Process Calculi for Protocol Verification

Midlands Graduate School 2015

Eike Ritter

University of Birmingham

Outline

Course content: Formal specification and verification of security protocols using a suitably modified process calculus

- Lecture 1: Security protocols
- Lecture 2: The Applied Pi-Calculus
- Lecture 3: Verification of Security properties in the Applied Pi-Calculus
- Lecture 4: Tools for automating this verification

Security Protocols

Security protocols are exchange of messages between parties in the presence of adversaries

Want to achieve security properties like

- confidentiality: Who can access the information?
- authenticity: Who has sent the information?
- integrity: Has the message been modified in transit?
- privacy: How much private information can other parties learn?

Require cryptography to achieve these properties

Examples

- Authentication protocols
  Verify credentials of users without without the attacker obtaining them (eg login)
- Confidential data transfer
  Only sender and receiver can learn data (eg https)
- Key exchange protocols
  Establish session keys between participants
  Design of such protocols surprisingly difficult

Capabilities of attacker

Protocol is exchange of messages between various parties
Unless otherwise specified, have adversary which is able to

- read all messages
- modify all messages
- inject any message (including previously read ones)
- delete any message

In this course, consider only attacks caused by incorrectly specified protocols

Do not consider attacks based on

- exploiting weaknesses in cryptography
  (ie we assume perfect cryptography)
- timing attacks
  (eg attacks to obtain keys stored on smart cards)
- weak random number generators (we assume random numbers are unguessable)
- social engineering (eg getting the keys by pretending to be someone else)

So-called Dolev-Yao attacker

Examples of attacks on Security protocols

Security protocols often consist only of small number of messages
Surprising number of attacks have been found, often years after being defined:

- Needham-Schroder authentication protocol:
  flaw found 17 years(!!) after definition
- Single Sign-on for Google apps
  Authentication failed
- Linkability in Mobile phones
  Can trace mobile phone users without their knowledge

Basic Cryptographic operations

Will use both symmetric and asymmetric cryptography

Symmetric cryptography

Have one key k and two operations:

- $\mathrm{enc}(M, k)$ encrypts message M using key k;
- $\mathrm{dec}(M, k)$ decrypts message M using key k

such that

$$\mathrm{dec}(\mathrm{enc}(M, k), k) = M$$

Important property: Given $\mathrm{dec}(M, k)$ it is computationally infeasible to deduce M without knowing k
Have fast implementations in SW and HW (eg AES)

Asymmetric cryptography

Big disadvantage of symmetric crytography:
key must be known to both parties
Can relax this assumption by using asymmetric cryptography
Idea: Replace common key by public/private key pair
public key known to everybody
private key must not be revealed
Have operations adec and aenc, pk such that

- pk generates public key from private key
- $\mathrm{adec}(\mathrm{aenc}(M, \mathrm{pk}(k)), k) = M$
- $\mathrm{adec}(\mathrm{aenc}(M, k), \mathrm{pk}(k)) = M$

First equation used for encryption, second one for signing
Asymmetric cryptography is expensive
$\Rightarrow$ use it to establish shared key, then use symmetric encryption
Example: SSL/TLS

Protocol Notation

Write Alice, Bob for honest participants
Write Eve for passive attacker (cannot modify or inject messages)
Write Mallory for general attacker
Write $A \rightarrow B \ : \ m$ for message m sent over public channel from Alice to Bob
Write pk_A for Alice's public key
Write nc for nonce (large unguessable random number, usually assumed to be fresh)
use $\{M\}_k$ for encryption (symmetric or asymmetric) of message M with key k

Example: Needham-Schroder protocol

Purpose: Establish shared session key between Alice and Bob
Protocol (1978):

$$\begin{aligned} A \to B &: \{(A, nc_A)\}_{\mathrm{pk}_B} \\ B \to A &: \{(nc_A, nc_B)\}_{\mathrm{pk}_A} \\ A \to B &: \{nc_B\}_{\mathrm{pk}_B} \end{aligned}$$

Now A and B share nonces nc_A and nc_B which they can use to generate shared session key
Security property: Only A and B know nc_A and nc_B

