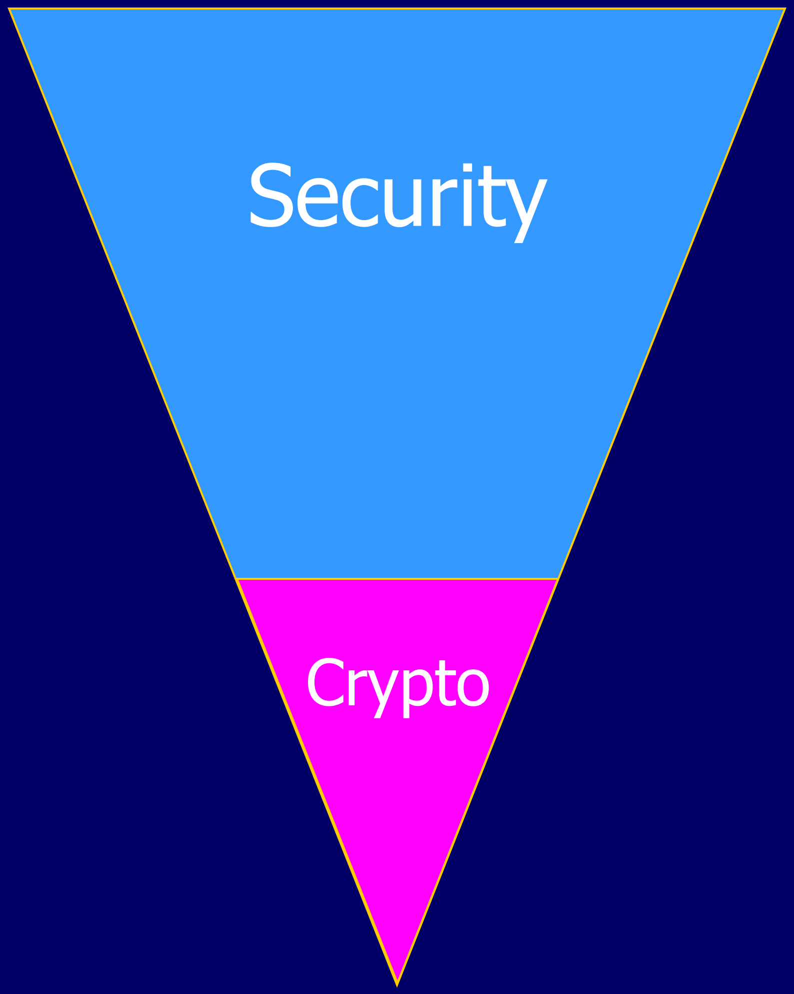
Symbolic Analysis of Computer Network Security Protocols

Andre Scedrov
University of Pennsylvania

Computer Security

Goal: protection of computer systems and digital information

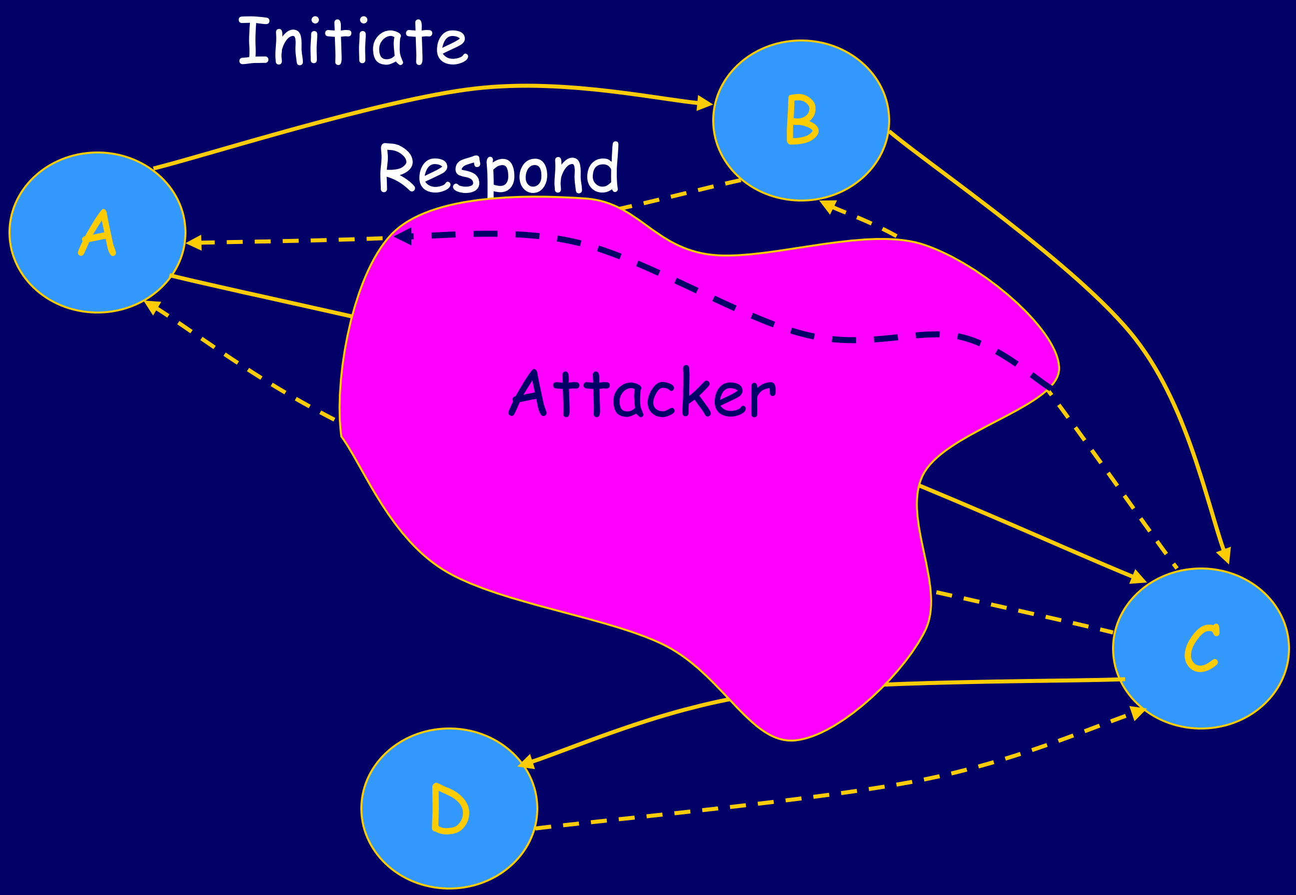
- Access control
- OS security
- Network security
- Cryptography
- …



