Programming Language Methods in Computer Security

John Mitchell
Stanford University

Plan

- Perspective on computer security
- Protocol security
  - Protocol examples
  - A basic rewriting model
  - Incorporating probability and complexity

Talk is deliberately too long – give impressionistic view of main ideas; you can read details later

Part I

Computer Security

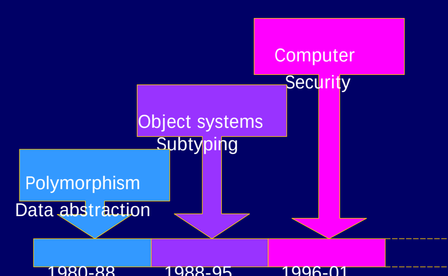
Orientation

- Computer security is
  - Branch of computer science concerned with the protection of computer systems and digital information
- Opportunistic view
  - This is a problem area
  - Not a solution technique

You can use methods you know to solve problems in computer security

And you may find yourself
living in a shotgun shack
And you may find yourself
in another part of the world
And you may find yourself
…
in a beautiful house,
with a beautiful wife
And you may ask yourself
Well…
How did I get here?

Personal POPL timeline



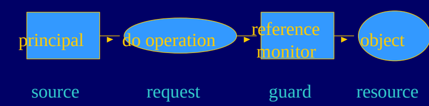
Personal turning point

- Is Java secure?
  - Proof is easy
    - Induction on structure of expressions, proving preservation of some property by structured operational semantics
    - (This was known to Curry, Hindley, …)
  - But what's the theorem?
    - Need to understand what "secure" means

Study some problems in computer security to see what this is all about

Topics in computer security

- Access control



- Operating systems security
- Network security
- Cryptography

Cryptography is wonderful, but almost all CERT advisories are software problems, not crypto.

Some references

- Books
  - D. Gollman, Computer Security, Wiley, 1999.
  - W. Stallings, Cryptography and Network Security …, Prentice-Hall, 1999.
  - A.J. Menzies, P.C. van Oorschot, and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1997.
  - D. Kahn, The Codebreakers, MacMillan, 1967.
- Periodicals and Journals
  - J. Computer Security
  - J. Cryptology

- Research Conferences
  - Crypto, EuroCrypt, AsiaCrypt (www.iacr.org)
  - IEEE Security and Privacy
  - IEEE Computer Security Foundations Workshop (CSFW)
  - ACM Computer and Communication Security
- On-line newsgroups, web sites
  - Comp.risks, Comp.lang.java.security
  - CERT
  - Internet RFCs
  - RSA FAQ, many many more

Security vs Correctness

- Correctness
  - Given expected input, system produces desired output
- Security
  - Given arbitrary input, system does not
    - reveal secrets
    - become corrupted
    - provide false guarantees

Security usually involves safety properties; adversary can often destroy liveness properties

Example: Protocol Security

- Cryptographic Protocol
  - Program distributed over network
  - Use cryptography to achieve goal
- Attacker
  - Read, intercept, replace messages and remember their contents
- Correctness
  - Attacker cannot learn protected secret or cause incorrect protocol completion

POPL relevance

- Modeling
  - Need to characterize possible behaviors of system and attacker
- Verification
  - Show that system has security property
- Language security issues
  - Sandboxing, Java security
  - Mobile code security

Example of POPL-relevant concept

- Folklore in security community
  - Security properties do not compose
- Why is this a problem?
  - Build secure system from secure parts
- Can this be correct?
  - IMH(B)O, this is based on naïveté of researchers in security community

Compositionality is fundamental in denotational semantics, programming language foundations

Outline of rest of talk

- Sample protocols
- Formulation of protocol security
  - Complexity, decidability results
- Process calculus approach to probability and complexity
  - Why secrecy does not compose
  - Observational congruence "solves" problem

