# Security Analysis of Network Protocols

John Mitchell

Reference: http://www.stanford.edu/class/cs259/

# Course organization

- Lectures
  - Tues, Thurs for approx first six weeks of quarter
  - Project presentations in 3 stages
- This is a project course
  - There will be one or two short homeworks
  - Most of your work will be project and presentation
  - Typically done in teams of 2

Please enroll if you are here!

# SCPD Students

- Everything you need will be on the class website
- Project presentations
  - If you are in town, come and present
  - If you are elsewhere, we will work something out
    - Web-based presentation software
    - Recorded video
    - Send us info and we will present
  - Plan: last two weeks of course

# Today

- Basics of formal analysis of security protocols
  - What is protocol analysis?
  - Needham Schroeder and the Murφ model checker
- CS259 Website
  - Tools
  - Past Projects, Project Suggestions
- HW#1 will be out Thursday, due 24th Jan
  - Take example Murφ model and modify it
  - Find project partner (including if you are SCPD)

# Computer Security

- Cryptography
  - Encryption, signatures, cryptographic hash, ...
- Security mechanisms
  - Access control policy
  - Network protocols
- Implementation
  - Cryptographic library
  - Code implementing mechanisms
    - Reference monitor and TCB
    - Protocol
  - Runs under OS, uses program library, network protocol stack

Analyze protocols, assuming crypto, implementation, OS correct

# Cryptographic Protocols

- Two or more parties
- Communication over insecure network
- Cryptography used to achieve goal
  - Exchange secret keys
  - Verify identity (authentication)

Crypto (class poll):

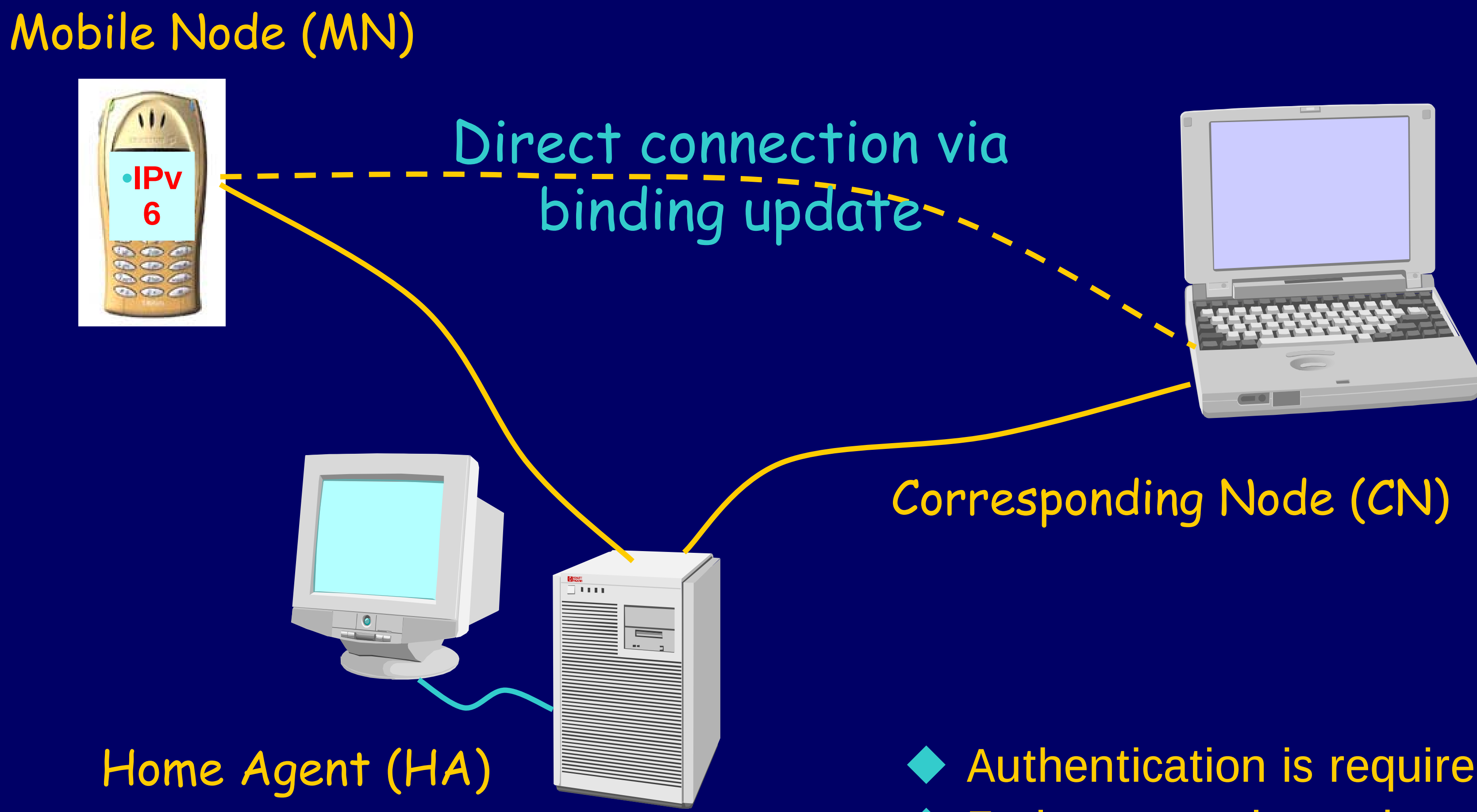
Public-key encryption, symmetric-key encryption, CBC, hash, signature, key generation, random-number generators

# Many Protocols

- Authentication
  - Kerberos
- Key Exchange
  - SSL/TLS handshake, IKE, JFK, IKEv2,
- Wireless and mobile computing
  - Mobile IP, WEP, 802.11i
- Electronic commerce
  - Contract signing, SET, electronic cash,

