Analysis of Security Protocols

John C. Mitchell
Stanford University

Outline

- Introduction to protocols and security
  - Security vs correctness
  - Examples and general concepts
- Standard approach: analyze protocol+intruder
  - Unifying protocol formalism
  - Communication and intruder model
- Specific projects based on standard approach
  - Theorem-proving, state enumeration methods, ...
- Mention some of own projects, results

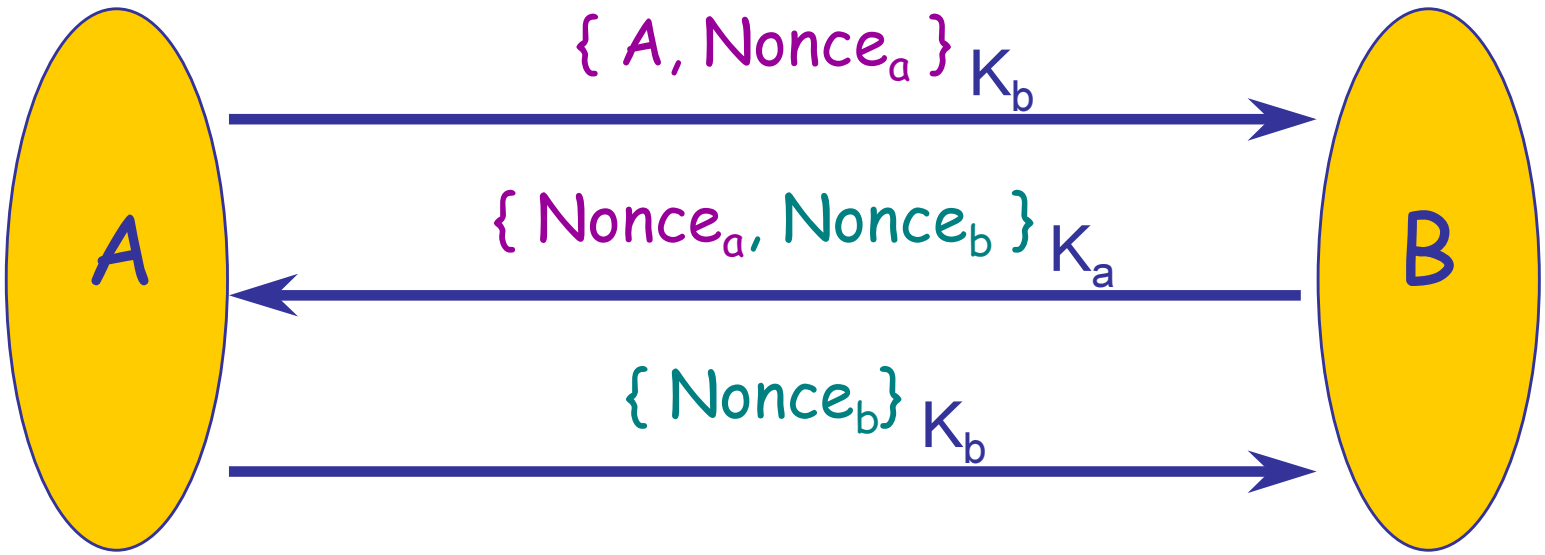
Correctness vs Security

- Program or System Correctness
  - Program satisfies specification
    - For reasonable input, get reasonable output
- Program or System Security
  - Program resists attack
    - For unreasonable input, output not completely disastrous
- Main differences
  - Active interference from environment
  - Refinement techniques may fail