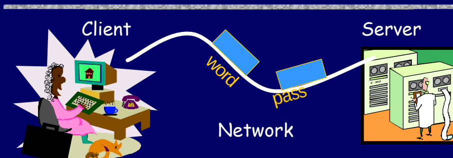
Part II

Security Protocols

Examples

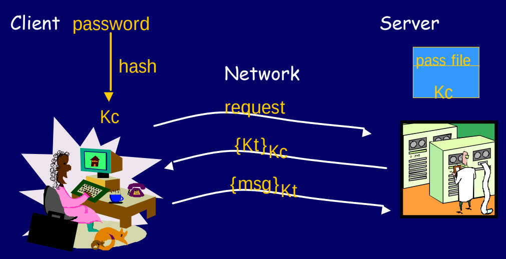
- Kerberos
  - Authentication protocol
  - Keep plaintext passwords off network
- SSL
  - Secure communication layer over TCP
  - Used for web transactions
- Contract signing protocols
  - Symmetric goal for asymmetric protocol
- Needham-Schroeder public key protocol
  - Simplified research-paper example

Motivation for Kerberos



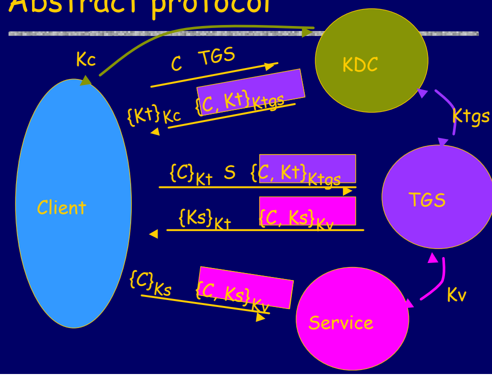
- Login, ftp connections require authentication
  - Intruders can run "packet sniffers"
- Keep passwords off the network
  - Challenge-response under shared secret key

Kerberos passwords and keys



Password shared with server but not transmitted

Abstract protocol



SSL: Secure Sockets Layer

Another
complicated
real-life
protocol

SSL Handshake Protocol

- Negotiate protocol version, crypto suite
  - Possible "version rollback attack"
- Authenticate client and server
  - Appeal to "certificate authority"
- Use public key to establish shared secret

Several underlying primitives:
public key crypto, signature, hash, private key crypto

One general idea in SSL

- Client, server communicate

- Compare hash of all messages
  - Compute hash(hi,hello,howareyou?) locally
  - Exchange hash values under encryption
- Abort if intervention detected

Handshake Protocol Description

ClientHello $C \to S$ $C, Ver_C, Suite_C, N_C$

ServerHello $S \to C$ $Ver_S, Suite_S, N_S, sign_{CA}\{ S, K_S \}$

ClientVerify $C \to S$ $sign_{CA}\{ C, V_C \}$
$\{ Ver_C, Secret_C \}_{K_S}$
$sign_C \{ Hash(Master(N_C, N_S, Secret_C) + Pad_2 + Hash(Msgs + C + Master(N_C, N_S, Secret_C) + Pad_1)) \}$

(Change to negotiated cipher)

ServerFinished $S \to C$ $\{ Hash(Master(N_C, N_S, Secret_C) + Pad_2 + Hash(Msgs + S + Master(N_C, N_S, Secret_C) + Pad_1)) \}_{Master(N_C, N_S, Secret_C)}$

ClientFinished $C \to S$ $\{ Hash(Master(N_C, N_S, Secret_C) + Pad_2 + Hash(Msgs + C + Master(N_C, N_S, Secret_C) + Pad_1)) \}_{Master(N_C, N_S, Secret_C)}$

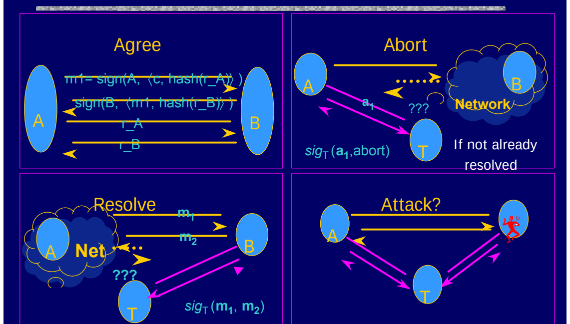
Contract signing

- Both parties want to sign the contract
- Neither wants to commit first

General protocol outline

- Trusted third party can force contract
  - Third party can declare contract binding if presented with first two messages.