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

Run of protocol

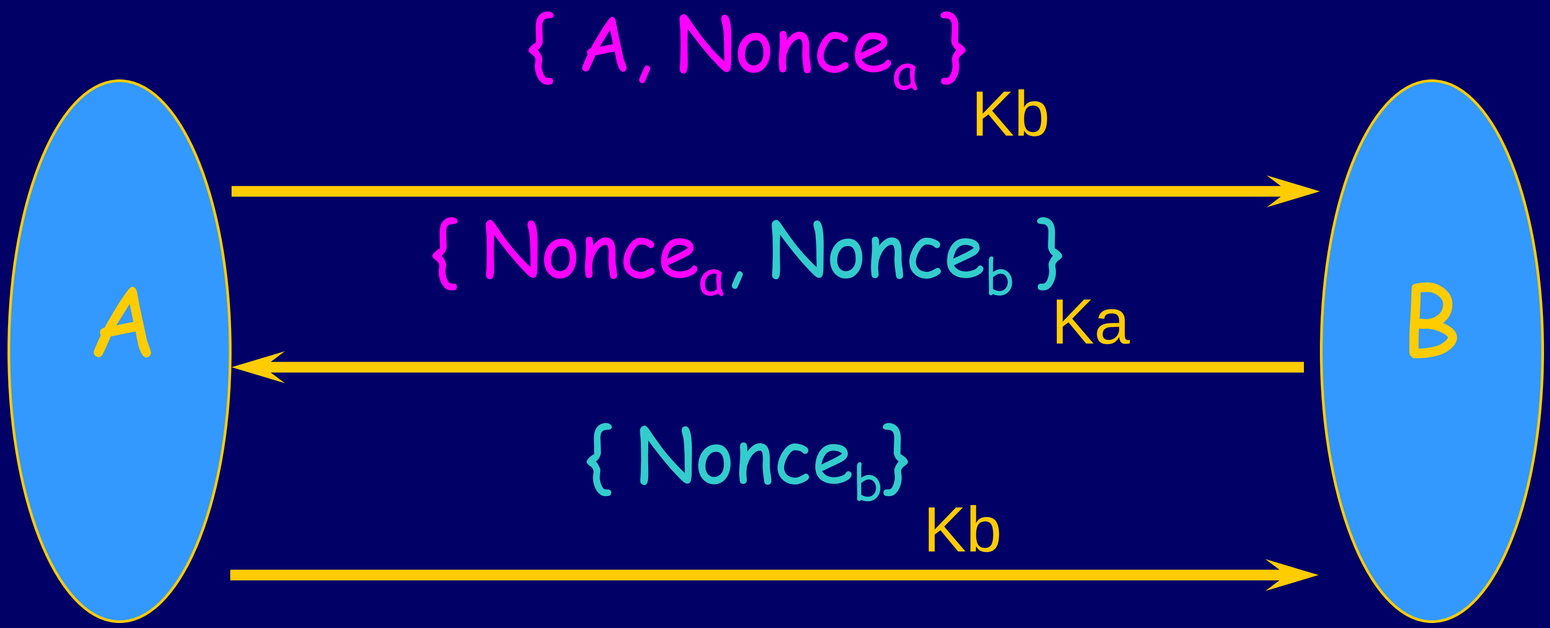


Correct if no security violation in any run

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

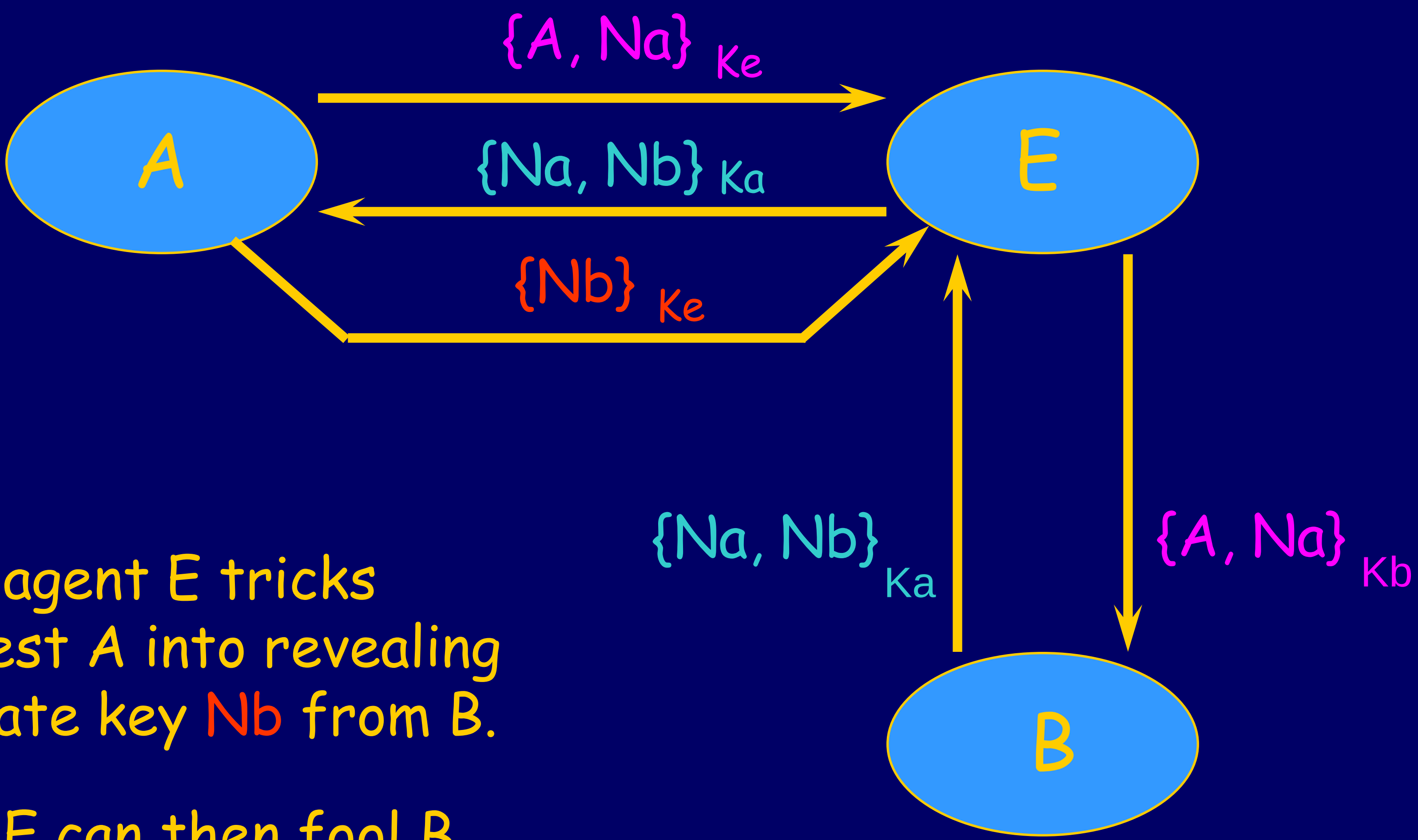
Needham-Schroeder Key Exchange



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

Anomaly in Needham-Schroeder

[Lowe]



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

Evil E can then fool B.

Kerberos Authentication Protocol

- Protocol goals
  - Repeatedly authenticate a client to multiple servers
  - Minimize use of client's long term key(s)
  - Does not guard against DOS attacks
- Kerberos 4 - 1989
- Kerberos 5
  - Specified in RFC 4120 (2005)
  - Subsequent revisions by working group
- A real world protocol
  - Windows 2000 and later (RFC 4120 + extensions)
  - User login, file access, printing, etc.

Kerberos 5

- Client C wants ticket for end server S
  - Tickets are encrypted – unreadable by C
- C first obtains long term (e.g., 1 day) ticket from a Kerberos Authentication Server K
  - Makes use of C's long term key
- C then obtains short term (e.g., 5 min.) ticket from a Ticket Granting Server T
  - Based on long term ticket from K
  - C sends this ticket to S

Contract Signing (Fair Exchange)

- Contract already agreed on
- Parties adversarial
- Both parties want to sign a contract
- Neither wants to sign first
- Fairness: each party gets the other's signature or neither does

