Security in Process Calculi
Overview

◆ Pi calculus
  • Core language for parallel programming
  • Modeling security via name scoping

◆ Applied pi calculus
  • Modeling cryptographic primitives with functions and equational theories
  • Equivalence-based notions of security
  • A little bit of operational semantics
  • Security as testing equivalence
Pi Calculus

[Milner et al.]

- **Fundamental language for concurrent systems**
  - High-level mathematical model of parallel processes
  - The "core" of concurrent programming languages
  - By comparison, lambda-calculus is the "core" of functional programming languages

- **Mobility is a basic primitive**
  - Basic computational step is the transfer of a communication link between two processes
  - Interconnections between processes change as they communicate

- **Can be used as a general programming language**
A Little Bit of History

- **1980: Calculus of communicating systems (CCS)**
- **1992: Pi calculus**
  - Ability to pass channel names between processes
- **1998: Spi calculus**
  - Adds cryptographic primitives to pi calculus
  - Security modeled as scoping
  - Equivalence-based specification of security properties
  - Connection with computational models of cryptography
- **2001: Applied pi calculus**
  - Generic functions, including crypto primitives
Pi Calculus Syntax

◆ Terms
  • $M, N ::= \text{x}$ \hspace{2cm} \text{variables}
    \hspace{2cm} \{ \text{Let } u \text{ range over names and variables} \}
    \hspace{2cm} \text{names}
    \hspace{2cm} | \ n

◆ Processes
  • $P, Q ::= \text{nil}$ \hspace{2cm} \text{empty process}
    \hspace{2cm} | \ u(x).P \hspace{2cm} \text{receive term from channel } P \text{ and assign to } x
    \hspace{2cm} | \ !P \hspace{2cm} \text{replicate process } P
    \hspace{2cm} | \ P | Q \hspace{2cm} \text{run processes } P \text{ and } Q \text{ in parallel}
    \hspace{2cm} | \ (\nu n)P \hspace{2cm} \text{restrict name } n \text{ to process } P
Modeling Secrecy with Scoping

- A sends M to B over secure channel c

\[
A(M) = \tilde{c}\langle M \rangle \\
B = c(x)\.nil \\
P(M) = (\nu c) (A(M) | B)
\]

This restriction ensures that channel c is “invisible” to any process except A and B (other processes don’t know name c)
Secrecy as Equivalence

\[ A(M) = \tilde{c}(M).\text{nil} \]
\[ B = c(x).\text{nil} \]
\[ P(M) = (\nu c) (A(M) | B) \]

- P(M) and P(M') are “equivalent” for any values of M and M'
  - No attacker can distinguish P(M) and P(M')

- Different notions of “equivalence”
  - Testing equivalence or observational congruence
  - Indistinguishability by any probabilistic polynomial-time Turing machine
Another Formulation of Secrecy

\[ A(M) = \overline{c}(M).\text{nil} \]
\[ B = c(x).\text{nil} \]
\[ P(M) = (\nu c) (A(M) | B) \]

No attacker can learn name n from P(n)

- Let Q be an arbitrary attacker process, and suppose it runs in parallel with P(n)
- Specification of secrecy:
  For any process Q in which n does not occur free, P(n) | Q will never output n
Modeling Authentication with Scoping

- A sends M to B over secure channel c
- B announces received value on public channel d

\[ A(M) = \bar{c}(M) \]
\[ B = c(x) \cdot \bar{d}(x) \]
\[ P(M) = (\nu c)(A(M) \mid B) \]
Specifying Authentication

\[ A(M) = \tilde{c}\langle M \rangle \]
\[ B = c(x) \cdot \tilde{d}\langle x \rangle \]
\[ P(M) = (\nu c) (A(M) | B) \]

Specification of authentication:
For any value of M, if B outputs M on channel \( d \), then A previously sent M on channel \( c \).
1. A and B have pre-established pairwise keys with server S
   - Model these keys as names of pre-existing communication channels
2. A creates a new key and sends it to S, who forwards it to B
   - Model this as creation of a new channel name
3. A sends M to B encrypted with the new key, B outputs M
Key Establishment in Pi Calculus

\[ A(M) = (\nu c_{AB}) c_{AS}(c_{AB}) . c_{AB}(M) \]
\[ S = c_{AS}(x) . c_{SB}(x) \]
\[ B = c_{SB}(x) . x(y) . d(y) \]
\[ P(M) = (\nu c_{AS}) (\nu c_{SB}) (A(M) | B | S) \]