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

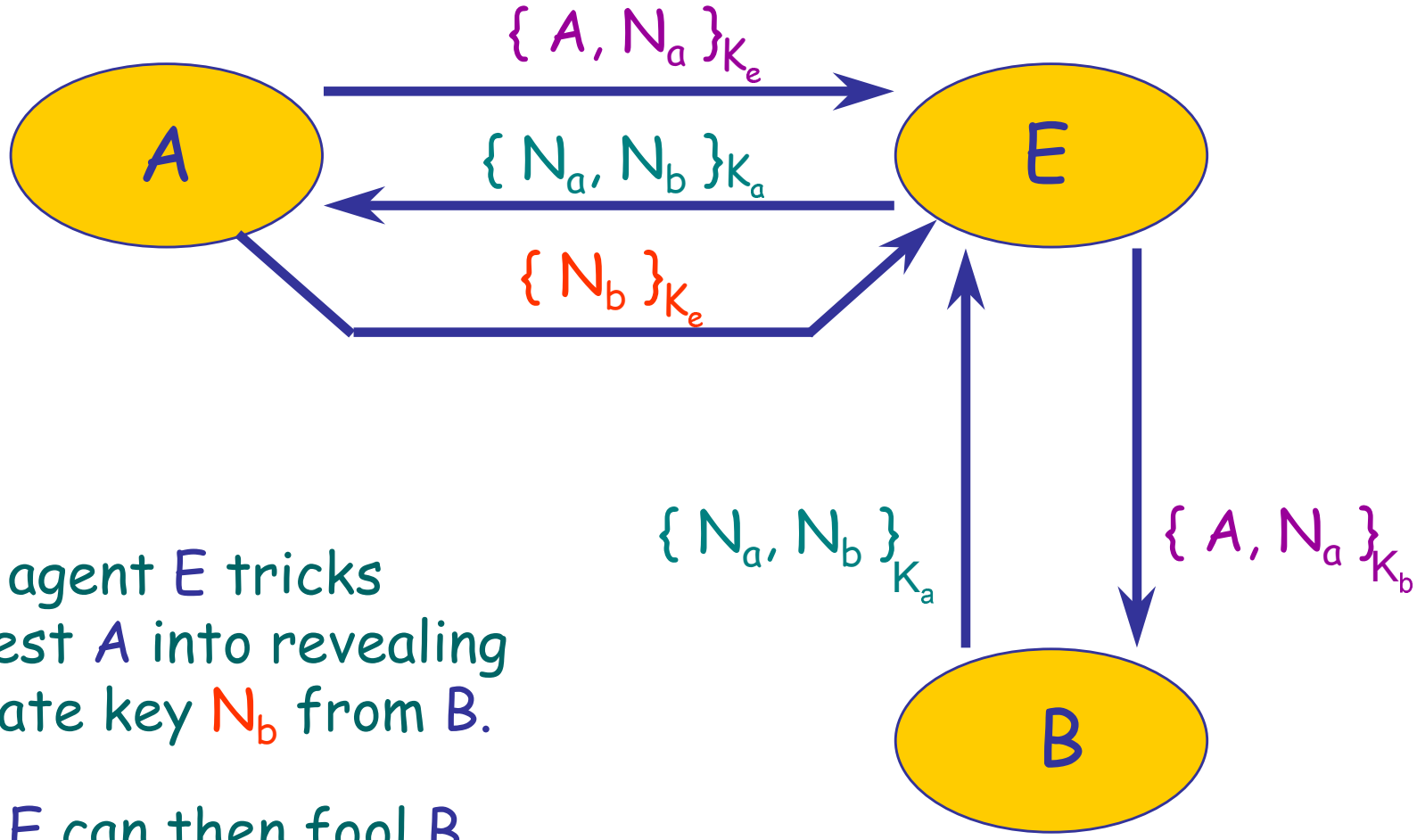
Needham-Schroeder Key Exchange



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.

Outline

- Introduction to protocols and security
  - Security vs correctness
  - Examples and general concepts
- ➡ Standard approach: analyze protocol+intruder
  - Unifying protocol formalism
  - Communication and intruder model
- Specific projects based on standard approach
  - Theorem-proving, state enumeration methods, ...
- Mention some of own projects, results