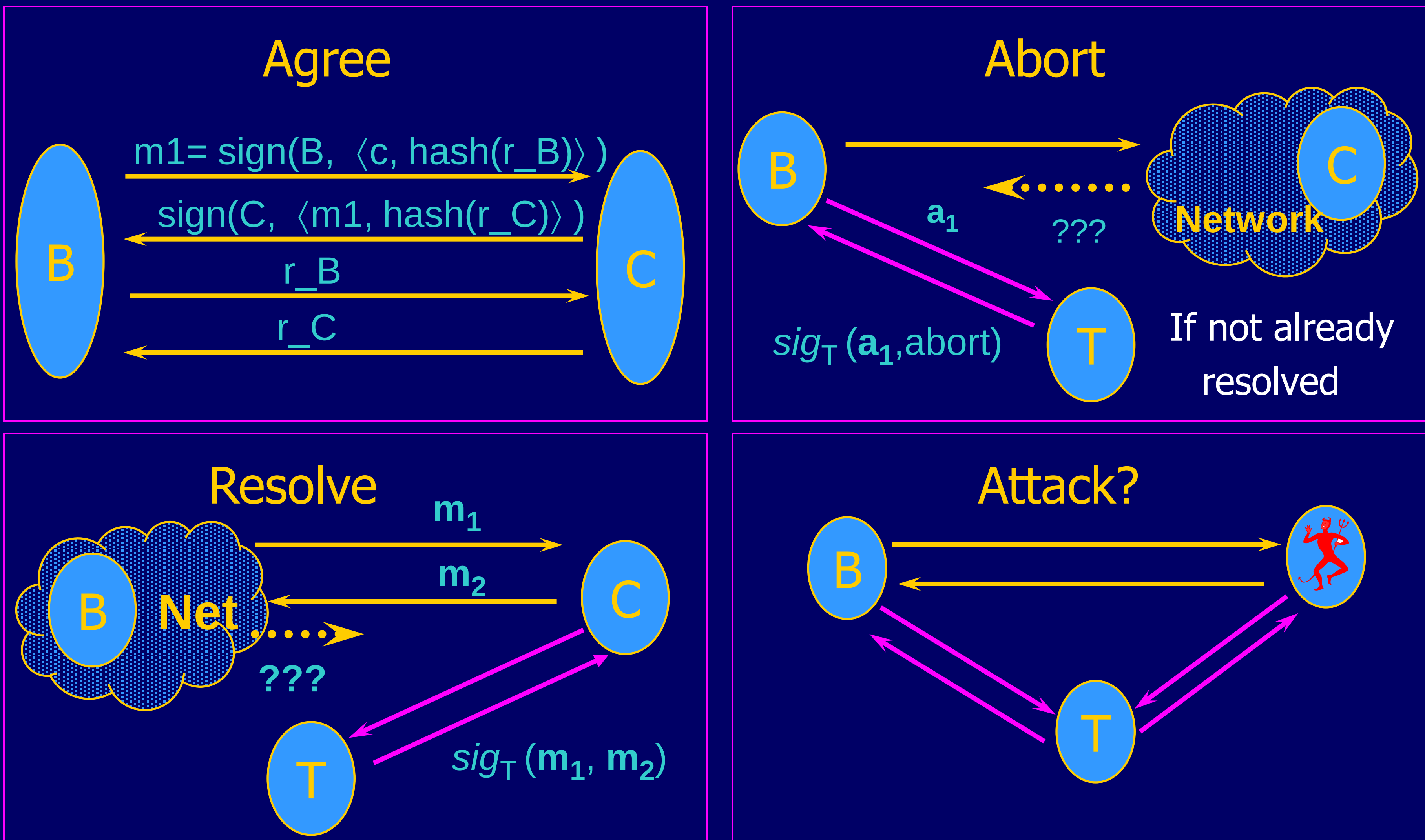
Scenario: Online Stock Trading

- Signed contracts for each trade
- Why include contracts:
  - Customer may want to prevent a broker who does not complete a requested trade from claiming that the request was never received
  - Broker may want proof that it is acting as requested

General protocol outline

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

Asokan-Shoup-Waidner protocol



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

Common Intruder Model

- Derived from positions taken in Needham-Schroeder [1978] and Dolev-Yao [1983]
- Idealization that makes protocol analysis palatable
  - 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…

Protocol Analysis Methods

- Non-formal approaches
  - Some crypto-based proofs [Bellare, Rogaway]
  - Communicating Turing Machines [Canetti]
- 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
    - Denotation of protocol = set of runs involving arbitrary number of principals plus intruder
- Protocol composition logic [Datta, Derek, Mitchell, Pavlovic]
- Cryptographic Library [Backes, Pfitzmann, Waidner]

Example projects and tools

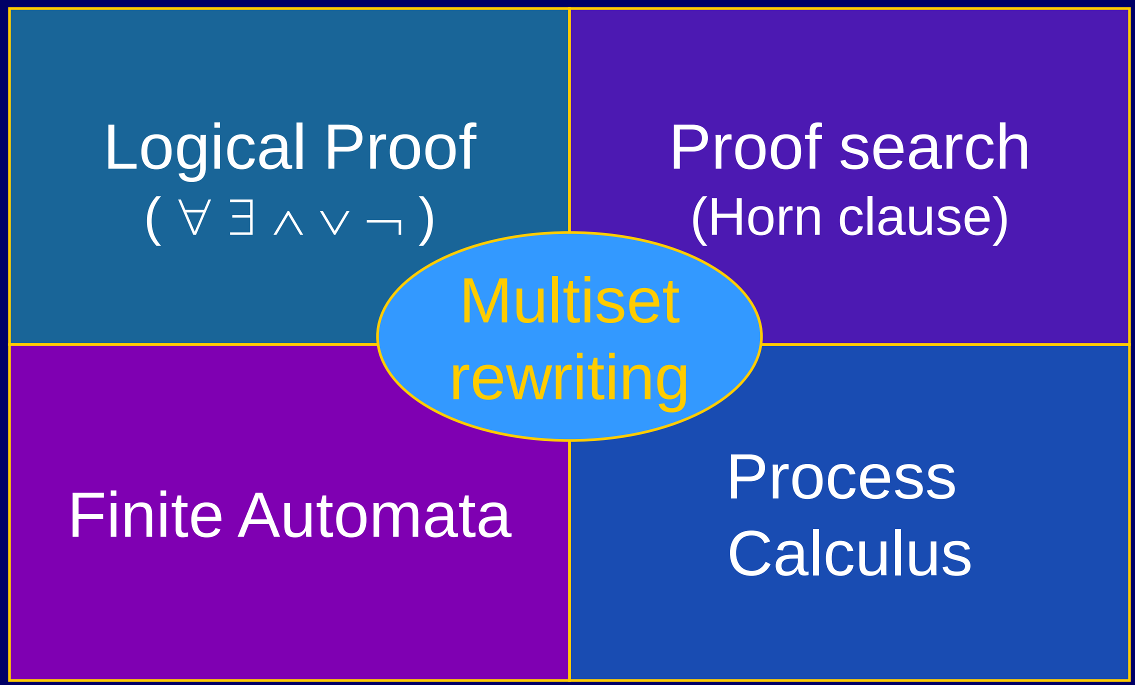
- Prove protocol correct
  - Paulson's "Inductive method", others in HOL, PVS,
  - MITRE - Strand spaces
  - Process calculus: Abadi-Gordon, Gordon-Jeffrey
- Search using symbolic representation of states
  - Meadows: NRL Analyzer, Millen: CAPSL
- Exhaustive finite-state analysis
  - FDR, based on CSP [Lowe, Roscoe, Schneider, …]
  - Murphi, CASPER, CAPSL, …

All depend on behavior of protocol in presence of attack

Multiset Rewriting Method

- A form of rewriting with
  - One associative, commutative operator (Banatre, LeMetayer; Chem Abs Machine)
  - $\exists$ to generate fresh data
- Conventions for modeling protocols, adversary using rewriting

A notation for inf-state systems



- Many previous models are buried in tools
- Define common model in tool-independent formalism

Modeling Requirements

- Express properties of protocols
  - Initialization
    - Principals and their private/shared data
  - Nonces
    - Generate fresh random data
- Model attacker
  - Characterize possible messages by attacker
  - Cryptography
- Set of runs of protocol under attack

Notation commonly found in literature

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

$B \rightarrow A : \{ Nonce_a, Nonce_b \}_{Ka}$

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

- The notation describes protocol traces
- Does not
  - specify initial conditions
  - define response to arbitrary messages
  - characterize possible behaviors of attacker

[Cervesato, Durgin, Lincoln, Mitchell, Scedrov]

Rewriting 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

Rewrite rules

[Cervesato, Durgin, Lincoln, Mitchell, Scedrov]

- 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
- Note
  - 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