Note communication on a channel with a dynamically generated name.
Applied Pi Calculus

- In pi calculus, channels are the only primitive
- This is enough to model some forms of security
  - Name of a communication channel can be viewed as an “encryption key” for traffic on that channel
    - A process that doesn’t know the name can’t access the channel
  - Channel names can be passed between processes
    - Useful for modeling key establishment protocols
- To simplify protocol specification, applied pi calculus adds functions to pi calculus
  - Crypto primitives modeled by functions and equations
Applied Pi Calculus: Terms

\[ M, N ::= \]

\[ \quad \text{Variable} \]
\[ \quad \text{x} \]
\[ \quad \text{Name} \]
\[ \quad n \]
\[ \quad \text{Function application} \]
\[ \quad f(M_1,\ldots,M_k) \]

◆ Standard functions
  - pair(), encrypt(), hash(), ...

◆ Simple type system for terms
  - Integer, Key, Channel\langle Integer\rangle, Channel\langle Key\rangle
Applied Pi Calculus: Processes

\[ P, Q ::= \text{nil} \quad \text{empty process} \]
\[ \quad | \ u\langle N\rangle.P \quad \text{send term } N \text{ on channel } u \]
\[ \quad | \ u(x).P \quad \text{receive from channel } P \text{ and assign to } x \]
\[ \quad | \ !P \quad \text{replicate process } P \]
\[ \quad | \ P|Q \quad \text{run processes } P \text{ and } Q \text{ in parallel} \]
\[ \quad | \ (\nu n)P \quad \text{restrict name } n \text{ to process } P \]
\[ \quad | \ \text{if } M = N \text{ then } P \text{ else } Q \quad \text{conditional} \]
Modeling Crypto with Functions

- Introduce special function symbols to model cryptographic primitives
- Equational theory models cryptographic properties
- Pairing
  - Functions `pair`, `first`, `second` with equations:
    \[
    \text{first}(\text{pair}(x,y)) = x \\
    \text{second}(\text{pair}(x,y)) = y
    \]
- Symmetric-key encryption
  - Functions `symenc`, `symdec` with equation:
    \[
    \text{symdec}(\text{symenc}(x,k),k) = x
    \]
More Equational Theories

◆ Public-key encryption
  • Functions $pk, sk$ generate public/private key pair $pk(x), sk(x)$ from a random seed $x$
  • Functions $pdec, penc$ model encryption and decryption with equation:
    $$pdec(penc(y,pk(x)),sk(x)) = y$$
  • Can also model “probabilistic” encryption:
    $$pdec(penc(y,pk(x),z),sk(x)) = y$$

◆ Hashing
  • Unary function $hash$ with no equations
  • $hash(M)$ models applying a one-way function to term $M$

Models random salt (necessary for semantic security)
Yet More Equational Theories

◆ Public-key digital signatures
  • As before, functions \( pk, sk \) generate public/private key pair \( pk(x), sk(x) \) from a random seed \( x \)
  • Functions \( \text{sign}, \text{verify} \) model signing and verification with equation:
    \[
    \text{verify}(y, \text{sign}(y, sk(x)), pk(x)) = y
    \]

◆ XOR
  • Model self-cancellation property with equation:
    \[
    \text{xor}(\text{xor}(x, y), y) = x
    \]
  • Can also model properties of cyclic redundancy codes:
    \[
    \text{crc}(\text{xor}(x, y)) = \text{xor}(\text{crc}(x), \text{crc}(y))
    \]
Dynamically Generated Data

- Use built-in name generation capability of pi calculus to model creation of new keys and nonces

\[ A(M) = \overline{c}\langle(M, s)\rangle \]

\[ B = c(x). \text{if } \text{second}(x) = s \]
\[ \text{then } \overline{d}\langle\text{first}(x)\rangle \]

\[ P(M) = (\nu s)\ (A(M) \mid B) \]

Models creation of fresh capability every time A and B communicate.
Better Protocol with Capabilities

\[
A(M) = \tilde{c}\langle (M, \text{hash}(s, M)) \rangle
\]

\[
B = c(x).\text{if second}(x) = \text{hash}(s, \text{first}(x)) \text{ then } \tilde{d}\langle \text{first}(x) \rangle
\]

\[
P(M) = (\nu s)(A(M) \mid B)
\]

Hashing protects integrity of M and secrecy of s