18739A: Foundations of Security and Privacy

Process Calculus and Security

Anupam Datta

Fall 2007-08

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.]

- Modeling language for concurrent systems
  - High-level mathematical model of parallel processes
  - A “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
  - In theory at least; see Pierce’s Pict implementation

A Little Bit of History

[Milner]

- 1980: Calculus of Communicating Systems (CCS)
- 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*
    | n — *names*

  } *Let u range over names and variables*

- **Processes**
  - P,Q ::= nil — *empty process*
    | $u\langle N\rangle.P$ — *send term N on channel u*
    | u(x).P — *receive term from channel P and assign to x*
    | !P — *replicate process P*
    | P|Q — *run processes P and Q in parallel*
    | $(\nu n)P$ — *restrict name n to process P*

Examples

- Process to send a message
  - $\bar{c}\langle M\rangle$
- Process to receive x and send x+1
  - $c(x).\ \bar{c}\langle x+1\rangle$
- Process to compute n factorial
  - $c(n,1) \mid ![\ c(x,y).$ if $x>0$ then $c\langle x-1, y*x\rangle$ else $\bar{d}\langle y\rangle]$
- With input and output from channel d
  - $d(z).\ (\nu\ c)(\ \ c(z,1) \mid ![\ \ldots$ if ... then ... else $\bar{d}\langle y\rangle]\ )$

Other processes can send, receive on d, but cannot see actions on private channel c

Modeling Secrecy with Scoping

- A sends M to B over secure channel c

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

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

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

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')

Another Formulation of Secrecy

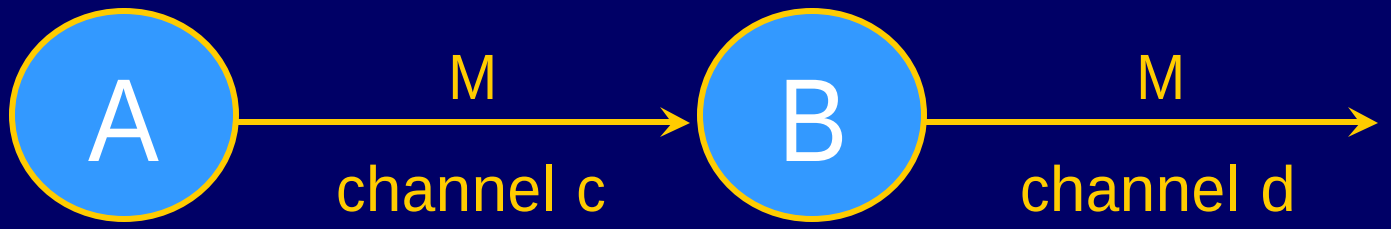
$$A(M) = \bar{c}\langle M \rangle$$
$$B = c(x).nil$$
$$P(M) = (\nu c)(A(M) \mid B)$$

- 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)
  - For any process Q in which n does not occur,

