A one message protocol using cryptography, where K_{AB} is a symmetric key shared between A and B for private communication.

$$A \longrightarrow B : \{M\}_{K_{AB}} \text{ on } c_{AB}$$

This can be represented as

$$\begin{aligned}
A &\triangleq \mathsf{send}_{c_{AB}}\langle\{M\}_{K_{AB}}\rangle; \mathsf{halt} \\
B &\triangleq \mathsf{recv}_{c_{AB}}(x); \mathsf{case}\ x\ \mathsf{of}\ \{y\}_{K_{AB}} : F(y) \\
P &\triangleq \mathsf{new}\ K_{AB}; (A \mid B)
\end{aligned}$$

The key K_{AB} is restricted, only A and B can use it.

The channel c_{AB} is public. Other principals may send messages on it or listen on it.

P can make silent transitions to $\mathsf{new}\ K_{AB}; F(M)$.

Formal semantics

We now need to define how processes execute.

For example we would like

$$\mathsf{send}_c\langle M\rangle; P \mid \mathsf{recv}_c(x); Q \xrightarrow{\tau} P \mid Q[M/x]$$

where τ denotes a silent action (internal communication).

Let $fn(M)$ and $fn(P)$ be the set of free names in term M and process P respectively.

Let $fv(M)$ and $fv(P)$ be the set of free variables in term M and process P respectively.

Closed processes are processes without any free variables.

Let $P \triangleq \mathsf{new}\ c; \mathsf{new}\ K; \mathsf{recv}_d(x); \mathsf{case}\ x\ \mathsf{of}\ \{y\}_{K'} : \mathsf{send}_d\langle\{y\}_K, z, c\rangle; \mathsf{halt}$.

We have

$fn(\mathsf{send}_d\langle\{y\}_K, z, c\rangle; \mathsf{halt}) = \{c, d, K\}$

$fv(\mathsf{send}_d\langle\{y\}_K, z, c\rangle; \mathsf{halt}) = \{y, z\}$

$fn(\mathsf{case}\ x\ \mathsf{of}\ \{y\}_{K'} : \mathsf{send}_d\langle\{y\}_K, z, c\rangle; \mathsf{halt}) = \{c, d, K, K'\}$

$fv(\mathsf{case}\ x\ \mathsf{of}\ \{y\}_{K'} : \mathsf{send}_d\langle\{y\}_K, z, c\rangle; \mathsf{halt}) = \{x, z\}$

$fn(P) = \{d, K'\}$

$fv(P) = \{z\}$

$fn(\{y\}_K) = \{K\}$

$fv(\{y\}_K) = \{y\}$

First we define reduction relation $>$ on closed processes:

$$
\begin{array}{rl}
\mathsf{repeat}\ P & > P \mid \mathsf{repeat}\ P \\
\mathsf{check}\ (M == M); P & > P \\
\mathsf{let}\ (x, y) = (M, N); P & > P[M/x, N/y] \\
\mathsf{case}\ 0\ \mathsf{of}\ 0 : P,\ \mathsf{succ}\ (x) : Q & > P \\
\mathsf{case}\ \mathsf{succ}\ (M)\ \mathsf{of}\ 0 : P,\ \mathsf{succ}\ (x) : Q & > Q[M/x] \\
\mathsf{case}\ \{M\}_N\ \mathsf{of}\ \{x\}_N : P & > P[M/x]
\end{array}
$$

When these rules cannot be applied, it means that the process cannot be simplified.

The following processes cannot be simplified, hence cannot be executed further.

$\mathsf{check}\ (0 == \mathsf{succ}\ (0); P$ (comparison fails).

$\mathsf{let}\ (x, y) = 0; P$ (unpairing fails)

$\mathsf{case}\ (M, N)\ \mathsf{of}\ 0 : P,\ \mathsf{succ}\ (x) : Q$ (not an integer)

$\mathsf{case}\ (M, N)\ \mathsf{of}\ \{x, y\}_K : P$ (not an encrypted message)

$\mathsf{case}\ \{M, N\}_{K'}\ \mathsf{of}\ \{x, y\}_K : P$ where $K \neq K'$

When these rules cannot be applied, it means that the process cannot be simplified.

The following processes cannot be simplified, hence cannot be executed further.

$\mathsf{check}\ (0 == \mathsf{succ}\ (0); P$ (comparison fails).