Asokan-Shoup-Waidner protocol



Abuse-Free Contract Signing

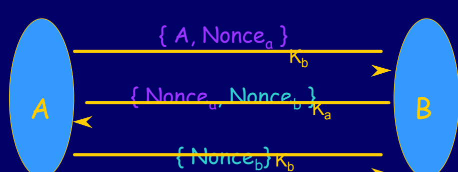
Abuse = Ability to determine the outcome + Ability to prove it

Not a trace property!

- Example
  - Alice agrees to buy Bob's house
  - Bob shows partially signed contract to Carol

Protocol analysis can benefit from sophisticated concurrency models

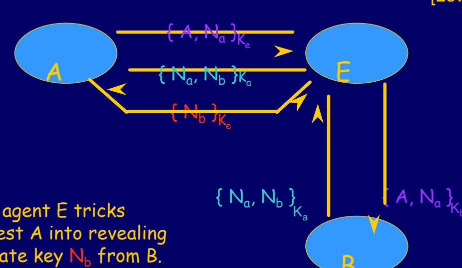
Needham-Schroeder Key Exchange



Result: A and B share two private numbers not known to any observer without K_a^{-1}, K_b^{-1}

Anomaly in Needham-Schroeder

[Lowe]



Evil agent E tricks honest A into revealing private key N_b from B.

Evil E can then fool B.

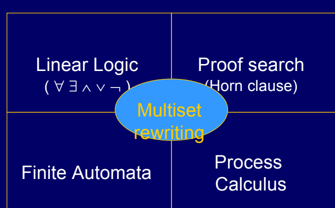
Part III

Multiset Rewriting Formulation

Analyzing Security Protocols

- Non-formal approaches (can be useful, but no tools…)
  - Some crypto-based proofs [Bellare, Rogaway]
- BAN and related logics
  - Axiomatic semantics of protocol steps
- Methods based on operational semantics
  - Intruder model derived from Dolev-Yao
  - Protocol gives rise to set of traces
  - Perfect encryption
    - Possible to include known algebraic properties

A notation for inf-state systems



- Define protocol, intruder in minimal framework

Protocol Modeling Decisions

- How powerful is the adversary?
  - Simple replay of previous messages
  - Decompose, reassemble and resend
  - Statistical analysis of network traffic
  - Timing attacks
- How much detail in underlying data types?
  - Plaintext, ciphertext and keys
    - atomic data or bit sequences
  - Encryption and hash functions
    - "perfect" cryptography
    - algebraic properties: encr(x*y) = encr(x) * encr(y) for RSA encrypt(k,msg) = msg^k mod N

Protocol Notation

- Non-deterministic infinite-state systems
- Facts
  - $F ::= P(t_1, \ldots, t_n)$
  - $t ::= x \mid c \mid f(t_1, \ldots, t_n)$
  - } Multi-sorted first-order atomic formulas
- States $\{ F_1, \ldots, F_n \}$
  - Multiset of facts
    - Includes network messages, private state
    - Intruder will see messages, not private state

State Transitions

- Transition
  - $F_1, \ldots, F_k \longrightarrow \exists x_1 \ldots \exists x_m.\ G_1, \ldots, G_n$
- What this means
  - If $F_1, \ldots, F_k$ in state σ, then a next state σ' has
    - Facts $F_1, \ldots, F_k$ removed
    - $G_1, \ldots, G_n$ added, with $x_1 \ldots x_m$ replaced by new symbols
    - Other facts in state σ carry over to σ'
  - Free variables in rule universally quantified
  - Pattern matching in $F_1, \ldots, F_k$ can invert functions
- Linear Logic: $F_1 \otimes \ldots \otimes F_k \multimap \exists x_1 \ldots \exists x_m (G_1 \otimes \ldots \otimes G_n)$

Simplified Needham-Schroeder

- Predicates
  - A_i, B_i, N_i
  - -- Alice, Bob, Network in state i
- Transitions
  - $\exists x.\ A_1(x)$
  - $A_1(x) \longrightarrow N_1(x),\ A_2(x)$
  - $N_1(x) \longrightarrow \exists y.\ B_1(x,y)$
  - $B_1(x,y) \longrightarrow N_2(x,y),\ B_2(x,y)$
  - $A_2(x),\ N_2(x,y) \longrightarrow A_3(x,y)$
  - $A_3(x,y) \longrightarrow N_3(y),\ A_4(x,y)$
  - $B_2(x,y),\ N_3(y) \longrightarrow B_3(x,y)$

