Communication security: Formal models and proofs

Hubert Comon

September 14, 2016
The context (I)

- credit cards
- contactless cards
- telephones
- online transactions
- cars, fridges,... Internet of Things
- Big Brother: NSA
- Biomedical applications
- ...

Introduction to protocol security
The context (II)

“It's not who votes that counts. It's who counts the votes.”

—apocryphally attributed to Josef Vissarionovich Stalin, Soviet revolutionary, political leader, party animal and all-around scary character.
The context (III)

- Security protocols
- Testing is not very useful
- Hiding the code is not a good idea
- The scope of formal methods
A simple handshake protocol

\[ A \rightarrow B : \ n, r \cdot \text{aenc}(\langle A, n \rangle, \text{pk}(\text{sk}_B), r) \]
\[ B \rightarrow A : \ r'. \text{aenc}(n, \text{pk}(\text{sk}_A), r') \]
The formal verification problem

\[ \forall A. \quad A \parallel P \models \phi \]
The formal verification problem

\[ \forall A. \ A \parallel P \models \phi \]

\[ \forall A. \ A \parallel P_1 \sim A \parallel P_2 \]
The formal verification problem

\[ \forall A. \ A \parallel P \models \phi \]

\[ \forall A. \ A \parallel P_1 \sim A \parallel P_2 \]

Universal quantification on \( A \): we cannot apply directly model-checking techniques.
The formal verification problem

\[ \forall A. \quad A \parallel P \models \phi \]

\[ \forall A. \quad A \parallel P_1 \sim A \parallel P_2 \]

Universal quantification on \( A \): we cannot apply directly model-checking techniques.

One important issue: range of \( A \) ?
Attacker models

The DY-attacker
Messages are terms, the attacker is defined through an equation theory or an inference system

The computational attacker
Messages are bitstrings, the attacker is a probabilistic polynomial time Turing machine

Other attackers
Attacker models

The DY-attacker
Messages are terms, the attacker is defined through an equation theory or an inference system
Attacker models

The DY-attacker
Messages are terms, the attacker is defined through an equation theory or an inference system

The computational attacker
Messages are bitstrings, the attacker is a probabilistic polynomial time Turing machine
Attacker models

The DY-attacker
Messages are terms, the attacker is defined through an equation theory or an inference system

The computational attacker
Messages are bitstrings, the attacker is a probabilistic polynomial time Turing machine

Other attackers
Goals of the lecture

Verification inputs

- Cryptographic libraries
- Protocol programs
- Attacker model
- Security property
Goals of the lecture

Verification inputs
- Cryptographic libraries
- Protocol programs
- Attacker model
- Security property

Goals of the lecture
Show how to derive the proof obligations in a parametric way, abstracting from crypto libraries, attacker models.
Focus on the semantics of protocols, for arbitrary libraries and attacker models.
Roadmap

4 successive versions of the calculus, by increasing expressiveness (we could have considered the last case only...)

1. Simple case
2. Adding events: required for agreement properties
3. Adding replication
4. Adding channel generation: required for computational semantics

Then indistinguishability properties (privacy).
Cryptographic libraries

Syntax

- An arbitrary set of cryptographic primitives $\mathcal{F}$: hash, public-key encryption(s), symmetric encryption(s), zkp,... represented by (typed) function symbols
- At least one random generation algorithm. Random numbers are represented by names $n, n_1, r, ...$ out of a set $\mathcal{N}$

Terms are built over variables, function symbols and names.
\( \mathcal{M} \) is an interpretation domain. Typically ground or constructor terms (the DY semantics) or bitstrings (the computational semantics).

\( \mathcal{M} \) includes error messages (exceptions) \( \text{Err} \).

If \( \sigma \) is an environment (mapping from variables to \( \mathcal{M} \)), \( u \) is a term,

\[ [u]_\sigma^{\mathcal{M}} \]

is the interpretation of \( u \) in \( \mathcal{M} \) w.r.t. \( \sigma \): \( \mathcal{M} \) is a (partial) \( \mathcal{F} \)-algebra.

The interpretation is strict:

\[ u_i \in \text{Err} \quad \Rightarrow \quad [f(u_1, \ldots, u_n)]_\sigma^{\mathcal{M}} \in \text{Err} \]
Cryptographic libraries
A possible set of function symbols

- $\text{aenc}(u, pk, r)$ is (supposed to be) the asymmetric encryption of $u$ with the public key $pk$ and random input $r$.
- $\text{dec}(u, sk)$ is (supposed to be) the decryption of $u$ with the secret key $sk$
- $\text{pk}(sk)$ is (supposed to be) the public key associated with the secret key $sk$
- $\langle u, v \rangle$
- $\pi_1(u), \pi_2(u)$
Cryptographic libraries
A DY model

\( \mathcal{M}_{DY} \) (messages) is the least set of ground terms such that:

- \( \mathcal{N} \subseteq \mathcal{M}_{DY} \)
- if \( u, v \in \mathcal{M}_{DY} \) then \( \langle u, v \rangle \in \mathcal{M}_{DY} \)
- if \( k \in \mathcal{N} \) then \( \text{pk}(k) \in \mathcal{M}_{DY} \)
- if \( u \in \mathcal{M}_{DY}, k, r \in \mathcal{N} \), then \( \text{aenc}(u, \text{pk}(k), r) \in \mathcal{M}_{DY} \).

\( \mathcal{M}_{DY} \) also includes special error terms \( \text{Err} \) (not messages).

\[
\text{dec}(\text{aenc}(u, \text{pk}(k), r), k) \rightarrow u \quad \text{For } k, r \in \mathcal{N}, u \text{ a message}
\]

\[
\pi_1(\langle u, v \rangle) \rightarrow u \quad u, v \text{ are messages}
\]

\[
\pi_2(\langle u, v \rangle) \rightarrow v \quad u, v \text{ are messages}
\]

\[
[ u ]_{\mathcal{M}_{DY}}^{\mathcal{M}_{DY}} = u \sigma \downarrow
\]

Any irreducible ground term, which is not a message, is an error.
Cryptographic libraries
Computational models

- $\eta \in \mathbb{N}$ is a security parameter
- $\tau$ maps $\mathcal{N}$ to $\{0, 1\}^\eta$
- $\mathcal{M}_c(\tau, \eta) \subseteq \{0, 1\}^*$
- $[n]^{\mathcal{M}_c(\tau, \eta)} = \tau(n)$
- $\text{aenc}(\_ , \_ , \_ ), \text{dec}(\_ , \_ ), \text{pk}(\_ )$ are interpreted as a public-key encryption scheme.
- with an interpretation of pairing/projections, $\mathcal{M}_c(\tau, \eta)$ is an $\mathcal{F}$-algebra
A simple process calculus

Syntax

\[ P ::= \]
\[ 0 \quad \text{null process (stalled)} \]
\[ \text{in}(x).P \quad \text{input of } x \text{ (binds } x) \]
\[ \text{out}(t).P \quad \text{output of } t \]
\[ \text{if } \text{EQ}(u, v) \text{ then } P \text{ else } P \quad \text{conditional branching} \]
\[ \text{let } y = u \text{ in } P \quad \text{evaluation (binds } y) \]
\[ \nu n.P \quad \text{random generation} \]
\[ P \parallel P \quad \text{parallel composition} \]

All variable occurrences are bound.
Example

The simple handshake protocol

\[ A \rightarrow B : \nu n, r. \text{aenc}(\langle A, n \rangle, \text{pk}(\text{sk}_B), r) \]
\[ B \rightarrow A : \nu r'. \text{aenc}(n, \text{pk}(\text{sk}_A), r') \]

\[ A(\text{sk}_a, \text{pk}(\text{sk}_b)) = \]

\[ B(\text{sk}_b) = \]
Example

The simple handshake protocol

\[ A \rightarrow B : \ nu \ n, r. \ aenc(\langle A, n \rangle, pk(sk_B), r) \]
\[ B \rightarrow A : \ nu \ r'. \ aenc(\ n, pk(sk_A), r') \]

\[ A(sk_a, pk(sk_b)) = \]
\[ nu \ n, r. \ out(aenc(\langle pk(sk_a), n \rangle, pk(sk_b), r)) \]

\[ B(sk_b) = \]
Example

The simple handshake protocol

\[ A \rightarrow B : \nu n, r. \text{aenc}(\langle A, n \rangle, \text{pk}(\text{sk}_B), r) \]
\[ B \rightarrow A : \nu r'. \text{aenc}(n, \text{pk}(\text{sk}_A), r') \]

\[ A(\text{sk}_a, \text{pk}(\text{sk}_b)) = \]
\[ \nu n, r. \text{out}(\text{aenc}(\langle \text{pk}(\text{sk}_a), n \rangle, \text{pk}(\text{sk}_b), r)). \]

\[ B(\text{sk}_b) = \]
\[ \nu r'. \text{in}(x). \text{let } y = \text{dec}(x, \text{sk}_b) \text{ in} \]
Example