$\mathsf{let}\ (x, y) = 0; P$ (unpairing fails)

$\mathsf{case}\ (M, N)\ \mathsf{of}\ 0 : P,\ \mathsf{succ}\ (x) : Q$ (not an integer)

$\mathsf{case}\ (M, N)\ \mathsf{of}\ \{x, y\}_K : P$ (not an encrypted message)

$\mathsf{case}\ \{M, N\}_{K'}\ \mathsf{of}\ \{x, y\}_K : P$ where $K \neq K'$

This is also based on the perfect cryptography assumption: distinct terms represent distinct messages.

A barb β is either

- a name n (representing input on channel n), or
- a co-name $\overline{n}$ (representing output on channel n)

An action is either

- a barb (representing input or output to the outside world), or
- τ (representing a silent action i.e. internal communication)

We write $P \xrightarrow{\alpha} Q$ to mean that P makes action α after which Q is the remaining process that is left to be executed.

Commitment relation Consider again $\mathsf{send}_c\langle M\rangle; P \mid \mathsf{recv}_c(x); Q$

Commitment relation Consider again $\mathsf{send}_c\langle M\rangle; P \mid \mathsf{recv}_c(x); Q$

The first subprocess makes an output action on channel c.

We will represent it as $\mathsf{send}_c\langle M\rangle; P \xrightarrow{\overline{c}} \langle M\rangle P$.

$\langle M\rangle P$ is called a concretion: it represents a commitment to output message M after which P will be executed.

Commitment relation Consider again $\mathsf{send}_c\langle M\rangle; P \mid \mathsf{recv}_c(x); Q$

The first subprocess makes an output action on channel c.

We will represent it as $\mathsf{send}_c\langle M\rangle; P \xrightarrow{\overline{c}} \langle M\rangle P$.

$\langle M\rangle P$ is called a concretion: it represents a commitment to output message M after which P will be executed.

The second subprocess makes an input action on channel c.

We will represent it as $\mathsf{recv}_c(x); Q \xrightarrow{c} (x)Q$.

$(x)Q$ is called an abstraction:it represents a commitment to input some x after which Q will be executed.

Commitment relation Consider again $\mathsf{send}_c\langle M\rangle; P \mid \mathsf{recv}_c(x); Q$

The first subprocess makes an output action on channel c.

We will represent it as $\mathsf{send}_c\langle M\rangle; P \xrightarrow{\overline{c}} \langle M\rangle P$.

$\langle M\rangle P$ is called a concretion: it represents a commitment to output message M after which P will be executed.

The second subprocess makes an input action on channel c.

We will represent it as $\mathsf{recv}_c(x); Q \xrightarrow{c} (x)Q$.

$(x)Q$ is called an abstraction:it represents a commitment to input some x after which Q will be executed.

Abstractions and concretions can be combined:

$\langle M\rangle P \mathbin{@} (x)Q = P \mid Q[M/x]$

Formally an abstraction F is of the form

$$(x_1, \ldots, x_k)P$$

where $k \geq 0$ and P is a process.

A concretion C is of the form

$$(\mathsf{new}\ n_1, \ldots, n_l)\langle M_1, \ldots, M_k\rangle P$$

where $n_1, \ldots, n_l$ are names, $l, k \geq 0$ and P is a process.

Formally an abstraction F is of the form

$$(x_1, \ldots, x_k)P$$

where $k \geq 0$ and P is a process.

A concretion C is of the form

$$(\mathsf{new}\ n_1, \ldots, n_l)\langle M_1, \ldots, M_k\rangle P$$

where $n_1, \ldots, n_l$ are names, $l, k \geq 0$ and P is a process.

For $F \triangleq (x_1, \ldots, x_k)P$ and $C \triangleq (\mathsf{new}\ n_1, \ldots, n_l)\langle M_1, \ldots, M_k\rangle Q$

with $\{n_1, \ldots, n_l\} \cap fn(P) = \emptyset$ we define interaction of F and C as

$$\boxed{\begin{array}{l} F \mathbin{@} C \triangleq \mathsf{new}\ n_1; \ldots \mathsf{new}\ n_l; (P[M_1/x_1, \ldots, M_k/x_k] \mid Q) \\ C \mathbin{@} F \triangleq \mathsf{new}\ n_1; \ldots \mathsf{new}\ n_l; (Q \mid P[M_1/x_1, \ldots, M_k/x_k]) \end{array}}$$

An agent A is an abstraction, concretion or a process.