  $A_1(n_a)$

  -- Alice in state 1 with nonce n_a

  $B_1(n_a, n_b)$

  -- Bob in state 1 with n_a, n_b

  $N_1(n_a)$

  -- Network contains message 1 with data n_a

- Transitions

  $\exists x.\ A_1(x)$

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

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

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

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

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

Sample Trace

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

$\exists x.\ A_1(x)$

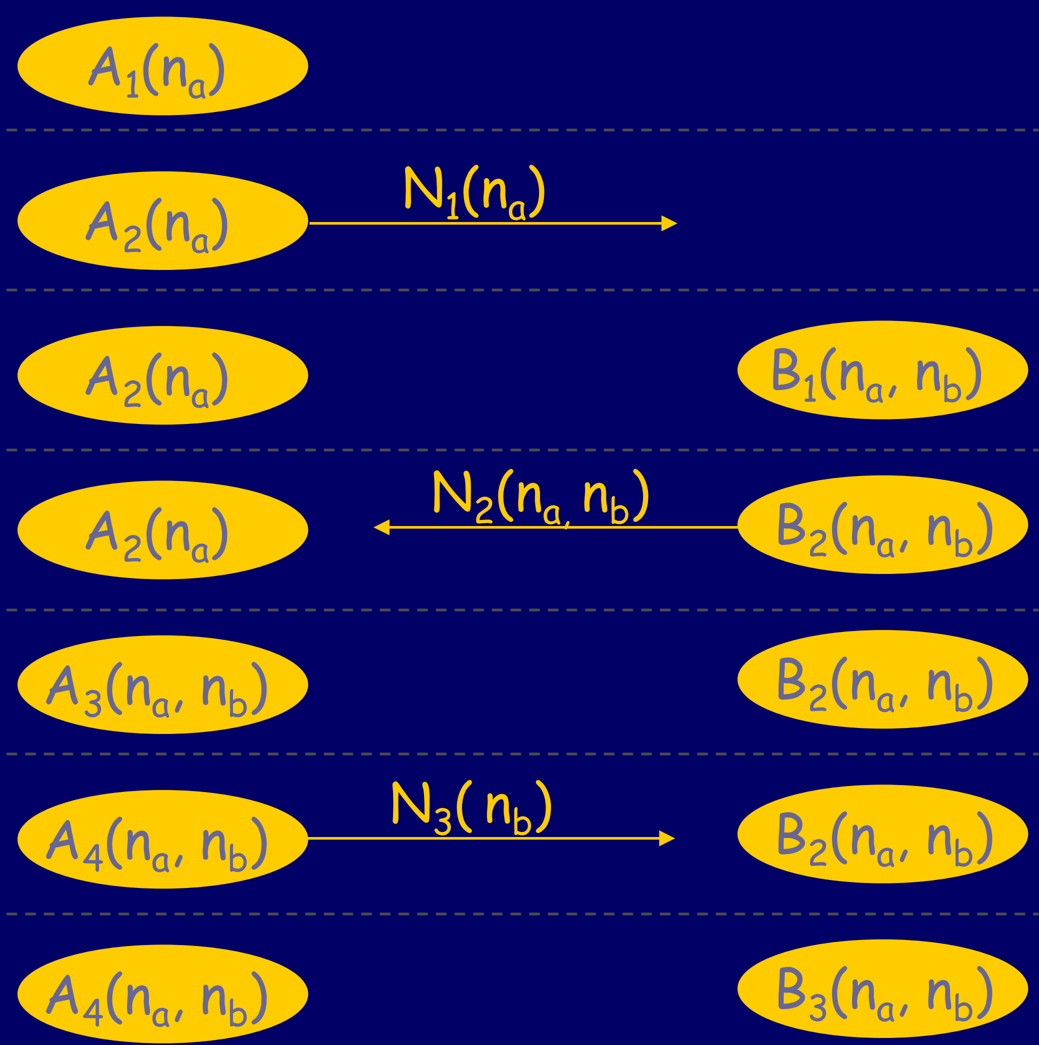
$A_1(x) \to A_2(x), N_1(x)$

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

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

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

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



Formalize Intruder Model

- Intercept, decompose and remember messages

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

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

- Decrypt if key is known

$$M(enc(k,x)), M(k) \longrightarrow M(x)$$

- 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

Protocol theory

- Initialization theory
  - Bounded theory that "precedes" protocol run
  - Example: $\exists$ key. Principal(key)
- Role generation theory
  - Principal(key) $\longrightarrow$ A_0(key), Principal(key)
  - Principal(key) $\longrightarrow$ B_0(key), Principal(key)
- Role theory
  - Finite ordered list of rules

$$A_i(\ldots),\ N_j(\ldots) \longrightarrow \exists \ldots\ A_k(\ldots),\ N_l(x) \qquad \text{where } i<k,\ j<l$$

  - Can also have persistent predicates on left/right

Two-phase intruder theory

- Avoid pointless looping by intruder
  - $M(x), M(y) \longrightarrow N(x,y), M(x), M(y)$
  - $N(x,y) \longrightarrow M(x), M(y)$
- Phase 1: Decomposition
- Phase 2: Composition

Thesis: MSR Model is accurate

- Captures "Dolev-Yao-Needham-Millen-Meadows- …" model
  - MSR defines set of traces protocol and attacker
  - Connections with approach in other formalisms
- Useful for protocol analysis
  - Errors shown by model are errors in protocol
  - If no error appears, then no attack can be carried out using only the actions allowed by the model

Attack on Simplified Protocol

$\exists x.\ A_1(x)$	$A_1(n_a)$		
$A_1(x) \rightarrow A_2(x), N_1(x)$	$A_2(n_a)$ $\xrightarrow{N_1(n_a)}$		
$N_1(x) \rightarrow M(x)$	$A_2(n_a)$	$M(n_a)$	
$\exists x.\ M(x)$	$A_2(n_a)$	$M(n_a), M(n_a')$	
$M(x) \rightarrow N_1(x), M(x)$	$A_2(n_a)$	$M(n_a), M(n_a')$ $\xrightarrow{N_1(n_a')}$	
$N_1(x) \rightarrow \exists y.\ B_1(x,y)$	$A_2(n_a)$	$M(n_a), M(n_a')$	$B_1(n_a', n_b)$

Continue “man-in-the-middle” to violate specification

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 $\times$ msg -> msg

    dec : d_key $\times$ 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
  - $Alice_1(e,d,x),\ N_{Announce}(e') \longrightarrow Alice_2(e,d,x,e')$
  - $Alice_2(e,d,x,e') \longrightarrow N_1(enc(e',\langle x,e\rangle)),\ Alice_3(e,d,x,e')$
- Bob decrypts message and generates nonce
  - $Bob_1(u,v),\ N_1(enc(u,\langle x,y\rangle)) \rightarrow \exists z.\ Bob_2(u,v,x,y,z)$

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 $M_{data}(data)$, $M_e(e\_key)$, $M_d(d\_key)$

Connections with logic and tools

- Search can find protocol errors
  - Backward search:
    - Interrogator [Millen]
    - NRL analyzer [Meadows]
    - ProVerif [Blanchet]
  - Forward search (model checking)
    - FDR [Roscoe], Casper [Lowe], Murphi [Mitchell^2 & Stern]
    - SMV [Marrero, Clarke, & Jha]
    - Athena [Song], TIPE [Denker, Meseguer, Talcott & Millen]
- Prove protocol properties
  - Poly-time prob. process calculus [Lincoln, Mitchell, Ramanathan, Scedrov,Teague]
    - CryptoVerif [Blanchet]
  - Inductive proof:
    - InaJo [Kemmerer], Coq [Bolignano]
    - Isabelle [Paulson, Basin], PVS[Dutertre, Schneider, Millen]

Conventional wisdom

- Find protocol errors
  - Model checking
  - Exhaustive search of finite-state system
- Prove protocol correct
  - Use theorem-proving system
  - Exhausting development of formal proof
- Are there decidable protocol cases?
  - Many are short programs with simple data
  - Ping-Pong protocols (D&Y: Ptime) too restrictive
  - What causes intractability for interesting protocols?