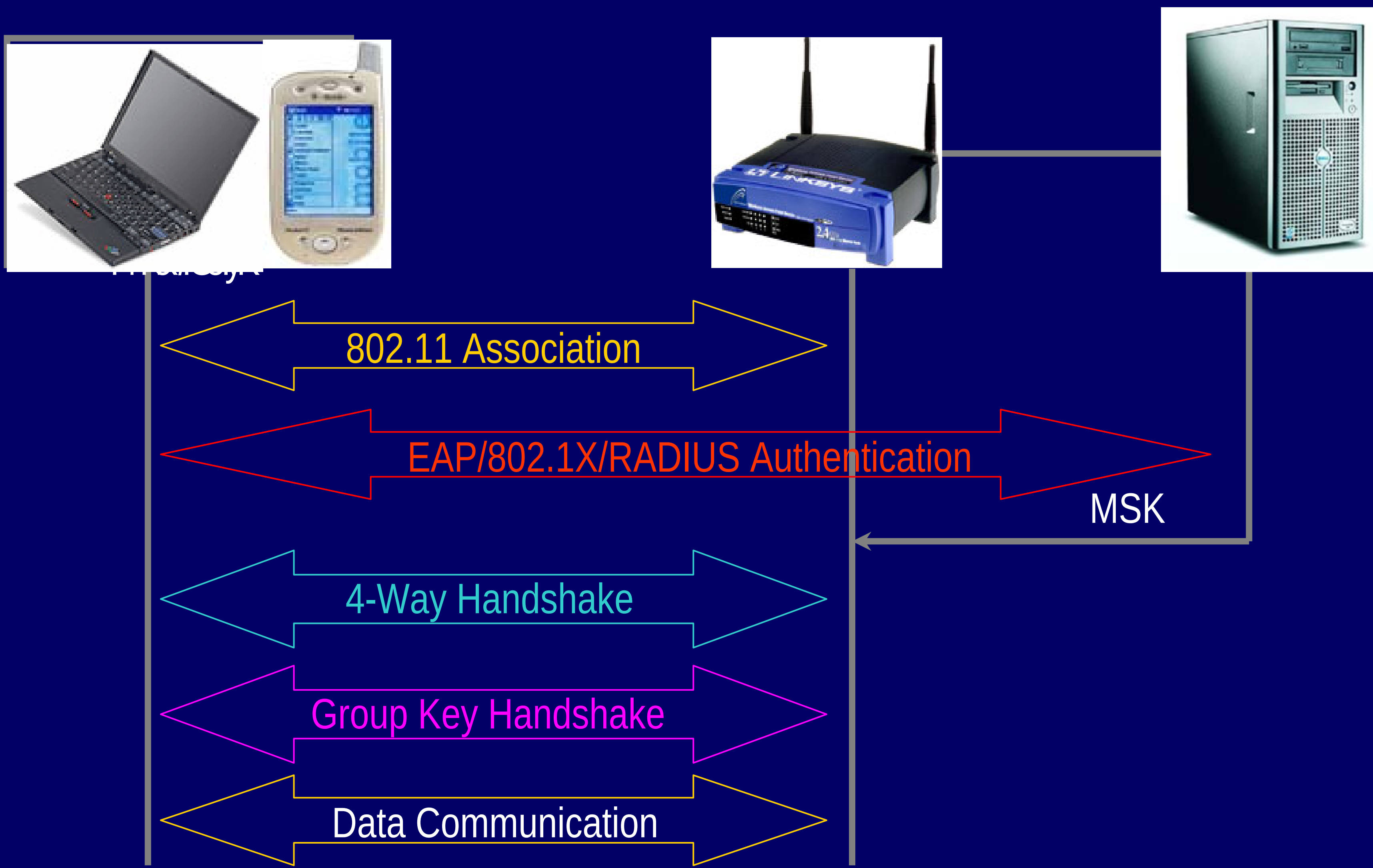
See http://www.lsv.ens-cachan.fr/spore/, http://www.avispa-project.org/library

