CS 395T

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) [Milner]
- 1992: Pi calculus [Milner, Parrow, Walker]
  - Ability to pass channel names between processes
- 1998: Spi calculus [Abadi, Gordon]
  - 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 [Abadi, Fournet]
  - Generic functions, including crypto primitives

Pi Calculus Syntax

- Terms
  - $M, N ::= x$ *variables*
    $\quad | \; n$ *names*
    $\Big\}$ *Let u range over names and variables*
- Processes
  - $P,Q ::= \mathrm{nil}$ *empty process*
    $\quad | \; \bar{u}\langle N \rangle.P$ *send term N on channel u*
    $\quad | \; u(x).P$ *receive term from channel P and assign to x*
    $\quad | \; !P$ *replicate process P*
    $\quad | \; P|Q$ *run processes P and Q in parallel*
    $\quad | \; (\nu n)P$ *restrict name n to process P*

Modeling Secrecy with Scoping

- A sends M to B over secure channel c

$$
\begin{aligned}
A(M) &= \bar{c}\langle M\rangle \\
B &= c(x).\mathrm{nil} \\
P(M) &= (\nu c)(A(M)\,|\,B)
\end{aligned}
$$

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

$$
\begin{aligned}
A(M) &= \bar{c}\langle M\rangle.\mathtt{nil} \\
B &= c(x).\mathtt{nil} \\
P(M) &= (\nu c)(A(M) \mid B)
\end{aligned}
$$

Without (νc), attacker could run process c(x) and tell the difference between P(M) and P(M’)

- 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

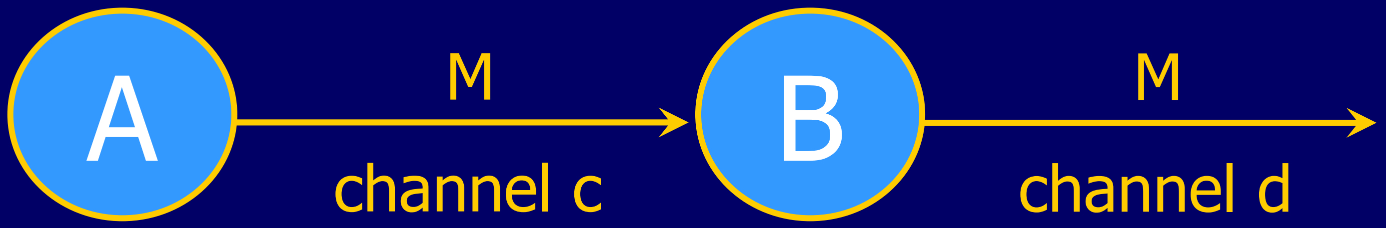
$$A(M) = \bar{c}\langle M\rangle.\mathrm{nil}$$
$$B = c(x).\mathrm{nil}$$
$$P(M) = (\nu c)(A(M)\,|\,B)$$

- 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



$$
\begin{aligned}
A(M) &= \bar{c}\langle M\rangle \\
B &= c(x).\bar{d}\langle x\rangle \\
P(M) &= (\nu c)(A(M)\,|\,B)
\end{aligned}
$$

