Abstractions for Network Security

Michele Bugliesi

joint work with

Riccardo Focardi

Padova, 13.3.2009
Two, often conflicting, requirements

Security ...
Calls for formal specifications of the low-level safeguards built against the threats such applications are exposed to.

Programming ...
Needs high-level abstractions and reasoning methods to hide away low-level details and focus on application-level properties.
### The iKP protocol family

| C = client, M = merchant, A = amex |

[IBM 1995]
An e-payment service

The $i$KP protocol family … with many dots to fill [IBM 1995]

$C = $ client, $M = $ merchant, $A = $ amex

- **INIT**: $C \rightarrow M$
- **INVC**: $C \leftarrow M$
- **PAYM**: $C \rightarrow M$
- **REQ**: $M \rightarrow A$
- **RESP**: $M \leftarrow A$
- **CONF**: $C \leftarrow M$
An e-payment service

The iKP protocol family ... with many dots to fill

C = client, M = merchant, A = amex

[IBM 1995]
An e-payment service

The iKP protocol family ... with many dots to fill [IBM 1995]

C = client, M = merchant, A = amex

Messy, ain’t it?

INIT\[C \quad \langle SAL T_C, CID \rangle \quad \rightarrow \quad M\]

INVC\[C \quad \langle ID, TID, DATE, NONCE, H(\ldots) \rangle \quad \leftarrow \quad M\]

PAYM\[C \quad ENC_A(DESC, H(\ldots), PIN, NONCE, \ldots) \quad \rightarrow \quad M\]

REQ\[M \quad ENC_A(DESC, H(\ldots), PIN, NONCE, \ldots) \quad \rightarrow \quad A\]

RESP\[M \quad Y / N, SIGN_A(Y / N, H(\ldots)) \quad \leftarrow \quad A\]

CONF\[C \quad Y / N, SIGN_A(Y / N, H(\ldots)) \quad \leftarrow \quad M\]
An e-payment service

Abstract specification

tid = transaction id, desc = order description, pin = client pin

INIT C  -----→ M
INVC C ┌─────┐ M
PAYM C  -----→ M

REQ M  -----→ A
RESP M  ┌─────┐ A
CONF C  ┌─────┐ M
Abstract specification

\( tid = \) transaction id, \( desc = \) order description, \( pin = \) client pin

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Source 1</th>
<th>Source 2</th>
<th>Target 1</th>
<th>Target 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT</td>
<td>C (\langle C\rangle)</td>
<td>C (\langle M^\circ:desc, tid\rangle)</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>INVC</td>
<td>C (\langle desc, pin, tid\rangle^A)</td>
<td>M</td>
<td>C</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>PAYM</td>
<td>C (\langle desc, pin, tid\rangle^A)</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>REQ</td>
<td>M (\langle M^\circ:desc, tid\rangle)</td>
<td>M</td>
<td>A</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>RESP</td>
<td>M (\langle desc, pin, tid\rangle^A)</td>
<td>A</td>
<td>M</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>CONF</td>
<td>C (\langle A^\circ:y/n\rangle)</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
</tbody>
</table>
An e-payment service

Abstract specification

tid = transaction id, desc = order description, pin = client pin
Wouldn’t it be better if we could reason like this?

\[
\begin{array}{llll}
\text{INIT} & C & \langle C \rangle & M \\
\text{INVC} & C & \langle M^{\circ}: \text{desc}, \text{tid} \rangle & M \\
\text{PAYM} & C & \langle \text{desc}, \text{pin}, \text{tid} \rangle^A & M \\
\text{REQ} & M & \langle M^{\circ}: \text{desc}, \text{tid} \rangle & A \\
\text{RESP} & M & \langle \text{desc}, \text{pin}, \text{tid} \rangle^A & A \\
\text{CONF} & C & \langle A^{\circ}: y/n \rangle & M
\end{array}
\]
Security based on private channels:

$$\nu n(\overline{\eta}<m> \mid n(x).P) \simeq P\{m/x\}$$

Scope rules guarantee that the environment cannot access data exchanged over a private channel.
Traditional process algebraic takes at specification

**pi calculus**

- Security based on private channels:

\[ \nu n(\overline{n}\langle m \rangle \mid n(x).P) \simeq P\{m/x\} \]

- Scope rules guarantee that the environment cannot access data exchanged over a **private channel**.

Simple, but ...
Traditional process algebraic takes at specification

**pi calculus**

- Security based on private channels:
  \[ \nu n(\overline{\langle m \rangle} \mid n(x).P) \equiv P\{m/x\} \]

- Scope rules guarantee that the environment cannot access data exchanged over a **private channel**.