The simple handshake protocol

\[ A \to B : \nu n, r. \; \text{aenc}(\langle A, n \rangle, \text{pk}(\text{sk}_B), r) \]
\[ B \to A : \nu r'. \; \text{aenc}(n, \text{pk}(\text{sk}_A), r') \]

\[ A(\text{sk}_a, \text{pk}(\text{sk}_b)) = \]
\[ \nu n, r. \; \text{out}(\text{aenc}(\langle \text{pk}(\text{sk}_a), n \rangle, \text{pk}(\text{sk}_b), r)). \]

\[ B(\text{sk}_b) = \]
\[ \nu r'. \; \text{in}(x).\; \text{let} \; y = \text{dec}(x, \text{sk}_b) \; \text{in} \]
\[ \text{let} \; y_1 = \pi_1(y) \; \text{in} \; \text{let} \; y_2 = \pi_2(y) \; \text{in} \]
\[ \text{out}(\text{aenc}(y_2, y_1, r')). \; 0. \]
Example

The simple handshake protocol

\[ A \rightarrow B : \ \nu n, r. \ aenc((A, n), pk(sk_B), r) \]
\[ B \rightarrow A : \ \nu r'. aenc(n, pk(sk_A), r') \]

\[ A(sk_a, pk(sk_b)) = \]
\[ \nu n, r. \ out(aenc((pk(sk_a), n), pk(sk_b), r)). \]
\[ in(z). \ let \ z_1 = \text{dec}(z, sk_a) \ in \]

\[ B(sk_b) = \]
\[ \nu r'. \ in(x). \ let \ y = \text{dec}(x, sk_b) \ in \]
\[ \text{let } y_1 = \pi_1(y) \text{ in } \text{let } y_2 = \pi_2(y) \text{ in } \]
\[ \text{out(aenc}(y_2, y_1, r')) \cdot 0. \]
Example

The simple handshake protocol

\[ A \rightarrow B : \nu n, r. \text{aenc}(\langle A, n \rangle, \text{pk}(\text{sk}_B), r) \]

\[ B \rightarrow A : \nu r'. \text{aenc}(n, \text{pk}(\text{sk}_A), r') \]

\[ A(\text{sk}_a, \text{pk}(\text{sk}_b)) = \]

\[ \nu n, r. \text{out}(\text{aenc}(\langle \text{pk}(\text{sk}_a), n \rangle, \text{pk}(\text{sk}_b), r)). \]

\[ \text{in}(z). \text{let } z_1 = \text{dec}(z, \text{sk}_a) \text{ in} \]

\[ \text{if } \text{EQ}(z_1, n) \text{ then } 0(\text{Success}) \text{ else } 0(\text{Fail}) \]

\[ B(\text{sk}_b) = \]

\[ \nu r'. \text{in}(x). \text{let } y = \text{dec}(x, \text{sk}_b) \text{ in} \]

\[ \text{let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in} \]

\[ \text{out}(\text{aenc}(y_2, y_1, r')). 0. \]
Example

The simple handshake protocol

\[
\begin{align*}
A & \rightarrow B : \nu n, r. \text{aenc}(\langle A, n \rangle, \text{pk}(sk_B), r) \\
B & \rightarrow A : \nu r'. \text{aenc}(n, \text{pk}(sk_A), r')
\end{align*}
\]

\[
A(sk_a, \text{pk}(sk_b)) = \\
\nu n, r. \text{out}(\text{aenc}(\langle \text{pk}(sk_a), n \rangle, \text{pk}(sk_b), r)). \\
in(z). \text{let } z_1 = \text{dec}(z, sk_a) \text{ in} \\
\text{if } \text{EQ}(z_1, n) \text{ then } 0(\text{Success}) \text{ else } 0(\text{Fail})
\]

\[
B(sk_b) = \\
\nu r'. \text{in}(x). \text{let } y = \text{dec}(x, sk_b) \text{ in} \\
\text{let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in} \\
\text{out}(\text{aenc}(y_2, y_1, r')). 0.
\]

\[
\nu sk_a, sk_b. \text{ out}(\langle \text{pk}(sk_a), \text{pk}(sk_b) \rangle). \ (A(sk_a, \text{pk}(sk_b)) \parallel B(sk_b))
\]

A simple version of the process calculus
Structural equivalence

\[
\begin{align*}
0 \parallel P & \equiv P \\
P \parallel Q & \equiv Q \parallel P \\
P \parallel (Q \parallel R) & \equiv (P \parallel Q) \parallel R \\
\nu n.P & \equiv \nu n'.P\{n \mapsto n'\} \\
\text{in}(x).P & \equiv \text{in}(x').P\{x \mapsto x'\} \\
\text{let } x = u \text{ in } P & \equiv \text{let } x' = u \text{ in } P\{x \mapsto x'\} \\
(\nu n.P) \parallel Q & \equiv \nu n'.(P \parallel Q) & \text{if } n \notin \text{freenames}(Q)
\end{align*}
\]
Operational semantics

States of the network are tuples \((\phi, \sigma, P)\), where

- \(\phi\) is a frame of the form \(\nu \bar{n}. m_1, \ldots, m_k\), where \(\bar{n}\) is a set of names (used so far) and \(m_1, \ldots, m_k\) is a sequence of values in \(\mathcal{M}\) (that have been sent out so far)

- \(\sigma\) is an environment: an assignment of the free variables to values in \(\mathcal{M}\)

- \(P\) is a process

The semantics is a labeled transition system, whose labels are the inputs provided by the attacker (sometimes, an empty input)
Operational semantics

The transition system (I)

\[(\phi, \sigma, \text{in}(x).P) \xrightarrow{u} (\phi, \sigma \uplus \{x \mapsto u\}, P)\]
Operational semantics

The transition system (I)

\[
(\phi, \sigma, \text{in}(x). P) \xrightarrow{u} (\phi, \sigma \uplus \{ x \mapsto u \}, P)
\]

\[
(\phi, \sigma, P) \xrightarrow{u} (\phi', \sigma', P')
\]

\[
(\phi, \sigma, \text{if EQ}(s, t) \text{ then } P \text{ else } Q) \xrightarrow{u} (\phi', \sigma', P')
\]

if \([s]_\sigma = [t]_\sigma \notin \text{Err}
\]
Operational semantics
The transition system (I)

\[(\phi, \sigma, \text{in}(x).P) \xrightarrow{u} (\phi, \sigma \cup \{x \mapsto u\}, P)\]

\[(\phi, \sigma, P) \xrightarrow{u} (\phi', \sigma', P')\]

\[(\phi, \sigma, \text{if EQ}(s, t) \text{ then } P \text{ else } Q) \xrightarrow{u} (\phi', \sigma', P')\]

\[\text{if}\llbracket s \rrbracket_{\sigma}^{M} = \llbracket t \rrbracket_{\sigma}^{M} \notin \text{Err}\]

\[(\phi, \sigma, Q) \xrightarrow{u} (\phi', \sigma', P')\]

\[(\phi, \sigma, \text{if EQ}(s, t) \text{ then } P \text{ else } Q) \xrightarrow{u} (\phi', \sigma', P')\]

\[\text{if}\llbracket s \rrbracket_{\sigma}^{M} \neq \llbracket t \rrbracket_{\sigma}^{M} \text{ or } \llbracket s \rrbracket_{\sigma}^{M} \in \text{Err} \text{ or } \llbracket t \rrbracket_{\sigma}^{M} \in \text{Err}\]
Operational semantics
The transition system (II)

\[(\phi, \sigma, \text{let } x = u \text{ in } P) \rightarrow (\phi, \sigma \sqcup \{x \mapsto w\}, P)\] if \([u]_\sigma^M = w \notin \text{Err} \]
Operational semantics

The transition system (II)

\[
(\phi, \sigma, \text{let } x = u \text{ in } P) \rightarrow (\phi, \sigma \uplus \{x \mapsto w\}, P)
\]

if \([u]_\sigma^M = w \notin \text{Err}

\[
(\nu \bar{n}.\theta, \sigma, \text{out}(s).P) \rightarrow (\nu \bar{n}.\theta \cdot [s]_\sigma^M, \sigma, P)
\]

