1}} \multimap\!\!\!\longrightarrow$$

$$(\nu s) < P_1\{s[1]/\alpha_1\}^{\rceil \mu} \mid \ldots \mid P_n\{s[n]/\alpha_n\}^{\rceil \mu} \, , \, s : \varepsilon >$$

where $\mu = \bigsqcup_{i \in \{1 \ldots n+1\}} \mu_i$

Need for the join:

$$s[2]?(1, x^{\top}).\mathsf{if}\ x^{\top}\ \mathsf{then}\ \bar{b}[2]\ \mathsf{else}\ \mathbf{0}$$

$$\mid\ b[1](\beta_1).\beta_1!\langle 2, \mathsf{true}^{\perp}\rangle.\mathbf{0} \ \mid\ b[2](\beta_2).\beta_2?(1, y^{\perp}).\mathbf{0}$$

Safety: 1st attempt

Let $|M|$ be the process obtained by erasing all monitoring levels in M.

Monitored process safety:

M is safe if for any monotone H such that $< |M| \, , \, H >$ is saturated:

If $< |M| \, , \, H > \longrightarrow (\nu\tilde{r}) < P \, , \, H' >$

then $< M \, , \, H > \multimap\!\!\longrightarrow (\nu\tilde{r}) < M' \, , \, H' >$, where $|M'| = P$ and M' is safe.

Process safety: A process P is safe if $P^{\rceil\perp}$ is safe.

Safety: definition

Let $|M|$ be the process obtained by erasing all monitoring levels in M.

Testers $\quad T ::= \mathbf{0} \mid \bar{a}[n] \mid a[\mathrm{p}](\alpha).\mathbf{0} \mid T \mid T$

Monitored process safety: M is safe if for any tester T and monotone H such that $< |M| \, , H >$ is saturated:

If $< |M| \mid T \, , H > \longrightarrow (\nu\tilde{r}) < P \, , H' >$

then $< M \mid T^{\rceil\perp} \, , H > \multimap\!\!\rightarrow (\nu\tilde{r}) < M' \, , H' >$, where $|M'| = P$ and M' is safe.

Process safety: A process P is safe if $P^{\rceil\perp}$ is safe.

Security

Observation defined as usual wrt a downward-closed set of levels $\mathscr{L}$.

What is $\mathscr{L}$-observable in $(\nu\tilde{r}) < P\,,\,H >$? Messages of level $\ell \in \mathscr{L}$ in H.

$\Longrightarrow$ session queues play the role of memories in imperative languages

$\mathscr{L}$-projection of **Q**-sets

$$(\mathrm{p}, \Pi, \vartheta) \Downarrow \mathscr{L} = \begin{cases} (\mathrm{p}, \Pi, \vartheta) & \text{if } lev(\vartheta) \in \mathscr{L} \\ \varepsilon & \text{otherwise} \end{cases}$$

extended pointwise to named queues and **Q**-sets (NB: $s : \varepsilon$ not observed)

$\mathscr{L}$-equality of **Q**-sets: $H =_{\mathscr{L}} K$ if $H \Downarrow \mathscr{L} = K \Downarrow \mathscr{L}$

Security (ctd)

$\mathscr{L}$-bisimulation on processes: symmetric relation $\mathscr{R}$ such that $P_1 \mathscr{R} P_2$ implies, for any tester T and for any pair of monotone H_1, H_2 such that $H_1 =_{\mathscr{L}} H_2$ and each $< P_i \, , \, H_i >$ is saturated:

If $< P_1 \mid T \, , \, H_1 > \longrightarrow (\nu \tilde{r}) < P_1' \, , \, H_1' >$, then there exist P_2', H_2' such that

$$< P_2 \mid T \, , \, H_2 > \longrightarrow^* \equiv (\nu \tilde{r}) < P_2' \, , \, H_2' >, \text{ where } H_1' =_{\mathscr{L}} H_2' \text{ and } P_1' \mathscr{R} P_2'$$

$\mathscr{L}$-equivalence: $P_1 \simeq_{\mathscr{L}} P_2$ if $P_1 \mathscr{R} P_2$ for some $\mathscr{L}$-bisimulation $\mathscr{R}$

$\mathscr{L}$-security: P is $\mathscr{L}$-*secure* if $P \simeq_{\mathscr{L}} P$

Main results

Safety implies absence of run-time errors

If P is safe, then every monitored computation:

$$< P^{\rceil\perp} \, , \emptyset > \; = \; < M_0 \, , H_0 > \multimap\!\!\rightarrow \cdots \multimap\!\!\rightarrow (\nu \tilde{r}_k) < M_k \, , H_k >$$

is such that $\neg < M_k \, , H_k > \dagger$.

Safety implies security

If P is safe, then P is $\mathscr{L}$-secure for any down-closed set of levels $\mathscr{L}$.

Main results (ctd)

Absence of run-time errors does not imply safety

Not safe

$$
\begin{aligned}
P &= \bar{a}[2] \mid a[1](\alpha_1).P_1 \mid a[2](\alpha_2).P_2 \\
P_1 &= \alpha_1!\langle 2, \mathsf{true}^{\top}\rangle.\alpha_1?(2, x^{\top}).\mathbf{0} \\
P_2 &= \alpha_2?(1, z^{\top}).\mathsf{if}\ z^{\top}\ \mathsf{then}\ \alpha_2!\langle 1, \mathsf{false}^{\top}\rangle.\mathbf{0}\ \mathsf{else}\ \alpha_2!\langle 1, \mathsf{true}^{\perp}\rangle.\mathbf{0}
\end{aligned}
$$

Security does not imply safety

Not safe

$$
s[1]?(2, x^{\top}).\mathsf{if}\ x^{\top}\ \mathsf{then}\ s[1]!\langle 2, \mathsf{true}^{\perp}\rangle.\mathbf{0}\ \mathsf{else}\ s[1]!\langle 2, \mathsf{true}^{\perp}\rangle.\mathbf{0}
$$

Conclusion and future work

- Complete the picture by showing typability => safety
- Explore monitored semantics with labelled transitions, to return informative error messages to the programmer.
- Attach reputation and trust to participants, and possibly use them to refine delegation.

[Submitted, full version soon on our web pages]