**Simple, but ...**

- hard to implement in distributed setting
- poorly expressive: only two communication modes (either full protection or no protection)
Traditional process algebraic takes at specification

spi/applied pi calculus

- Security based on formal cryptography
- Scope rules guarantee that the environment cannot access data protected by private keys. Traffic observable:

\[ \nu k \langle \text{net}(m) \rangle | \text{net}(x).\text{decrypt } x \text{ as } \{y\}_k \text{ in } P \not\equiv P\{m/y\} \]
Traditional process algebraic takes at specification

spi/applied pi calculus

- Security based on formal cryptography
- Scope rules guarantee that the environment cannot access data protected by private keys. Traffic observable:

\[ \nu k(\text{net}(\{m\}_k) \mid \text{net}(x) . \text{decrypt } x \text{ as } \{y\}_k \text{ in } P) \not\cong P\{m/y\} \]

More realistic, direct implementation, but ...
Traditional process algebraic takes at specification

**spi/applied pi calculus**

- Security based on formal cryptography
- Scope rules guarantee that the environment cannot access data protected by private keys. Traffic observable:

\[
\nu_k(\text{net}\langle\{m\}_k\rangle | \text{net}(x).\text{decrypt } x \text{ as } \{y\}_k \text{ in } P) \not\equiv P\{m/y\}
\]

More realistic, direct implementation, but ...

- high-level specifications easily cluttered with details of underlying cryptographic protocols
- costly verification techniques and proofs
Desiderata

- Encapsulate cryptographic protocols into higher-level programming and specification abstractions
- Extract high-level (security) properties on data from the underlying protocols:

\[ \text{in}(x : \text{AuthenticFrom}(y)).P\{x, y\} \]
Need more adequate abstractions

Desiderata

- Encapsulate cryptographic protocols into higher-level programming and specification abstractions
- Extract high-level (security) properties on data from the underlying protocols:

  \[ \text{in}(x : \text{AuthenticFrom}(y)).P\{x, y\} \]

- Rephrasing a successful slogan:

  **language-based network security**

- A more effective framework for the analysis of authorization policies, trust-establishment protocols, ... etc.
Need more adequate abstractions

**Desiderata**

- Encapsulate cryptographic protocols into higher-level programming and specification abstractions
- Extract high-level (security) properties on data from the underlying protocols:

\[
\text{in}(x : \text{AuthenticFrom}(y)).P\{x, y\}
\]

- Rephrasing a successful slogan:

  **language-based network security**

- A more effective framework for the analysis of authorization policies, trust-establishment protocols, ... etc.

Ideas and motivations shared with others of course:

[Abadi, Fournet, Gonthier 2000, 2002]

[Adão, Fournet 2006]
A calculus (of course:)
- library of communication primitives;
- convey powerful built-in security guarantees;
- convenient for high-level application design and programming;
- amenable to cryptographic implementations

A framework for security verification in adversarial settings
- Dolev-Yao intruders, but without explicit cryptography
- verification methods based on bisimulation equivalences
- co-inductive proofs, and up-to techniques
Introducing the

A pi calculus for network security

Security verification and Intruder models

Conclusions
An asynchronous pi calculus

Syntax

\[ H, K ::= 0 \mid H\mid K \mid (\nu a)K \mid \text{if } u = v \text{ then } H \text{ else } K \mid A\langle u \rangle \mid \overline{u}\langle a : v \rangle^\eta \mid a\langle u : y \rangle^\eta.H \]

- \text{pi calculus}
- \text{matching}
- \text{guarded rec defs}
- \text{output (a name)}
- \text{input (a name)}

Names act (also) as principal identities
An asynchronous pi calculus

Syntax

\[ H, K ::= \begin{align*}
0 & | H|K & | (\nu a)K \\
& | \text{if } u = v \text{ then } H \text{ else } K \\
& | A\langle u \rangle \\
& | \overline{u}\langle a : v \rangle^\eta \\
& | a(u : y)^\eta.H
\end{align*} \]

- \( \overline{u}\langle a : v \rangle^\eta \): \( a \) sends an authenticated message \( v \) to \( u \)
- \( a(u : y)^\eta \): \( a \) receives from \( u \) an authenticated message \( y \)
- sender ‘−’ implies no authentication
- \( \eta \in \{\circ, \bullet\} \) signals secrecy mode