We write the commitment relation as $P \xrightarrow{\alpha} A$ where P is a closed process, A is a closed agent ($fv(A) = \emptyset$) and α is an action.

An agent A is an abstraction, concretion or a process.

We write the commitment relation as $P \xrightarrow{\alpha} A$ where P is a closed process, A is a closed agent ($fv(A) = \emptyset$) and α is an action.

$$\mathsf{send}_m\langle M_1, \ldots, M_k\rangle; P \xrightarrow{\overline{m}} (\mathsf{new}\ )\langle M_1, \ldots, M_k\rangle P$$

An agent A is an abstraction, concretion or a process.

We write the commitment relation as $P \xrightarrow{\alpha} A$ where P is a closed process, A is a closed agent ($fv(A) = \emptyset$) and α is an action.

$$\mathsf{send}_m\langle M_1, \ldots, M_k\rangle; P \xrightarrow{\overline{m}} (\mathsf{new}\ )\langle M_1, \ldots, M_k\rangle P$$

$$\mathsf{recv}_m(x_1, \ldots, x_k); P \xrightarrow{m} (x_1, \ldots, x_k)P$$

An agent A is an abstraction, concretion or a process.

We write the commitment relation as $P \xrightarrow{\alpha} A$ where P is a closed process, A is a closed agent ($fv(A) = \emptyset$) and α is an action.

$$\mathsf{send}_m\langle M_1, \ldots, M_k\rangle; P \xrightarrow{\overline{m}} (\mathsf{new}\ )\langle M_1, \ldots, M_k\rangle P$$

$$\mathsf{recv}_m(x_1, \ldots, x_k); P \xrightarrow{m} (x_1, \ldots, x_k)P$$

$$\frac{P \xrightarrow{m} F \quad Q \xrightarrow{\overline{m}} C}{P \mid Q \xrightarrow{\tau} F\ @\ C}$$

An agent A is an abstraction, concretion or a process.

We write the commitment relation as $P \xrightarrow{\alpha} A$ where P is a closed process, A is a closed agent ($fv(A) = \emptyset$) and α is an action.

$$\mathsf{send}_m\langle M_1, \ldots, M_k\rangle; P \xrightarrow{\overline{m}} (\mathsf{new}\ )\langle M_1, \ldots, M_k\rangle P$$

$$\mathsf{recv}_m(x_1, \ldots, x_k); P \xrightarrow{m} (x_1, \ldots, x_k)P$$

$$\frac{P \xrightarrow{m} F \quad Q \xrightarrow{\overline{m}} C}{P \mid Q \xrightarrow{\tau} F \mathbin{@} C}$$

$$\frac{P \xrightarrow{\overline{m}} C \quad Q \xrightarrow{m} F}{P \mid Q \xrightarrow{\tau} C \mathbin{@} F}$$

Example

Define

$P \triangleq \mathsf{send}_c\langle \mathsf{succ}\ (0)\rangle; \mathsf{halt}$

$Q \triangleq \mathsf{recv}_c(x); \mathsf{case}\ x\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$

From our rules we have

Example

Define

$P \triangleq \mathsf{send}_c\langle \mathsf{succ}\ (0)\rangle; \mathsf{halt}$

$Q \triangleq \mathsf{recv}_c(x); \mathsf{case}\ x\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$

From our rules we have

$$P \quad \xrightarrow{\overline{c}} \langle \mathsf{succ}\ (0)\rangle \mathsf{halt}$$

$$(\langle M_1, \ldots, M_k\rangle P' \text{ denotes } (\mathsf{new}\ )\langle M_1, \ldots, M_k\rangle P')$$

Example

Define

$P \triangleq \mathsf{send}_c\langle \mathsf{succ}\ (0)\rangle; \mathsf{halt}$

$Q \triangleq \mathsf{recv}_c(x); \mathsf{case}\ x\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$

From our rules we have

$$P \xrightarrow{\overline{c}} \langle \mathsf{succ}\ (0)\rangle \mathsf{halt}$$

$$(\langle M_1, \ldots, M_k\rangle P' \text{ denotes } (\mathsf{new}\ )\langle M_1, \ldots, M_k\rangle P')$$

$$Q \xrightarrow{c} (x)\mathsf{case}\ x\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$$

Example

Define

$P \triangleq \mathsf{send}_c\langle \mathsf{succ}\ (0)\rangle; \mathsf{halt}$

$Q \triangleq \mathsf{recv}_c(x); \mathsf{case}\ x\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$