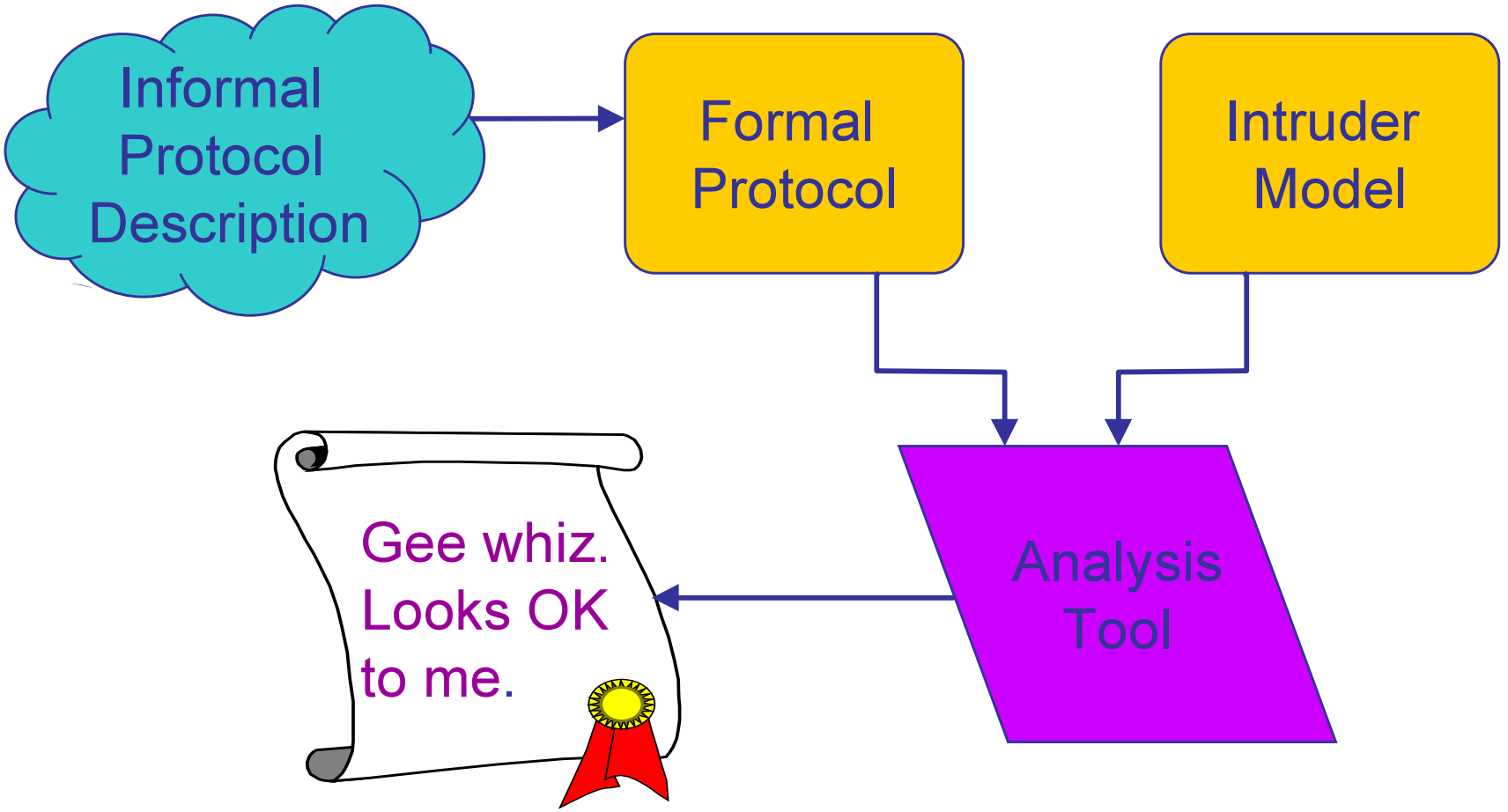
Analyzing Security Protocols

- 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

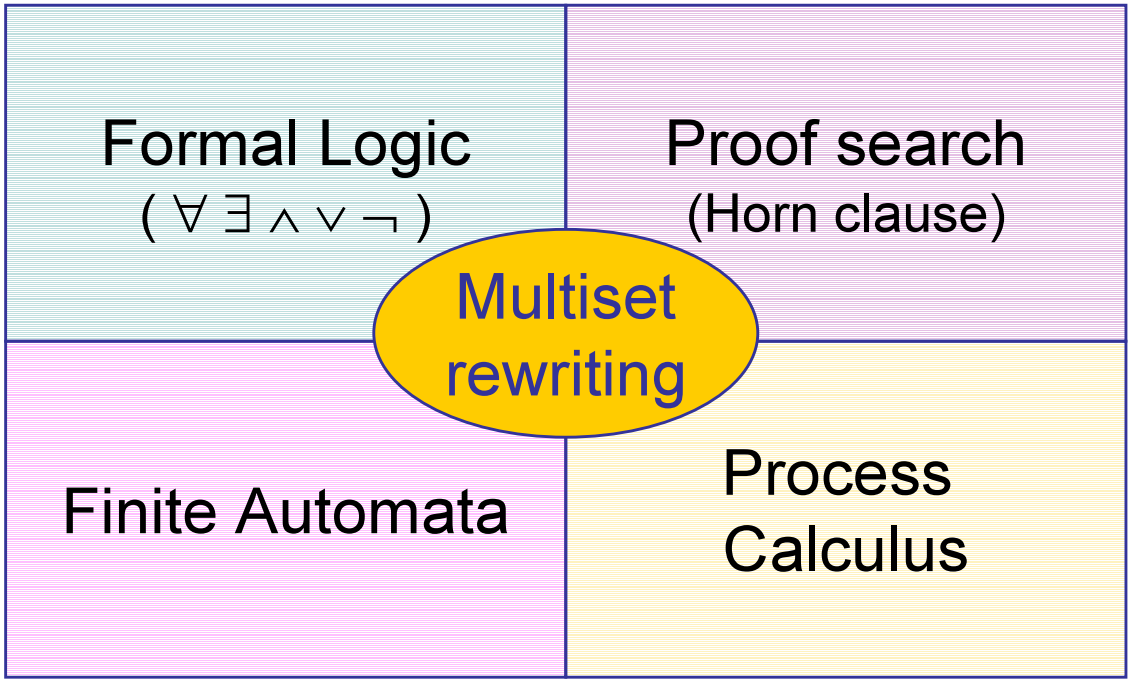
Example projects and tools

- Prove protocol correct
  - Paulson's "Inductive method", others in HOL, PVS, etc.
  - Bolignano -- Abstraction methods
  - MITRE -- Strand spaces
  - Process calculus approach: Abadi-Gordon spi-calculus
- Search using symbolic representation of states
  - Meadows: NRL Analyzer, Millen: Interrogator
- Exhaustive finite-state analysis
  - FDR, based on CSP [Lowe, Roscoe, Schneider, …]
  - Murφ -- specialized input language
  - Clarke et al. -- state search with axiomatic intruder model

Explicit Intruder Method



A notation for inf-state systems



- Define protocol, intruder in minimal framework
- Disadvantage: need to introduce new notation