# Mobile IPv6 Architecture

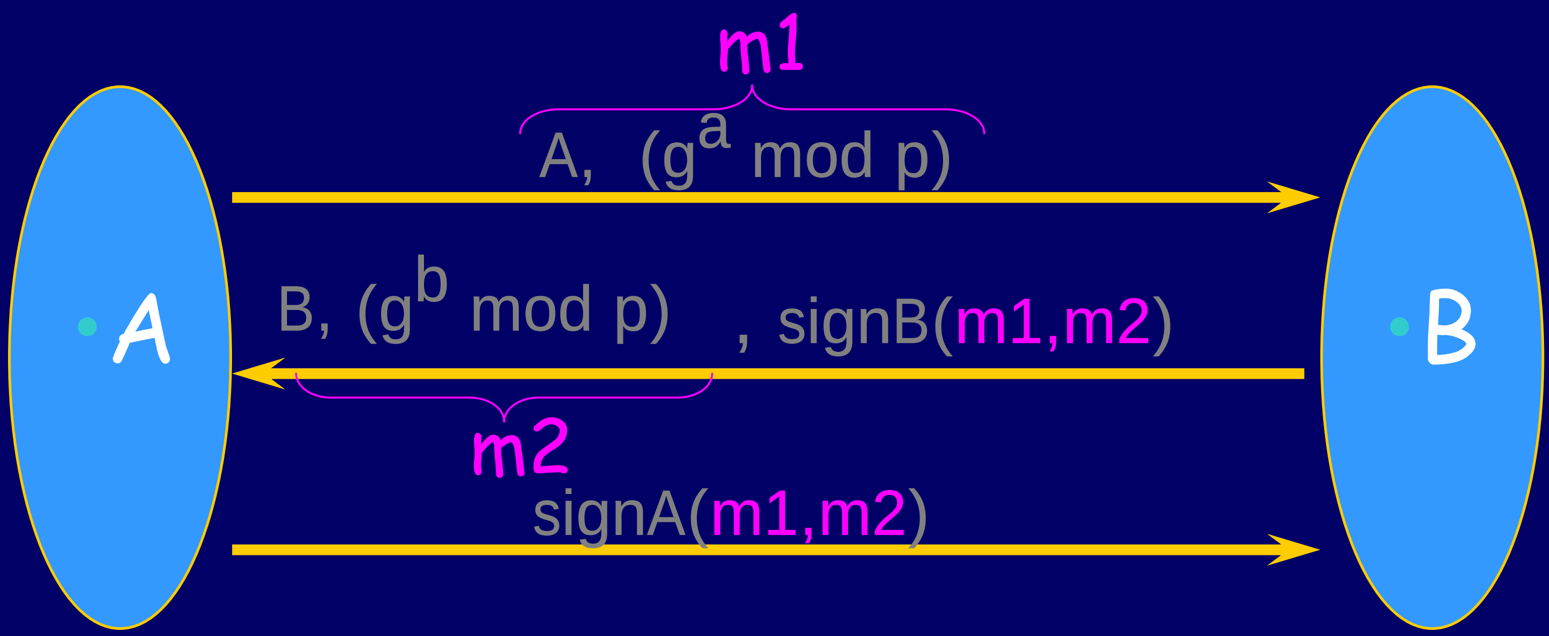



- Authentication is required
- Early proposals weak

# 802.11i Wireless Authentication



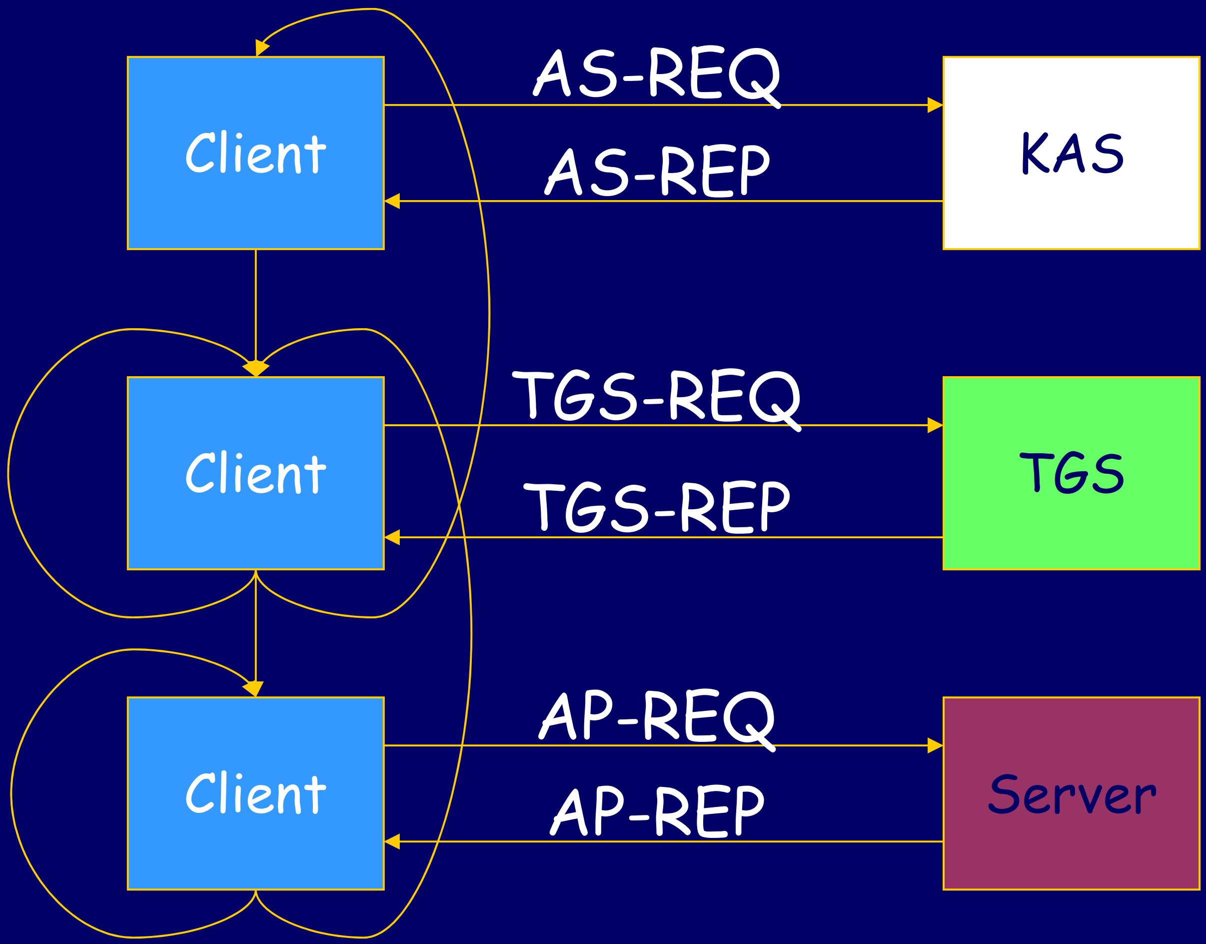
# IKE subprotocol from IPSEC




Result: A and B share secret $g^{ab} \bmod p$

Analysis involves probability, modular exponentiation, complexity, digital signatures, communication networks

# Kerberos Protocol




Used in Stanford WebAuth

# Correctness vs Security

- Program or System Correctness
  - Program satisfies specification
    - For reasonable input, get reasonable output
- Program or System Security
  - Program properties preserved in face of attack
    - For unreasonable input, output is not completely disastrous
- Main differences
  - Active interference from adversary
  - Refinement techniques may fail
    - More functionality can be *worse*

# Protocol Attacks

- Kerberos [Scederov et. Al.]
  - Public key version - lack of identity in message causes authentication failure
- WLAN 802.11i [He , Mitchell]
  - Lack of authentication in msg causes dos vulnerability
  - Proved correct using PCL [ Datta , Derek, Sundararajan]
- GDOI [meadows – Pavlovic]
  - Authorization failure
- SSL [Mitchell – Shmatikov]
  - Version roll-back attack, authenticator confusion between main and resumption protocol
- Needham-Schroeder [Lowe]
  - We will look at this today

# Security Analysis

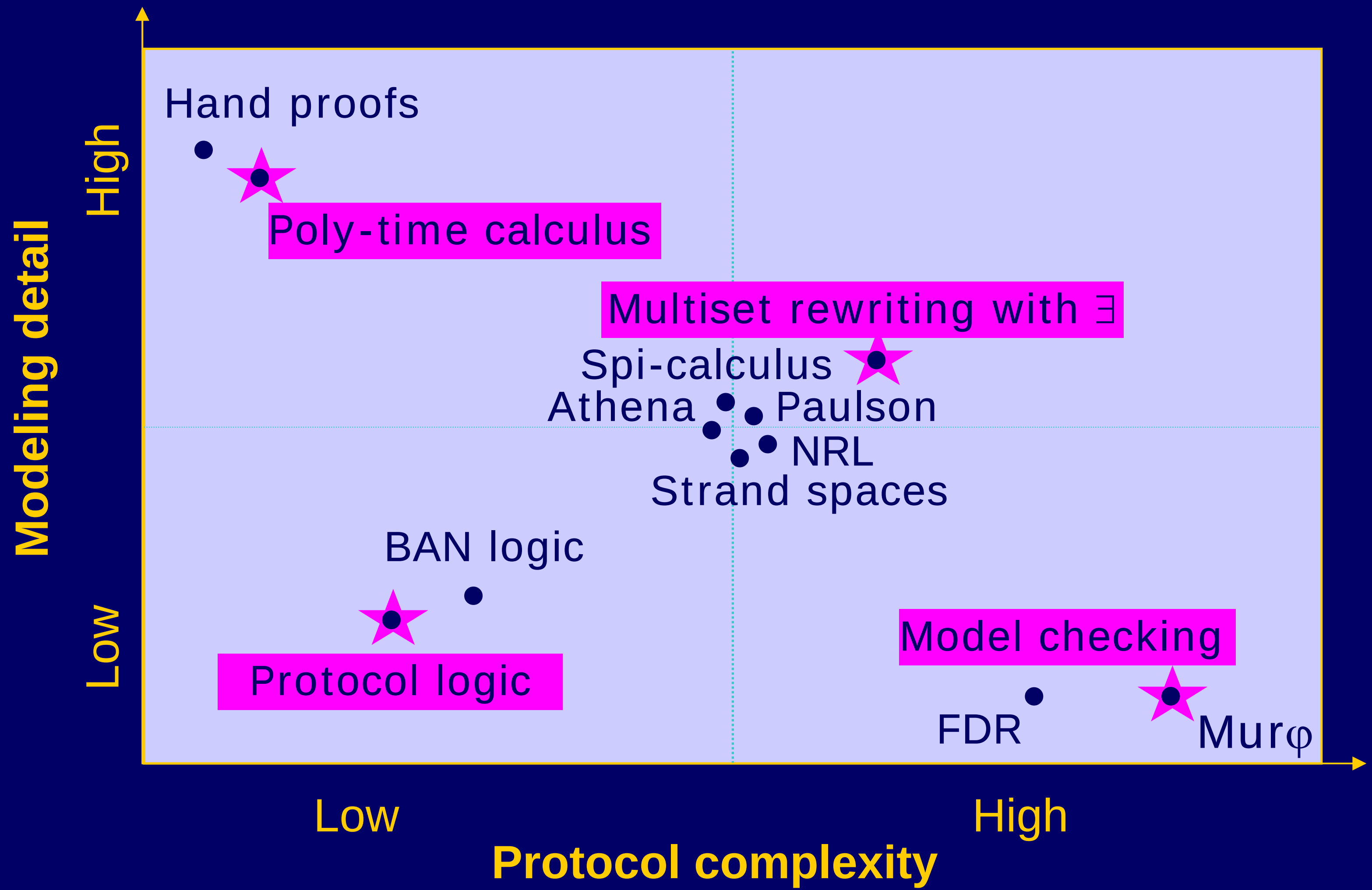
- Model system
- Model adversary
- Identify security properties
- See if properties are preserved under attack