From our rules we have

$$P \xrightarrow{\bar{c}} \langle \mathsf{succ}\ (0)\rangle \mathsf{halt}$$

$$(\langle M_1, \ldots, M_k\rangle P' \text{ denotes } (\mathsf{new}\ )\langle M_1, \ldots, M_k\rangle P')$$

$$Q \xrightarrow{c} (x)\mathsf{case}\ x\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$$

$$P \mid Q \xrightarrow{\tau} \mathsf{halt} \mid \mathsf{case}\ \mathsf{succ}\ (0)\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$$

Example

Define

$P \triangleq \mathsf{send}_c\langle \mathsf{succ}\ (0)\rangle; \mathsf{halt}$

$Q \triangleq \mathsf{recv}_c(x); \mathsf{case}\ x\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$

From our rules we have

$$P \xrightarrow{\bar{c}} \langle \mathsf{succ}\ (0)\rangle \mathsf{halt}$$

$(\langle M_1, \ldots, M_k\rangle P'$ denotes $(\mathsf{new}\ )\langle M_1, \ldots, M_k\rangle P')$

$$Q \xrightarrow{c} (x)\mathsf{case}\ x\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$$

$$P \mid Q \xrightarrow{\tau} \mathsf{halt} \mid \mathsf{case}\ \mathsf{succ}\ (0)\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt})$$

$$\xrightarrow{\bar{d}} \langle 0\rangle(\mathsf{halt} \mid \mathsf{halt})$$ using the following rules. . .

$$\frac{P > Q \quad Q \xrightarrow{\alpha} A}{P \xrightarrow{\alpha} A}$$

$$\frac{P \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} A \mid Q} \qquad \frac{Q \xrightarrow{\alpha} A}{P \mid Q \xrightarrow{\alpha} P \mid A}$$

where

$$P_1 \mid (x_1, \ldots, x_k)P_2 \triangleq (x_1, \ldots, x_k)(P_1 \mid P_2)$$

$$P_1 \mid (\mathsf{new}\ n_1, \ldots, n_k)\langle M_1, \ldots, M_l\rangle P_2 \triangleq (\mathsf{new}\ n_1, \ldots, n_k)\langle M_1, \ldots, M_l\rangle(P_1 \mid P_2)$$

provided that $x_1, \ldots, x_k \notin fv(P_1)$ and $n_1, \ldots, n_k \notin fn(P_1)$

For the previous example we have:

$$\mathsf{case\ succ}\ (0)\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt}) > \mathsf{send}_d\langle 0\rangle; \mathsf{halt}$$

and

$$\mathsf{send}_d\langle 0\rangle; \mathsf{halt} \xrightarrow{\bar{d}} \langle 0\rangle \mathsf{halt}$$

hence

$$\mathsf{case\ succ}\ (0)\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt}) \xrightarrow{\bar{d}} \langle 0\rangle \mathsf{halt}$$

hence

$$\mathsf{halt} \mid \mathsf{case\ succ}\ (0)\ \mathsf{of}\ 0 : \mathsf{halt},\ \mathsf{succ}\ (y) : (\mathsf{send}_d\langle y\rangle; \mathsf{halt}) \xrightarrow{\bar{d}} \mathsf{halt} \mid \langle 0\rangle \mathsf{halt}$$
$$= \langle 0\rangle(\mathsf{halt} \mid \mathsf{halt})$$

Consider $P \triangleq (\mathsf{recv}_c(x); P_1) \mid \mathsf{new}\ c; (\mathsf{send}_c\langle 0\rangle; P_2 \mid \mathsf{recv}_c(x); P_3)$

We would like $P \xrightarrow{\tau} (\mathsf{recv}_c(x); P_1) \mid \mathsf{new}\ c; (P_2 \mid P_3[0/x])$

but not $P \xrightarrow{\tau} P_1[0/x] \mid \mathsf{new}\ n; (P_2 \mid \mathsf{recv}_c(x); P_3)$

Consider $P \triangleq (\mathsf{recv}_c(x); P_1) \mid \mathsf{new}\ c; (\mathsf{send}_c\langle 0\rangle; P_2 \mid \mathsf{recv}_c(x); P_3)$