$A \rightarrow B$: $\{n_a, A\}_{Kb}$
$B \rightarrow A$: $\{n_a, n_b\}_{Ka}$
$A \rightarrow B$: $\{n_b\}_{Kb}$

- Authentication
  - $A_4(x,y) \wedge B_3(x,y') \supset y=y'$

picture next slide

Sample Trace

$A \rightarrow B$: $\{n_a, A\}_{Kb}$
$B \rightarrow A$: $\{n_a, n_b\}_{Ka}$
$A \rightarrow B$: $\{n_b\}_{Kb}$

$\exists x.\ A_1(x)$

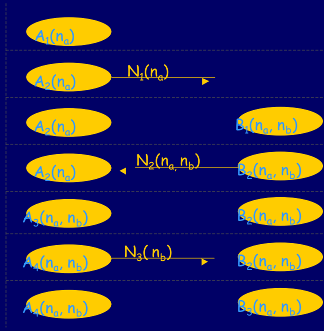
$A_1(x) \rightarrow A_2(x),\ N_1(x)$

$N_1(x) \rightarrow \exists y.\ B_1(x,y)$

$B_1(x,y) \rightarrow N_2(x,y),\ B_2(x,y)$

$A_2(x),\ N_2(x,y) \rightarrow A_3(x,y)$

$A_3(x,y) \rightarrow N_3(y),\ A_4(x,y)$



What does this accomplish?

- Represent protocols precisely
  - High-level program that defines how protocol agent responds to any message
- Represent intruder precisely
  - Capture Dolev-Yao model
- Define classes of protocols
  - Finite length, bounded message size, etc.
- Study upper, lower bounds on protocol security

Common Intruder Model

- Derived from Dolev-Yao model [NS 78, DY 89]
  - Adversary is nondeterministic process
  - Adversary can
    - Block network traffic
    - Read any message, decompose into parts
    - Decrypt if key is known to adversary
    - Insert new message from data it has observed
  - Adversary cannot
    - Gain partial knowledge
    - Guess part of a key
    - Perform statistical tests, …

Formalize Intruder Model

- Intercept, decompose and remember messages
  - $N_1(x) \longrightarrow M(x)$ $\quad N_2(x,y) \longrightarrow M(x),\ M(y)$
  - $N_3(x) \longrightarrow M(x)$
- Compose and send messages from "known" data
  - $M(x) \longrightarrow N_1(x),\ M(x)$
  - $M(x),\ M(y) \longrightarrow N_2(x,y),\ M(x),\ M(y)$
  - $M(x) \longrightarrow N_3(x),\ M(x)$
- Generate new data as needed
  - $\exists x.\ M(x)$

Highly nondeterministic, same for any protocol

Restricted class of protocols

- Finite number of roles (participant rules)
- Finite number of steps
  - Each participant does $\leq$n steps
- Bounded message size
  - Fixed number of fields in message
  - Fixed set of message constants
  - Fixed depth encryption (1 or 2 enough)
  - Nonces (but no only "create new", and =?)
- Everything fixed or constant, except nonces

Upper and lower bounds

[Durgin, Lincoln, Mitchell, Scedrov]

<table>
<tr><th></th><th></th><th>Bounded # of roles</th><th>Bounded # of ∃</th><th>Unbounded # of ∃</th></tr>
<tr><td rowspan="2">I with ∃</td><td>≠, =</td><td rowspan="4">NP – complete</td><td></td><td rowspan="4">Undecidable</td></tr>
<tr><td>= only</td><td rowspan="3">DExp – time</td></tr>
<tr><td rowspan="2">I w/o ∃</td><td>≠, =</td></tr>
<tr><td>= only</td></tr>
</table>

All: finite # of different roles, finite length roles, bounded message size

One Idea:

Attack requires exponential runs

A: Broadcast $\{0, 0, 0, 0\}_k$

B1: $\{x_0, x_1, x_2, 0\}_k \longrightarrow \{x_0, x_1, x_2, 1\}_k$
B2: $\{x_0, x_1, 0, 1\}_k \longrightarrow \{x_0, x_1, 1, 0\}_k$
B3: $\{x_0, 0, 1, 1\}_k \longrightarrow \{x_0, 1, 0, 0\}_k$
B4: $\{0, 1, 1, 1\}_k \longrightarrow \{1, 0, 0, 0\}_k$