- Basic concept
  - No “absolute security”
  - Security means: under given assumptions about system, no attack of a certain form will destroy specified properties.

# Important Modeling Decisions

- How powerful is the adversary?
  - Simple replay of previous messages
  - Block messages; Decompose, reassemble and resend
  - Statistical analysis, partial info from 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 analysis spectrum



# Four "Stanford" approaches

SRI, U Penn,
U Texas, Kiel,
INRIA, ...

- Finite-state analysis
  - Case studies: find errors, debug specifications
- Symbolic execution model: Multiset rewriting
  - Identify basic assumptions
  - Study optimizations, prove correctness
  - Complexity results
- Process calculus with probability and complexity
  - More realistic intruder model
  - Interaction between protocol and cryptography
  - Equational specification and reasoning methods
- Protocol logic
  - Axiomatic system for modular proofs of protocol properties

# Some other projects and tools

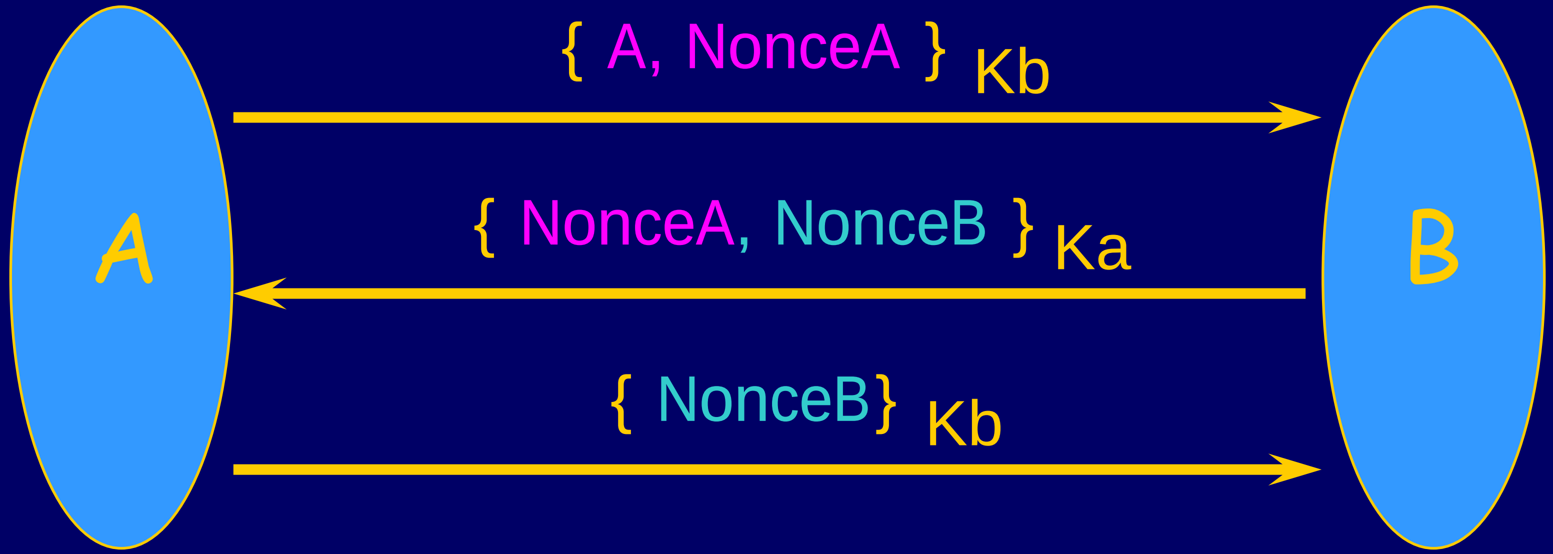
- Exhaustive finite-state analysis
  - FDR, based on CSP [Lowe, Roscoe, Schneider, ...]
- Search using symbolic representation of states
  - Meadows: NRL Analyzer, Millen: Interrogator
- Prove protocol correct
  - Paulson's "Inductive method", others in HOL, PVS, ...
  - MITRE -- Strand spaces
  - Process calculus approach: Abadi-Gordon spi-calculus, applied pi-calculus, ...
  - Type-checking method: Gordon and Jeffreys, ...

Many more – this is just a small sample

# Example: Needham-Schroeder

- Famous simple example
  - Protocol published and known for 10 years
  - Gavin Lowe discovered unintended property while preparing formal analysis using FDR system
- Background: Public-key cryptography
  - Every agent A has
    - Public encryption key Ka
    - Private decryption key $Ka^{-1}$
  - Main properties
    - Everyone can encrypt message to A
    - Only A can decrypt these messages