We would like $P \xrightarrow{\tau} (\mathsf{recv}_c(x); P_1) \mid \mathsf{new}\ c; (P_2 \mid P_3[0/x])$

but not $P \xrightarrow{\tau} P_1[0/x] \mid \mathsf{new}\ n; (P_2 \mid \mathsf{recv}_c(x); P_3)$

Hence we have the rule

$$\boxed{\frac{P \xrightarrow{\alpha} A \quad \alpha \notin \{n, \overline{n}\}}{\mathsf{new}\ n; P \xrightarrow{\alpha} \mathsf{new}\ n; A}}$$

where

$$(\mathsf{new}\ m)(x_1, \ldots, x_k)P \triangleq (x_1, \ldots, x_k)\mathsf{new}\ m; P$$

$$(\mathsf{new}\ m)(\mathsf{new}\ m_1, \ldots, m_k)\langle M_1, \ldots, M_l\rangle P \triangleq (\mathsf{new}\ m, m_1, \ldots, m_k)\langle M_1, \ldots, M_l\rangle P$$

provided that $m \notin \{m_1, \ldots, m_k\}$

We have $\mathsf{send}_c\langle 0\rangle; P_2 \xrightarrow{\overline{c}} \langle 0\rangle P_2$

and $\mathsf{recv}_c(x); P_3 \xrightarrow{c} (x) P_3$

hence $\mathsf{send}_c\langle 0\rangle; P_2 \mid \mathsf{recv}_c(x); P_3 \xrightarrow{\tau} \langle 0\rangle P_2 \mathbin{@} (x) P_3 = P_2 \mid P_3[0/x]$

Since $\tau \notin \{\overline{c}, c\}$

hence $\mathsf{new}\ c; (\mathsf{send}_c\langle 0\rangle; P_2 \mid \mathsf{recv}_c(x); P_3) \xrightarrow{\tau} \mathsf{new}\ c; (P_2 \mid P_3[0/x])$

Hence $(\mathsf{recv}_c(x); P_1) \mid \mathsf{new}\ c; (\mathsf{send}_c\langle 0\rangle; P_2 \mid \mathsf{recv}_c(x); P_3)$
$$\xrightarrow{\tau} (\mathsf{recv}_c(x); P_1) \mid \mathsf{new}\ c; (P_2 \mid P_3[0/x])$$

Consider $P \triangleq (\mathsf{new}\ K; \mathsf{send}_c\langle K\rangle; \mathsf{halt}) \mid (\mathsf{recv}_c(x); \mathsf{send}_d\langle x\rangle; \mathsf{halt})$

We have $\mathsf{send}_c\langle K\rangle; \mathsf{halt} \xrightarrow{\bar{c}} (\mathsf{new}\ )\langle K\rangle\mathsf{halt}$

hence $\mathsf{new}\ K; \mathsf{send}_c\langle K\rangle; \mathsf{halt} \xrightarrow{\bar{c}} \mathsf{new}\ K; (\mathsf{new}\ )\langle K\rangle\mathsf{halt} = (\mathsf{new}\ K)\langle K\rangle\mathsf{halt}$

Also $\mathsf{recv}_c(x); \mathsf{send}_d\langle x\rangle; \mathsf{halt} \xrightarrow{c} (x)\mathsf{send}_d\langle x\rangle; \mathsf{halt}$

Hence
$P \xrightarrow{\tau} (\mathsf{new}\ K)\langle K\rangle\mathsf{halt}\ @\ (x)\mathsf{send}_d\langle x\rangle; \mathsf{halt} = (\mathsf{new}\ K)(\mathsf{halt} \mid \mathsf{send}_d\langle K\rangle; \mathsf{halt})$

Equivalence on processes

A test is of the form (Q, β) where Q is a closed process and β is a barb.

A process P passes the test (Q, β) iff

$$(P \mid Q) \xrightarrow{\tau} Q_1 \ldots \xrightarrow{\tau} Q_n \xrightarrow{\beta} A$$

for some $n \geq 0$, some processes $Q_1, \ldots, Q_n$ and some agent A.

Q is the "environment" and we test whether the process together with the environment inputs or outputs on a particular channel.

Testing preorder $P_1 \sqsubseteq P_2$ iff for every test (Q, β), if P_1 passes (Q, β) then P_2 passes (Q, β).

Testing equivalence $P_1 \simeq P_2$ iff $P_1 \sqsubseteq P_2$ and $P_2 \sqsubseteq P_1$.

Secrecy

Consider process P with only free variable x.