Protocol Notation

- Non-deterministic *infinite*-state systems
- Facts

  $F ::= P(t_1, \ldots, t_n)$

  $t ::= x \mid c \mid f(t_1, \ldots, t_n)$

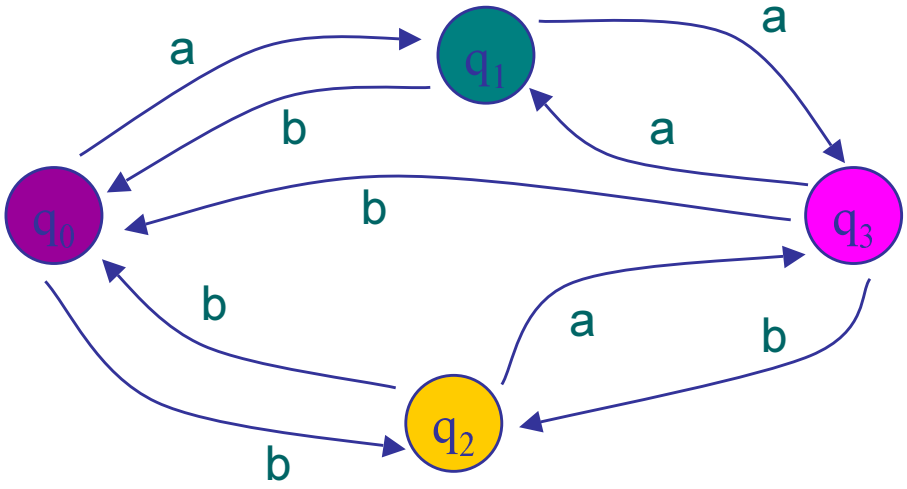
  } Multi-sorted first-order atomic formulas

- 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

Finite-State Example



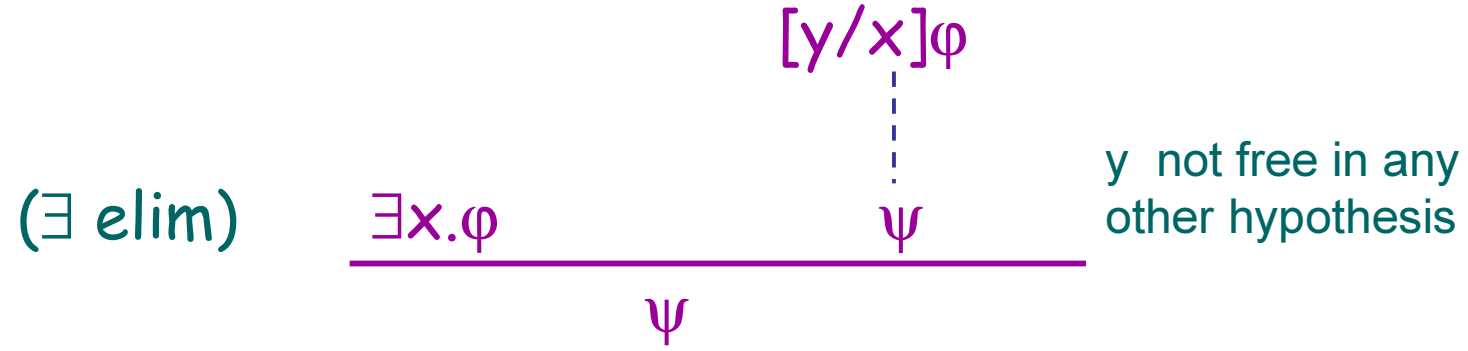
- Predicates: State, Input
- Function: •
- Constants: q_0, q_1, q_2, q_3, a, b, nil
- Transitions: State(q_0), Input(a • x) $\rightarrow$ State(q_1), Input(x)

  State(q_0), Input(b • x) $\rightarrow$ State(q_2), Input(x)

  …

Existential Quantification

- Natural-deduction proof rule



- Summary: for proof from ∃x.φ, choose new symbol and proceed from [y/x]φ

Turing Machine

- Predicates
  - Current(state,cell) -- current state, tape pos.
  - Contents(cell, symbol) -- contents of tape cell
  - Adjacent(cell, cell) -- keep cells in order
- Constants
  - $q_0, q_1, q_2, \ldots$ -- finite set of states
  - c_0, c_{eot} -- initial tape cells
  - "0", "1", "b" -- tape symbols

Turing Machine (II)

- Transitions

infinite linear tape

- Adjacent(c_0, c_{eot})
- Adjacent(c, c_{eot}) $\longrightarrow$ $\exists$c'. Adjacent(c,c'), Adjacent(c',c_{eot})

sample move right

- Current(q_i,c), Contents(c,"0"), Adjacent(c,c')

$\longrightarrow$ Current(q_k,c'), Contents(c,"1"), Adjacent(c,c')

sample move left

- Current(q_i,c), Contents(c,"1"), Adjacent(c',c)

$\longrightarrow$ Current(q_k,c'),Contents(c,"0"), Adjacent(c,c')