The flaw in the Needham-Schroder protocol

Consider the following interaction between Alice, Bob and Mallory:

$$\begin{aligned} A \to M &: \{(A, nc_A)\}_{\mathrm{pk}_M} \\ M \to B &: \{(A, nc_A)\}_{\mathrm{pk}_B} \\ B \to A &: \{(nc_A, nc_B)\}_{\mathrm{pk}_A} \\ A \to M &: \{nc_B\}_{\mathrm{pk}_M} \\ M \to B &: \{nc_B\}_{\mathrm{pk}_B} \end{aligned}$$

Result: B thinks he has established common secrets with A but in fact secrets are known to Mallory
so-called Man-in-the-middle Attack

Fix

proposed by Lowe (1995) (!!)

$$\begin{aligned} A \to B &: \{(A, nc_A)\}_{\mathrm{pk}_B} \\ B \to A &: \{(nc_A, nc_B, B)\}_{\mathrm{pk}_A} \\ A \to B &: \{nc_B\}_{\mathrm{pk}_B} \end{aligned}$$

How do we know that this is now secure?

Another example: Voting

Want to achieve privacy property:
No-one (not even election officers) can link particular vote with voter
Need further cryptographic primitive: Blind Signatures
Have operations blind, unblind, sign such that

$$\text{unblind}(\text{sign}(\text{blind}(M, r), k), r) = \text{sign}(M, k)$$

(text M blinded with factor r signed with k and then unblinding with factor r produces signed text)

Have following protocol:

$$V \to O \ : \ \mathrm{sign}(\mathrm{blind}((v, n), r), sk_V)$$
$$O \to V \ : \ \mathrm{sign}(\mathrm{blind}((v, n), r), sk_O)$$
Synchronisation
$$V \to O \ : \ \mathrm{sign}((v, n), sk_O)(\text{Submitted anonymously})$$

Protocol achieves vote privacy (but otherwise inadequate)

Intuition

Want formal languages to model security protocols so that automatic verifiation can be done
Here: use Applied Pi-Calculus, a suitable adaptation of process calculi
Intuition:
Processes correspond to agents (Alice, Bob, Mallory etc.)
Sending messages modelled as communication in process calculus
Attacker modelled as arbitrary process which runs in parallel with processes modelling Alice and Bob

Applied pi calculus: Grammar

Terms

$$
\begin{array}{lll}
M, N ::= & & \\
 & a, b, c, k, m, n, s, t, r, \ldots & \text{name} \\
 & x, y, z & \text{variable} \\
 & g(M_1, \ldots, M_l) & \text{function}
\end{array}
$$

Equational theory

Applied pi calculus: Grammar

Processes

$P, Q, R ::=$		processes
	0	null process
	$P \mid Q$	parallel comp.
	$!P$	replication
	$\nu\, n.P$	name restriction
	$u(x).P$	message input
	$\overline{u}\langle M\rangle.P$	message output
	if $M = N$ then P else Q	cond'nl

$A, B, C ::=$		extended processes
	P	plain process
	$A \mid B$	parallel comp.
	$\nu\, n.A$	name restriction
	$\nu\, x.A$	variable restriction
	$\{M/x\}$	active substitution (attacker knowledge)

Structural equivalence

Par-0	A	$\equiv$	$A \mid 0$
Par-A	$A \mid (B \mid C)$	$\equiv$	$(A \mid B) \mid C$
Par-C	$A \mid B$	$\equiv$	$B \mid A$
Repl	$!P$	$\equiv$	$P \mid !P$
New-0	$\nu\, n.0$	$\equiv$	0
New-C	$\nu\, u.\nu\, w.A$	$\equiv$	$\nu\, w.\nu\, u.A$
New-Par	$A \mid \nu\, u.B$	$\equiv$	$\nu\, u.(A \mid B)$ where $u \notin \mathrm{fv}(A) \cup \mathrm{fn}(A)$
Alias	$\nu\, x.\{^M/_x\}$	$\equiv$	0
Subst	$\{^M/_x\} \mid A$	$\equiv$	$\{^M/_x\} \mid A\{^M/_x\}$
Rewrite	$\{^M/_x\}$	$\equiv$	$\{^N/_x\}$ where $M =_E N$