Names act (also) as principal identities
Semantics – informally

Plain exchange: $\overline{u}(\neg : m)^\circ$

No secrecy/authentication guarantee

Intruder

\[ \begin{array}{c}
? \\
\downarrow m \\
\downarrow m \\
\downarrow m \\
\downarrow m' \\
\downarrow m' \\
\downarrow m' \\
\end{array} \]
Authentic exchanges: $\overline{u} \langle a : m \rangle^o$

Authenticate source: $a$ must be a name (prevent impersonation). Protects against replays, no secrecy.
Semantics – informally

Secret exchanges: $\overline{u}(\langle - : m \rangle)$

Seals $m$. Protects against eavesdropping, no authentication
Semantics – informally

Secure exchanges: $\overline{u}\langle a : m \rangle^*$

Combines guarantees of secure and authentic outputs

\[ \text{Intruder} \]

\[ \text{Intruder} \]

\[ \text{Intruder} \]

\[ \text{Intruder} \]
An example

Simple e-banking transaction

Client $\rightarrow$ Bank

Client $\leftarrow$ Bank

Client $\rightarrow$ Bank

Client and Bank establish new session to carry out transaction
Simple e-banking transaction

\[
\nu c \text{bank} \langle - : c, id, pin \rangle^* \rightarrow \text{Bank}
\]

\[
\text{Client} \leftarrow \text{Bank}
\]

\[
\text{Client} \rightarrow \text{Bank}
\]

\text{Client} and \text{Bank} establish new session to carry out transaction
An example

Simple e-banking transaction

\[
\begin{align*}
\text{CLIENT} & \xrightarrow{(νc)\overline{\text{bank}}\langle− : c, id, pin\rangle^*} \text{BANK} \\
\text{CLIENT} & \xleftarrow{(νb)\overline{\text{c}}\langle\text{bank} : b\rangle^\circ} \text{BANK} \\
\text{CLIENT} & \xrightarrow{} \text{BANK}
\end{align*}
\]

CLIENT and BANK establish new session to carry out transaction
Simple e-banking transaction

\[
\begin{align*}
\text{CLIENT} & \quad (\nu c)\text{bank} \langle - : c, id, pin \rangle^* \\
\text{CLIENT} & \quad (\nu b)\overline{c} \langle \text{bank} : b \rangle^\circ \\
\text{CLIENT} & \quad \overline{b} \langle c : "deposit", amount \rangle^\circ \\
\end{align*}
\]

\text{CLIENT and BANK establish new session to carry out transaction}
Simple e-banking transaction

\[
\text{TRANSACTION}(id, pin) \overset{\text{def}}{=} \text{CLIENT}(id, pin) \mid \text{BANK}(id, pin)
\]

\[
\text{CLIENT}(id, pin) \overset{\text{def}}{=} (\nu c)\text{bank}(\neg : c, id, pin) \cdot | c(\text{bank} : x_b) \cdot \overline{x_b}(c : "deposit", amount) \cdot
\]

\[
\text{BANK}(id, pin) \overset{\text{def}}{=} \text{bank}(\neg : x_c, x_id, x_pin) \cdot
\]

if \((x_id = id \land x_pin = pin)\) then
\[
(\nu b)\overline{x_c} \langle \text{bank} : b \rangle \cdot | b(c : x_{op}, x_{amount}) \cdot \ldots
\]
Calculus provides identity-based security primitives

- Idealized abstractions for programming secure distributed interactions
- Convenient, but security guarantees must be tested in adversarial settings
Calculus provides identity-based security primitives

- Idealized abstractions for programming secure distributed interactions
- Convenient, but security guarantees must be tested in adversarial settings

Introduce networks, a lower-level calculus providing all capabilities that must be assumed available to the intruder
Network-level communications

\[ M, N ::= \overline{u}\langle a : v \parallel t\rangle^\eta \quad \text{network output} \]
\[ \quad \mid a(u : y \parallel z)^\eta . M \quad \text{network input} \]
\[ \quad \mid \ldots \]

Same rationale as high-level primitives
Network-level communications

\[
M, N ::= \overline{u}(a : v \parallel t)^{\eta} \quad \text{network output}
\]
\[
| \quad a(u : y \parallel z)^{\eta} \cdot M \quad \text{network input}
\]
\[
| \quad \ldots
\]

Same rationale as high-level primitives

- \( t \) in output represents the network-level view of payload \( v \);
- on input \( y \) and \( z \) get bound to the payload and its network-level view
Intruder specific capabilities

\[ M, N ::= \ldots \]
\[ \uparrow z(x : y \parallel w)^\eta_i M \quad \text{intercept} \]
\[ !i.M \quad \text{forward/replay} \]