Simplified Needham-Schroeder

- Predicates

  A_i, B_i, N_i

  -- Alice, Bob, Network in state i

- Transitions

  $\exists x.\ A_1(x)$

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

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

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

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

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

  $B_2(x,y),\ N_3(y) \longrightarrow B_3(x,y)$

picture next slide

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

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

$A \rightarrow B$: $\{n_b\}_{Kb}$

- Authentication

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

Sample Trace

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

$\exists x.\ A_1(x)$

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

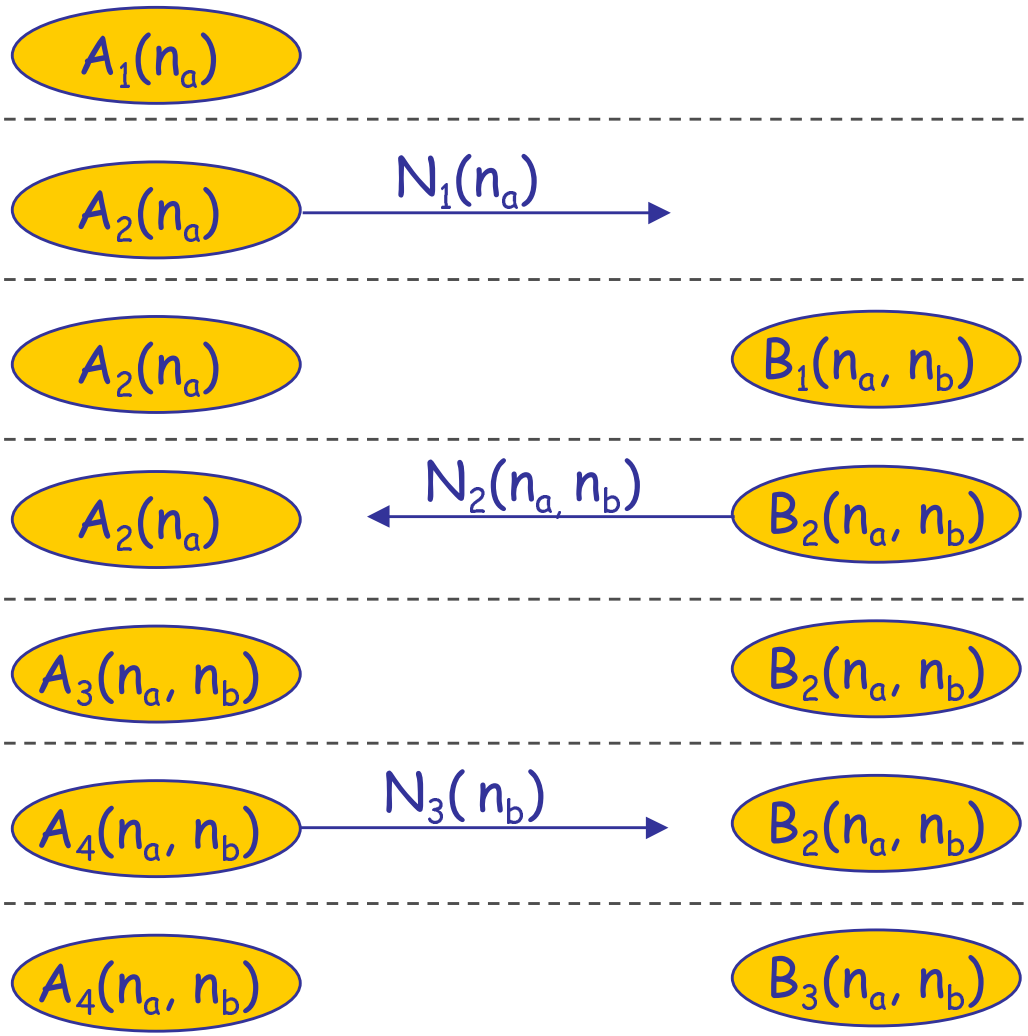
$N_1(x) \rightarrow \exists y.\ B_1(x,y)$

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

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

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

$B_2(x,y), N_3(y) \rightarrow B_3(x,y)$



Common Intruder Model

- Derived from Dolev-Yao model [1989]
  - 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 and remember messages