General protocols are undecidable

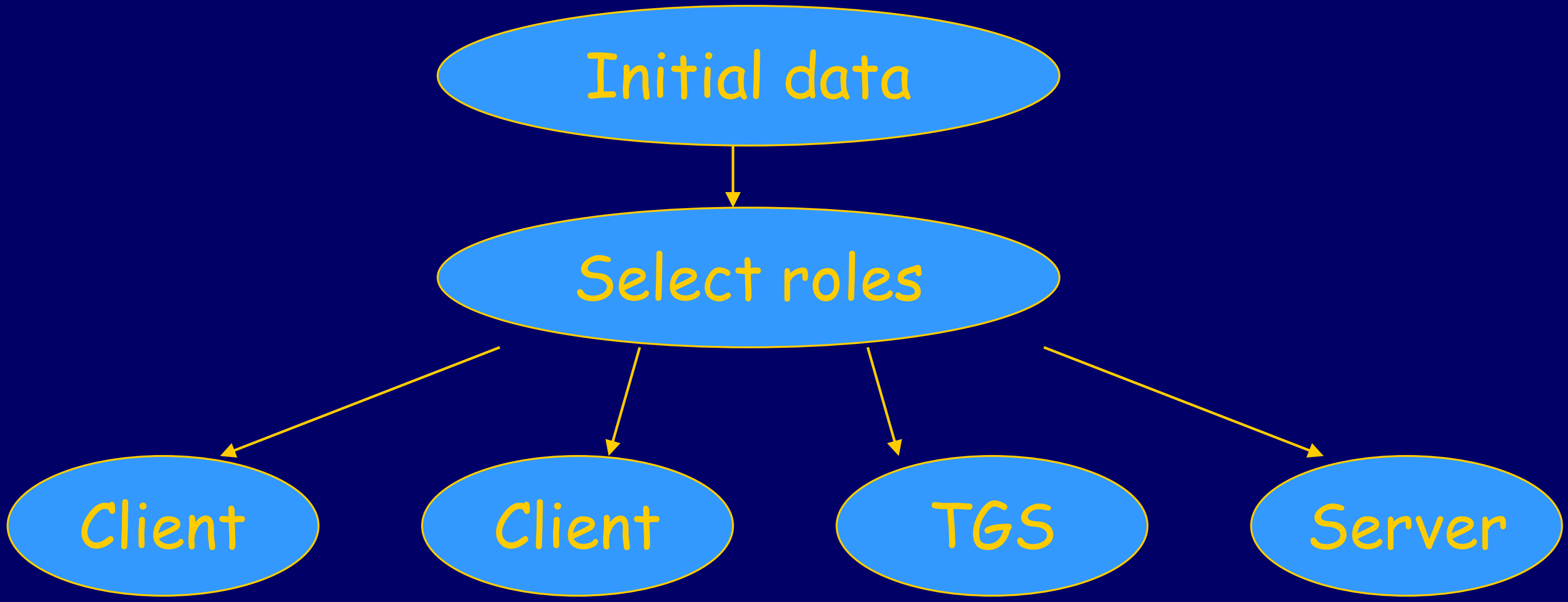
- Even and Goldreich 1983, Heintze and Tygar 1996, …
- Idea: Post Correspondence Problem
  - Given an indexed finite set of pairs of strings (U_i,V_i) , is there a sequence of indices $i_1, \ldots, i_n$ so that $U_{i1} \ldots U_{in} = V_{i1} \ldots V_{in}$
- Security: Intruder never learns SECRET
  - Unreachability of state including M(SECRET)

General protocols are undecidable

- Post Correspondence Problem as a protocol:
  - Good guy appends pair (U_i,V_i) to end of sequence
  - If top and bottom read the same, spill secret
    - A -> B: $\{empty, empty\}_k$
    - B -> A: $\{X,Y\}_k \Rightarrow \{(X\ U_i), (Y\ V_i)\}_k$
    - A -> B: $\{X,X\}_k \Rightarrow$ if $X \neq$ empty, send SECRET

Protocols vs Rewrite rules

- Can axiomatize any computational system
- But -- protocols are not arbitrary programs



Bounded message size

- Prohibit arithmetic
  - Some protocols use successor:
    - A -> B: $\{Nonce\}_k$
    - B -> A: $\{Nonce + 1\}_k$
  - Successor and equality test lead to undecidability
- Prohibit nested encryption
  - Some protocols use nested encryption:
    - A -> B: $\{\{m\}_k, Nonce\}_{k'}$
  - Arbitrary depth encryption allows undecidability
    - A -> B: $\{\{m\}_k, \{\{\{m\}_k\}_k\}_k, Q\}_k$
  - State is Q, two counters are 1 and 3.

What about a "realistic" restricted class of protocols ?

- Finite number of principals
- Each role has finite number of steps
  - But a principal may repeat any number of roles
- Bounded message size
  - Fixed number of fields in message
  - Fixed set of message constants
  - Fixed depth encryption
  - Allow nonces (but only "create new nonce", and =)
- Everything constant, except number of roles and number of new nonces

Protocol theory

- Initialization theory
  - Describes initial conditions such as key generation or other shared information
- Role generation theory
  - Designates possibly multiple roles that each participant may play (such as initiator, responder, client, or server)
- Agent theory
  - Disjoint union of bounded subtheories that each characterize a possible role

Protocol theory

- Initialization theory
  - Bounded theory that "precedes" protocol run
  - Example: $\exists$ key. Principal(key)
- Role generation theory
  - Principal(key) $\longrightarrow$ A_0(key), Principal(key)
  - Principal(key) $\longrightarrow$ B_0(key), Principal(key)
- Role theory
  - Finite ordered list of rules

    $A_i(\ldots), N_j(\ldots) \longrightarrow \exists\ldots\ A_k(\ldots), N_l(x)$ where $i<k$, $j<l$
  - Can also have persistent predicates on left/right

Two-phase intruder theory

- Avoid pointless looping by intruder
  - $M(x), M(y) \longrightarrow N(x,y), M(x), M(y)$
  - $N(x,y) \longrightarrow M(x), M(y)$
- Phase 1: Decomposition
- Phase 2: Composition

Thesis: MSR Model is accurate

- Captures "Dolev-Yao-Needham-Millen-Meadows- …" model
  - MSR defines set of traces protocol and attacker
  - Connections with approach in other formalisms
- Useful for protocol analysis
  - Errors shown by model are errors in protocol
  - If no error appears, then no attack can be carried out using only the actions allowed by the model

Secrecy still undecidable

- There is no algorithm for deciding whether a given protocol in restricted form, run in combination with the standard intruder, allows the intruder to gain access to a given initial secret.
  - Represent existential Horn theories as protocol theories
  - Existential Horn theories w/o function symbols are undecidable: *Vardi ICALP'81, Chandra, Lewis, and Makowsky STOC'81*

Direct encoding

- Turing machines, Cook's Theorem
  - but use *nonces* instead of propositional variables

Start | 0 | 0 | 1 | q_2•0 | 0 | 1 | 1 | 0 | End

Start | 0 | 0 | q_5•1 | 1 | 0 | 1 | 1 | 0 | End

Start | 0 | 0 | 0 | q_6•1 | 0 | 1 | 1 | 0 | End

Turing machine

1 | q•0 | 0

1

Constant (3) piece of state at time N determines state of cell at time N+1

Start | 0 | 0 | 1 | q_2•0 | 0 | 1 | 1 | 0 | End

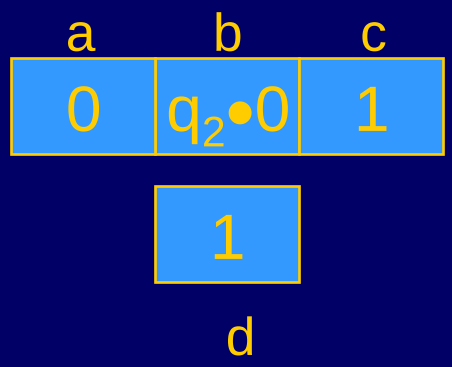
Start | 0 | 0 | q_5•1 | 1 | 0 | 1 | 1 | 0 | End

Start | 0 | 0 | 0 | q_6•1 | 0 | 1 | 1 | 0 | End

Turing machine

- Predicates
  - Cell(name, symbol, right) -- *contents of tape cell*
  - Below(cell, cell) -- *next row of tableau*
- Rules
  - Cell(a,0,b), Cell(b, $q_2\bullet 0$,c), Cell(c,1,…)

  $\longrightarrow$ ∃ d. Below(b,d), Cell(d,1,…),...