We will consider x as secret if for all terms M, M' we have $P[M/x] \simeq P[M'/x]$.

I.e. an observer cannot detect any changes in the value of x.

Example Consider $P \triangleq \mathsf{send}_c\langle x\rangle; \mathsf{halt}$.

x is being sent out on a public channel. Consider test $(Q, \overline{d})$ where

environment $Q \triangleq \mathsf{recv}_c(x); \mathsf{check}\ (x == 0); \mathsf{send}_d\langle 0\rangle; \mathsf{halt}$.

We have $P[0/x] \mid Q \xrightarrow{\tau} \mathsf{halt} \mid \mathsf{send}_d\langle 0\rangle; \mathsf{halt} \xrightarrow{\overline{d}} \langle 0\rangle(\mathsf{halt} \mid \mathsf{halt})$.

Hence $P[0/x]$ passes the test. However $P[\mathsf{succ}\ (0)/x]$ fails the test.

Hence P does not preserve secrecy of x.

Information flow analysis for the Spi-calculus

We classify data into three classes

secret	data which should not be leaked
public	data which can be communicated to anyone
any	arbitrary data

Subsumption relation on classes:

$$\begin{array}{lll} \mathsf{secret} & \preceq \mathsf{any} & \\ \mathsf{public} & \preceq \mathsf{any} & \\ T & \preceq T & \text{for } T \in \{\mathsf{secret}, \mathsf{public}, \mathsf{any}\} \end{array}$$

An environment E provides information about the classes to which names and variables belong.

We define typing rules for the following kinds of judgments

	$\vdash E$	environment E is well formed
E	$\vdash M : T$	term M is of class T in environment E
E	$\vdash P$	process P is well typed in environment E

E.g. secret data should not be sent on public channels.

Data of level any should be protected as if it is of level secret, but can be exploited only as if it had level public.

Our goal is to define typing rules to filter out processes that leak secrets.

Informally we would like to show that if environment E has only any variables and public names and $E \vdash P$ then P does not leak any variables $x \in dom(E)$.

Our previous example:

$P \triangleq \mathsf{send}_c\langle x \rangle; \mathsf{halt}$

Consider $E = \{x : \mathsf{any}, c : \mathsf{public} :: L_1, d : \mathsf{public} :: L_2\}$

(L_1 and L_2 will be explained later.)

x is of level any but is sent out on c of level public, which will be forbidden by our typing rules.

Consider the protocol

$A \longrightarrow S : A, B$

$S \longrightarrow A : \{A, B, Na, \{Nb\}_{K_{sb}}\}_{K_{sa}}$

$A \longrightarrow B : \{Nb\}_{K_{sb}}$

A principal X may play the role of A in one session and of B in another session.

We need a clear way of distinguishing the messages received and their components.

This is important only for messages sent on secret channels and for messages encrypted with secret keys.

We adopt the following standard format:

Messages sent on secret channels should have three components of levels secret, any and public respectively.

Consider protocol

$B \longrightarrow A : Nb$

$A \longrightarrow B : \{M, Nb\}_{K_{ab}}$

By replaying nonces, an attacker can find out whether the same M is sent more than once, or different ones. Hence he gets

some partial information about the contents of the messages.

To prevent this we include an extra fresh nonce (confounder) in each message encrypted with secret keys.

$A \longrightarrow B : \{M, Nb, Na\}_{K_{ab}}$

We adopt the following standard format for messages encrypted with secret keys: $\{M_1, M_2, M_3, n\}_K$

where M_1 has level secret, M_2 has level any, M_3 has level public,

and n is the confounder.

n can be used as confounder only in this term and nowhere else.

This information is remembered by the environment E.

I.e. if $n : T :: \{M_1, M_2, M_3, n\}_K \in E$ then

we know that n is used as a confounder only in that message.

The typing rules

The empty environment is denoted $\emptyset$.