    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



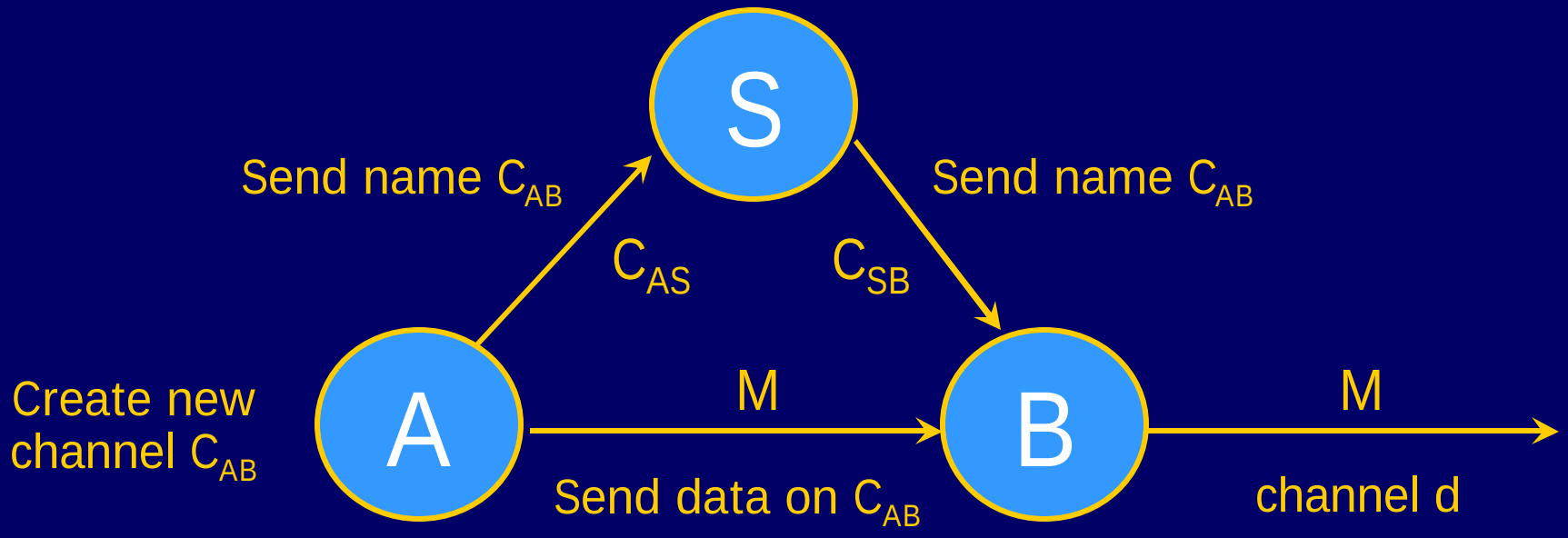
$$A(M) = \bar{c}\langle M\rangle$$
$$B = c(x).\bar{d}\langle x\rangle$$
$$P(M) = (\nu c)(A(M)|B)$$

Specifying Authentication

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

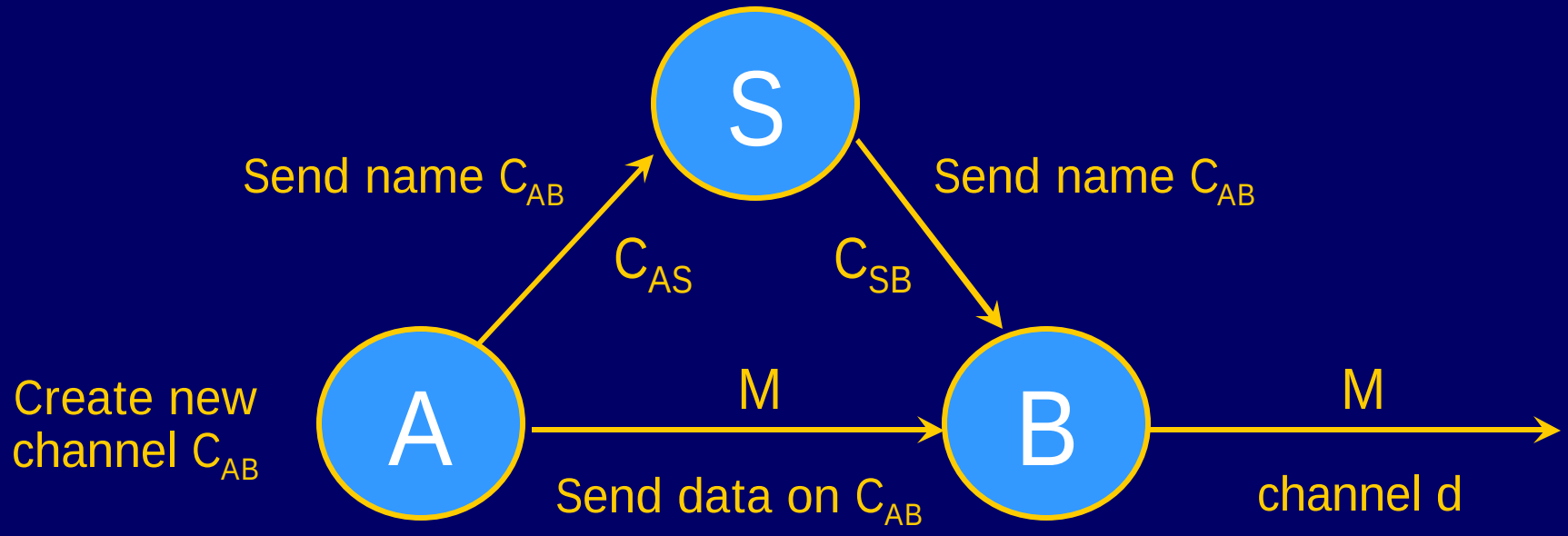
- For any value of M, if B outputs M on channel d, then A previously sent M on channel c