Turing machine

Turing machine move

- Cell(a,da, b), Cell(b,db, c), Cell(c,dc, d), Below(b,b’)

  $\longrightarrow$ ∃c’. Below(c,c’), Cell(b’,F(da,db,dc),c’)

Copy to Next Time

- Below(a,a’), Cell(a,Start,b)

  $\longrightarrow$ ∃a’’,b’: Below(a’,a’’), Cell(a’,Start, b’)

Extend Tape

- Below(a,a’), Cell(a,End,b)

  $\longrightarrow$ ∃b’, c’: Cell(a’,0, b’), Cell(b’, End, c’)

Start and End

- $\longrightarrow$ ∃a,a’,b,c,d,e: Cell(a,Start,b), Cell(b,Qinit,c), Cell(c, 0, d), Cell(d,End,e), Below(a,a’)
- Cell(a,Qfinal,b) $\longrightarrow$ Broadcast(Secret)

Turing machine discussion

- Each move is a protocol role
  - Finite length protocol
- Attacker replays and routes messages
  - To prevent malicious alteration, encrypt all messages will shared private key:

    $\{ \text{Cell}(a,da, b) \}_k$
- Machine steps in standard protocol form

  $A_i(\ldots), N_m(\ldots) \rightarrow A_k(\ldots), N_l(\ldots)$

  Role reads hypotheses one at a time, saving data in internal state.

Undecidability

- Finite length protocols with
  - bounded number of principals
  - bounded message size

  have undecidable behavior if
  - principals can repeat roles arbitrarily many times
  - runs can generate new atomic data

- What happens if we
  - Bound ability to generate new data?
  - Restrict number of roles ?

Attack requires exponential run

- Sender role broadcasts initial message
  - A: Broadcast $\{0, 0, 0, 0\}_k$
- n responder roles modify secret messages
  - B1: $\{x, y, z, 0\}_k \longrightarrow \{x, y, z, 1\}_k$
  - B2: $\{x, y, 0, 1\}_k \longrightarrow \{x, y, 1, 0\}_k$
  - B3: $\{x, 0, 1, 1\}_k \longrightarrow \{x, 1, 0, 0\}_k$
  - B4: $\{0, 1, 1, 1\}_k \longrightarrow \{1, 0, 0, 0\}_k$
- Server broadcasts key on specific message
  - C : $\{1, 1, 1, 1, 1\}_k \longrightarrow$ Broadcast(k)
- Attack requires 2^n steps and 2^n messages.

Security DEXP-time complete

- No new data, but repeat roles arbitrarily
- Same encoding of Horn clauses (DATALOG)
  - Axiomatize bounded Turing machine tableau
- Use counters instead of nonces to name cells
  - Cell(name, data, neighbor) as before
  - Represent name by pair of numbers
    - Cell($\underbrace{0,1,0,\ldots,0}_{\text{n bits}}$, $\underbrace{0,0,1,\ldots,1}_{\text{n bits}}$, data, neighbor),
  - $2^n \times 2^n$ tableau using messages of size 4n

Testing for $a = b$, $c \neq d$

- $x \neq y$ atomic
- Conditional transition rule

  $a_1=b_1, \ldots , a_i=b_i, c_1 \neq d_1 , \ldots , c_j \neq d_j ;\; F_1, \ldots, F_k$

  $\longrightarrow \exists x_1 \ldots \exists x_m .\; G_1, \ldots , G_n$
- What this means
  - If $F_1, \ldots, F_k$ in state σ, and if $a_1=b_1, \ldots , a_i=b_i$, $c_1 \neq d_1 , \ldots , c_j \neq d_j$ are true, 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

Complexity results using MSR

<table>
<tr><th colspan="2"></th><th>Bounded # of roles</th><th>Bounded use of ∃</th><th>Unbounded use of ∃</th></tr>
<tr><td rowspan="2">Intruder 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">Intruder w/o ∃</td><td>≠, =</td></tr>
<tr><td>= only</td></tr>
</table>

All: Finite number of different roles, each role of finite length, bounded message size

Key insight: existential quantification (∃) captures cryptographic nonce; main source of complexity

[Durgin, Lincoln, Mitchell, Scedrov]

Lower bounds from Horn clauses

		Bounded # of roles	Bounded # of $\exists$	Unbounded # of $\exists$
Intruder with $\exists$	$\neq$, =	NP-complete: Provable by bounded-length proof	??	Undecidable: Datalog + $\exists$
	= only		Dexptime: Datalog	
Intruder w/o $\exists$	$\neq$, =			
	= only			

All: Finite number of different roles, each role of finite length, bounded message size

Need to show that hard instances of Horn clause inference can be be represented in the restricted form of a security protocol

[Durgin, Lincoln, Mitchell, Scedrov]

Additional decidable cases

- Bounded role instances, unbounded msg size
  - Huima 99: decidable
  - Amadio, Lugiez: NP w/ atomic keys
  - Rusinowitch, Turuani: NP-complete, composite keys
  - Other studies, e.g., Kusters: unbounded # data fields
- Constraint systems
  - Cortier, Comon: Limited equality test
  - Millen, Shmatikov: Finite-length runs

All: bound number of role instances

Lessons

- Symbolic notation for unrestricted protocols
  - Nonce becomes existentially quantified variable
  - Translations to process calculus, strands, HOL, ...
  - Fragment of linear logic
    - Protocol search is proof search
    - Formal proofs using linear-logic proof theory, tools
- Study decision problems (secrecy, authenticity)
  - Undecidable if protocols generate new data
  - DEXP-time complete with bounded new data
  - NP-complete if bounded number of roles

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

- Our 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 in general
  - Given only the ciphertext it may be easy to generate a *different ciphertext* so that the respective *plaintexts are related*
  - Attacks may exploit this: adversary computes enc(f(x)) given only enc(x)

Malleability [Dolev,Dwork,Naor]

- RSA
  - $enc(k,msg) = msg^k \bmod N$
  - property $enc(x \cdot y) = enc(x) \cdot enc(y)$
  - trivial to compute $enc(2x)$ given only $enc(x)$
- Model
  - $Network(enc(k,x)) \longrightarrow M(\ldots) \ldots \longrightarrow Network(enc(k,c \cdot x))$

  Can send encrypted message without "knowing" message
- Non-malleable crypto [Dolev,Dwork,Naor]

[Butler, Cervesato, Jaggard, Scedrov]

Kerberos 5 Analysis: Goals

- Give precise statement and formal analysis of a real world protocol
  - Find a real world protocol – Kerberos 5
  - Pick favorite formalization method - MSR
- Identify and formalize protocol goals
- Give proofs of achieved protocol goals
  - Gain experience in reasoning with MSR
- Note any anomalous behavior
  - Suggest possible fixes, test these

Kerberos 5

- Client C wants ticket for end server S
  - Tickets are encrypted – unreadable by C
- C first obtains long term (e.g., 1 day) ticket from a Kerberos Authentication Server K
  - Makes use of C's long term key
- C then obtains short term (e.g., 5 min.) ticket from a Ticket Granting Server T
  - Based on long term ticket from K
  - C sends this ticket to S

Protocol Messages

Please give me ticket for T

C ⟶ K

Ticket for C to give to T

C ⟵ K

Ticket from K, one for S?