Well formed environments:

$$\vdash \emptyset$$

$$\frac{\vdash E \quad x \notin dom(E)}{\vdash E, x : T}$$

$$\frac{\begin{array}{cc} \vdash E & n \notin dom(E) \\ E \vdash M_1 : T_1 \ldots E \vdash M_k : T_k & E \vdash N : R \end{array}}{\vdash E, n : T :: \{M_1, \ldots, M_k, n\}_N}$$

Environment lookups and subsumption:

$$\frac{E \vdash M : T \qquad T \sqsubseteq R}{E \vdash M : R}$$

$$\frac{\vdash E \qquad x : T \in E}{E \vdash x : T}$$

$$\frac{\vdash E \qquad n : T :: \{M_1, \ldots, M_k, n\}_N \in E}{E \vdash n : T}$$

$$\frac{\vdash E}{E \vdash 0 : \mathsf{public}}$$

$$\frac{E \vdash M : T}{E \vdash \mathsf{succ}\,(M) : T}$$

$$\frac{E \vdash M : T \qquad E \vdash N : T}{E \vdash \langle M, N \rangle : T}$$

Encryption

$$\frac{E \vdash M_1 : T \quad \ldots \quad E \vdash M_k : T \quad E \vdash N : \textsf{public} \quad T = \textsf{public} \text{ if } k = 0}{E \vdash \{M_1, \ldots, M_k\}_N : T}$$

$$\frac{\begin{array}{ccc} E \vdash M_1 : \textsf{secret} & E \vdash M_2 : \textsf{any} & E \vdash M_3 : \textsf{public} \\ E \vdash N : \textsf{secret} & & n : T :: \{M_1, M_2, M_3, n\}_N \in E \end{array}}{E \vdash \{M_1, M_2, M_3, n\}_N : \textsf{public}}$$

$$\frac{E \vdash M : \mathsf{public} \quad E \vdash M_1 : \mathsf{public} \quad \ldots \quad E \vdash M_k : \mathsf{public} \quad E \vdash P}{E \vdash \mathsf{send}_M \langle M_1, \ldots, M_k \rangle ; P}$$

$$\frac{E \vdash M : \mathsf{secret} \quad E \vdash M_1 : \mathsf{secret} \quad E \vdash M_2 : \mathsf{any} \quad E \vdash M_3 : \mathsf{public} \quad E \vdash P}{E \vdash \mathsf{send}_M \langle M_1, M_2, M_3 \rangle ; P}$$

Only public data may be sent on public channels.

On secret channels, data is always sent in the standard format we have agreed upon.

We consider pairing as left-associative.

For example (M_1, M_2, M_3, M_4) is same as $((M_1, M_2), M_3, M_4)$

Similar rules for inputs.

$$\frac{E \vdash M : \mathsf{public} \quad E, x_1 : \mathsf{public}, \ldots, x_k : \mathsf{public} \vdash P}{E \vdash \mathsf{recv}_M(x_1, \ldots, x_k); P}$$

$$\frac{E \vdash M : \mathsf{secret} \quad E, x_1 : \mathsf{secret}, x_2 : \mathsf{any}, x_3 : \mathsf{public} \vdash P}{E \vdash \mathsf{recv}_M(x_1, x_2, x_3); P}$$

The appropriate class information for the input variables is added to the environment, and the new environment is used for typing the remaining process.

$$\frac{\vdash E}{E \vdash \mathsf{halt}}$$

$$\frac{E \vdash P \qquad E \vdash Q}{E \vdash P \mid Q}$$

$$\frac{E \vdash P}{E \vdash \mathsf{repeat}\ P}$$

$$\frac{E, n : T :: L \vdash P}{E \vdash \mathsf{new}\ n; P}$$

The newly created name can be chosen to be kept secret or can be revealed, and can be chosen to used as a confounder in some message.

$$\boxed{\frac{E \vdash M : T \qquad E \vdash N : R \qquad E \vdash P \qquad T, R \in \{\mathsf{public}, \mathsf{secret}\}}{E \vdash \mathsf{check}\ (M == N); P}}$$

Equality checks are not allowed on data of class any to prevent implicit information flow.

Example Consider $P \triangleq \mathsf{recv}_c(y); \mathsf{check}\ (x == y); \mathsf{send}_c\langle 0\rangle; \mathsf{halt}$ where x is the data whose secrecy we are interested in.

Secrecy of x is not maintained. $P[M/x]$ and $P[M'/x]$ are not equivalent for $M \neq M'$.

Consider test $(Q, \overline{d})$ where $Q \triangleq \mathsf{send}_c\langle M\rangle; \mathsf{recv}_c(z); \mathsf{send}_d\langle 0\rangle; \mathsf{halt}$.