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

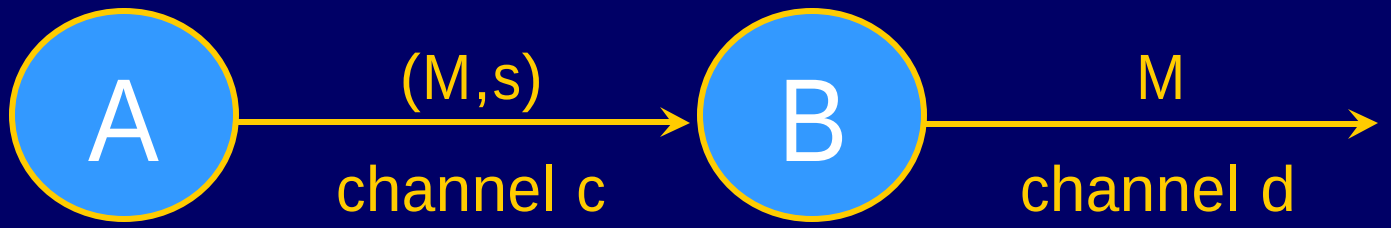
    verify(y,sign(y,sk(x)),pk(x)) = y
- XOR
  - Model self-cancellation property with equation:

    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

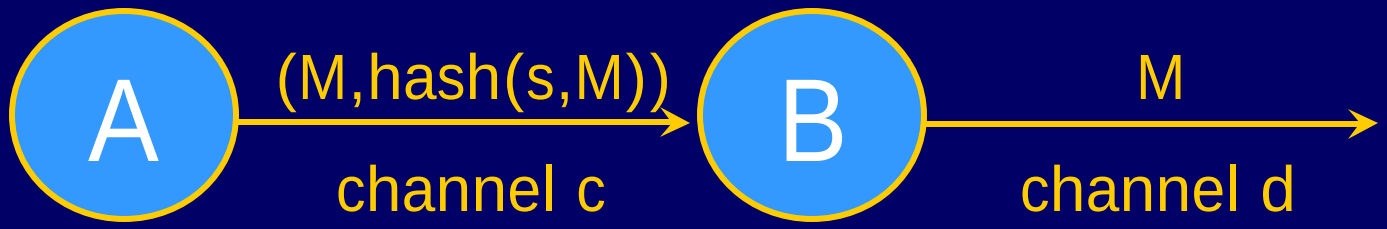


$$A(M) = \bar{c}\langle(M,s)\rangle$$
$$B = c(x).\text{if } second(x)=s \text{ then } \bar{d}\langle first(x)\rangle$$
$$P(M) = (\nu s)(A(M)|B)$$

