Information flow safety in multiparty sessions

Sara Capecchi, Ilaria Castellani and Mariangiola Dezani-Ciancaglini

(TORINO University and INRIA Sophia Antipolis Méditerranée)
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 $\mathcal{u}^l$ can only depend on a receive or test of a value $\mathcal{u}^{l_0}$ with $l_0 \leq l$
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

  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:

- **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:

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

- **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:

\[ s[1](2, x^\top). \text{ if } x^\top \text{ then } s[1]!(2, \text{true}^\bot) \text{ else } s[1]!(2, \text{false}^\bot) \]

- **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:

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

- **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
  ↓?
- **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
- **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 $\alpha$

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

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

- Security levels $\ell, \ell'$, forming a finite lattice $(\mathcal{L}, \leq)$.
- 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 $s[p], p \in \{1, \ldots, n\}$.

\[
\begin{align*}
\text{value} & :\quad v ::= \text{true} \mid \text{false} \mid \ldots \\
\text{expression} & :\quad e ::= x^\ell \mid v^\ell \mid \text{not } e \mid e \text{ and } e' \mid \ldots \\
\text{identifier} & :\quad u ::= \zeta \mid a \\
\text{channel} & :\quad c ::= \alpha \mid s[p]
\end{align*}
\]
Syntax: processes

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

Asynchronous communication: messages transiting in queues

\[
\begin{align*}
H & ::= H \cup \{s : h\} \mid \emptyset \\
   & \quad \text{Q-set} \\
\empty{h} & ::= m \cdot h \mid \varepsilon \\
   & \quad \text{queue} \\
m & ::= (p, \Pi, \vartheta) \\
   & \quad \text{message in transit} \\
\vartheta & ::= v^l \mid s[p]^l \mid \lambda^l \mid a^l \\
   & \quad \text{message content}
\end{align*}
\]

Independent message commutation:

\[
(p, \Pi, \vartheta) \cdot (p', \Pi', \vartheta') \cdot h \equiv (p', \Pi', \vartheta') \cdot (p, \Pi, \vartheta) \cdot h
\]

if \( p \neq p' \) or \( \Pi \cap \Pi' = \emptyset \)
Semantics: configurations

In the semantics, Q-sets will be the observable part of process behaviour
⇒ need to be separated from the rest of the process.

Configurations

\[ C ::= < P, H > \mid (\nu \tilde{r}) < P, H > \mid C \parallel C \]

Reduction semantics:

transitions of the form \( < P, H > \rightarrow (\nu \tilde{r}) < P', H' > \)
Semantics: computational rules

Session initiation:

\[
< a[\alpha_1](P_1). \mid \ldots \mid a[\alpha_n](P_n). \mid b[n], \emptyset > \rightarrow
\]

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

Value exchange:

\[
< s[p]!(\Pi, e).P, s : h > \rightarrow < P, s : h : (p, \Pi, v^\ell) > \quad (e \downarrow v^\ell) \quad [\text{Send}]
\]

\[
< s[q]?((p, x^\ell).P, s : (p, q, v^\ell) \cdot h) > \rightarrow < P\{v^\ell/x^\ell\}, s : h > \quad [\text{Rec}]
\]
Semantics: choice

Selection / branching:

\[ < s[p] \oplus^\ell \langle \Pi, \lambda \rangle. P, s : h > \rightarrow < P, s : h \cdot (p, \Pi, \lambda^\ell) > \]  

[Label]

\[ < s[q] \&^\ell (p, \{ \lambda_i : P_i \}_{i \in I}), s : (p, q, \lambda^\ell_k) \cdot h > \rightarrow < P_k, s : h > \quad (k \in I) \]  

[Branch]
Online medical service

\[ I = \bar{a}[2] \]
\[ U = a[1](\alpha_1). \quad \text{if simple-info} \]
\[ \quad \text{then } \alpha_1 \oplus \langle 2, sv1 \rangle. \alpha_1 !\langle 2, que \rangle. \alpha_1 ?(1, ans).0 \]
\[ \quad \text{else } \alpha_1 \oplus \langle 2, sv2 \rangle. \alpha_1 !\langle 2, pwd^T \rangle. \alpha_1 ?(2, form^T). \]
\[ \quad \quad \text{if gooduse}(form^T) \]
\[ \quad \quad \text{then } \alpha_1 !\langle 2, que^T \rangle. \alpha_1 ?(2, ans^T).0 \]
\[ \quad \quad \text{else } \alpha_1 !\langle 2, que \rangle. \alpha_1 ?(2, ans).0 \]

\[ S = a[2](\alpha_2). \quad \alpha_2 \& \langle 1, \{ sv1 : \alpha_2 ?(1, que). \alpha_2 !\langle 1, ans \rangle.0, \sv2 : \alpha_2 ?(1, pwd^T). \alpha_2 !\langle 1, form^T \rangle. \alpha_2 ?(1, que^T). \alpha_2 !\langle 1, ans^T \rangle.0 \} \]
User may accidentally leak data (sending in clear a secret question):

\[ U = \ldots \text{ if } \text{gooduse}(\text{form}^\top) \]
\[ \text{then } \ldots \]
\[ \text{else } \alpha_1 ! (2, \text{que}^\perp).\alpha_1 ? (2, \text{ans}^\perp).0 \]

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

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

Monitored processes (where $\mu \in \mathcal{L}$):

$$M ::= P^\mu \mid M \mid M \mid (\forall \tilde{r})M \mid \text{def } D \text{ in } M$$