Internal reductions

Intuition: Describes dynamic behaviour of processes

$\textsc{Comm}$ $\overline{u}\langle x\rangle.P \mid u(x).Q \rightarrow P \mid Q$

$\textsc{Then}$ if $N = N$ then P else $Q \rightarrow P$

$\textsc{Else}$ if $L = M$ then P else $Q \rightarrow Q$
for ground terms L, M where $L \neq_E M$

Observational Equivalence

Idea: two processes are observationally equivalent if they have the same behaviour when run in parallel with any other process

Definition

An *evalution context* C is a process with a hole $\_$ not under a replication, a conditional, an input or an output.

We write $A \Downarrow a$ when A can send an arbitrary message on channel a.

Definition

Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes such that $A \mathcal{R} B$ implies

(i) if $A \Downarrow a$, then $B \Downarrow a$;

(ii) if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$

(iii) $C[A] \mathcal{R} C[B]$ for all closing evalution contexts $C[\_]$

Verification of observational equivalence

Key problem: have to consider all contexts when veryfying observational equivalence
Better strategy: characterise all possible ways of interaction between process and environment
Interactions captured by notion $A \xrightarrow{\alpha} A'$
Define labelled bisimulation as a relation which is satisfied if processes have the same way of interacting with the environment
Finally show: labelled bisimilarity and observational equivalence coincide

Static Equivalence

Intuition: Active substitutions $\{M_1/x_1, \ldots, M_n/x_n\}$ (shortcut for $\{M_1/x_1\} \mid \cdots \{M_n/x_n\}$), represent intruder knowledge
Important concept: when do two active substitutions represent the same intruder knowledge (called static equivalence, written $\approx_s$)
Key point: If two active substitutions are statically equivalent, intruder cannot apply any test to distinguish them
Example:

$$\nu k.\{\mathrm{enc}(a, k)/x\} \approx_s \nu k.\{\mathrm{enc}(b, k)/x\}$$

but

$$\{\mathrm{enc}(a, k)/x\} \not\approx_s \nu k.\{\mathrm{enc}(b, k)/x\}$$

Definition

(i) A *frame* ϕ is a process $\nu\vec{n}.\{M_1/x_1, \ldots, M_n/x_n\}$;

(ii) Two terms M and N are called *equal in the frame* ϕ, written $(M = N)\phi$ if $\phi = \nu\vec{n}.\sigma$ and $M\sigma = N\sigma$ and $\vec{n} \cap (FN(M) \cup FN(N)) = \emptyset$;

(iii) Two frames $\phi = \nu\vec{n}.\sigma$ and $\psi = \nu\vec{n}.\tau$ are called *statically equivalent* if for all terms M and N, $(M = N)\phi$ iff $(M = N)\psi$.

Labelled reductions

Intuition: model interaction of process with environment

IN $$u(x).P \xrightarrow{u(M)} P\{^M/_x\}$$

OUT-ATOM $$\overline{u}\langle x\rangle.P \xrightarrow{\overline{u}\langle x\rangle} P$$

SCOPE $$\frac{A \xrightarrow{\alpha} A' \qquad u \text{ does not occur in } \alpha}{\nu\, u.A \xrightarrow{\alpha} \nu\, u.A'}$$

PAR $$\frac{A \xrightarrow{\alpha} A' \qquad \mathrm{bv}(\alpha) \cap \mathrm{fv}(B) = \mathrm{bn}(\alpha) \cap \mathrm{fn}(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

STRUCT $$\frac{A \equiv B \qquad B \xrightarrow{\alpha} B' \qquad B' \equiv A'}{A \xrightarrow{\alpha} A'}$$

Labelled reductions, continued

$$\textsc{Open-Atom} \quad \frac{A \xrightarrow{\overline{u}\langle x\rangle} A' \qquad x \neq u}{\nu\, x.A \xrightarrow{\nu\, x.\overline{u}\langle x\rangle} A'}$$