Intercepting \( \overline{b(a : m \parallel n)}^\circ \):
- caches copy indexed by fresh \( i \), binds \( \{ b/z, a/x, n/w \} \)
- binding for \( y \) depends on \( \eta \) – more below
- All outputs may be intercepted

\( !i \): use index to reply/forward the cached copy
- push-back cached message if authentic
- produce a replica of cached copy if non-authentic
### Compilation

<table>
<thead>
<tr>
<th>Expression</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[\overline{u} \langle - : v \rangle^\circ]$</td>
<td>$\triangleq \overline{u} \langle - : v \parallel v \rangle^\circ$</td>
</tr>
<tr>
<td>$[\overline{u} \langle a : v \rangle^\circ]$</td>
<td>$\triangleq (\nu c) \overline{u} \langle a : v \parallel c \rangle^\circ$</td>
</tr>
<tr>
<td>$[\overline{u} \langle a : v \rangle^\bullet]$</td>
<td>$\triangleq (\nu c) \overline{u} \langle a : v \parallel c \rangle^\bullet$</td>
</tr>
<tr>
<td>$[b(u : x)^\eta.H]$</td>
<td>$\triangleq b(u : x \parallel y)^\eta.[H]$ \hspace{1cm} (y \cap fv(H) = \emptyset)$</td>
</tr>
</tbody>
</table>
Principals as network processes

Compilation

\[
\begin{align*}
\bar{u}\langle - : v \rangle & \triangleq \bar{u}\langle - : v \parallel v \rangle \\
\bar{u}\langle a : v \rangle & \triangleq (\nu c)\bar{u}\langle a : v \parallel c \rangle \\
\bar{u}\langle a : v \rangle^\bullet & \triangleq (\nu c)\bar{u}\langle a : v \parallel c \rangle^\bullet \\
[b(\bar{u} : x)^\eta . H] & \triangleq b(\bar{u} : x \parallel y)^\eta . [H] \quad (y \cap fv(H) = \emptyset)
\end{align*}
\]

Payload and its network view coincide in plain outputs
Principals as network processes

Compilation

\[
\begin{align*}
[u(− : v)°] & \triangleq u(− : v \parallel v)° \\
[u(a : v)°] & \triangleq (v c)u(a : v \parallel c)° \\
[u(a : v)⋆] & \triangleq (v c)u(a : v \parallel c)⋆ \\
[b(u : x)^η.H] & \triangleq b(u : x \parallel y)^η.[H] \quad (y \cap fv(H) = \emptyset)
\end{align*}
\]

Payload and its network view coincide in plain outputs

A fresh \(c\) as network view of secret/authentic outputs

- \(c \sim\) randomized encryption in secret outputs
- \(c \sim\) time-variant signatures in authentic inputs
Partition identities into *Trusted* $N_t$ and *Untrusted* $N_u$. 
Partition identities into Trusted $N_t$ and Untrusted $N_u$.

<table>
<thead>
<tr>
<th>Trusted Principals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network processes that only impersonate identities in the set $N_t$ and only create fresh names in the same set.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intruders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network processes that only impersonate identities in the set $N_u$ and only creates fresh names in the same set.</td>
</tr>
</tbody>
</table>
Partition identities into Trusted $N_t$ and Untrusted $N_u$.

<table>
<thead>
<tr>
<th>Trusted Principals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network processes that only impersonate identities in the set $N_t$ and only create fresh names in the same set.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intruders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network processes that only impersonate identities in the set $N_u$ and only creates fresh names in the same set.</td>
</tr>
</tbody>
</table>

- $P$ impersonates an identity $a$ iff $P$ uses $a$ as the subject of an input or the source of an authentic output:
  - syntactic (implementation sound) restrictions rule out dynamic impersonation
Communication

\[ \overline{b \langle a : m \parallel c \rangle^\eta} \ | \ b(a : y \parallel z)^\eta.N \rightarrow N\{m/y, c/z\} \]

- secrecy mode \( \eta \) at end-point must match;
- sender must be known to the receiver in case the message is authentic
Intercept

\[ \overline{b} \langle a : m \parallel c \rangle^\bullet \mid \uparrow z(x : y \parallel w)^\bullet_i . N \rightarrow \overline{b} \langle a : m \parallel c \rangle^\bullet_i \mid N\{b/z, a/x, c/y, c/w\} \quad (b \in N_t) \]

\[ \overline{b} \langle a : m \parallel c \rangle^\eta \mid \uparrow z(x : y \parallel w)^\eta_i . N \rightarrow \overline{b} \langle a : m \parallel c \rangle^\eta_i \mid N\{b/z, a/x, m/y, c/w\} \quad (b \in N_u \text{ or } \eta = \circ) \]