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

$$A(M) = \bar{c}\langle(M, hash(s,M))\rangle$$
$$B = c(x).\text{if } second(x) = hash(s, first(x)) \text{ then } \bar{d}\langle first(x)\rangle$$
$$P(M) = (\nu s)(A(M) \mid B)$$

Operational Semantics

- Reduction $\rightarrow$ is the smallest relation on closed processes that is closed by structural equivalence and application of evaluation contexts such that

$$a\langle M\rangle.P \mid a(x).Q \rightarrow P \mid Q[M/x]$$

models P sending M to Q on channel a

$$\text{if } M = M \text{ then } P \text{ else } Q \rightarrow P$$

$$\text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q$$

for any ground M, N s.t. M $\neq$ N in the equational theory

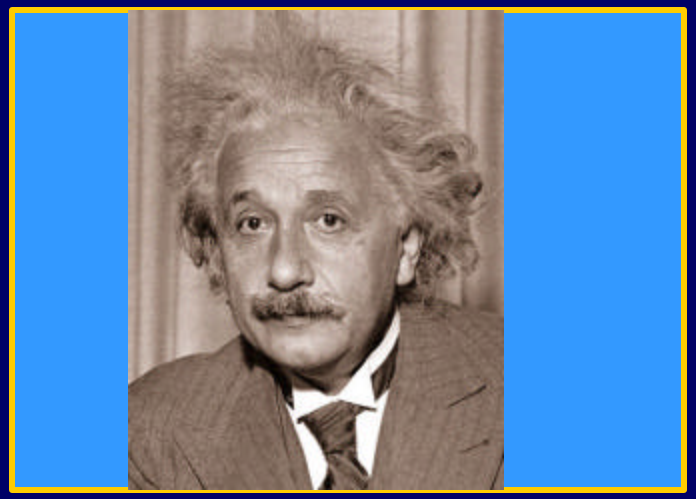
Outline

- Applied Pi Calculus
  - Syntax
  - Operational Semantics
  - Expressing and proving security properties

Proving Security

- "Real" protocol
  - Process-calculus specification of the actual protocol
- "Ideal" protocol
  - Achieves the same goal as the real protocol, but is secure by design
  - Uses unrealistic mechanisms, e.g., private channels
  - Represents the desired behavior of real protocol
- To prove the real protocol secure, show that no attacker can tell the difference between the real protocol and the ideal protocol
  - Proof will depend on the model of attacker observations

Is Bart Smart?



Who is in the box?

Can’t tell => Both equally smart

Example: Challenge-Response

- Challenge-response protocol

  $A \rightarrow B \quad \{i\}_k$

  $B \rightarrow A \quad \{i+1\}_k$

- This protocol is secure if it is indistinguishable from this “ideal” protocol

  $A \rightarrow B \quad \{random_1\}_k$

  $B \rightarrow A \quad \{random_2\}_k$

Example: Authentication

- Authentication protocol

  $A \rightarrow B$ $\{i\}_k$

  $B \rightarrow A$ $\{i+1\}_k$

  $A \rightarrow B$ "Ok"

- This protocol is secure if it is indistinguishable from this "ideal" protocol

  $A \rightarrow B$ $\{random_1\}_k$

  $B \rightarrow A$ $\{random_2\}_k$

  $B \rightarrow A$ $random_1$, $random_2$ on a magic secure channel

  $A \rightarrow B$ "Ok" if numbers on real & magic channels match

Security as Observational Equivalence

- Need to prove that two processes are observationally equivalent to the attacker
- Complexity-theoretic model
  - Prove that two systems cannot be distinguished by any probabilistic polynomial-time adversary

    [Beaver '91, Goldwasser-Levin '90, Micali-Rogaway '91]
- Abstract process-calculus model
  - Cryptography is modeled by abstract functions
  - Prove testing equivalence between two processes
  - Proofs are easier, but it is nontrivial to show computational completeness [Abadi-Rogaway '00]