Derived rule $\quad \overline{u}\langle M\rangle.P \xrightarrow{\nu x.\ \overline{u}\langle x\rangle} P \mid \{M/x\}$

Labelled bisimulation

Definition

Labelled bisimilarity ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes such that $A \mathcal{R} B$ implies

(i) $A \approx_s B$;

(ii) if $A \to A'$, then $B \to^* B'$ and $A' \mathcal{R} B'$;

(iii) if $A \xrightarrow{\alpha} A'$ then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A' \mathcal{R} B'$ (subject to some sanity conditions on bound variables)

Main theorem

Theorem

Observational equivalence and labelled bisimilarity coincide.

Proof.

Non-trivial but standard in process calculi. □

Automatic verification of protocol

One tool available, called Proverif
may be used to check both reachability properties as well as (limited) observational equivalences
Have input language modelled closely on applied pi-calculus

Capabilities of Proverif

Can verify

- Reachability properties: Is a certain event reachable (eg leaking secret keys to the attacker)
- Correspondence assertions: If event e has been executed, then event e' has been previously been executed
- Observational equivalences: same as in applied Pi-calculus
  However, Proverif can only verify special cases, namely where processes differ only via terms used and behave in an identical way

Limitations of Proverif

Translation of applied Pi-calculus into Horn clauses is only approximation

$\Rightarrow$ Proverif may report false attacks

Sources of false attacks:

- Private names in one session may be re-used in next session
- synchronisation using private channels not modelled

Proverif may also not terminate

How Proverif works

Proverif translates protocol into Horn clauses specifying the power of the attacker

Key predicates:

- $\mathtt{att}(M)$ means attacker knows term M
- $\mathtt{mess}(c, M)$ means message M was sent on channel c

Protocol rules are translated into Horn clauses

input generates hypothesis

output of M on channel c generates clause stating that current hypotheses implies $\mathtt{mess}(c, M)$

Reachability checked by testing whether predicate $\mathtt{att}(M)$ is derivable

Definition

We define a plain process by

P, Q	$::=$	0	null process
		$P \mid Q$	parallel composition
		$!P$	replication
		$\nu n.P$	name restriction
		$\texttt{if } M = N \texttt{ then } P \texttt{ else } Q$	conditional
		$u(x).P$	input
		$\overline{u}\langle M \rangle.P$	output

Definition

We define a *canonical process*, ranged over by A, B, C, to be an expression

$$\nu\vec{n}.(\sigma, \mathcal{P})$$

where

- $\nu\vec{n}$ is a set of bound names;
- σ is a substitution $\{\{M_1/x_1\}, \ldots, \{M_n/x_n\}\}$
- $\mathcal{P}$ is a multiset $\{P_1, \ldots, P_k\}$ where P_i is a plain process

A canonical process $\nu\vec{n}.(\sigma, \mathcal{P})$ is called *closed* if each variable is either bounded or defined by σ.

Structural reductions:

$$\begin{aligned}
\nu\vec{n}.(\sigma, \mathcal{P} \cup \{!P\}) &\rightarrow \nu\vec{n}.(\sigma, \mathcal{P} \cup \{!P, P\}) \\
\nu\vec{n}.(\sigma, \mathcal{P} \cup \{P \mid Q\}) &\rightarrow \nu\vec{n}.(\sigma, \mathcal{P} \cup \{P, Q\}) \\
\nu\vec{n}.(\sigma, \mathcal{P} \cup \{0\}) &\rightarrow \nu\vec{n}.(\sigma, \mathcal{P}) \\
\nu\vec{n}.(\sigma, \mathcal{P} \cup \{\nu m.P\}) &\rightarrow \nu\vec{n}, m.(\sigma, P \cup \{P\}) \\
&\quad \text{if } m \notin \vec{n} \cup FV(\sigma) \cup FV(\mathcal{P})
\end{aligned}$$