Intruder gets access to payload \( m \) just in case

- it has controls over the receiving identity
- output is not secret
Intruder reductions

**Forward and Replay**

\[
\bar{b} \langle a : m \parallel c \rangle_i^n \parallel !i.N \quad \rightarrow \quad \bar{b} \langle a : m \parallel c \rangle_i^n \parallel N
\]

\[
\bar{b} \langle - : m \parallel c \rangle_i^n \parallel !i.N \quad \rightarrow \quad \bar{b} \langle - : m \parallel c \rangle_i^n \parallel \bar{b} \langle - : m \parallel c \rangle_i^n \parallel N
\]

No surprise.
Introduction

A pi calculus for network security
  - Security verification and Intruder models
  - Conclusions
Barbed Equivalence $\simeq$

The largest symmetric relation $\simeq$ such that

- $M \simeq N$ and $M \downarrow b$ imply $N \downarrow b$.
- $M \simeq N$ and $M \rightarrow M'$ imply $N \rightarrow N'$ with $M' \simeq N'$.
- $M \simeq N$ implies $M \mid I \simeq N \mid I$ for all (closed) intruders $I$ and $(\nu \tilde{n}) M \simeq (\nu \tilde{n}) N$ for all names $\tilde{n} \in \mathbf{N}_u$

Notation: write $H \simeq K$ to note $[H] \simeq [K]$
Intruder equivalence

Barbed Equivalence \(\simeq\)

The largest symmetric relation \(\simeq\) such that

- \(M \simeq N\) and \(M \downarrow b\) imply \(N \downarrow b\).
- \(M \simeq N\) and \(M \rightarrow M'\) imply \(N \rightarrow N'\) with \(M' \simeq N'\).
- \(M \simeq N\) implies \(M \mid I \simeq N \mid I\) for all (closed) intruders \(I\)
  and \((\nu \tilde{n})M \simeq (\nu \tilde{n})N\) for all names \(\tilde{n} \in N_u\)

Notation: write \(H \simeq K\) to note \([H] \simeq [K]\)

Remarks

- Not fully contextual, only close under intruder contexts
- Processes equivalent if no intruder can distinguish them
Intruder equivalence

Only a restricted form of closure under composition with trusted principals.

**Theorem (Weak Compositionality)**

Let $P$, $Q$ be trusted principals. $P \simeq Q$ implies

- $(\nu n)P \simeq (\nu n)Q$, for all $n \in \mathbb{N}_t$;
- $P|R \simeq Q|R$, for all trusted principals $R$ which do not impersonates identities in $\text{fn}(P) \cup \text{fn}(Q)$. 
Traffic analysis

Even in *secure* mode, traffic is observable, and does not guarantee delivery.

\[ \nu n(\overline{n}(a : m)^\cdot | n(a : y)^\cdot.H) \not\equiv H\{m/x\} \]
Distinguishing (in) equalities

Traffic analysis

Even in *secure* mode, traffic is observable, and does not guarantee delivery.

\[ \nu n(\overline{n}(a:m)^* | n(a:y)^*.H) \not\equiv H\{m/x\} \]

Contrast with pi/spi

\[ \nu n(\overline{n}(m) | n(x).P) \simeq_{\text{pi}} P\{m/x\} \]
\[ \nu k(\overline{n}\{m\}_k | n(x).\text{decrypt } x \text{ as } \{y\}_k \text{ in } P) \not\equiv_{\text{spi}} P\{m/y\} \]
Distinguishing equalities

Forward Secrecy

- $\nu n(\overline{n}\langle a : m\rangle^\bullet | n(a : y)^\bullet.\overline{p}(\neg : n))$
  $\simeq \nu n(\overline{n}\langle a : m'\rangle^\bullet | n(a : y)^\bullet.\overline{p}(\neg : n))$

- making a channel public does not leak previous exchanges
Distinguishing equalities

**Forward Secrecy**

- $\nu n(\overline{n} \langle a : m \rangle \cdot | n(a : y) \cdot \overline{p}(\overline{\_} : n))$
  $\simeq\nu n(\overline{n} \langle a : m' \rangle \cdot | n(a : y) \cdot \overline{p}(\overline{\_} : n))$

- making a channel public does not leak previous exchanges
- syntactic restrictions on input guarantee forward secrecy
Distinguishing equalities

Forward Secrecy

- \( \nu n(\overline{n}(a : m) \cdot | n(a : y) \cdot \overline{p}(- : n)) \)
- \( \simeq \nu n(\overline{n}(a : m') \cdot | n(a : y) \cdot \overline{p}(- : n)) \)
- making a channel public does not leak previous exchanges
- syntactic restrictions on input guarantee forward secrecy