# Needham-Schroeder Key Exchange




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

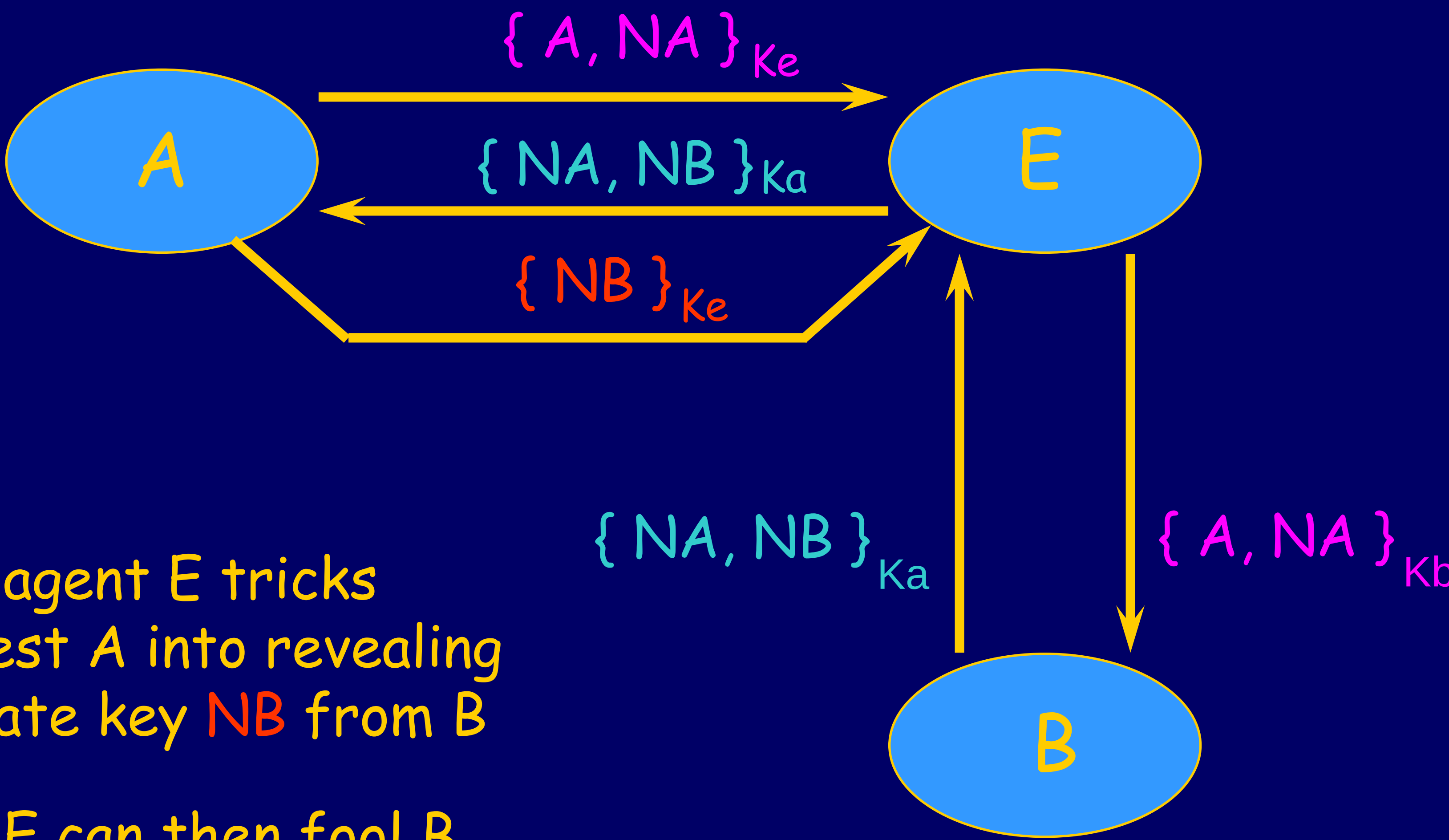
# Needham Schroeder properties

- **Responder correctly authenticated**
  - If initiator A completes the protocol, believes Honest B is responder, then B must think he responded to A.
- **Initiator correctly authenticated**
  - If responder B completes the protocol, believes Honest A was initiator, then A must thinks she initiated the protocol with B.
- **Nonce secrecy**
  - When honest initiator completes the protocol with honest peer, attacker does not know either nonce.

Honest: follows steps of the protocol (only)

[Lowe]