C ⟶ T

Ticket for C to give to S

C ⟵ T

Ticket from T

C ⟶ S

Confirmation (optional)

C ⟵ S

Overview of Results

- Formalized Kerberos 5 at different levels of detail
- Observed anomalous behavior
  - Some properties of Kerberos 4 do not hold for Kerberos 5
  - Proved authentication properties that do hold for Kerberos 5
- Proofs of properties which do hold
  - Methods adapted from Schneider
- Interactions with Kerberos working group

Related Kerberos Work

- Kerberos 4 - Bella & Paulson
  - Inductive approach using theorem prover Isabelle
  - Proofs of authentication and confidentiality
  - Incorporated timestamps and temporal checks
- Bella & Riccobene
  - Gurevich's Abstract State Machine
- Kerberos 5 - Mitchell, Mitchell, & Stern

Related Formal Work

- MultiSet Rewriting (MSR) formalism
  - Lincoln, Mitchell, Scedrov, Durgin, and Cervesato
  - Extended to Typed MSR by Cervesato
- Rank functions
  - Defined by Schneider
  - Our proof methods adapted from this idea

Abstract Formalization

- Contains core protocol
  - Other formalization refines this one
- Exhibits an anomaly
  - This appears to be structural and not due to omitted detail
- Allows us to prove authentication results

Messages in Abstract Level

$$C \xrightarrow{\quad C,T,n_1 \quad} K$$

$$C \xleftarrow{\quad C,\{k_{CT},C\}_{k_T},\ \{k_{CT},n_1,T\}_{k_C} \quad} K$$

$$C \xrightarrow{\quad \{k_{CT},C\}_{k_T},\{C\}_{k_{CT}},C,S,n_2 \quad} T$$

$$C \xleftarrow{\quad C,\{k_{CS},C\}_{k_S},\{k_{CS},n_2,S\}_{k_{CT}} \quad} T$$

$$C \xrightarrow{\quad \{k_{CS},C\}_{k_S},\{C,t\}_{k_{CS}} \quad} S$$

$$C \xleftarrow{\quad \{t\}_{k_{CS}} \quad} S$$

Detailed Formalization

- Uses richer message structure
  - Adds some fields for options
    - E.g., anonymous tickets
  - Models encryption type
  - Adds checksums
- Exhibits anomalies
  - Encryption type option specific to this level
  - Structural anomaly also seen at abstract level
    - Also variations which use added detail

Messages in Detailed Level

$$C \xrightarrow{KOpts,C,T,n_1,e_1} K$$

$$C \xleftarrow{C,\{Tflags,k_{CT},C\}_{k_T}, \{k_{CT},n_1,Tflags,T\}^{e_1'}_{k_C}} K$$

$$C \xrightarrow{\{Tflags,k_{CT},C\}_{k_T},\{C,MD,t\}_{k_{CT}},Topts,C,S,n_2,e_2} T$$

$$C \xleftarrow{C,\{Sflags,k_{CS},C\}_{k_S},\{k_{CS},n_2, Sflags,S\}^{e_2'}_{k_{CT}}} T$$

$$C \xrightarrow{SOpts,\{Sflags,k_{CS},C\}_{k_S},\{C,MD',t'\}_{k_{CS}}} S$$

$$C \xleftarrow{[\{t'\}^{e}_{k_{CS}}]} S$$

$$C \xleftarrow{KRB\_ERROR,[-|t|t'],t_{err},ErrCode,C,(K|T|S)} K|T|S$$

Encryption Type Anomaly

- Kerberos 5 allows C to specify encryption types that she wants used in K's response

$$C \xrightarrow[\text{etype (sent unencrypted)}]{\text{Please give me ticket for } T \text{ using}} K$$

$$C \xleftarrow[\text{info encrypted using etype)}]{\text{Ticket for } C \text{ to give to } T \text{ (other}} K$$

- C's key associated with the etype e_{bad} is k_{bad}
  - Intruder I learns k_{bad}
  - C knows this and attempts to avoid e_{bad}/k_{bad}
  - I can still force k_{bad} to be used

Ticket Anomaly

$C \xleftarrow{\text{Ticket for } C \text{ to give to } T} K$

- **Kerberos 4:**
  - Ticket is enclosed in another encryption

$\xleftarrow{\{\text{Ticket, Other data}\}_{k_C}}$

- **Kerberos 5:**
  - Ticket is separate from other encryption

$\xleftarrow{\text{Ticket, }\{\text{Other data}\}_{k_C}}$

Ticket Anomaly

$C \xrightarrow{\text{Please give me a ticket for } T} K$

$I \xleftarrow{\text{Ticket for } T\text{, \{Other data\}}_{k_C}} K$

$C \xleftarrow{X\text{, \{Other data\}}_{k_C}} I$ (cuts Ticket)

$C \xrightarrow{X\text{, Ticket for } S\text{?}} I$

$I \xrightarrow{\text{Ticket from } K\text{, ticket for } S\text{?}} T$

$C \xleftarrow{\text{Ticket for } S\text{.}} T$

Ticket Anomaly

- T grants the client C a ticket for S
- C has never sent a proper request for a ticket
  - C never has the ticket for T
  - C thinks she has sent a proper request
  - C's view of the world is inaccurate
  - Some properties of Kerberos 4 don't hold here
- Seen in both formalizations
  - Variations possible using added detail

Anonymous Ticket Anomaly

The AS Exchange takes place as usual, producing TGT and k_{TC}:

$C \xleftrightarrow[\text{KRB-AS-REP (TGT, } k_{TC}\text{)}]{\text{KRB-AS-REQ}} K$

The client C requests a regular and an anonymous ticket (both for S) using TGT:

$C \xrightarrow{\text{KRB-TGS-REQ (Regular, based on TGT)}} T$

$C \xrightarrow{\text{KRB-TGS-REQ (Anonymous, based on TGT)}} T$

The TGS T replies, but the intruder I switches the tickets (undetected by C):

$C \xleftarrow{\{SK_{Anon}, Anon, \ldots\}k_S, \{SK_C, \ldots\}k_{TC}} I \xleftarrow{\{SK_C, C, \ldots\}k_S, \{SK_C, \ldots\}k_{TC}} T$

$C \xleftarrow{\{SK_C, C, \ldots\}k_S, \{SK_{Anon}, \ldots\}k_{TC}} I \xleftarrow{\{SK_{Anon}, Anon, \ldots\}k_S, \{SK_{Anon}, \ldots\}k_{TC}} T$

- **C has wrong beliefs about data**
- Undesirable, but doesn't violate design goals. However, …

- SK_C and SK_{Anon} are service keys generated for regular and anonymous tickets.
- {m}k is the encryption of m with k.

Options for Final Step

1. C's name is leaked when she tries to contact S anonymously:

$$C \xrightarrow{\{SK_C, C, \ldots\}k_S,\ \{Anon, t\}SK_{Anon}} S$$

Intruder actions required if this message's integrity is protected [Tom Yu].

2. Alternatively, C sends each type of request. The request with anonymous ticket gives error, but I fixes other request by replaying first authenticator.

$$C \xrightarrow{\{SK_{Anon}, Anon, \ldots\}k_S,\ \{C, t\}SK_C} S$$

$$C \xrightarrow{\{SK_C, C, \ldots\}k_S,\ \{Anon, t\}SK_{Anon}} I \xrightarrow{\{SK_C, C, \ldots\}k_S,\ \{C, t\}SK_C} S$$

I then tampers with error message so that it names C. C believes anonymous request accepted (no error), regular request failed; reverse is true instead.

- **C's name is leaked or she has wrong beliefs about which type of request succeeded/failed.**

Discussion

- No violations of authentication or confidentiality, but anomalous behavior
  - Possible to leak *C*'s name (even if link to *S* is integrity protected)
  - Possible for *C* to have reversed view of which type of request has been accepted
- Are these (or related issues) of practical concern?
- We should be aware of possibility for these types of problems.