$N_1(x) \longrightarrow M(x)$ $\quad$ $N_2(x,y) \longrightarrow M(x), M(y)$

$N_3(x) \longrightarrow M(x)$

- Send messages from "known" data

$M(x) \longrightarrow N_1(x), M(x)$

$M(x), M(y) \longrightarrow N_2(x,y), M(x), M(y)$

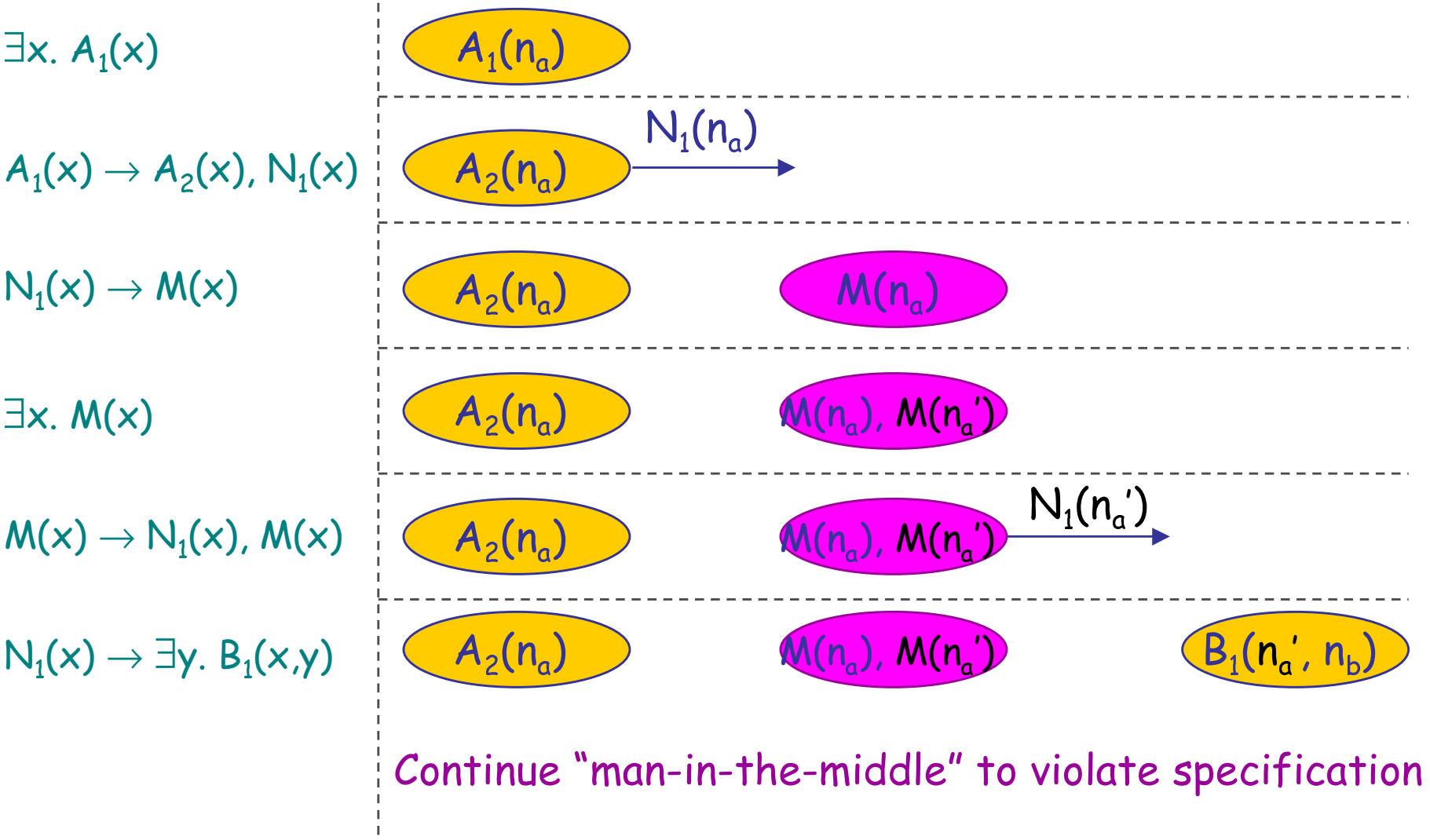
$M(x) \longrightarrow N_3(x), M(x)$

- Generate new data as needed

$\exists x.\ M(x)$

Highly nondeterministic, same for any protocol

Attack on Simplified Protocol



Modeling Perfect Encryption

- Encryption functions and keys
  - For public-key encryption
    - two key sorts: e_key, d_key
    - predicate Key_pair(e_key, d_key)
  - Functions

    enc : e_key × msg -> msg

    dec : d_key × msg -> msg (implicit in pattern-matching)
- Properties of this model
  - Encrypt, decrypt only with appropriate keys
  - Only produce enc(key, msg) from key and msg *(!!!)*
    - *This is not true for some encryption functions*

Steps in public-key protocol

- Bob generates key pair and publishes
  - $\exists_{e_key}\ u.\ \exists_{d_key}\ v.\ Bob_1(u,v)$
  - $Bob_1(u,v) \longrightarrow N_{Announce}(u),\ Bob_2(u,v)$
- Alice sends encrypted message to Bob
  - $Alice_1(e,d,x),\ N_{Announce}(e') \longrightarrow Alice_2(e,d,x,e')$
  - $Alice_2(e,d,x,e') \longrightarrow N_1(enc(e',\langle x,e\rangle)),\ Alice_3(u,v,x,w)$
- Bob decrypts
  - $Bob_1(u,v),\ N_1(enc(u,\langle x,y\rangle)) \rightarrow \exists y.\ Bob_1(u,v,\ x,y)$

Intruder Encryption Capabilities

- Intruder can encrypt with encryption key
  - $M_e(k), M_{data}(x) \longrightarrow N_i(enc(k,x)), M_e(k), M_{data}(x)$
- Intruder can decrypt with decryption key
  - $N_j(enc(k,x)), Key_pair(k,k'), M_d(k'), \longrightarrow M_{data}(x), \ldots$
- Add to previous intruder model

Assumes sorts data, e_key, d_key with typed predicates $M_{data}(data)$, $M_e(e_key)$, $M_d(d_key)$

Intruder: power and limitations

- Can find some attacks
  - Needham-Schroeder by exhaustive search
- 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 prove erroneous protocol correct
  - Requires property that crypto does not provide

Malleability [Dolev,Dwork,Naor]

- Idealized assumption
  - If intruder produces Network(enc(k,x)) then either
    - Network(enc(k,x)) $\longrightarrow$ M (enc(k,x)) (replay)
    - M(k), M(x) $\longrightarrow$ M (enc(k,x)) (knows parts)
- Not true for RSA
  - encrypt(k,msg) = msg^k mod N
  - property encr(x*y) = encr(x) * encr(y)
- Model
  - Network(enc(k,x)) $\longrightarrow$ M (…) … $\longrightarrow$ Network (enc(k,c*x))

Can send encrypted message without "knowing" message

Finite state ?