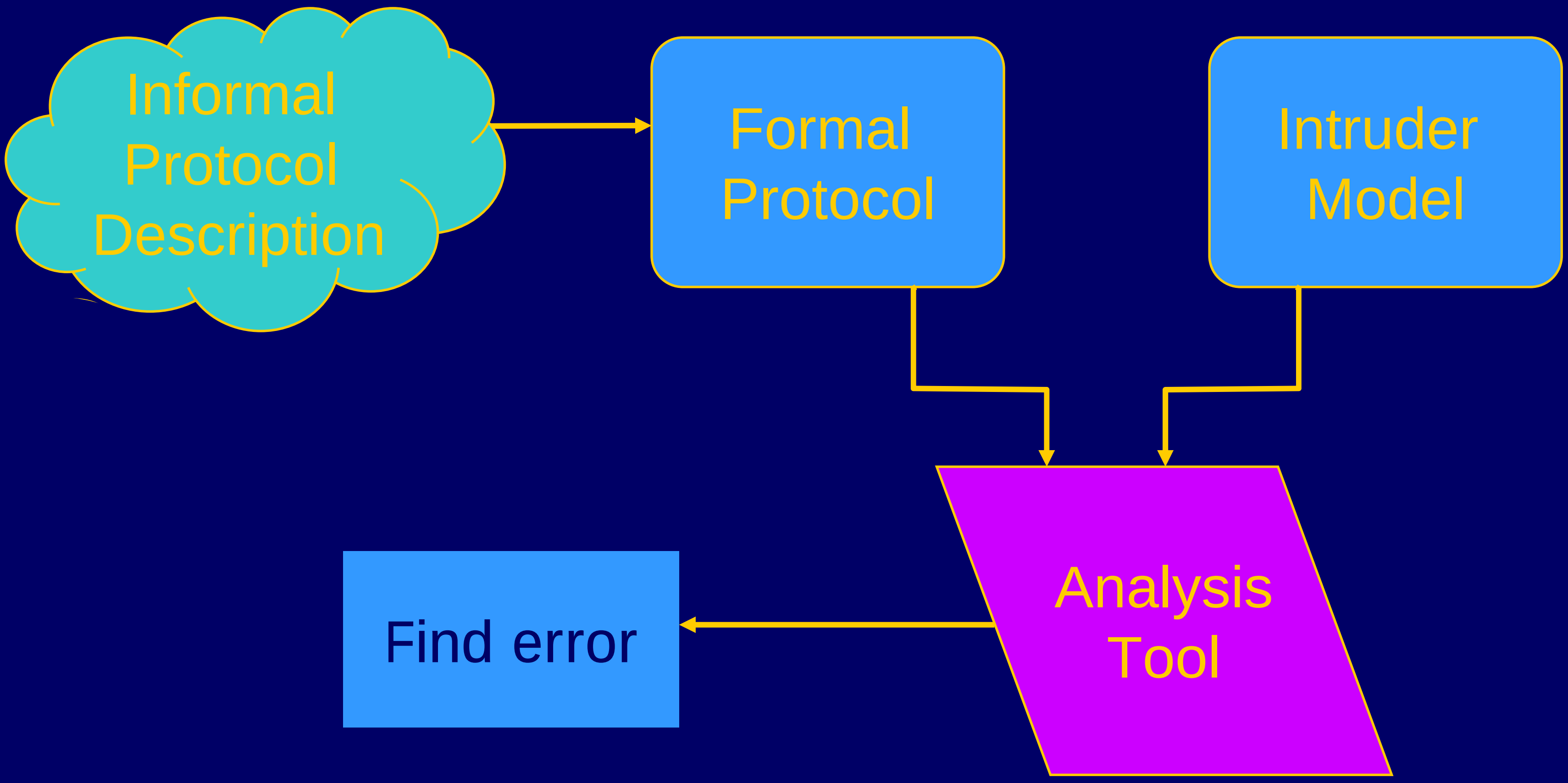
# Anomaly in Needham-Schroeder



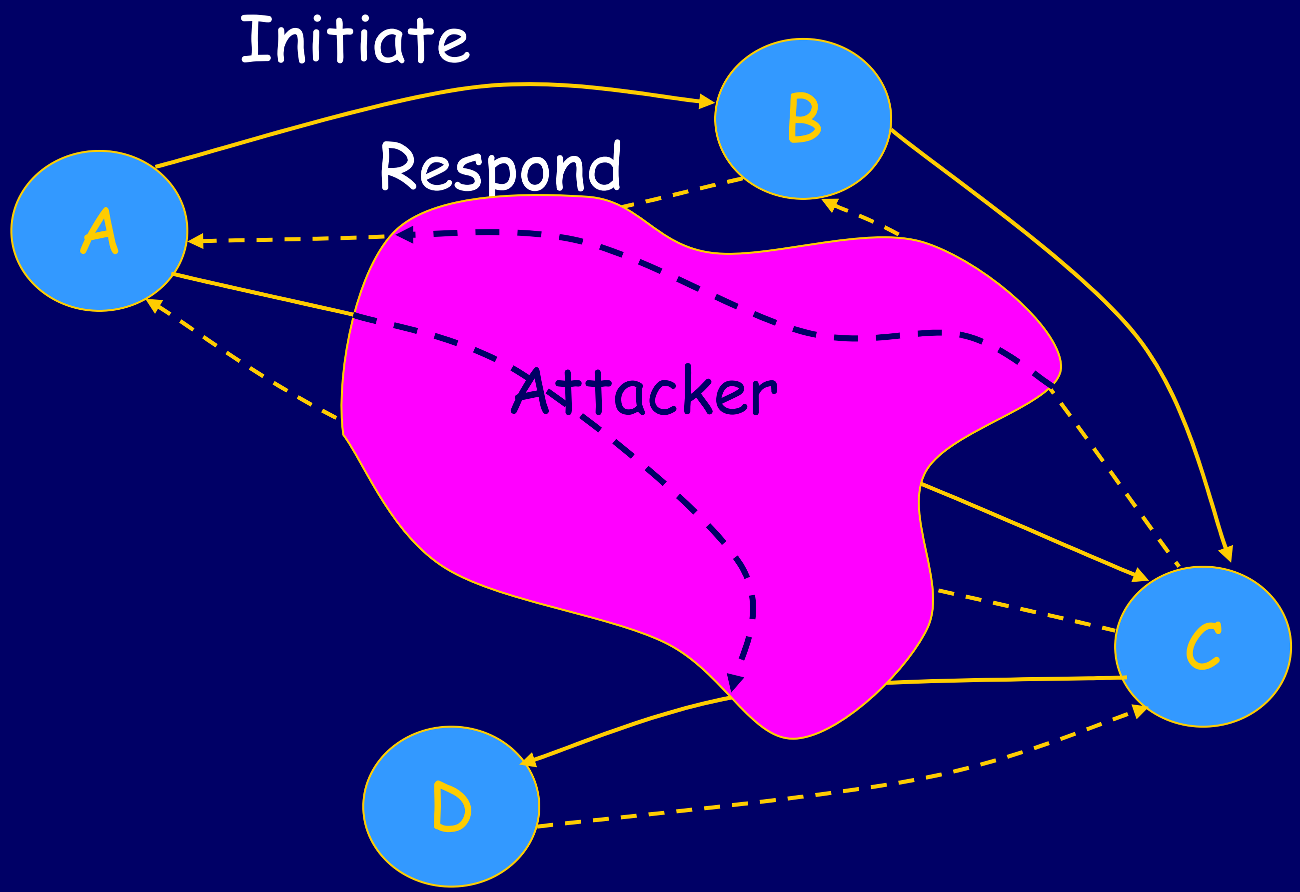

Evil agent E tricks honest A into revealing private key NB from B

Evil E can then fool B

# Explicit Intruder Method

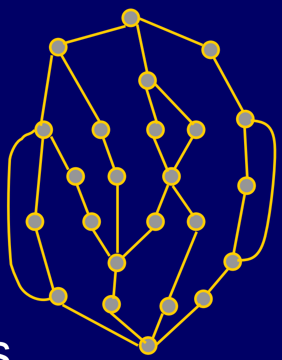


# Run of protocol




Correct if no security violation in any run

# Automated Finite-State Analysis



- Define finite-state system
  - Bound on number of steps
  - Finite number of participants
  - Nondeterministic adversary with finite options
- Pose correctness condition
  - Can be simple: authentication and secrecy
  - Can be complex: contract signing
- Exhaustive search using “verification” tool
  - Error in finite approximation $\Rightarrow$ Error in protocol
  - No error in finite approximation $\Rightarrow$ ???

# Finite-state methods

- Two sources of infinite behavior
  - Many instances of participants, multiple runs
  - Message space or data space may be infinite
- Finite approximation
  - Assume finite participants
    - Example: 2 clients, 2 servers
  - Assume finite message space
    - Represent random numbers by r1, r2, r3, ...
    - Do not allow unbounded encrypt(encrypt(encrypt(...)))