Contrast with pi/spi

- \( \nu n(\overline{n}(m) \cdot | n(x) \cdot \overline{p}(n)) \simeq_{pi} \nu n(\overline{n}(m') \cdot | n(x) \cdot \overline{p}(n)) \)
- \( \nu k(\overline{n}\{m\}_k) \cdot | \text{decrypt } x \text{ as } \{y\}_k \text{ in } \overline{n}(k) \)
  \( \not\simeq_{spi} \nu n(\overline{n}\{m'\}_k) \cdot | \text{decrypt } x \text{ as } \{y\}_k \text{ in } \overline{n}(k) \)
Assume $b$ trusted
Assume $b$ trusted

- Secret output guarantees secrecy of payload

$$b\langle a : m \rangle \approx b\langle a : m' \rangle$$
Assume $b$ trusted

- Secret output guarantees secrecy of payload

\[ \overline{b}a : m \cdot \simeq \overline{b}a : m' \cdot \]

- Sending a secret is equivalent to sending a nonce

\[ \overline{b} (- : m) \cdot \simeq (\nu n) \overline{b} (- : n) \cdot \]
Assume $b$ trusted

- Secret output guarantees secrecy of payload
  \[
  \overline{b}\langle a : m \rangle \leadsto \overline{b}\langle a : m' \rangle
  \]

- Sending a secret is equivalent to sending a nonce
  \[
  \overline{b}\langle - : m \rangle \leadsto (\nu n)\overline{b}\langle - : n \rangle
  \]

- Spi-calculus characterization of secrecy
  \[
  \overline{b}\langle a : m \rangle | b(a : x) . H(x) \leadsto \overline{b}\langle a : m' \rangle | b(a : x) . H(x)
  \]
  whenever $H(m) \leadsto H(m')$
Assume a trusted.
Authentication specific equalities

Assume a trusted.

- Authentic payload may not be received from intruder

\[ b(a : x)^\eta . H \simeq 0 \]
Assume a trusted.

- Authentic payload may not be received from intruder

\[ b(a : x)^\eta . H \simeq 0 \]

- Authentic messages from trusted identities cannot be replayed

\[ \overline{b} \langle a : m \rangle^\eta \mid b(a : x)^\eta . b(a : y)^\eta . H \simeq \overline{b} \langle a : m \rangle^\eta \mid b(a : x)^\eta . 0 \]
Assume a trusted.

- Authentic payload may not be received from intruder
  \[ b(a : x)^\eta.H \simeq 0 \]

- Authentic messages from trusted identities cannot be replayed
  \[ \overline{b}\langle a : m\rangle^\eta | b(a : x)^\eta.b(a : y)^\eta.H \simeq \overline{b}\langle a : m\rangle^\eta | b(a : x)^\eta.0 \]

- Classical characterization of authentication (and integrity)
  \[ \overline{b}\langle a : m\rangle^\eta | b(a : x)^\eta.H(x) \simeq \overline{b}\langle a : m\rangle^\eta | b(a : x)^\eta.H(m) \]
Based on LTS as usual, but LTS defined in two steps.
Coinductive proof techniques

Based on LTS as usual, but LTS defined in two steps.

Internal Transitions

Standard interpretation: encode arbitrary contexts.

\[ N \xrightarrow{\alpha} N' \]
Coinductive proof techniques

Based on LTS as usual, but LTS defined in two steps.

Internal Transitions

Standard interpretation: encode arbitrary contexts.

\[ N \xrightarrow{\alpha} N' \]

External/Observable Transitions

A subset of internal transitions, encoding intruder contexts only.

\[ N \xleftarrow{\alpha} N' \]
Internal transitions (sample)

Interception

(Secret Output Intercepted)

\[
\text{b trusted}
\]

\[
\overline{b} \langle a : \tilde{m} \parallel \tilde{c} \rangle \quad (i) \nrightarrow \overline{b} \langle a : \tilde{m} \parallel \tilde{c} \rangle
\]

(Output Intercepted)

\[
\text{b untrusted or } \eta = \circ
\]

\[
\overline{b} \langle a : \tilde{m} \parallel \tilde{c} \rangle \eta \quad (i) \nrightarrow \overline{b} \langle a : \tilde{m} \parallel \tilde{c} \rangle \eta
\]