$P[M/x] \mid Q$ passes the test:

$P[M/x] \mid Q \xrightarrow{\tau} \mathsf{check}\ (M = M); \mathsf{send}_c\langle 0\rangle; \mathsf{halt} \mid \mathsf{recv}_c(z); \mathsf{send}_d\langle 0\rangle; \mathsf{halt} \xrightarrow{\tau}$
$\mathsf{halt} \mid \mathsf{send}_d\langle 0\rangle; \mathsf{halt} \xrightarrow{\overline{d}} \langle 0\rangle(\mathsf{halt} \mid \mathsf{halt})$

$P[M'/x] \mid Q$ does not pass the test.

Similarly, case analysis on data of class any are disallowed.

$$\frac{E \vdash M : T \qquad E, x : T, y : T \vdash P \qquad T \in \{\mathsf{public}, \mathsf{secret}\}}{E \vdash \mathsf{let}\ (x, y) = M; P}$$

$$\frac{E \vdash M : T \qquad E \vdash P \qquad E, x : T \vdash Q \qquad T \in \{\mathsf{secret}, \mathsf{public}\}}{E \vdash \mathsf{case}\ M\ \mathsf{of}\ 0 : P,\ \mathsf{succ}\ (x) : Q}$$

Decryption

$$\frac{E \vdash L : T \quad E \vdash N : \mathsf{public} \quad E, x_1 : T, \ldots, x_k : T \vdash P \quad T \in \{\mathsf{secret}, \mathsf{public}\}}{E \vdash \mathsf{case}\ L\ \mathsf{of}\ \{x_1, \ldots, x_k\}_N : P}$$

$$\frac{\begin{array}{c} E \vdash L : T \qquad E \vdash N : \mathsf{secret} \qquad T \in \{\mathsf{secret}, \mathsf{public}\} \\ E, x_1 : \mathsf{secret}, x_2 : \mathsf{any}, x_3 : \mathsf{public}, x_4 : \mathsf{any} \vdash P \end{array}}{E \vdash \mathsf{case}\ L\ \mathsf{of}\ \{x_1, x_2, x_3, x_4\}_N : P}$$

The confounder x_4 in the second rule is assumed to be of type any because we have no more information about it.

Typing implies noleak of information

Suppose

- $\vdash E$
- all variables in $dom(E)$ are of level any and all names in $dom(E)$ are of level public.
- $E \vdash P$
- P has free variables $x_1, \ldots, x_k$
- $fn(M_i), fn(M'_i) \subseteq dom(E)$ for $1 \leq i \leq k$.

then $P[M_1/x_1, \ldots, M_k/x_k] \simeq P[M'_1/x_1, \ldots, M'_k/x_k]$

Well typed processes maintain secrecy of the free variables $(x_1, \ldots, x_k)$, i.e. they are not leaked.

Our previous example $P \triangleq \mathsf{recv}_c(y); \mathsf{check}\ (x == y); \mathsf{send}_c\langle 0\rangle; \mathsf{halt}$

We take $E \triangleq \{x : \mathsf{any}, c : \mathsf{public} :: \{n\}_0\}$. c is not meant to be used as a confounder, hence we have the dummy term $\{n\}_0$.

We have $\vdash E$.

In order to show $E \vdash P$ we need to find some T such that

$E, y : \mathsf{public} \vdash \mathsf{check}\ (x == y); \mathsf{send}_c\langle 0\rangle; \mathsf{halt}$.

But this is impossible because equality checks should not involve data of class any.

Hence the process doesn't type-check, as required.

Consider $P \triangleq \mathsf{new}\ K; \mathsf{new}\ m; \mathsf{new}\ n; \mathsf{send}_c\langle\{m, x, 0, n\}_K\rangle; \mathsf{halt}.$

We take $E \triangleq \{x : \mathsf{any}, c : \mathsf{public} :: \{n\}_0\}$. We have $\vdash E$.

To show $E \vdash P$ we choose

$E' \triangleq E, K : \mathsf{secret} :: \{K\}_0, m : \mathsf{secret} :: \{m\}_0, n : \mathsf{secret} :: \{m, x, 0, n\}_K$

and show that $E' \vdash \mathsf{send}_c\langle\{m, x, 0, n\}_K\rangle; \mathsf{halt}.$

This is ok because $E' \vdash m : \mathsf{secret}$, $E' \vdash x : \mathsf{any}$, $E' \vdash 0 : \mathsf{public}$, $E' \vdash n : \mathsf{secret}$, $E' \vdash K : \mathsf{secret}$ and $E' \vdash \mathsf{halt}$.