Outline

- Introduction to protocols and security
  - Security vs correctness
  - Examples and general concepts
- Standard approach: analyze protocol+intruder
  - Unifying protocol formalism
  - Communication and intruder model

➡ Specific projects based on standard approach

  - Theorem-proving, state enumeration methods, ...

- Mention some of own projects, results

Analysis of Protocol+Intruder

- Prove properties of protocols
  - Unbounded # of participants, message space
  - Prove that system satisfies specification
    - Paulson, etc: prove invariant holds at all reachable states
    - Spi-calculus: prove protocol equivalent ideal protocol
- Symbolic search with pruning
  - Search backward from error
  - Prune search by proving forward invariants
- Exhaustive finite-state methods
  - Approximate infinite-state system by finite one
  - Search all states, perhaps with optimizations

Example description languages

- First- or Higher-order Logic
  - Define set of traces, prove protocol correct
- Horn-clause Logic $\forall x\ldots (A_1 \wedge A_2 \wedge \ldots \supset B)$
  - Symbolic search methods
- Process calculus
  - FDR model checker based on CSP
  - Spi-calculus proof methods based on pi-calculus
- Additional formalisms
  - CAPSL protocol description language [Millen]
  - Murφ language for finite-state systems

Paulson's Inductive Method

- Define set TR of traces of protocol+intruder
  - Similar to traces in unifying formalism
  - Transition $F_1, \ldots, F_k \longrightarrow \exists x_1 \ldots \exists x_m.\ G_1, \ldots, G_n$ gives one way of extending trace
- Auxiliary functions mapping traces to sets
  - Analz(trace) = data visible to intruder
  - Synth(trace) = messages intruder can synthesize
- Definitions and proofs use induction
  - Similar inductive arguments for many protocols

Symbolic Search Methods

- Examples: NRL Protocol Analyzer, Interrogator
- Main idea
  - Write protocol as set of Horn clauses
    - Transition $F_1, \ldots, F_k \longrightarrow \exists x_1 \ldots \exists x_m.\ G_1, \ldots, G_n$ can be Skolemized and translated to Prolog clauses
  - Search back from possible error for contradiction
    - This is usual Prolog refutation procedure
- Important pruning technique
  - Prove invariants by forward reasoning
  - Use these to avoid searching unreachable states

Process Calculus Description

- Protocol defined by set of processes
  - Each process gives one step of one principal
  - Can derive by translation from unifying notation
    - $F_1, \ldots, F_k \longrightarrow \exists x_1 \ldots \exists x_m.\ G_1, \ldots, G_n$ is one process
    - Replace predicates by port names
    - Replace pattern-matching by explicit destructuring
    - In pi-calculus, use ν in place of $\exists$
  - Example
    - $B_1(x,y) \longrightarrow N_2(x,y), B_2(x,y)$
    - $b_1(p)$. let x=fst(p) and y=snd(p) in $\overline{n_2}\langle x,y\rangle \mid \overline{b_2}\langle x,y\rangle$ end

Spi-Calculus [AG97, ...]

- Write protocol in process calculus
- Express security using observ. equivalence
  - Standard relation from programming language theory

    $P \approx Q$ iff for all contexts $C[\ ]$, same observations about $C[P]$ and $C[Q]$
  - Context (environment) represents adversary
- Use proof rules for $\approx$ to prove security
  - Protocol is secure if no adversary can distinguish it from an idealized version of the protocol

Finite-state methods

- Two sources of infinite behavior
  - Many instances of participants, multiple runs
  - Message space or data space may be infinite
- Finite approximation
  - Transitions: $F_1, \ldots, F_k \longrightarrow \exists x_1 \ldots \exists x_m.\ G_1, \ldots, G_n$ choose fixed number of Skolem constants
  - Terms: restrict repeated functions f(f(f(f(x))))
- Can express finite-state protocol + intruder in
  - CSP : FDR-based model checking projects
  - Other notations: Murφ project, Clarke et al., ...

Outline

- Introduction to protocols and security
  - Security vs correctness
  - Examples and general concepts
- Standard approach: analyze protocol+intruder
  - Unifying protocol formalism
  - Communication and intruder model
- Specific projects based on standard approach
  - Theorem-proving, state enumeration methods, ...