A simple version of the process calculus
Operational semantics
The transition system (II)

\[
\text{if } \mathbb{u}\mathcal{M}_\sigma = w \notin \text{Err}
\]

\[
(\phi, \sigma, \text{let } x = u \text{ in } P) \rightarrow (\phi, \sigma \uplus \{x \mapsto w\}, P)
\]

\[
(\nu \bar{n}.\theta, \sigma, \text{out}(s).P) \rightarrow (\nu \bar{n}.\theta \cdot \llbracket s \rrbracket_\sigma^\mathcal{M}, \sigma, P)
\]

\[
(\phi, \sigma, P) \xrightarrow{u} (\phi', \sigma', P')
\]

\[
(\phi, \sigma, P \parallel Q) \xrightarrow{u} (\phi', \sigma', P' \parallel Q)
\]
Operational semantics

The transition system (II)

\[(\phi, \sigma, \text{let } x = u \text{ in } P) \rightarrow (\phi, \sigma \cup \{x \mapsto w\}, P)\]

\[\text{if } [u]_\sigma^\mathcal{M} = w \notin \text{Err}\]

\[(\nu \bar{n}.\theta, \sigma, \text{out}(s).P) \rightarrow (\nu \bar{n}.\theta \cdot [s]_\sigma^\mathcal{M}, \sigma, P)\]

\[(\phi, \sigma, P) \xrightarrow{u} (\phi', \sigma', P')\]

\[(\phi, \sigma, P \parallel Q) \xrightarrow{u} (\phi', \sigma', P' \parallel Q)\]

\[\text{if } n \notin \bar{n} \cup \text{freename}(\theta)\]

\[(\nu \bar{n}.\theta, \sigma, \nu n.P) \rightarrow \nu \bar{n} \cup n.\theta, \sigma, P)\]

A simple version of the process calculus
Example

On the black board.
Restricting the feasible transitions

\[ (\phi_1, \sigma_1, P_1) \xrightarrow{u_1} \cdots \xrightarrow{u_{k-1}} (\phi_k, \sigma_k, P_k) \]

is possible w.r.t. model \( \mathcal{M} \) and an attacker \( \mathcal{A} \) if, for every \( i \),

\[ \mathcal{A}(\llbracket \phi_i \rrbracket_{\sigma_i}, P_i) = \llbracket u_i \rrbracket_{\sigma_i} \]

Note: could include a state in \( \mathcal{A} \).
Example DY

There is a DY attacker \( \mathcal{A} \) such that \( \mathcal{A}(\phi) = [u]_{\sigma}^{M_{DY}} \) iff

\[
\phi \vdash_I u \sigma \downarrow
\]

where \( I \) is defined by:

\[
\phi \vdash u_1 \cdots \phi \vdash u_n \\
\phi \vdash f(u_1, \ldots, u_n) \downarrow
\]

For every \( f \in \mathcal{F} \)

\[
\nu \bar{n}. u_1, \ldots, u_n \vdash u_i
\]

\[
\nu \bar{n}. \theta \vdash n'
\]

if \( n' \in N \setminus \bar{n} \).
Exercise

In the simple handshake example, describe all feasible transition sequences in the DY model (assume the name extrusion, let, conditionals and outputs are always performed before inputs). Is the nonce $n$ secret?
A is a Probabilistic Polynomial Time Turing machine (PPT). Some inputs that were not possible in the DY model might now be possible.

A typical example

A might be able to compute (with a significant probability) $\mathcal{M}_c(\tau, \eta)$ from $\mathcal{M}_c(\tau, \eta)$
Example computational

$A$ is a Probabilistic Polynomial Time Turing machine (PPT).
Some inputs that were not possible in the DY model might now be possible.

A typical example

$A$ might be able to compute (with a significant probability)
$\llbracket \text{aenc}(u, \text{pk}(k_1), r_1) \rrbracket^{M_c(\tau, \eta)}$ from $\llbracket \text{aenc}(v, \text{pk}(k_1), r_1) \rrbracket^{M_c(\tau, \eta)}$

$\exists A, \quad \text{Prob}\{\tau, \rho : A(\llbracket \text{aenc}(v, \text{pk}(k_1), r_1) \rrbracket^{M_c(\tau, \eta)}) = \llbracket \text{aenc}(u, \text{pk}(k_1), r_1) \rrbracket^{M_c(\tau, \eta)} \} > \epsilon(\eta)$

$\epsilon$ is non-negligible: there is a polynomial $\text{Pol}$ such that

$$\lim inf_{\eta \to +\infty} \epsilon(\eta) \times \text{Pol}(\eta) > 1$$
Confidentiality

In the DY case
Is there a DY attacker $\mathcal{A}$ and a feasible transition sequence

$$(\emptyset, \emptyset, P) \xrightarrow{\ast} (\phi, \sigma, Q)$$

such that $\mathcal{A}(\phi, Q) = s$?
Confidentiality

In the DY case
Is there a DY attacker \( A \) and a feasible transition sequence

\[
(\emptyset, \emptyset, P) \xrightarrow{\star} (\phi, \sigma, Q)
\]

such that \( A(\phi, Q) = s \) ? This problem is in NP
Confidentiality

In the DY case
Is there a DY attacker $\mathcal{A}$ and a feasible transition sequence

$$((\emptyset, \emptyset, P) \xrightarrow{*} (\phi, \sigma, Q))$$

such that $\mathcal{A}(\phi, Q) = s$? **This problem is in NP**

In the computational case
Is there a PPT $\mathcal{A}$ such that, for every computational model $\mathcal{M}_c(\tau, \eta)$, the probability that there is a feasible sequence

$$((\emptyset, \emptyset, P) \xrightarrow{*} (\phi, \sigma, Q))$$

such that $\mathcal{A}(\phi, Q) = s$ is negligible in $\eta$? **This requires in general assumptions on the libraries**
Exercises

In the following cases, give reasonable processes $A, B$ and either give an attack on the confidentiality of $s$ or prove that there is no such attack in the DY model.

1. 
   
   $A \rightarrow B : \nu s, \nu r. \langle \text{pk}(sk_A), \text{aenc}(s, \text{pk}(sk_B), r) \rangle$
   
   $B \rightarrow A : \nu r'. \langle \text{pk}(sk_B), \text{aenc}(s, \text{pk}(sk_A), r') \rangle$

   \[
P = \nu sk_a, \nu sk_b. \text{out}(\langle \text{pk}(sk_A), \text{pk}(sk_B) \rangle) \cdot (A(sk_a, \text{pk}(sk_B)) \parallel B(sk_b))
   \]

2. 
   
   $A \rightarrow B : \nu s, r_1, r_2. \text{aenc}(\langle \text{pk}(sk_A), \text{aenc}(s, \text{pk}(sk_B), r_1) \rangle, \text{pk}(sk_B), r_2)$
   
   $B \rightarrow A : \nu r_3, r_4. \text{aenc}(\langle \text{pk}(sk_B), \text{aenc}(s, \text{pk}(sk_A), r_3) \rangle, \text{pk}(sk_A), r_4)$

   \[
P = \nu sk_a, \nu sk_b. \text{out}(\langle \text{pk}(sk_A), \text{pk}(sk_B) \rangle) \cdot (A(sk_a, \text{pk}(sk_B)) \parallel B(sk_b) \parallel B(sk_b))
   \]
Gathering feasibility conditions

States of the network are tuples \((\phi, \sigma, P, \theta)\), where

- \(\phi, \sigma, P\) as before
- \(\theta\) is a constraint: equalities, disequalities and computational constraints of the form \(\phi \triangleright u\).
Gathering feasibility conditions

States of the network are tuples \((\phi, \sigma, P, \theta)\), where

- \(\phi, \sigma, P\) as before
- \(\theta\) is a constraint: equalities, disequalities and computational constraints of the form \(\phi \triangleright u\).

\[
(\phi, \sigma, \text{in}(x).P, \theta) \rightarrow (\phi, \sigma, P, \theta \land \phi \triangleright x)
\]
Gathering feasibility conditions

States of the network are tuples \((\phi, \sigma, P, \theta)\), where

- \(\phi, \sigma, P\) as before
- \(\theta\) is a constraint: equalities, disequalities and computational constraints of the form \(\phi \triangleright u\).

\[
(\phi, \sigma, \text{in}(x).P, \theta) \rightarrow (\phi, \sigma, P, \theta \land \phi \triangleright x)
\]

\[
(\phi, \sigma, \text{if EQ}(s, t) \text{ then } P \text{ else } Q, \theta) \rightarrow (\phi, \sigma, P, \theta \land \text{EQ}(s, t))
\]
Gathering feasibility conditions

States of the network are tuples \((\phi, \sigma, P, \theta)\), where

- \(\phi, \sigma, P\) as before
- \(\theta\) is a constraint: equalities, disequalities and computational constraints of the form \(\phi \triangleright u\).

\[
(\phi, \sigma, \text{in}(x).P, \theta) \rightarrow (\phi, \sigma, P, \theta \land \phi \triangleright x)
\]

\[
(\phi, \sigma, \text{if EQ}(s, t) \text{ then } P \text{ else } Q, \theta) \rightarrow (\phi, \sigma, P, \theta \land \text{EQ}(s, t))
\]

\[
(\phi, \sigma, \text{if EQ}(s, t) \text{ then } P \text{ else } Q, \theta) \rightarrow (\phi, \sigma, P, \theta \land \neg\text{EQ}(s, t))
\]
Consequences

Advantages

- A finite transition system (regardless of the model)
- Confidentiality reduces to constraint satisfaction

\[ \theta \land \phi_f \triangleright s \]

in NP in the DY model
Consequences

Computational case

Specify the assumptions on the libraries: impossibility conditions.