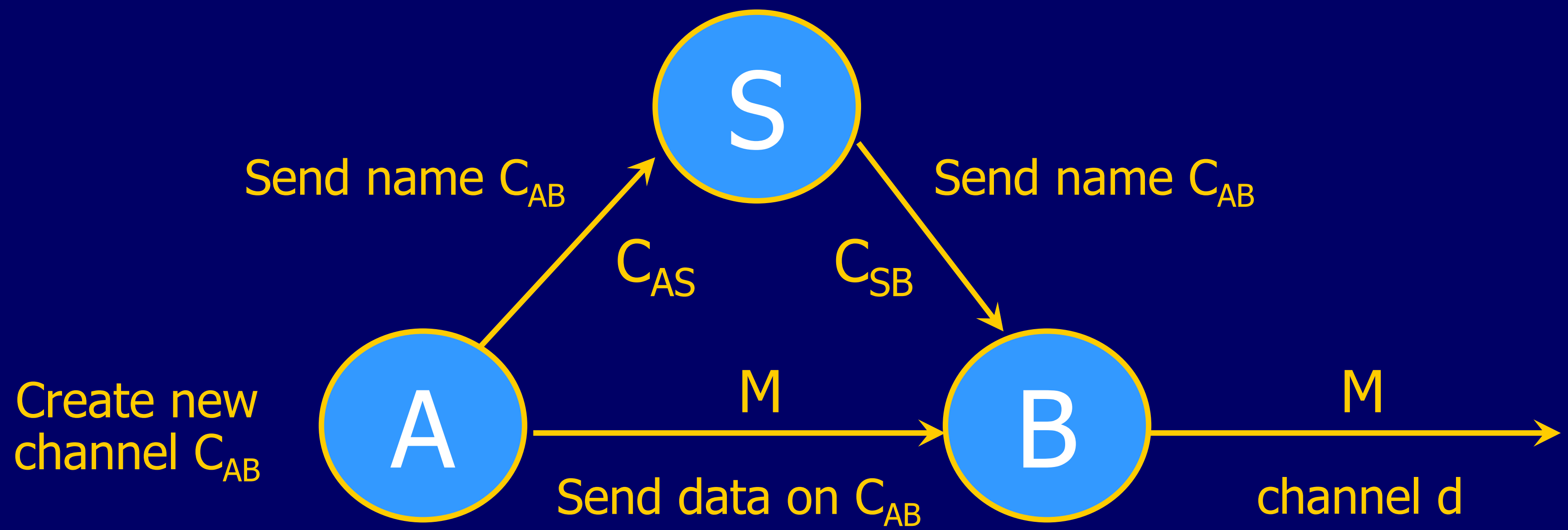
Specifying Authentication

$$A(M) = \bar{c}\langle M\rangle$$
$$B = c(x).\bar{d}\langle x\rangle$$
$$P(M) = (\nu c)(A(M)|B)$$

- Specification of authentication:

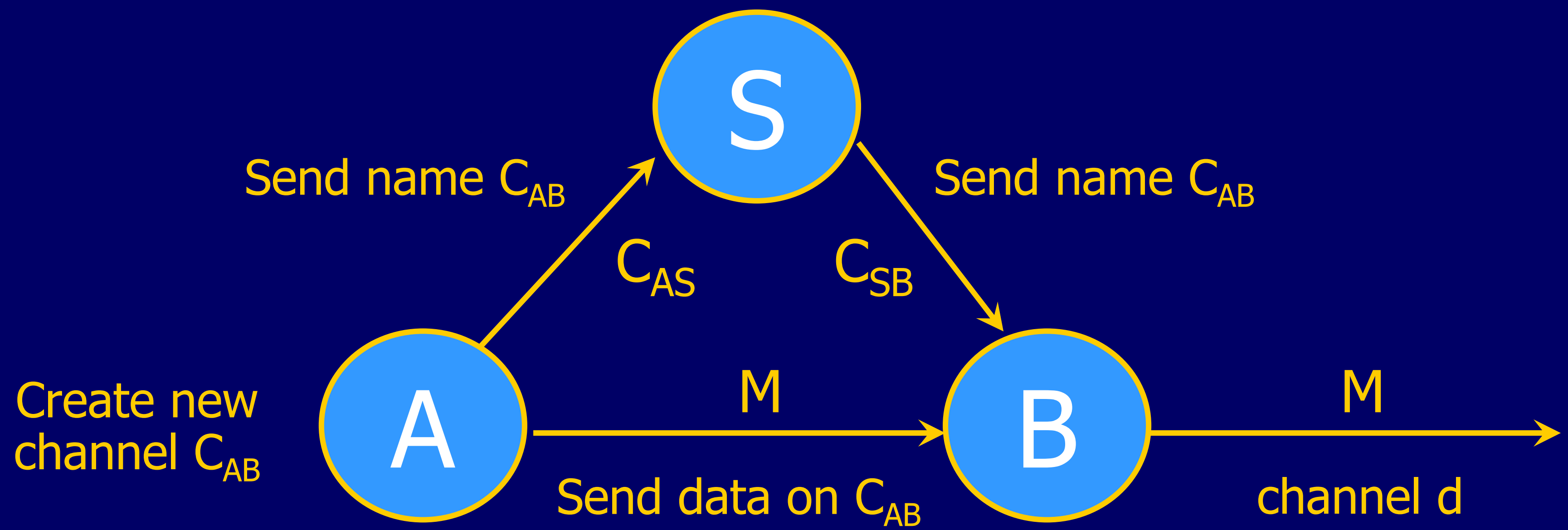
  For any value of M, if B outputs M on channel d, then A previously sent M on channel c

A Key Establishment Protocol



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



$$A(M) = (\nu c_{AB})\,\overline{c_{AS}}\langle c_{AB}\rangle.\overline{c_{AB}}\langle M\rangle$$

$$S = c_{AS}(x).\overline{c_{SB}}\langle x\rangle \_$$

$$B = c_{SB}(x).x(y).d\langle y\rangle$$

$$P(M) = (\nu c_{AS})(\nu c_{SB})(A(M)|B|S)$$

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 ::=	x	*Variable*
|	n	*Name*
|	$f(M_1,\ldots,M_k)$	*Function application*