Monitored transitions

$$< M, H > \xrightarrow{\text{(v$s$)}} < M', H' >$$

Error predicate

$$< M, H > \dagger$$

New structural rules:

$$(P_1 \mid P_2)^\mu \equiv P_1^\mu \mid P_2^\mu$$

$$C \dagger \land C \equiv C' \implies C' \dagger$$
Monitored semantics rules

Conditional:

\[
\begin{align*}
\text{if } e \text{ then } P \text{ else } Q|\mu & \rightarrow P|\mu \sqcup e \quad \text{if } e \downarrow \text{ true}^\ell \\
\text{if } e \text{ then } P \text{ else } Q|\mu & \rightarrow Q|\mu \sqcup e \quad \text{if } e \downarrow \text{ false}^\ell
\end{align*}
\]

Value input:

\[
\begin{align*}
\text{if } \mu \leq \ell & \quad \text{then} \quad <s[q](p,x^\ell).P|\mu \ , \ s : (p,q,v^\ell) \cdot h > \rightarrow \langle P\{v/x\}|\ell \ , \ s : h \rangle \\
\text{else} & \quad <s[q](p,x^\ell).P|\mu \ , \ s : (p,q,v^\ell) \cdot h > \dagger
\end{align*}
\]
Monitored semantics rules (ctd)

Session initiation:

\[
\begin{align*}
& a[1](\alpha_1).P_1^{\mu_1} \mid \ldots \mid a[n](\alpha_n).P_n^{\mu_n} \mid \bar{a}[n]^{\mu_{n+1}} \longrightarrow \\
& (\forall s) < P_1\{s[1]/\alpha_1\}^{\mu} \mid \ldots \mid P_n\{s[n]/\alpha_n\}^{\mu}, s : \varepsilon > \\
& \text{where } \mu = \bigsqcup_{i \in \{1, \ldots, n+1\}} \mu_i
\end{align*}
\]

Need for the join:

\[
\begin{align*}
& s[2]?(1, x^\top).\text{if } x^\top \text{ then } b[2] \text{ else } 0 \\
& \mid b[1](\beta_1).\beta_1!(2, \text{true}^\perp).0 \mid b[2](\beta_2).\beta_2?(1, y^\perp).0
\end{align*}
\]
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 $< |M|, H > \rightarrow (\forall \bar{r}) < P, H' >$

then $< M, H > \rightarrow (\forall \bar{r}) < M', H' >$, where $|M'| = P$ and $M'$ is safe.

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

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

Testers

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

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

If $< |M|, T, H > \rightarrow (v\tilde{r}) < P, H' >$

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

Process safety: A process $P$ is safe if $P^\perp$ is safe.
Security

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

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

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

$\mathcal{L}$-projection of Q-sets

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

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

$\mathcal{L}$-equality of Q-sets: $H =_{\mathcal{L}} K$ if $H \downarrow \mathcal{L} = K \downarrow \mathcal{L}$
\( \mathcal{L} \)-bisimulation on processes: symmetric relation \( \mathcal{R} \) such that \( P_1 \mathcal{R} P_2 \) implies, for any tester \( T \) and for any pair of monotone \( H_1, H_2 \) such that \( H_1 \equiv H_2 \) and each \( < P_i, H_i > \) is saturated:

If \( < P_1 | T, H_1 > \rightarrow (v\bar{r}) < P'_1, H'_1 > , \) then there exist \( P'_2, H'_2 \) such that
\[ < P_2 | T, H_2 > \rightarrow^* (v\bar{r}) < P'_2, H'_2 > , \]
where \( H'_1 \equiv H'_2 \) and \( P'_1 \mathcal{R} P'_2 \)

\( \mathcal{L} \)-equivalence: \( P_1 \sim \mathcal{L} P_2 \) if \( P_1 \mathcal{R} P_2 \) for some \( \mathcal{L} \)-bisimulation \( \mathcal{R} \)

\( \mathcal{L} \)-security: \( P \) is \( \mathcal{L} \)-secure if \( P \sim \mathcal{L} P \)
Main results

Safety implies absence of run-time errors

If $P$ is safe, then every monitored computation:
\[
< P \downarrow , \emptyset > \rightarrow < M_0 , H_0 > \rightarrow \cdots \rightarrow (\nu \tilde{r}_k) < M_k , H_k >
\]

is such that $\not\rightarrow < M_k , H_k > \dagger$.

Safety implies security

If $P$ is safe, then $P$ is $\mathcal{L}$-secure for any down-closed set of levels $\mathcal{L}$. 
Main results (ctd)

Absence of run-time errors does not imply safety

Not safe

\[ P = \alpha_1[2] | a[1](\alpha_1).P_1 | a[2](\alpha_2).P_2 \]
\[ P_1 = \alpha_1!\langle 2, \text{true}^T \rangle.\alpha_1?\langle 2, x^T \rangle.0 \]
\[ P_2 = \alpha_2?\langle 1, z^T \rangle.\text{if } z^T \text{ then } \alpha_2!\langle 1, \text{false}^T \rangle.0 \text{ else } \alpha_2!\langle 1, \text{true}^\perp \rangle.0 \]

Security does not imply safety

Not safe

\[ s[1]?\langle 2, x^T \rangle.\text{if } x^T \text{ then } s[1]!\langle 2, \text{true}^\perp \rangle.0 \text{ else } s[1]!\langle 2, \text{true}^\perp \rangle.0 \]
Conclusion and future work

- Complete the picture by showing $\text{typability} \Rightarrow \text{safety}$

- Explore monitored semantics with $\text{labelled transitions}$, to return informative error messages to the programmer.

- Attach $\text{reputation}$ and $\text{trust}$ to participants, and possibly use them to refine delegation.

[Submitted, full version soon on our web pages]