Internal reductions:

$$\begin{array}{rcl} \nu\vec{n}.(\sigma, \mathcal{P} \cup \{a(x).P)\} \cup \{\overline{a}\langle M\rangle.Q\}) & \rightarrow & \nu\vec{n}.(\sigma, \mathcal{P} \cup \{(P\{M/x\}, Q\})) \\ \nu\vec{n}.(\sigma, \mathcal{P} \cup \{\texttt{if } M = N \texttt{ then } P \texttt{ else } Q\}) & \rightarrow & \nu\vec{n}.(\sigma, \mathcal{P} \cup \{P\}) \\ & & \text{if } M =_\Sigma N \\ \nu\vec{n}(\sigma, \mathcal{P} \cup \{\texttt{if } M = N \texttt{ then } P \texttt{ else } Q\}) & \rightarrow & \nu\vec{n}(\sigma, \mathcal{P} \cup \{Q\}) \\ \text{if } M \neq_\Sigma N \text{ and } M, N \text{ ground terms} & & \end{array}$$

Labelled transitions:

$$\nu\vec{n}.(\sigma, \mathcal{P} \cup \{u\sigma(x).P\}) \xrightarrow{u(M)} \nu\vec{n}.(\sigma, \mathcal{P} \cup \{(P\{M\sigma/x\}\}) \quad \text{if } (\mathrm{fn}(M) \cup \{u\}) \cap \vec{n} = \emptyset$$

$$\nu\vec{n}.(\sigma, \mathcal{P} \cup \{\overline{u\sigma}\langle M\rangle.P\}) \xrightarrow{(\overline{u},x)} \nu\vec{n}.(\sigma \cup \{\{M/x\}\}, \mathcal{P} \cup \{P\}) \quad \text{if } u \notin \vec{n} \text{ and } x \notin \mathrm{dom}(\sigma) \cup FV(\sigma) \cup FV(P)$$

Proverif translates protocol into Horn clauses specifying the power of the attacker
Key predicates:

- $\texttt{att}(M)$ means attacker knows term M
- $\texttt{mess}(c, M)$ means message M was sent on channel c

Translation uses

- ρ is environment (mapping names and variables of process language to terms of clause language)
- H is a set of formulae containing hypotheses
- ℓ accumulates variables that have been input. Used to partially differentiate new names in different sessions

$$
\begin{aligned}
[\![0]\!]\rho H \ell &= \emptyset \\
[\![P|Q]\!]\rho H \ell &= [\![P]\!]\rho H \ell \cup [\![Q]\!]\rho H \ell \\
[\![!P]\!]\rho H \ell &= [\![P]\!]\rho H \ell \\
[\![\nu a.P]\!]\rho H \ell &= [\![P]\!](\rho \cup \{a \mapsto a[\ell]\}) H \ell \\
[\![u(x).P]\!]\rho H \ell &= [\![P]\!](\rho \cup x \mapsto x)(H \wedge \mathtt{mess}(u\rho, x)(x :: \ell) \\
[\![\overline{u}\langle M\rangle.P]\!]\rho H \ell &= \{H \supset \mathtt{mess}(u\rho, M\rho)\} \cup [\![P]\!]\rho H \ell \\
[\![\mathtt{if}\ M = N\ \mathtt{then}\ P & \\
\mathtt{else}\ Q]\!] &= [\![P]\!](\rho\omega)H(\omega)(\ell\omega) \cup [\![Q]\!]\rho H \ell \\
& \quad \omega \text{ mgu of } M\rho \text{ and } N\rho
\end{aligned}
$$

Formulation of attacker knowledge:

- for each function symbol f with arity n, we have a rule

$$(\mathtt{att}(x_1) \wedge \cdots \wedge \mathtt{att}(x_n)) \supset \mathtt{att}(f(x_1, \ldots, x_n)$$

(attacker can apply functions)

- $$(\mathtt{mess}(x, y) \wedge \mathtt{att}(x)) \supset \mathtt{att}(y)$$

(attacker can observe messages on channels he knows)

- $$(\mathtt{att}(x) \wedge \mathtt{att}(y)) \supset \mathtt{mess}(x, y)$$

(attacker can construct messages if he knows channel and value)

Reachability properties (eg secrecy) is now question whether $\mathtt{att}(M)$ is derivable