M. Bugliesi Abstractions for Network Security
Interception

(Secret Output Intercepted)

\[ \overline{b}\langle a : \tilde{m} \parallel \tilde{c} \rangle^\bullet \xrightarrow{(i)\dagger}\overline{b}\langle a : \tilde{m} \parallel \tilde{c} \rangle^\bullet \]

(Output Intercepted)

\[ \overline{b}\langle a : \tilde{m} \parallel \tilde{c} \rangle^\eta \xrightarrow{(i)\dagger}\overline{b}\langle a : \tilde{m} \parallel \tilde{c} \rangle^\eta \]

\( b \) trusted

\( b \) untrusted or \( \eta = \circ \)
Co-forward and replay

(Co-forward)

\[
\overline{b} \langle a : \tilde{m} \parallel \tilde{c} \rangle^\eta_i \xrightarrow{(i)} \overline{b} \langle a : \tilde{m} \parallel \tilde{c} \rangle^\eta
\]

(Co-replay)

\[
\overline{b} \langle - : \tilde{m} \parallel \tilde{c} \rangle^\eta_i \xrightarrow{(i)} \overline{b} \langle - : \tilde{m} \parallel \tilde{c} \rangle^\eta \quad \mid \quad \overline{b} \langle - : \tilde{m} \parallel \tilde{c} \rangle^\eta
\]
Observable transitions

A subset of the internal transitions

\[
\begin{align*}
N & \xrightarrow{\alpha} N' \\
\alpha & \not\in \left\{ b\langle a : \tilde{m} \parallel \tilde{c} \rangle^\circ \mid a \in N_t \right\} \\
N & \not\xrightarrow{\alpha} N'
\end{align*}
\]

- intruder cannot read messages directed to trusted identities (but can intercept those messages)
- intruder cannot impersonate trusted identities on output
Standard Asynchronous bisimilarity

The largest symmetric relation $\sim_a$ such that whenever $M \sim_a N$ and $M \xrightarrow{\alpha} M'$ with $bn(\alpha) \cap fn(N) = \emptyset$ one has:

- if $\alpha$ is an $\uparrow o \tau$ action, then $N \xrightarrow{\alpha} N'$ and $M' \sim_a N'$;
- if $\alpha$ is an input action, then $N \xrightarrow{\alpha} N'$ and $M' \sim_a N'$ or $N \xrightarrow{\tau} N'$ and $M' \sim_a N' \mid \overline{\alpha}$. 
The largest symmetric relation $\sim_a$ such that whenever $M \sim_a N$ and $M \xrightarrow{\alpha} M'$ with $bn(\alpha) \cap fn(N) = \emptyset$ one has:

- if $\alpha$ is an $\uparrow o \tau$ action, then $N \xrightarrow{\alpha} N'$ and $M' \sim_a N'$;
- if $\alpha$ is an input action, then $N \xrightarrow{\alpha} N'$ and $M' \sim_a N'$ or $N \xrightarrow{\tau} N'$ and $M' \sim_a N' | \bar{\alpha}$.

**Theorem**

*Soundness* $P \sim_a Q$ implies $P \simeq Q$, for all $P, Q$ trusted principals.
Problem: replicas get in the way
Consider $\overline{b} \langle - : m \rangle^\bullet \sim \overline{b} \langle - : m' \rangle^\bullet$

Simple enough, but one needs:

$$S = \{(\overline{b} \langle - : m \rangle^\bullet, \overline{b} \langle - : m' \rangle^\bullet), (\overline{b} \langle - : m || c \rangle^\bullet, \overline{b} \langle - : m' || c \rangle^\bullet),$$

$$\bigcup \{((\prod_{k} \overline{b} \langle - : m || c \rangle^\bullet, \prod_{k} \overline{b} \langle - : m' || c \rangle^\bullet) \mid k \geq 0\}\}$$
Lazy intruders to the rescue!

New LTS:

(Co-replay)

\[
N \xrightarrow{b(-:\tilde{m}\parallel\tilde{c})^\circ} N' \\
\overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle_i \parallel N \xrightarrow{(i)} \overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle_i \parallel N'
\]

(Co-forward)

\[
N \xrightarrow{b(a:\tilde{m}\parallel\tilde{c})^\circ} N' \\
\overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle_i \parallel N \xrightarrow{(i)} N'
\]
Lazy intruders to the rescue!

New LTS: replicas generated “on demand”

(Co-replay)

\[ N \xrightarrow{b(-:\tilde{m}||\tilde{c})^\circ} N' \]

\[ \overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle_i^\circ | N \xrightarrow{(i)} \overline{b}\langle - : \tilde{m} \parallel \tilde{c}\rangle_i^\circ | N' \]

(Co-forward)

\[ N \xrightarrow{b(a:\tilde{m}||\tilde{c})^\circ} N' \]

\[ \overline{b}\langle a : \tilde{m} \parallel \tilde{c}\rangle_i^\circ | N \xrightarrow{(i)} N' \]
Lazy intruders to the rescue!