C : $\{1, 1, 1, 1\}_k \longrightarrow$ Broadcast(k)

Protocol (without nonces) with n roles B1, …, Bn requires attack with 2^n steps

Part IV

Probability, Complexity, and Process Calculus

Limitations of Standard Model

- Can find some attacks
  - Successful analysis of industrial protocols
- Other attacks are outside model
  - Interaction between protocol and encryption
- Some protocols cannot be modeled
  - Probabilistic protocols
  - Steps that require specific property of encryption
- Possible to "OK" an erroneous protocol

Language Approach

[Abadi, Gordon]

- Write protocol in process calculus
- Express security using observational equivalence
  - Standard relation from programming language theory
    P ≈ Q iff for all contexts C[ ], same observations about C[P] and C[Q]
  - Context (environment) represents adversary
- Use proof rules for ≈ to prove security
  - Protocol is secure if no adversary can distinguish it from some idealized version of the protocol

Probabilistic Poly-time Analysis

[Lincoln, Mitchell, Mitchell, Scedrov]

- Adopt spi-calculus approach, add probability
- Probabilistic polynomial-time process calculus
  - Protocols use probabilistic primitives
    - Key generation, nonce, probabilistic encryption, ...
  - Adversary may be probabilistic
  - Modal type system guarantees complexity bounds
- Express protocol and specification in calculus
- Study security using observational equivalence
  - Use probabilistic form of process equivalence

Needham-Schroeder Private Key

- Analyze part of the protocol P
  - A → B: { i } $_K$
  - B → A: { f(i) } $_K$
- "Obviously" secret protocol Q (zero knowledge)
  - A → B: { random_number } $_K$
  - B → A: { random_number } $_K$
- Analysis: P ≈ Q reduces to crypto condition related to non-malleability [Dolev, Dwork, Naor]
  - Fails for RSA encryption, f(i) = 2i

Technical Challenges

- Language for prob. poly-time functions
  - Extend Hofmann language with rand
- Replace nondeterminism with probability
  - Otherwise adversary is too strong ...
- Define probabilistic equivalence
  - Related to poly-time statistical tests ...
- Develop specification by equivalence
  - Several examples carried out
- Proof systems for probabilistic equivalence
  - Work in progress

Basic example

- Sequence generated from random seed
  - P_n: let b = n^k-bit sequence generated from n random bits
    in PUBLIC ⟨b⟩ end
- Truly random sequence
  - Q_n: let b = sequence of n^k random bits
    in PUBLIC ⟨b⟩ end
- P is crypto strong pseudo-random generator
  - P ≈ Q
  - Equivalence is asymptotic in security parameter n

Compositionality

- Property of observational equiv

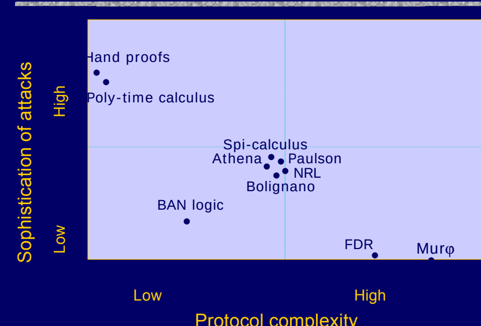
$$\frac{A \approx B \quad C \approx D}{A|C \approx B|D}$$

similarly for other process forms

Current State of Project

- New framework for protocol analysis
  - Determine crypto requirements of protocols !
  - Precise definition of crypto primitives
- Probabilistic ptime language
- Pi-calculus-like process framework
  - replaced nondeterminism with rand
  - equivalence based on ptime statistical tests
- Proof methods for establishing equivalence
- Future: tool development

Formal Analysis Techniques



Conclusion

- Computer security is fun
  - Lots of technical problems
  - High cocktail-party quotient
- Programming language methods can work
  - Model systems and attackers
  - Define and analyze security properties
  - Methods for verifying security
  - Increase sophistication of security research
    - Resolve issues like compositionality problem