\( S, \text{aenc}(n, \text{pk}(k), r) \triangleright n \Rightarrow S \triangleright n \)

\( S_1 \triangleright x \land S_2, x \triangleright y \Rightarrow S_1, S_2 \triangleright y \)

\( S_1 \triangleright x_1 \land \ldots \land S_n \triangleright x_n \Rightarrow S_1, \ldots, S_n \triangleright f(x_1, \ldots, x_n) \)

\( S, S_1, S_2 \) are finite sets of terms.
Consequences

Computational case

Specify the assumptions on the libraries: impossibility conditions.

\[ S, \text{aenc}(n, \text{pk}(k), r) \triangleright n \implies S \triangleright n \]

\[ S_1 \triangleright x \land S_2, x \triangleright y \implies S_1, S_2 \triangleright y \]

\[ S_1 \triangleright x_1 \land \ldots \land S_n \triangleright x_n \implies S_1, \ldots, S_n \triangleright f(x_1, \ldots, x_n) \]

\[ S, S_1, S_2 \text{ are finite sets of terms.} \]

Check the constraint satisfiability, together with \( \phi \triangleright s \) and the above axioms (in PTIME !!)
Back to the simple handshake protocol. Study its security in the computational model, assuming the properties of the cryptographic libraries that are described in the lecture.
Exercise

solution sketch

\[ A(\text{sk}_a, \text{pk}(\text{sk}_b)) = \]
\[ \nu n, r. \text{out}(\text{aenc}(\langle \text{pk}(\text{sk}_a), n \rangle, \text{pk}(\text{sk}_b), r)). \]
\[ \text{in}(z). \text{let } z_1 = \text{dec}(z, \text{sk}_a) \text{ in} \]
\[ \text{if } \text{EQ}(z_1, n) \text{ then } 0(\text{Success}) \text{ else } 0(\text{Fail}) \]

\[ B(\text{sk}_b) = \]
\[ \nu r'. \text{in}(x). \text{let } y = \text{dec}(x, \text{sk}_b) \text{ in} \]
\[ \text{let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in} \]
\[ \text{out}(\text{aenc}(y_2, y_1, r')). 0. \]

\[ P = \nu \text{sk}_a, \text{sk}_b. \text{out}(\langle \text{pk}(\text{sk}_a), \text{pk}(\text{sk}_b) \rangle). (A(\text{sk}_a, \text{pk}(\text{sk}_b)) \parallel B(\text{sk}_b)) \]

\[ (\emptyset, \emptyset, P, \top) \xrightarrow{\cdot} \]
\[ (\nu \text{sk}_a, \text{sk}_b, n, r, r'. \langle \text{pk}(\text{sk}_a), \text{pk}(\text{sk}_b) \rangle, \text{aenc}(\langle \text{pk}(\text{sk}_a), n \rangle, \text{pk}(\text{sk}_b), r), \emptyset, Q, \top) \]
\[ \phi_1 \]

\[ Q = \text{in}(z). \text{let } z_1 = \text{dec}(z, \text{sk}_a) \text{ in} \text{ if } \text{EQ}(z_1, n) \text{ then } 0 \text{ else } 0 \]
\[ \parallel \text{in}(x). \text{let } y = \text{dec}(x, \text{sk}_b) \text{ in let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in out}(\text{aenc}(y_2, y_1, r)) \]

Symbolic (Abstract) semantics
Exercise
solution sketch

Two complete traces

\[
(\phi_1, \emptyset, Q, \top) \rightarrow^* (\phi_2, \emptyset, 0, \phi_1 \triangleright z \land \phi_1 \triangleright x \land \theta)
\]

\[
(\phi_1, \emptyset, Q, \top) \rightarrow^* (\phi_2, \emptyset, 0, \phi_2 \triangleright z \land \phi_1 \triangleright x \land \theta)
\]

where

\[
\phi_2 = \phi_1, \text{aenc}(y_2, y_1, r')
\]

\[
\theta = \text{EQ}(y, \text{dec}(x, \text{sk}_b)) \land \text{EQ}(y_1, \pi_1(y)) \land \text{EQ}(y_2, \pi_2(y)) \land \text{EQ}(z_1, \text{dec}(z, \text{sk}_a))
\]
Exercise
solution sketch

Two complete traces

\[(\phi_1, \emptyset, Q, \top) \rightarrow^* (\phi_2, \emptyset, 0, \phi_1 \triangleright z \land \phi_1 \triangleright x \land \theta)\]

\[(\phi_1, \emptyset, Q, \top) \rightarrow^* (\phi_2, \emptyset, 0, \phi_2 \triangleright z \land \phi_1 \triangleright x \land \theta)\]

where

\[\phi_2 = \phi_1, \text{aenc}(y_2, y_1, r')\]

\[\theta = \text{EQ}(y, \text{dec}(x, \text{sk}_b)) \land \text{EQ}(y_1, \pi_1(y)) \land \text{EQ}(y_2, \pi_2(y)) \land \text{EQ}(z_1, \text{dec}(z, \text{sk}_a))\]

Solve the constraint

\[\phi_2 \triangleright z \land \phi_1 \triangleright x \land \theta \land \phi_2 \triangleright n\]
Exercise
solution sketch

Two complete traces

\[(\phi_1, \emptyset, Q, \top) \xrightarrow{*} (\phi_2, \emptyset, 0, \phi_1 \triangleright z \land \phi_1 \triangleright x \land \theta)\]

\[(\phi_1, \emptyset, Q, \top) \xrightarrow{*} (\phi_2, \emptyset, 0, \phi_2 \triangleright z \land \phi_1 \triangleright x \land \theta)\]

where

\[\phi_2 = \phi_1, \text{aenc}(y_2, y_1, r')\]

\[\theta = \text{EQ}(y, \text{dec}(x, sk_b)) \land \text{EQ}(y_1, \pi_1(y)) \land \text{EQ}(y_2, \pi_2(y)) \land \text{EQ}(z_1, \text{dec}(z, sk_a))\]

Solve the constraint

\[\phi_2 \triangleright z \land \phi_1 \triangleright x \land \theta \land \phi_2 \triangleright n\]

Solve the constraint

\[\phi_2 \triangleright \text{aenc}(z_1, \text{pk}(sk_a), z_2) \land \phi_1 \triangleright \text{aenc}(\langle y_2, y_1 \rangle, \text{pk}(sk_b), y_3) \land \phi_2 \triangleright n\]
Solving the first constraint

\[ \phi_1 \triangleright aenc(\langle y_2, y_1 \rangle, pk(sk_b), y_3) \]

\[ \phi_1 = \nu sk_a, sk_b, n, r, r'. \langle pk(sk_a), pk(sk_b) \rangle, aenc(\langle pk(sk_a), n \rangle, pk(sk_b), r) \]

- Either \( y_2 = pk(sk_a), \ y_1 = n, \ y_3 = r \)
- Or \( \phi_1 \triangleright y_2, \ \phi_1 \triangleright y_1, \ \phi_1 \triangleright y_3 \)
**Adding events to the process calculus**

**syntax**

\[ P ::= \]

\[
\begin{align*}
0 & \quad \text{null process (stalled)} \\
\text{in}(x).P & \quad \text{input of } x \text{ (binds } x) \\
\text{out}(t).P & \quad \text{output of } t \\
\text{if } \text{EQ}(u, v) \text{ then } P \text{ else } P & \quad \text{conditional branching} \\
\text{let } y = u \text{ in } P & \quad \text{evaluation} \\
\nu \bar{u}.P & \quad \text{random generation} \\
P \parallel P & \quad \text{parallel composition} \\
Eve(\bar{u}) \cdot P & \quad \text{the event } Eve \text{ is raised} \\
\end{align*}
\]