New LTS: replicas generated “on demand”

(\text{Co-replay})

\[
\begin{array}{c}
N \xrightarrow{b(-:\tilde{m} \parallel \tilde{c})^\circ} N' \\
\hline
\bar{b}\langle - : \tilde{m} \parallel \tilde{c} \rangle_i \mid N \xrightarrow{(i)} \bar{b}\langle - : \tilde{m} \parallel \tilde{c} \rangle_i \mid N'
\end{array}
\]

(\text{Co-forward})

\[
\begin{array}{c}
N \xrightarrow{b(a:\tilde{m} \parallel \tilde{c})^\circ} N' \\
\hline
\bar{b}\langle a : \tilde{m} \parallel \tilde{c} \rangle_i \mid N \xrightarrow{(i)} N'
\end{array}
\]

\textbf{Note} \sim \text{ the resulting bisimilarity}
Good news!

\[ b\langle - : m \rangle^* \sim b\langle - : m' \rangle^* \]

Need a simpler candidate now

\[ S = \{(b\langle - : m \rangle^*, b\langle - : m' \rangle^*), (\overline{b}\langle - : m || c \rangle^*, \overline{b}\langle - : m' || c \rangle^*)\} \]
Good news!

\[ \sim \text{ candidates are more convenient to use} \]

\[ \overline{b}(- : m) \sim \overline{b}(- : m') \]

Need a simpler candidate now

\[ S = \{(\overline{b}(- : m), \overline{b}(- : m') \}, (\overline{b}(- : m || c)_{\downarrow}, \overline{b}(- : m' || c)_{\downarrow})\} \]

Lazyness does not affect discriminating power

**Theorem**

*On trusted principals, \( \sim = \sim \)***
Dolev Yao Intruder

Must be assumed to have complete control of the network
Dolev Yao Intruder

Must be assumed to have complete control of the network
Dolev Yao Intruder

Must be assumed to have complete control of the network
More on intruders

How about our intruder?

Must be assumed to have complete control of the network
How about our intruder?

Uh oh ... some exchanges may go unnoticed!
More on intruders

How about our intruder?

Uh oh ... some exchanges may go unnoticed!

Our Dolev-Yao intruder is NOT in full control!
Men in the middle

- Easily characterized: simply disregard $\tau$-transitions
- Call $\sim^*$ the resulting labelled bisimilarity
Men in the middle

- Easily characterized: simply disregard $\tau$-transitions
- Call $\sim^*$ the resulting labelled bisimilarity

Dolev-Yao and Oblivious intruders have same discriminating power

Theorem ($\sim$ vs $\sim^*$)

On trusted principals, $\sim = \sim^*$. 
Men in the middle

- Easily characterized: simply disregard $\tau$-transitions
- Call $\sim^*$ the resulting labelled bisimilarity

Dolev-Yao and Oblivious intruders have same discriminating power

Theorem ($\sim$ vs $\sim^*$)

On trusted principals, $\sim = \sim^*$.

- $\sim^*$ is Adão and Fournet’s bisimilarity, extended to our calculus
- as in that case, it is synchronous (remember? no $\tau$’s)
- more convenient to use in proofs
Outline

- Introduction
- A pi calculus for network security
- Security verification and Intruder models
  - Conclusions
What we have done

Designed an API for secure distributed communication
- reasonably high-level to be used for programming
- certainly useful as intermediate language
- designed with implementation in mind

Developed Verification Techniques
- based on standard techniques
- as well as on analysis of intruder discriminating power
- a hierarchy of intruders ordered by their discriminating power
Present and Future Work

[with Paolo Modesti]

Fully-abstract translation to spi/applied-pi calculus
  - strong authentication non-trivial
Language design and implementation based on core API
  - local secret channels, more structure for principals, ...
  - types, sessions ...

More abstractions
  - “indirect” authentication/secrecy
  - hashing and data correlation, ...

Can we really do without explicit cryptography?
  - ... and let the compiler take care of it? To what extent?
Thank You!
Implementation of $\overline{B} \langle A : m \rangle$ and $B(A : x)$

Alice

New S

Bob

New S'

Timeout

B $\ll -$:A,s,START $>$>

S $\ll$ B:S’,CHALL $>$>

S’ $\ll$ A:m,RESP $>$>

S $\ll$ B:S’,ABORT $>$>