- Standard functions
  - pair(), encrypt(), hash(), …
- Simple type system for terms
  - Integer, Key, Channel⟨Integer⟩, Channel⟨Key⟩

Applied Pi Calculus: Processes

$P,Q ::= \mathrm{nil}$	*empty process*
$\mid \bar{u}\langle N\rangle.P$	*send term N on channel u*
$\mid u(x).P$	*receive from channel P and assign to x*
$\mid\ !P$	*replicate process P*
$\mid P\mid Q$	*run processes P and Q in parallel*
$\mid (\nu n)P$	*restrict name n to process P*
$\mid \text{if } M = N \text{ then } P \text{ else } Q$	*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:

    first(pair(x,y)) = x

    second(pair(x,y)) = y
- Symmetric-key encryption
  - Functions symenc, symdec with equation:

    symdec(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

Models random salt
(necessary for semantic security)

◆ Hashing

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

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 sign,verify model signing and verification with equation:

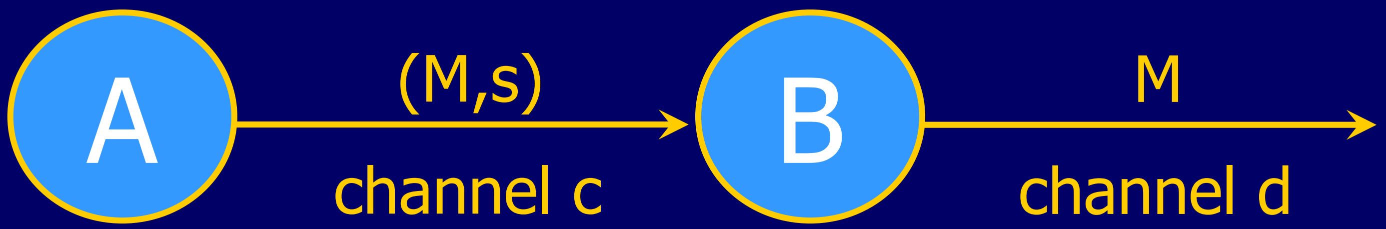
    verify(y,sign(y,sk(x)),pk(x)) = y
- XOR
  - Model self-cancellation property with equation:

    xor(xor(x,y),y) = x
  - Can also model properties of cyclic redundancy codes:

    crc(xor(x,y)) = xor(crc(x),crc(y))

Dynamically Generated Data

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

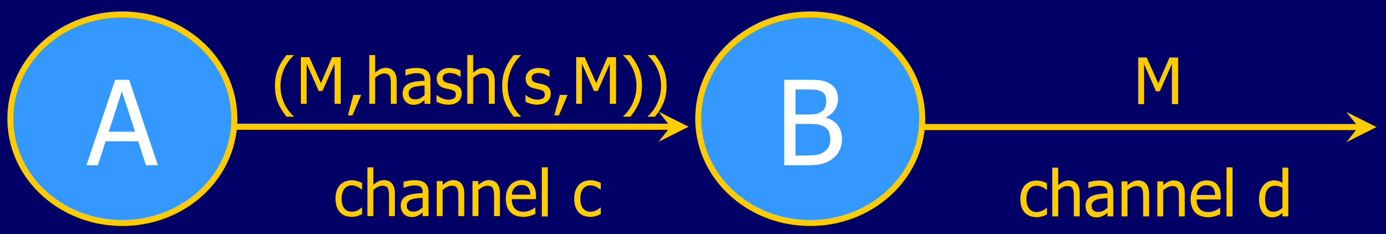


$$\begin{aligned}
A(M) &= \bar{c}\langle (M,s) \rangle \\
B &= c(x).\text{if } second(x)=s \\
&\qquad\quad \text{then } \bar{d}\langle first(x) \rangle \\
P(M) &= (\nu s)(A(M) \mid B)
\end{aligned}$$

Models creation of fresh capability every time A and B communicate

capability s may be intercepted!

Better Protocol with Capabilities



Hashing protects integrity of M and secrecy of s

$$
\begin{aligned}
A(M) &= \bar{c}\langle (M, \mathtt{hash}(s,M)) \rangle \\
B &= c(x).\mathtt{if}\ \mathtt{second}(x) = \mathtt{hash}(s, \mathtt{first}(x)) \\
&\qquad \mathtt{then}\ \bar{d}\langle \mathtt{first}(x) \rangle \\
P(M) &= (\nu s)(A(M) \mid B)
\end{aligned}
$$