# Murφ [Dill et al.]

- Describe finite-state system
  - State variables with initial values
  - Transition rules
  - Communication by shared variables
- Scalable: choose system size parameters
- Automatic exhaustive state enumeration
  - Space limit: hash table to avoid repeating states
- Research and industrial protocol verification

# Applying Murφ to security protocols

- Formulate protocol
- Add adversary
  - Control over "network"  (shared variables)
  - Possible actions
    - Intercept any message
    - Remember parts of messages
    - Generate new messages, using observed data and initial knowledge (e.g. public keys)

# Needham-Schroeder in Murφ (1)

```
const
  NumInitiators:  1;   -- number of initiators
  NumResponders:  1;   -- number of responders
  NumIntruders:   1;   -- number of intruders
  NetworkSize:    1;   -- max. outstanding msgs in network
  MaxKnowledge:  10;   -- number msgs intruder can remember

type
  InitiatorId:  scalarset (NumInitiators);
  ResponderId:  scalarset (NumResponders);
  IntruderId:   scalarset (NumIntruders);

  AgentId:   union {InitiatorId, ResponderId, IntruderId};
```

# N-S message format in Murφ

```
MessageType : enum {          -- types of messages
  M_NonceAddress,             --  {Na, A}Kb  nonce and addr
  M_NonceNonce,               --  {Na,Nb}Ka  two nonces
  M_Nonce                     --  {Nb}Kb     one nonce
};

Message : record
    source:   AgentId;         -- source of message
    dest:     AgentId;         -- intended destination of msg
    key:      AgentId;         -- key used for encryption
    mType:    MessageType;     -- type of message
    nonce1:   AgentId;         -- nonce1
    nonce2:   AgentId;         -- nonce2 OR sender id OR empty
end;
```

# N-S protocol action in Murφ

```
ruleset i: InitiatorId do
  ruleset j: AgentId do
    rule "initiator starts protocol"
      ini[i].state = I_SLEEP &
          multisetcount (l:net, true) < NetworkSize ==>
    var
      outM: Message;    -- outgoing message
    begin
      undefine outM;
      outM.source  := i; outM.dest     := j;
      outM.key     := j; outM.mType    := M_NonceAddress;
      outM.nonce1  := i; outM.nonce2   := i;
      multisetadd (outM,net); ini[i].state :=I_WAIT;
      ini[i].responder := j;
    end; end; end;
```

# Adversary Model

- Formalize “knowledge”
  - initial data
  - observed message fields
  - results of simple computations
- Optimization
  - only generate messages that others read
  - time-consuming to hand simplify
- Possibility: automatic generation

# N-S attacker action in Murφ

```
-- intruder i sends recorded message
ruleset i: IntruderId do          -- arbitrary choice of
  choose j: int[i].messages do        --  recorded message
    ruleset k: AgentId do             --  destination
      rule "intruder sends recorded message"
        !ismember(k, IntruderId) &    -- not to intruders
        multisetcount (l:net, true) < NetworkSize
      ==>
       var  outM: Message;
       begin
          outM        := int[i].messages[j];
          outM.source := i;
          outM.dest   := k;
          multisetadd (outM,net);
end; end; end; end;
```

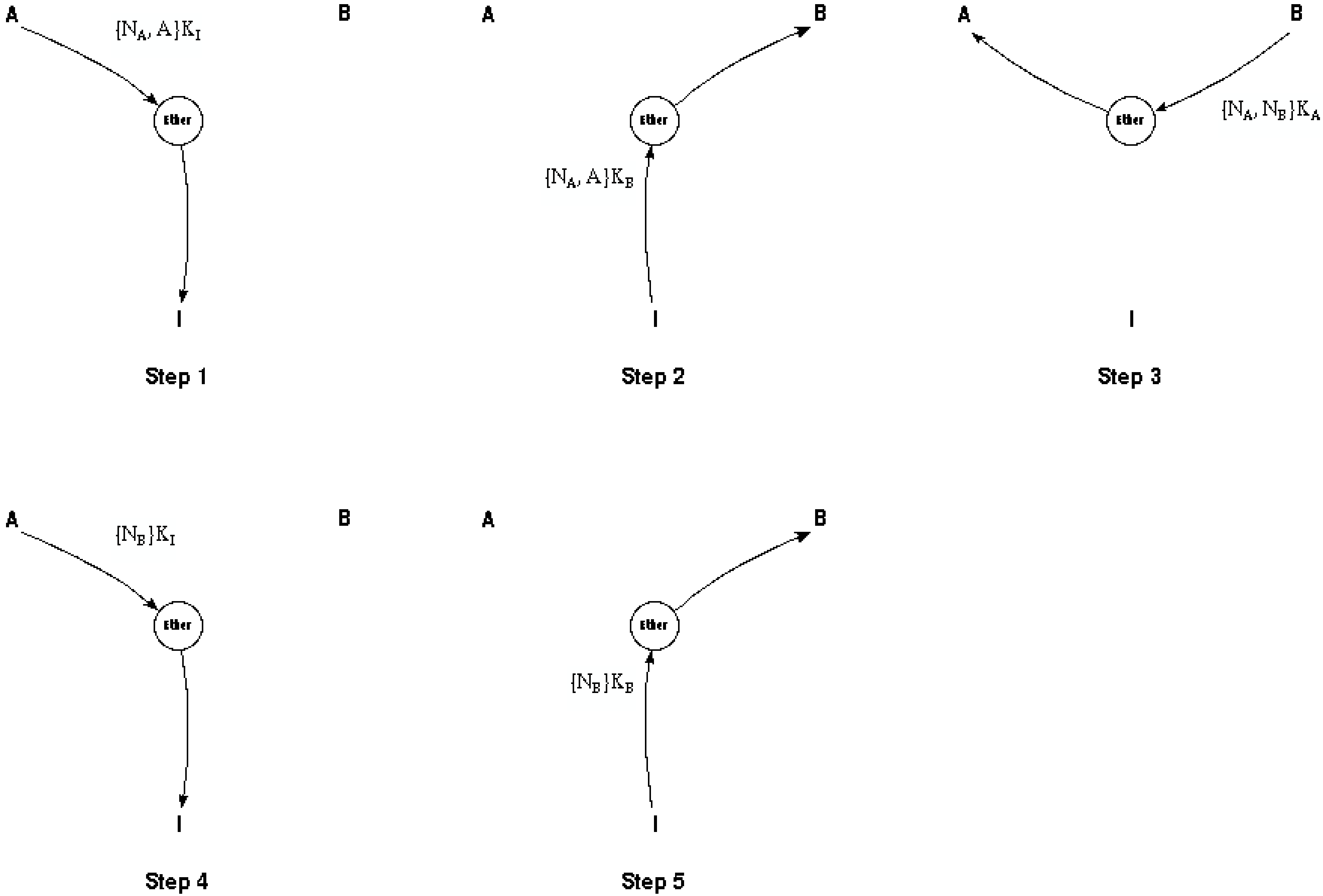
# Modeling Properties

```
invariant "responder correctly authenticated"
  forall i: InitiatorId do
    ini[i].state = I_COMMIT &
    ismember(ini[i].responder, ResponderId)
    ->
    res[ini[i].responder].initiator = i &
    ( res[ini[i].responder].state = R_WAIT |
      res[ini[i].responder].state = R_COMMIT )
  end;
```

# Run of Needham-Schroeder

- Find error after 1.7 seconds exploration
- Output: trace leading to error state
- Murφ times after correcting error:

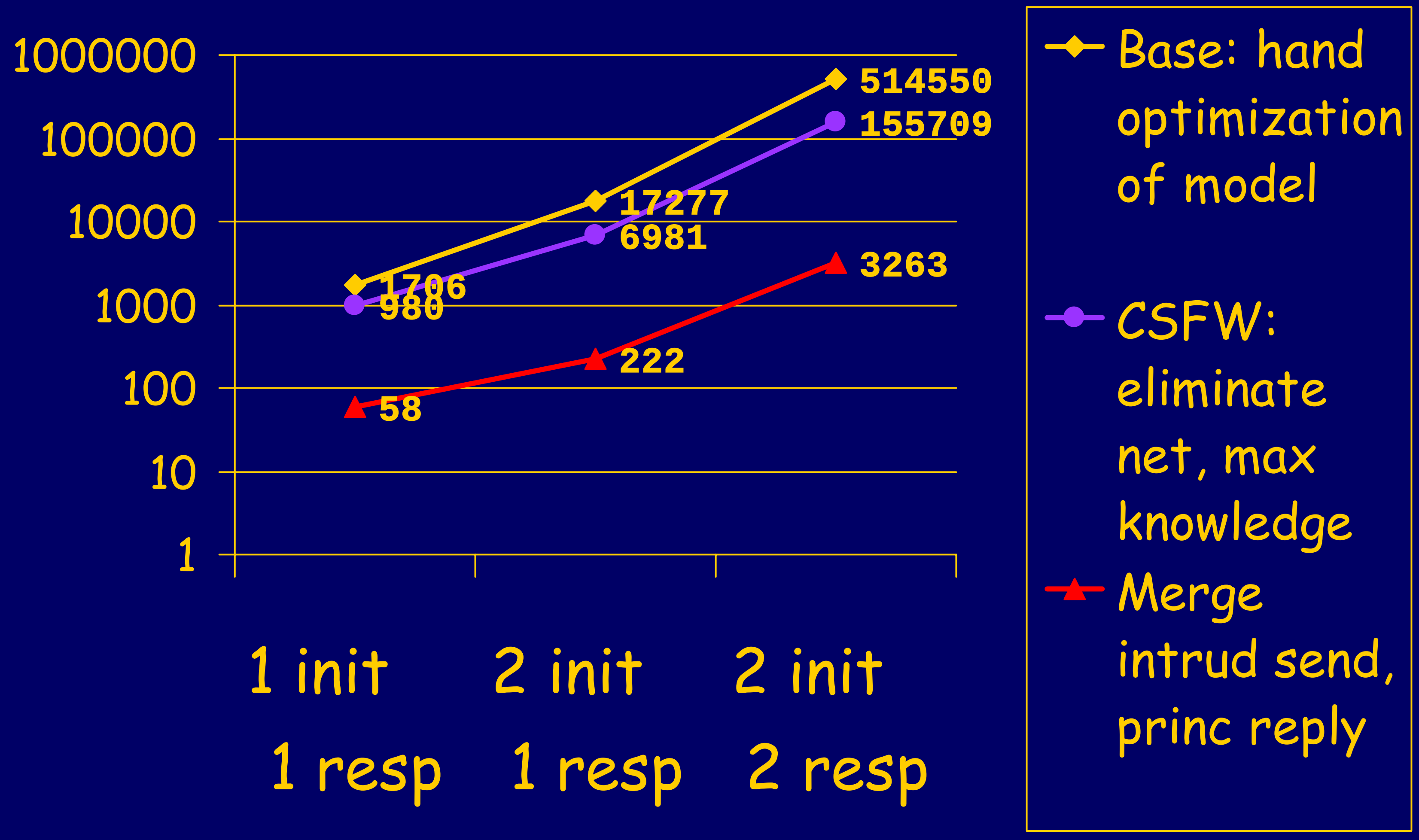
| number of | | | size of | | |
|---|---|---|---|---|---|
| ini. | res. | int. | network | states | time |
| 1 | 1 | 1 | 1 | 1706 | 3.1s |
| 1 | 1 | 1 | 2 | 40207 | 82.2s |
| 2 | 1 | 1 | 1 | 17277 | 43.1s |
| 2 | 2 | 1 | 1 | 514550 | 5761.1s |


A
{N_A, A}K_I
B
Ether
I
Step 1
A
B
Ether
{N_A, A}K_B
I
Step 2
A
B
Ether
{N_A, N_B}K_A
I
Step 3
A
{N_B}K_I
B
Ether
I
Step 4
A
B
Ether
{N_B}K_B
I
Step 5

# Limitations

- System size with current methods
  - 2-6 participants
    - Kerberos: 2 clients, 2 servers, 1 KDC, 1 TGS
  - 3-6 steps in protocol
  - May need to optimize adversary
- Adversary model
  - Cannot model randomized attack
  - Do not model adversary running time

# State Reduction on N-S Protocol



# Plan for this course

- Protocols
  - Authentication, key establishment, assembling protocols together, fair exchange, wireless ...
- Tools
  - Finite-state and probabilistic model checking, constraint-solving, process calculus, temporal logic, proof systems, game theory, poly-time computability...
- Projects (You do this later on your own!)
  - Choose a protocol or other security mechanism
  - Choose a tool or method and carry out analysis
  - Hard part: formulating security requirements

# CS259 Term Projects - 2006

| | | |
|---|---|---|
| Security Analysis of OTRv2 | *Formalization of HIPAA* | Security analysis of SIP |
| *Onion Routing* | *Analysis of ZRTP* | MOBIKE - IKEv2 Mobility and Multihoming Protocol |
| *802.16e Multicast-Broadcast Key Distribution Protocols* | *Short-Password Key Exchange Protocol* | *Analysis of the IEEE 802.16e 3-way handshake* |
| *Analysis of Octopus and Related Protocols* | | |

http://www.stanford.edu/class/cs259/

# CS259 Term Projects - 2004

| | | |
|---|---|---|
| *iKP protocol family* | *Electronic voting* | *XML Security* |
| *IEEE 802.11i wireless handshake protocol* | *Onion Routing* | *Electronic Voting* |
| *Secure Ad-Hoc Distance Vector Routing* | *An Anonymous Fair Exchange E-commerce Protocol* | *Key Infrastructure* |
| *Secure Internet Live Conferencing* | *Windows file-sharing protocols* | |

http://www.stanford.edu/class/cs259/

# Reference Material (CS259 web site)

- Protocols
  - Clarke-Jacob survey
  - Use Google; learn to read an RFC
- Tools
  - Murphi
    - Finite-state tool developed by David Dill's group at Stanford
  - PRISM
    - Probabilistic model checker, University of Birmingham
  - MOCHA
    - Alur and Henzinger; now consortium
  - Constraint solver using prolog
    - Shmatikov and Millen
  - Isabelle
    - Theorem prover developed by Larry Paulson in Cambridge, UK
    - A number of case studies available on line
- Will consider additional systems, tools (e.g. Prolog)