\text{\textit{Eva, Evb, ...} is a set of event symbols.}

Agreement properties: instrumenting the processes
Adding events to the process calculus semantics

The states have now an event component $\mathcal{E}$, a sequence of event symbols together with values. $\mathcal{E}$ is only modified when an event occurs in the process:

$$(\phi, \sigma, \text{Eve}(\overline{u}) \cdot P, \mathcal{E}) \rightarrow (\phi, \sigma, P, \mathcal{E} \cdot \text{Eve}(\overline{u})^M)$$
Example

\[ A(\text{sk}_a, \text{pk}(\text{sk}_b)) = \]

\[ \nu n, r. \text{out}(aenc(\langle \text{pk}(\text{sk}_a), n \rangle, \text{pk}(\text{sk}_b), r)). \]

\[ \text{in}(z). \text{let } z_1 = \text{dec}(z, \text{sk}_a) \text{ in} \]

\[ \text{if EQ}(z_1, n) \text{ then } \text{Eva}(n, \text{pk}(\text{sk}_b), \text{pk}(\text{sk}_a)) \text{ else } 0 \]

\[ B(\text{sk}_b) = \]

\[ \nu r'. \text{in}(x). \text{let } y = \text{dec}(x, \text{sk}_b) \text{ in} \]

\[ \text{let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in} \]

\[ \text{out}(aenc(y_2, y_1, r')). \text{Ev}(y_2, \text{pk}(\text{sk}_b), y_1) 0. \]
Example

\[A(\text{sk}_a, \text{pk}(\text{sk}_b)) = \nu n, r. \ \text{out}(\text{aenc}(\langle \text{pk}(\text{sk}_a), n \rangle, \text{pk}(\text{sk}_b), r).\] 
\[\text{in}(z). \ \text{let } z_1 = \text{dec}(z, \text{sk}_a) \text{ in}
\[\text{if } \text{EQ}(z_1, n) \text{ then } \text{Eva}(n, \text{pk}(\text{sk}_b), \text{pk}(\text{sk}_a)) \text{ else } 0\]

\[B(\text{sk}_b) = \nu r'. \ \text{in}(x). \ \text{let } y = \text{dec}(x, \text{sk}_b) \text{ in}
\[\text{let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in}
\[\text{out}(\text{aenc}(y_2, y_1, r')).\text{Evb}(y_2, \text{pk}(\text{sk}_b), y_1) \ 0.\]

\[\forall x, y, z. \ \text{Eva}(x, y, z) \Rightarrow \text{Evb}(x, y, z)\]
Agreement property

$$\forall x, y, z. \ Eva(x, y, z) \Rightarrow Evb(x, y, z)$$

More generally

$$\forall \vec{x}. \ Eva_1(u_1), \ldots, Eva_k(u_k) \Rightarrow Eva(\vec{u})$$

For every feasible trace, ending with an event set $E$, for any assignment $\sigma$, if $Eva_1(\vec{u}_1\sigma), \ldots, Eva_k(\vec{u}_k\sigma) \in E$, then $Eva(\vec{u}\sigma) \in E$
Exercise

In the DY-mode, does the simple handshake protocol satisfy the agreement property?

What happens if

1. we move forward the *Eva*
2. *B* replies with $y_2$ (instead of the encrypted version)?
Adding replication in the calculus

\[ P ::= \]
\[ \begin{array}{ll}
0 & \text{null process} \\
\text{in}(x).P & \text{input of } x \ (\text{binds } x) \\
\text{out}(t).P & \text{output of } t \\
\text{if } \text{EQ}(u, v) \text{ then } P \text{ else } P & \text{conditional branching} \\
\text{let } y = u \text{ in } P & \text{evaluation} \\
\nu \vec{n}.P & \text{random generation} \\
P \parallel P & \text{parallel composition} \\
\text{Eve}(\vec{u}) \cdot P & \text{the event } \text{Eve} \text{ is raised} \\
!P & \text{with a sequence of values } \vec{u}
\end{array} \]
Adding replication in the calculus

\[ P ::= \]

\[ \begin{align*}
0 & \quad \text{null process} \\
\text{in}(x).P & \quad \text{input of } x \text{ (binds } x) \\
\text{out}(t).P & \quad \text{output of } t \\
\text{if } \text{EQ}(u, v) \text{ then } P \text{ else } P & \quad \text{conditional branching} \\
\text{let } y = u \text{ in } P & \quad \text{evaluation} \\
\nu n.\! P & \quad \text{random generation} \\
P \parallel P & \quad \text{parallel composition} \\
\text{Eve}(\overline{u}) \cdot P & \quad \text{the event } \text{Eve} \text{ is raised with a sequence of values } \overline{u} \\
!P &
\end{align*} \]

Operational semantics:

\[ !P \equiv P \parallel !P \]
Example

The handshake protocol

!(νsk_a, sk_b. out(⟨pk(sk_a), pk(sk_b)⟩)). (!A(sk_a, pk(sk_b)) || !B(sk_b))
Verification in the DY-model

- This is undecidable
- Most popular: approximation and translation into Horn clauses
Translation into Horn clauses

Attacker clauses

\[
\begin{align*}
\text{Att}(a) & \iff \\
\text{Att}(\langle x_1, x_2 \rangle) & \iff \text{Att}(x_1), \text{Att}(x_2) \\
\text{Att}(\text{pk}(x)) & \iff \text{Att}(x) \\
\text{Att}(\text{aenc}(x_1, x_2, x_3)) & \iff \text{Att}(x_1) \text{Att}(x_2), \text{Att}(x_3) \\
\text{Att}(x_1) & \iff \text{Att}(\text{aenc}(x_1, \text{pk}(x_2), x_3)), \text{Att}(x_2) \\
\text{Att}(x_1) & \iff \text{Att}(\langle x_1, x_2 \rangle) \\
\text{Att}(x_2) & \iff \text{Att}(\langle x_1, x_2 \rangle)
\end{align*}
\]

\(a\) is a free name.
Translation into Horn clauses

Protocol clauses

\[ T(P) = [P]_{\emptyset, \emptyset} \text{ where } [P]_{\rho, H} \text{ is defined as follows} \]

\[
\begin{align*}
[0] & = \emptyset \\
[P \parallel Q]_{\rho, H} & = [P]_{\rho, H} \cup [Q]_{\rho, H} \\
[!P]_{\rho, H} & = [P]_{\rho, H} \\
[\nu n. P]_{\rho, H} & = [P]_{\rho \cup \{n \mapsto n(H)\}, H} \\
[in(x).P]_{\rho, H} & = [P]_{\rho \cup \{x\}} \\
[out(s).P]_{\rho, H} & = [P]_{\rho, H} \cup \{\text{Att}(s\rho) \leftarrow \text{Att}(s_1\rho), \ldots\} \\
[\text{if EQ}(s, t) \text{ then } P \text{ else } Q]_{\rho, H} & = [P]_{\rho \sigma, H \sigma} \cup [Q]_{\rho, H} \\
[\text{let } x = s \text{ in } P]_{\rho, H} & = [P]_{\rho \sigma_{x,s}, H\sigma_{x,s}} \\
[\text{let } x = s \text{ in } P]_{\rho, H} & = \emptyset
\end{align*}
\]
Examples

\[ B = \nu r . \text{in}(x) \cdot \text{let } y = \text{dec}(x, sk_b) \text{ in let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in out(aenc(y_2, y_1, r))} \]
Examples

\[ B = \nu r \cdot \text{in}(x) \cdot \text{let } y = \text{dec}(x, \text{sk}_b) \text{ in let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in out(aenc(y_2, y_1, r))} \]

\[ x = \text{aenc}(\langle y_1, y_2 \rangle, \text{pk}(...)) : B = \nu r \cdot \text{in}(...). \text{out}(\text{aenc}(y_2, y_1, r)) \]
Examples

\[ B = \nu r . \text{in}(x) . \text{let } y = \text{dec}(x, \text{sk}_B) \text{ in let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in } \text{out}(\text{aenc}(y_2, y_1, r)) \]

\[ x = \text{aenc}(\langle y_1, y_2 \rangle, \text{pk}(\text{sk}_B), z) : \]

\[ B = \nu r . \text{in}(\text{aenc}(\langle y_1, y_2 \rangle, \text{pk}(\text{sk}_B), z)) . \text{out}(\text{aenc}(y_2, y_1, r)) \]

\[ \text{Att}(\text{aenc}(y_2, y_1, r(\cdots))) \iff \text{Att}(\text{aenc}(\langle y_1, y_2 \rangle, \text{pk}(\text{sk}_B), z))) \]
Examples

\[ B = \nu r . \text{in}(x) . \text{let } y = \text{dec}(x, \text{sk}_b) \text{ in let } y_1 = \pi_1(y) \text{ in let } y_2 = \pi_2(y) \text{ in out(aenc}(y_2, y_1, r)) \]

\[ x = \text{aenc}(\langle y_1, y_2 \rangle, \text{pk}(_B), z) : \]

\[ B = \nu r . \text{in(aenc}(\langle y_1, y_2 \rangle, \text{pk}(_B), z)) . \text{out(aenc}(y_2, y_1, r)) \]

\[ \text{Att(aenc}(y_2, y_1, r(\cdots))) \iff \text{Att(aenc}(\langle y_1, y_2 \rangle, \text{pk}(_B), z))) \]

\[ \text{Att(aenc(}\langle \text{pk}(_A), n \rangle, \text{pk}(_B), r')) \iff \]

Translation into Horn clauses
A tool available online, whose main developer is Bruno Blanchet.