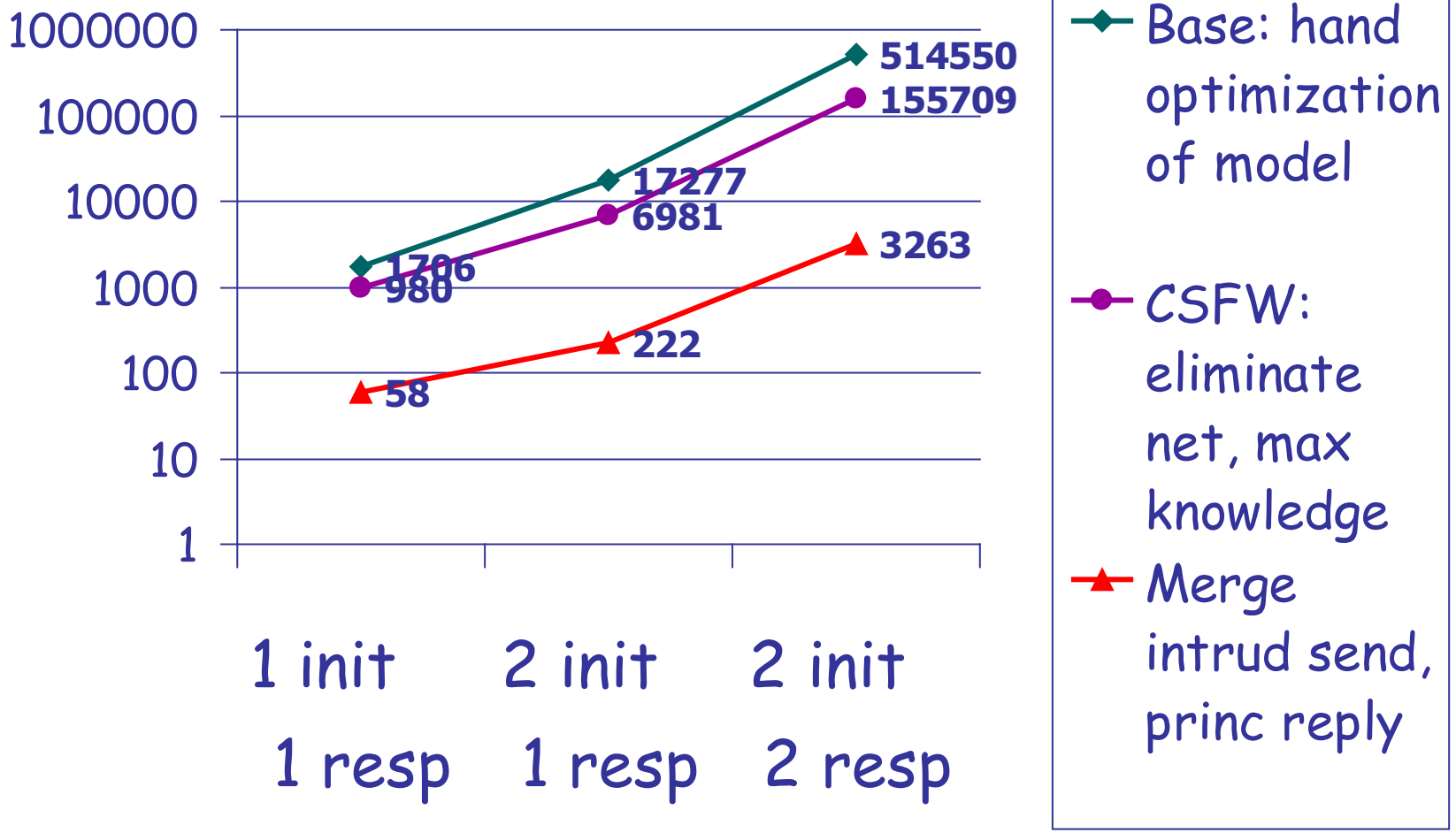
➡ Mention some of own projects, results

Security Protocols in Murφ

- Standard "benchmark" protocols
  - Needham-Schroeder, TMN, …
  - Kerberos
- Study of Secure Sockets Layer (SSL)
  - Versions 2.0 and 3.0 of handshake protocol
  - Include protocol resumption
- Discovered all known or suspected attacks
- Recent work on tool optimization

[Shmatikov, Stern, …]

State Reduction on N-S Protocol



Another current project

- Probabilisitic polynomial-time analysis
  - Drop simplifying assumption of perfect crypto
  - Use complexity-based assumptions instead
- Use spi-calculus framework
  - Probabilistic poly-time process calculus
  - Specification by observational equivalence
- Many technical problems
- Some promise

  Given protocol, identify properties required of cryptographic primitives [Lincon, Mitchell^2, Scedrov]

Authentication and Secrecy for Handshake Protocols

- How many protocols are there to verify?
  - Average length 7 steps
  - Data fields per message 5 fields
  - Distinct ways to fill a field $\times$ 50 entries
  - Number of possible combinations 1750 protocols
- Research directions
  - Get the monkeys and typewriters going
  - Easier description and specification, faster tools
  - Improved analysis of timestamps, ...
  - Interaction between protocol and crypto primitives

Beyond "standard" protocols

- Protocols for other purposes
  - Electronic commerce, fair commitment, ...
- Modularity
  - How to compose secure systems?
  - Which module performs security check?
- Isolation and Infiltration
  - Firewalls and tunneling
  - Intrusion detection, survival, recovery
  - Mobile code
    - New forms of interaction, new risks; need new model
- Economic security: more costly to attack than worth

Conclusion

- Security problems can be fun
- Many scientific, engineering challenges
  - Model intruder, properties of cryptography
- Results that matter

See short paper in proceedings, web site
for references to other work

Murφ DEMO !