Structural Equivalence

$$P \mid \mathrm{nil} \equiv P$$

$$P \mid Q \equiv Q \mid P$$

$$P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

$$!P \equiv P \mid !P$$

$$(\nu m)\,(\nu n)P \equiv (\nu n)\,(\nu m)P$$

$$(\nu n)\mathrm{nil} \equiv \mathrm{nil}$$

$$(\nu n)(P \mid Q) \equiv P \mid (\nu n)Q$$ *if n is not a free name in P*

$$P[M/x] \equiv P[N/x]$$ *if M=N in the equational theory*

Equivalence in Process Calculus

- Standard process-calculus notions of equivalence such as bisimulation are not adequate for cryptographic protocols
  - Different ciphertexts leak no information to the attacker who does not know the decryption keys
- $(\nu k)\bar{c}\langle \mathrm{symenc}(M,k)\rangle$ and $(\nu k)\bar{c}\langle \mathrm{symenc}(N,k)\rangle$ send different messages, but they should be treated as equivalent when proving security
  - In each case, a term is encrypted under a fresh key

Note

- The next few slides are quite technical
- Will revisit these concepts in a later lecture with examples

Observational Equivalence

Definition 1 Observational equivalence *($\approx$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A\ \mathcal{R}\ B$ implies:*

1. *if $A \Downarrow a$, then $B \Downarrow a$;*
2. *if $A \rightarrow^* A'$, then $B \rightarrow^* B'$ and $A'\ \mathcal{R}\ B'$ for some B';*
3. *$C[A]\ \mathcal{R}\ C[B]$ for all closing evaluation contexts $C[\_]$.*

Static Equivalence

- Frames are static knowledge exported by a process to the execution environment
  - Assignment of values to variables
    - $\{x=M, y=enc_k(M,x), \ldots\}$
  - Attacker (i.e., environment) learns these values
- Two frames φ and ψ are statically equivalent if they map the same variables to equal values
  - $Dom(\varphi)=Dom(\psi)$ and $\forall$ terms M, N $(M=N)\varphi$ iff $(M=N)\psi$
- Two processes are statically equivalent if they export the same knowledge to the environment
  - $A \approx_s B$ if their frames are statically equivalent

Labeled Bisimilarity

- Labeled bisimilarity is the largest symmetric relation *R* on closed processes s.t. A *R* B implies
  1. $A \approx_s B$
  2. If $A \rightarrow A'$, then $B \rightarrow^* B'$ and A' *R* B' for some B'
  3. If $A \stackrel{\alpha}{\Rightarrow} A'$ and $\text{freevars}(\alpha) \subseteq \text{dom}(A)$ and $\text{boundnames}(\alpha) \cap \text{freenames}(B) = \emptyset$, then $B \rightarrow^* \stackrel{\alpha}{\rightarrow} \rightarrow^* B'$ and A' *R* B' for some B'
- Why labeled bisimilarity?
  - Congruence: $\forall$ context C[], $A \approx_l B$ implies $C[A] \approx_l C[B]$
  - Easier to check than direct observational equivalence: only care about steps that export values to environment

Advantages and Disadvantages

- Proving testing equivalence is hard
  - Need to quantify over all possible attacker processes and all tests they may perform
- Testing equivalence is a congruence
  - Can compose protocols like building blocks

Bibliography

- Robin Milner. “Communication and Concurrency”. Prentice-Hall, 1989.
  - Calculus of communicating systems (CCS)
- Robin Milner. “Communicating and Mobile Systems: the π-Calculus”. Cambridge University Press, 1999.
  - Pi calculus
- Martin Abadi and Andrew Gordon. “A calculus for cryptographic protocols: the spi-calculus”. Information and Computation 148(1), 1999.
  - Spi calculus
- Martin Abadi and Cedric Fournet. “Mobile values, new names, and secure communication”. POPL 2001.
  - Applied pi calculus

Acknowledgement

- Lecture based on slides from J. Mitchell and V. Shmatikov