- Translate the specification of primitives into rewriting system + Attacker clauses
- Translate the protocol into clauses
- Negate the security goal
- Check for satisfiability (Ordered resolution strategy).
  Three possible outputs:
  1. Unsatisfiable: the protocol is secure
  2. Satisfiable: there might be an attack (but might also be secure)
  3. Non termination

Very successful tool: works well in practice.
False attacks

Using twice the same protocol clause (rigid variables !)

\[ A \to B : \nu n_1, n_2, r_1, r_2. \langle \text{aenc}(n_1, \text{pk}(\text{sk}_B), r_1), \text{aenc}(n_2, \text{pk}(\text{sk}_B), r_2) \rangle \]
\[ B \to A : \nu n, r_3. \text{aenc}(n, \text{pk}(\text{sk}_A), r_3) \]
\[ A \to B : \nu s. \langle n, \text{senc}(s, \langle n_1, n_2 \rangle) \rangle \]
False attacks

Using twice the same protocol clause (rigid variables !)

\[ A \rightarrow B : \ \nu n_1, n_2, r_1, r_2. \ \langle \text{aenc}(n_1, \text{pk}(<sk_B>), r_1), \text{aenc}(n_2, \text{pk}(<sk_B>), r_2) \rangle \]
\[ B \rightarrow A : \ \nu n, r_3. \ \text{aenc}(n, \text{pk}(<sk_A>), r_3) \]
\[ A \rightarrow B : \ \nu s. \ \langle n, \text{senc}(s, <n_1, n_2>) \rangle \]

Approximating names

\[ A \rightarrow B : \ \nu n.\text{aenc}(n, \text{pk}(<sk_B>), r) \]
\[ B \rightarrow A : \ \nu n'.\text{aenc}(n', \text{pk}(<sk_A>), r') \]
\[ A \rightarrow B : \ \nu s.\text{in}(<\text{aenc}(x, \text{pk}(<sk_A>), y)>.\text{if } x = n \text{ then out}(s) \text{ else out}(n)) \]
The determinacy issue

Currently the calculus is non-deterministic, while the computational security is probabilistic. As security is an asymptotic property, it is harmless when the (symbolic) transition system is finite (for a fixed attacker).
The determinacy issue

Currently the calculus is non-deterministic, while the computational security is probabilistic. As security is an asymptotic property, it is harmless when the (symbolic) transition system is finite (for a fixed attacker).

Instead of non-determinism, we let the attacker choose the transition.
The determinacy issue

Currently the calculus is non-deterministic, while the computational security is probabilistic. As security is an asymptotic property, it is harmless when the (symbolic) transition system is finite (for a fixed attacker).

Instead of non-determinism, we let the attacker choose the transition.

Communications are using different communication channels, that solve the non-determinism and are chosen by the attacker.
Extending the calculus with channels

c, c_1, \ldots \text{ are channel names}

$$P ::= \begin{array}{l}
\quad 0 \quad \text{null process (stalled)} \\
\mid \text{in}(c, x).P \quad \text{input of } x \text{ (binds } x) \\
\mid \text{out}(c, t).P \quad \text{output of } t \\
\mid \text{if } \text{EQ}(u, v) \text{ then } P \text{ else } P \quad \text{conditional branching} \\
\mid \text{let } y = u \text{ in } P \quad \text{evaluation} \\
\mid \nu \bar{n}.P \quad \text{random generation} \\
\mid P \parallel P \quad \text{parallel composition} \\
\mid \text{Eve}(\bar{u}) \cdot P \quad \text{the event } \text{Eve} \text{ is raised with a sequence of values } \bar{u} \\
\mid !P \\
\mid \nu c.P
\end{array}$$
Operational semantics

Only known channels can be observed

\[(\phi, \sigma, \text{out}(c, u) \cdot P, E, \theta) \rightarrow (\phi \cdot \llbracket u \rrbracket^M_\sigma, \sigma, P, E, \theta \land \phi \triangleright c)\]
Operational semantics

Only known channels can be observed

\[(\phi, \sigma, \text{out}(c, u) \cdot P, \mathcal{E}, \theta) \rightarrow (\phi \cdot \llbracket u \rrbracket_{\sigma}^{\mathcal{M}}, \sigma, P, \mathcal{E}, \theta \land \phi \triangleright c)\]

Inputs are only possible on the designated channels

\[(\phi, \sigma, \text{in}(c, x) \cdot P, \mathcal{E}, \theta) \rightarrow (\phi, \sigma \uplus \{ x \mapsto \llbracket u \rrbracket_{\sigma}^{\mathcal{M}} \}, P, \mathcal{E}, \theta)\]

If \( A(\phi, \gamma) = \llbracket c \rrbracket^{\mathcal{M}} \) (\( \gamma \) is the global control state).
Internal synchronizations

\[(\phi, \sigma, \text{in}(c, x) \cdot P \parallel \text{out}(c, u) \cdot Q, E, \theta) \rightarrow (\phi, \sigma \uplus \{ x \mapsto [u]_\sigma^M \}, P \parallel Q, E, \theta)\]
Restrictions on the processes

We assume that, for any reachable process

\[
\text{in}(c_1, x_1).P_1 \parallel \cdots \parallel \text{in}(c_k, x_k).P_k \parallel Q
\]

\(c_1, \ldots, c_k\) are pairwise distinct.
Restrictions on the processes

We assume that, for any reachable process

\[ \text{in}(c_1, x_1).P_1 \parallel \cdots \parallel \text{in}(c_k, x_k).P_k \parallel Q \]

\(c_1, \ldots, c_k\) are pairwise distinct.

**Lemma**

Any process \(P\) without channel can be translated into a process \(P'\) such that \(P'\) satisfies the above restriction and any \(P\) has the same (DY) operational semantics as forgetting the channels in the semantics of \(P'\).
Example

\[ P = \text{in}(x).\text{out}(t) \]

\[ Q = \text{in}(y).\text{out}(u) \]

\[ \text{Trans}(P \parallel Q) = \text{in}(c_P, x).\text{out}(c_P, t) \parallel \text{in}(c_Q, y).\text{out}(c_Q, u) \]
Example

\[
P = \text{in}(x).\text{out}(t)
\]

\[
Q = \text{in}(y).\text{out}(u)
\]

\[
\text{Trans}(P \parallel Q) = \text{in}(c_P, x).\text{out}(c_P, t) \parallel \text{in}(c_Q, y).\text{out}(c_Q, u)
\]

\[
\text{Trans}(!(P \parallel Q)) = !\text{in}(c_{!(P \parallel Q)}, ?).\nu c_P, c_Q.\text{out}(c_{!(P \parallel Q)}, \langle c_P, c_Q \rangle).
\]

\[
\text{Trans}(P \parallel Q)
\]
Indistinguishability

\[ \text{Tr}(M, A, P) = m_1, \ldots, m_k, \ldots \]

the (unique) sequence of outputs of \( P \), with attacker \( A \) in model \( M \).
Indistinguishability

\[ \text{Tr}(\mathcal{M}, A, P) = m_1, \ldots, m_k, \ldots \]

the (unique) sequence of outputs of \( P \), with attacker \( A \) in model \( \mathcal{M} \).

Informally: \( P_1 \) is indistinguishable from \( P_2 \), for a (family of) model(s) \( \mathcal{M} \), if, for every \( A_1, A_2, A_3 \),

\[ A_2(\text{Tr}(\mathcal{M}, A_1, P_1)) = A_3(\text{Tr}(\mathcal{M}, A_1, P_1)) \]

iff

\[ A_2(\text{Tr}(\mathcal{M}, A_1, P_2)) = A_3(\text{Tr}(\mathcal{M}, A_1, P_2)) \]
Examples in the DY case (I)

\( A_1 \) gives (deducible) inputs to the process and \( A_2, A_3 \) observe some identities on the output.

**Example 1**

\[
P_1 = \nu_n.\text{out}(c,n) \\
P_2 = \nu_n'.\text{out}(c,n')
\]

\( P_1 \sim P_2 \)

\( A_1 \) can't input anything and \( A_2(n) = A_3(n) \) iff \( A_2 = A_3 \)

**Example 2**

\[
P_1 = \text{out}(c,\text{ok}) \\
P_2 = \nu_n.\text{out}(c,n)
\]

\( P_1 \not\sim P_2 \)

\( A_2 \) is the identity, \( A_3 \) computes \( \text{ok} \).
Examples in the DY case (I)

$A_1$ gives (deducible) inputs to the process and $A_2, A_3$ observe some identities on the output.

**Example 1**

$P_1 = \nu n.\text{out}(c, n)$ \hspace{1cm} $P_2 = \nu n'.\text{out}(c, n')$
Examples in the DY case (I)

$A_1$ gives (deducible) inputs to the process and $A_2, A_3$ observe some identities on the output.

**Example 1**

\[
P_1 = \nu n.\text{out}(c, n) \quad P_2 = \nu n'.\text{out}(c, n')
\]

$P_1 \sim P_2$

$A_1$ can’t input anything and $A_2(n) = A_3(n)$ iff $A_2 = A_3$
Examples in the DY case (I)

$A_1$ gives (deducible) inputs to the process and $A_2, A_3$ observe some identities on the output.

**Example 1**

\[ P_1 = \nu n.\text{out}(c, n) \quad P_2 = \nu n'.\text{out}(c, n') \]

$P_1 \sim P_2$

$A_1$ can’t input anything and $A_2(n) = A_3(n)$ iff $A_2 = A_3$

**Example 2**

\[ P_1 = \text{out}(c, \text{ok}) \quad P_2 = \nu n.\text{out}(c, n) \]
Examples in the DY case (I)

$A_1$ gives (deducible) inputs to the process and $A_2, A_3$ observe some identities on the output.

**Example 1**

$P_1 = \nu n.\text{out}(c, n) \quad P_2 = \nu n'.\text{out}(c, n')$

$P_1 \sim P_2$

$A_1$ can’t input anything and $A_2(n) = A_3(n)$ iff $A_2 = A_3$

**Example 2**

$P_1 = \text{out}(c, \text{ok}) \quad P_2 = \nu n.\text{out}(c, n)$

$P_1 \not\sim P_2$

$A_2$ is the identity, $A_3$ computes $\text{ok}$.
Examples in the DY case (II)

Example 3

\[
P_1 = \nu k \cdot \text{out}(pk(k)) \cdot \text{out}(c, \text{aenc}(n_1, pk(k), r))
\]

\[
P_2 = \nu k \cdot \text{out}(pk(k)) \cdot \text{out}(c, \text{aenc}(n_2, pk(k), r))
\]
Examples in the DY case (II)

Example 3

\[ P_1 = \nu k. \text{out}(pk(k)).\text{out}(c, \text{aenc}(n_1, pk(k), r)) \]

\[ P_2 = \nu k. \text{out}(pk(k)).\text{out}(c, \text{aenc}(n_2, pk(k), r)) \]

\[ P_1 \not\sim P_2 \]
Examples in the DY case (II)

Example 3

\[ P_1 = \nu k. \text{out}(pk(k)).\text{out}(c, \text{aenc}(n_1, pk(k), r)) \]
\[ P_2 = \nu k. \text{out}(pk(k)).\text{out}(c, \text{aenc}(n_2, pk(k), r)) \]

\[ P_1 \not\sim P_2 \]

Example 4

\[ P_1 = \text{in}(c, x) \cdot \text{out}(c, \langle x, n \rangle) \]
\[ P_2 = \text{in}(c, x) \cdot \text{out}(c, \langle n, x \rangle) \]
Examples in the DY case (II)

Example 3

\[ P_1 = \nu k. \text{out}(\text{pk}(k)).\text{out}(c, \text{aenc}(n_1, \text{pk}(k), r)) \]
\[ P_2 = \nu k. \text{out}(\text{pk}(k)).\text{out}(c, \text{aenc}(n_2, \text{pk}(k), r)) \]

\( P_1 \not\sim P_2 \)

Example 4

\[ P_1 = \text{in}(c, x) \cdot \text{out}(c, \langle x, n \rangle) \]
\[ P_2 = \text{in}(c, x) \cdot \text{out}(c, \langle n, x \rangle) \]

\( P_1 \not\sim P_2 \)
Results in the DY case

Indistinguishability is decidable in the DY case (for processes without replication). The complexity is unknown.

Verification tool: \texttt{APTE}.

An approximated equivalence is considered in \texttt{ProVerif} (works with replication)
Computational indistinguishability (I)

For any PPT $A_1, A_2, A_3$,

$$\text{Prob}\{\tau, \rho : A_2(\text{Tr}(M_c(\tau, \eta), A_1, P_1)) = A_3(\text{Tr}(M_c(\tau, \eta), A_1, P_1))\} - \text{Prob}\{\tau, \rho : A_2(\text{Tr}(M_c(\tau, \eta), A_1, P_2)) = A_3(\text{Tr}(M_c(\tau, \eta), A_1, P_2))\}$$

is negligible.

Example 1

$P_1 = \nu_{\text{out}}(c, n)$.

$P_2 = \nu_{\text{out}}(c, n')$.

$P_1 \sim P_2$.

The difference of probabilities is 0!!

Equivalence properties 58/65
Computational indistinguishability (I)

For any PPT $A_1, A_2, A_3$,

$$\text{Prob}\{\tau, \rho : A_2(\text{Tr}(\mathcal{M}_c(\tau, \eta), A_1, P_1)) = A_3(\text{Tr}(\mathcal{M}_c(\tau, \eta), A_1, P_1))\} - \text{Prob}\{\tau, \rho : A_2(\text{Tr}(\mathcal{M}_c(\tau, \eta), A_1, P_2)) = A_3(\text{Tr}(\mathcal{M}_c(\tau, \eta), A_1, P_2))\}$$

is negligible.

**Example 1**

$$P_1 = \nu n.\text{out}(c, n) \quad P_2 = \nu n'.\text{out}(c, n')$$
Computational indistinguishability (I)

For any PPT $A_1, A_2, A_3$,

$$\text{Prob}\{\tau, \rho : A_2(\text{Tr}(M_c(\tau, \eta), A_1, P_1)) = A_3(\text{Tr}(M_c(\tau, \eta), A_1, P_1))\} - \text{Prob}\{\tau, \rho : A_2(\text{Tr}(M_c(\tau, \eta), A_1, P_2)) = A_3(\text{Tr}(M_c(\tau, \eta), A_1, P_2))\}$$

is negligible.

**Example 1**

$P_1 = \nu n . \text{out}(c, n)$ \hspace{1cm} $P_2 = \nu n' . \text{out}(c, n')$

$P_1 \sim P_2$

The difference of probabilities is 0 !!
Computational indistinguishability (II)

Example 2

\[ P_1 = \nu k_1, r_1, r_2. \text{out}(c, \text{aenc}(ok, pk(k_1), r_1)).\text{out}(c, \text{aenc}(ok, pk(k_1), r_2)) \]

\[ P_2 = \nu k_1, k_2, r_1, r_2. \text{out}(c, \text{aenc}(ok, pk(k_1), r_1)).\text{out}(c, \text{aenc}(ok, pk(k_2), r_2)) \]
Computational indistinguishability (II)

Example 2

\[ P_1 = \nu k_1, r_1, r_2.\text{out}(c, \text{aenc}(\text{ok}, \text{pk}(k_1), r_1)).\text{out}(c, \text{aenc}(\text{ok}, \text{pk}(k_1), r_2)) \]

\[ P_2 = \nu k_1, k_2, r_1, r_2.\text{out}(c, \text{aenc}(\text{ok}, \text{pk}(k_1), r_1)).\text{out}(c, \text{aenc}(\text{ok}, \text{pk}(k_2), r_2)) \]

\[ P_1 \sim P_2 \] depends on assumptions on the crypto libraries
Computational indistinguishability (II)

Example 2

\[ P_1 = \nu k_1, r_1, r_2.\text{out}(c, \text{aenc(ok, pk(k_1), r_1)}).\text{out}(c, \text{aenc(ok, pk(k_1), r_2)}) \]

\[ P_2 = \nu k_1, k_2, r_1, r_2.\text{out}(c, \text{aenc(ok, pk(k_1), r_1)}).\text{out}(c, \text{aenc(ok, pk(k_2), r_2)}) \]

\[ P_1 \sim P_2 \] depends on assumptions on the crypto libraries

Example 3

\[ P_1 = \text{in}(c, x).\text{if } x = 0 \text{ then out}(c, 0) \text{ else out}(c, x) \]

\[ P_2 = \text{in}(c, x).\text{out}(c, x) \]
Computational indistinguishability (II)

Example 2

\[ P_1 = \nu k_1, r_1, r_2.\text{out}(c, \text{aenc}(\text{ok}, \text{pk}(k_1), r_1)).\text{out}(c, \text{aenc}(\text{ok}, \text{pk}(k_1), r_2)) \]

\[ P_2 = \nu k_1, k_2, r_1, r_2.\text{out}(c, \text{aenc}(\text{ok}, \text{pk}(k_1), r_1)).\text{out}(c, \text{aenc}(\text{ok}, \text{pk}(k_2), r_2)) \]

\( P_1 \sim P_2 \) depends on assumptions on the crypto libraries

Example 3

\[ P_1 = \text{in}(c, x) \text{.if } x = 0 \text{ then out}(c, 0) \text{ else out}(c, x) \]

\[ P_2 = \text{in}(c, x) \text{.out}(c, x) \]

\( P_1 \sim P_2 \)
Attacker’s inputs

Example 4

\[ P_1 = \nu n_1, n_2. \text{out}(\langle n_1, n_2 \rangle).\text{in}(x).\text{if } \pi_1(x) = n_1 \text{ then } \text{out}(n_2) \text{ else } 0 \]

\[ P_2 = P_1 \]

We cannot use the constraints \( \phi \triangleright x \) any more.
Example 4

\[ P_1 = \nu n_1, n_2.\, \text{out}(\langle n_1, n_2 \rangle).\text{in}(x).\, \text{if} \, \pi_1(x) = n_1 \, \text{then} \, \text{out}(n_2) \, \text{else} \, 0 \]

\[ P_2 = P_1 \]

We cannot use the constraints $\phi \triangleright x$ any more.

We use fresh function symbols $g_1, g_2 \cdots$ for successive attacker's inputs.

\[ P_1 = \nu n_1, n_2.\, \text{out}(\langle n_1, n_2 \rangle).\, \text{if} \, \pi_1(g_1(\langle n_1, n_2 \rangle)) = n_1 \, \text{then} \, \text{out}(n_2) \, \text{else} \, 0 \]
Attacker’s inputs

Examples

\[ P_1 = \text{out}(u).\text{in}(x).\text{out}(v). \]
\[ \quad \text{if } b \text{ then } \text{in}(y).Q_1 \text{ else } \text{in}(z).Q_2 \]

\[ P_2 = \text{out}(u').\text{in}(x').\text{out}(v'). \]
\[ \quad \text{in}(y').Q' \]

Equivalence properties
Attacker’s inputs

Examples

\[ P_1 = \text{out}(u)\cdot\text{in}(x)\cdot\text{out}(v)\]
\[ \quad \text{if } b \text{ then } \text{in}(y) \cdot Q_1 \text{ else } \text{in}(z) \cdot Q_2 \]
\[ P_2 = \text{out}(u')\cdot\text{in}(x')\cdot\text{out}(v')\]
\[ \quad \text{in}(y') \cdot Q' \]

\[ P_1 = \text{out}(u)\cdot\text{out}(v\{x \mapsto g_1(u)\}))\]
\[ \quad \text{if } b \text{ then } Q_1\{y \mapsto g_2(u, w)\} \text{ else } Q_2\{z \mapsto g_2(u, w)\} \]

where \( w = v\{x \mapsto g_1(u)\} \)
Attacker’s inputs

Examples

\[ P_1 = \text{out}(u)\cdot\text{in}(x)\cdot\text{out}(v) \]
\[ \quad \text{if } b \text{ then } \text{in}(y)\cdot Q_1 \text{ else } \text{in}(z)\cdot Q_2 \]
\[ P_2 = \text{out}(u')\cdot\text{in}(x')\cdot\text{out}(v') \]
\[ \quad \text{in}(y') \cdot Q' \]

\[ P_1 = \text{out}(u)\cdot\text{out}(v\{x \mapsto g_1(u)\}) \]
\[ \quad \text{if } b \text{ then } Q_1\{y \mapsto g_2(u, w)\} \text{ else } Q_2\{z \mapsto g_2(u, w)\} \]

where \( w = v\{x \mapsto g_1(u)\} \)

\[ P_2 = \text{out}(u')\cdot\text{out}(v'\{x' \mapsto g_1(u')\}) \]
\[ \quad Q'\{y' \mapsto g_2(u', w')\} \]

where \( w' = v'\{x' \mapsto g_1(u')\} \)
The folding trick

Example

\[ P_1 = \text{in}(x) \cdot \text{if \ EQ}(x, 0) \text{ then } \text{out}(c, 0) \text{ else } \text{out}(c, \langle x, 0 \rangle) \]

\[ t_{P_1} = \text{if \ EQ}(g_1(), 0) \text{ then } 0 \text{ else } \langle g_1(), 0 \rangle \]

if \_ then \_ else \_ is now a function symbol!
The folding trick

Example

\[ P_1 = \text{in}(x).\text{if } \text{EQ}(x, 0) \text{ then } \text{out}(c, 0) \text{ else } \text{out}(c, \langle x, 0 \rangle) \]

\[ t_{P_1} = \text{if } \text{EQ}(g_1(), 0) \text{ then } 0 \text{ else } \langle g_1(), 0 \rangle \]

if \_ then \_ else \_ is now a function symbol!

Example

\[ P = \text{in}(c, x) \cdot \text{out}(c, u) || \text{in}(c', y) \cdot \text{out}(c', v) \]

\[ t_P = \text{if } \text{to}() = c \text{ then } u\{x \mapsto g_1()\} \text{ else if } \text{to}() = c' \text{ then } \cdots \text{ else } 0 \]
The folding trick

Example

\[ P_1 = \text{in}(x).\text{if\ }\text{EQ}(x, 0)\ \text{then\ out}(c, 0)\ \text{else\ out}(c, \langle x, 0 \rangle) \]

\[ t_{P_1} = \text{if\ }\text{EQ}(g_1(), 0)\ \text{then\ 0\ else\ }\langle g_1(), 0 \rangle \]

if _ then _ else _ is now a function symbol!

Example

\[ P = \text{in}(c, x) \cdot \text{out}(c, u) \parallel \text{in}(c', y) \cdot \text{out}(c', v) \]

\[ t_P = \text{if\ }\text{to}() = c\ \text{then\ }u\{x \mapsto g_1()\}\ \text{else\ if\ }\text{to}() = c'\ \text{then\ }\cdots\ \text{else\ 0} \]

In general, \( t_P \) is a sequence of terms using additional functions \( g_1, g_2, \ldots, to, \ldots \) and if, EQ
Examples of axioms

Some axioms independent of the libraries

\[
\begin{align*}
\text{if } x \text{ then } y \text{ else } y & \sim y \\
\text{if } x \text{ then if } x \text{ then } y \text{ else } y' \text{ else } z & \sim \text{if } x \text{ then } y \text{ else } z \\
\text{if EQ}(u, v) \text{ then } C[u] \text{ else } w & \sim \text{if EQ}(u, v) \text{ then } C[v] \text{ else } w
\end{align*}
\]

\[
x_1 \sim y_1 \land \cdots \land x_n \sim y_n \implies f(x_1, \ldots, x_n) \sim f(y_1, \ldots, y_n)
\]

An IND-CPA axiom (assumption on the library)

\[
\overline{u}, \text{aenc}(x, \text{pk}(k), r) \sim \overline{u}, \text{aenc}(y, \text{pk}(k), r)
\]

if \( k \), does not occur (as plaintext) in \( \overline{u}, x, y, r \) does not occur in \( \overline{u}, x, y \).
Reducing the existence of an attack to satisfiability

$P_1$ is indistinguishable from $P_2$ iff

\[ t_{P_1} \not\sim t_{P_2} \quad \land \quad \text{Axioms} \]

is unsatisfiable. (The symbols $g_1, g_2, \ldots$ are part of the interpretation).
Reducing the existence of an attack to satisfiability

$P_1$ is indistinguishable from $P_2$ iff

$$t_{P_1} \not\sim t_{P_2} \land \text{Axioms}$$

is unsatisfiable. (The symbols $g_1, g_2, \cdots$ are part of the interpretation).

Is it decidable (for processes without replication)?
The main ideas

- Several possible interpretation domains/attacker models
- We cannot use the model-checking tools
- Tricky security properties
- The abstract symbolic semantics
- Axiomatizing what is *not possible*: Security is FO inconsistency