An Authentication Theorem

- If T processes the message

  $\{k_{CT},C\}_{k_T},\{C\}_{k_{CT}},C,S,n_2$

  then some K sent the message

  $C,\{k_{CT},C\}_{k_T}, \{k_{CT},n_1,T\}_{k_C}$

  and C sent *some* message

  $X,\{C\}_{k_{CT}},C,S',n'_2$

- Authenticate data origin using rank
  - Show ticket $\{k_{CT},C\}_{k_T}$ originates with some K
  - Show authenticator $\{C\}_{k_{CT}}$ originates with C
    - This makes use of a corank argument for confidentiality

Comments from Kerberos Designers

- Generally positive response
  - Should look at protocol extensions
- Anomalies
  - These scenarios can occur
  - Practical concern unclear
  - Anonymous ticket variation of interest
    - Status of this option may change
    - Good to highlight possible concerns here

Rank and Corank

- Inspired by work of Schneider
- Define functions on MSR facts
  - k-Rank – encryptions by k
    - Data origin authentication
  - E-Corank – level of protection by keys in E
    - Secrecy
- Proofs
  - State desired property
  - Find applicable (co)rank functions
  - Determine effect of MSR rules on these functions

(End glimpse of Kerberos 5 analysis)

[Chadha, Kanovich, Scedrov]

A glimpse of contract signing

- Each party enters contract with goal
  - Party who wants contract acts to complete the contract
- Correctness is relative to goal
  - Do not want well-intentioned party to suffer
- Leads to game-theoretic notions
  - If A follows strategy S, then B cannot achieve win over A
  - Or, A follows strategy from some class …

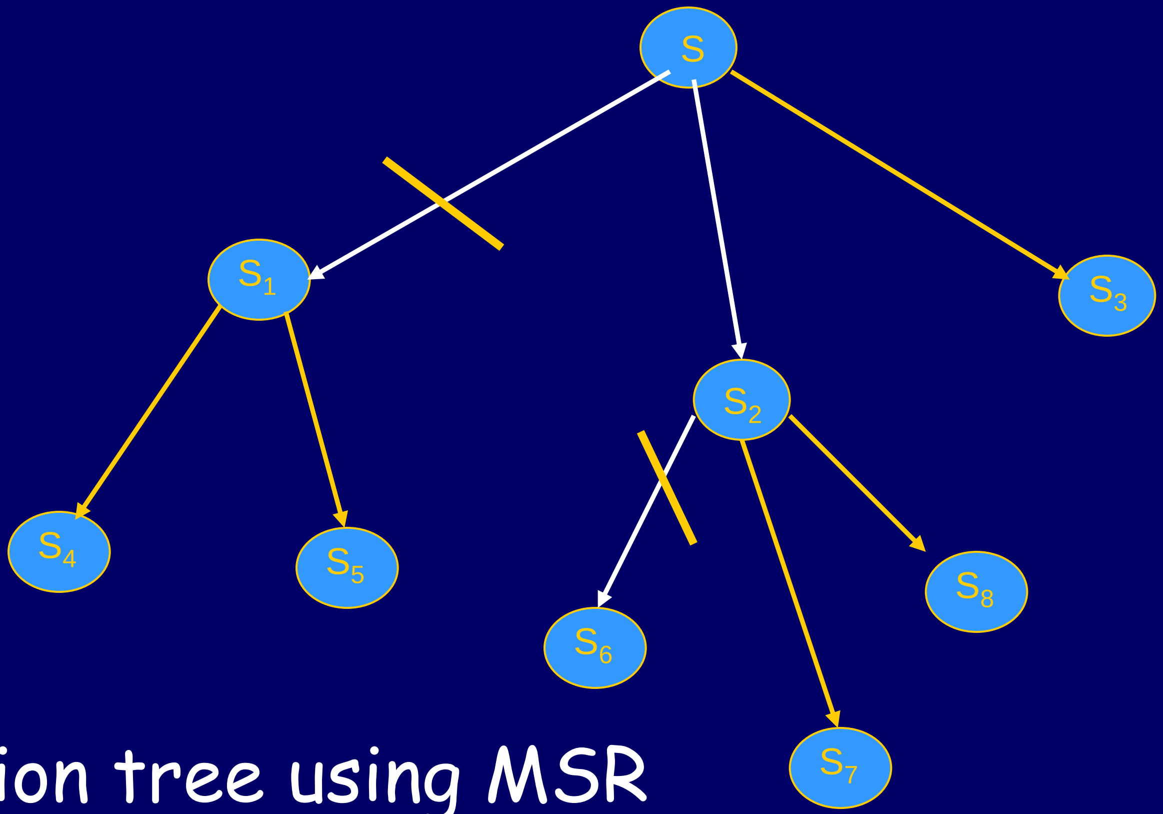
General protocol outline

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

Optimism and Advantage

- Once customer commits to the purchase, he cannot use the commited funds for other purposes
- Customer likely to wait for some time for broker to respond, since contacting TTP to force the contract is costly and can cause delays
- Since broker can request abort from TTP, this waiting period may give broker a way to profit: see if shares are available at a lower price
- The longer the customer is willing to wait, the greater chance the broker has to pair trades at a profit

Strategy: example



- Define execution tree using MSR
- Prune tree according to assumed strategy
- Determine correctness

Honest participant

- Principal P (B or C) is said to be honest if
  - P moves only according to protocol

Equivalent: P's key not known to adversary

Power to Abort

- $tr \backslash E$ is an abort tree for P if every leaf node is labeled by a state which is aborted for P
- Q has the power to abort at σ if there is an E such that $tr \backslash E$ is an abort tree for P
- Balance for honest P: For any reachable configuration σ, and for all bounds on the number of steps the intruder can take, at σ, Q does not have both the power to abort and the power to complete

Advantage

- Advantage
  - Power to abort and power to complete
- Balance
  - Potentially dishonest Q never has an advantage against an honest P
- Reflect natural bias of honest P
  - P is interested in completing a contract, so P is likely to wait before asking TTP for an abort or for a resolve
  - Formulate properties stronger than balance

Optimistic participant

- Honest P (B or C) is said to be optimistic if
  - Whenever P can choose between
    - waiting for a message from other participant Q
    - contacting TTP for any purpose

    P waits and allows Q to move next

[Chadha, Mitchell, Scedrov, Shmatikov]

Advantage

- Q is said to have the power to abort against an optimistic P the protocol in S
  - if Q can always drive the protocol to a configuration that is aborted for P
- Q is said to have the power to resolve against an optimistic P the protocol in S
  - if Q can always drive the protocol to a configuration that is complete for P and Q has P's signature
- Q has advantage against an optimistic P if Q has both the power to abort and the power to complete

Hierarchy

Advantage against honest P *H-adv*

$\Downarrow$

Advantage against optimistic P O-adv

MSR model lets us define execution tree

Define strategies, correctness over execution model

Advantage flow

B

C

O-adv

I am willing to sell at this price →

O-adv

I am willing to buy at this price ←

O-adv

Here is my signature →

Here is my signature ←

[Chadha, Mitchell, Scedrov, Shmatikov]

Impossibility Theorem

- In any optimistic, fair, and timely contract-signing protocol, any potentially dishonest participant will have an advantage at some point if the other participant is optimistic
- 3-valued version of:
  - Even's impossibility of deterministic two-party contract signing
  - Fischer-Lynch-Paterson impossibility of consensus in distributed systems

No evidence of advantage

- If
  - Q can provide evidence of P's participation to an outside observer X,

  then
  - Q does not have advantage against an optimistic P
- Evidence: what does X *know*
- X *knows* fact φ in state σ
  - φ is true in any state consistent with X's observations in σ

(End glimpse of contract signing)

Example projects and tools

- Prove protocol correct
  - Paulson's "Inductive method", others in HOL, PVS,
  - MITRE - Strand spaces
  - Process calculus: Abadi-Gordon, Gordon-Jeffrey
- Search using symbolic representation of states
  - Meadows: NRL Analyzer, Millen: CAPSL
- Exhaustive finite-state analysis
  - FDR, based on CSP [Lowe, Roscoe, Schneider, …]
  - Murphi, CASPER, CAPSL, …

All depend on behavior of protocol in presence of attack

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 our 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, ProVerif
- 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

Strands [Guttman et al.]

- Present information about causal interactions among protocol participants
- Events
  - message sent, message received
- Strands
  - finite sequences of events

    $s_1 \Rightarrow s_2 \Rightarrow \ldots \Rightarrow s_k$, each s_j an event
- Parametric strands
  - messages may contain variables (some marked "fresh")

Sample ~~Trace~~ Strand

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

$\exists x.\ A_1(x)$

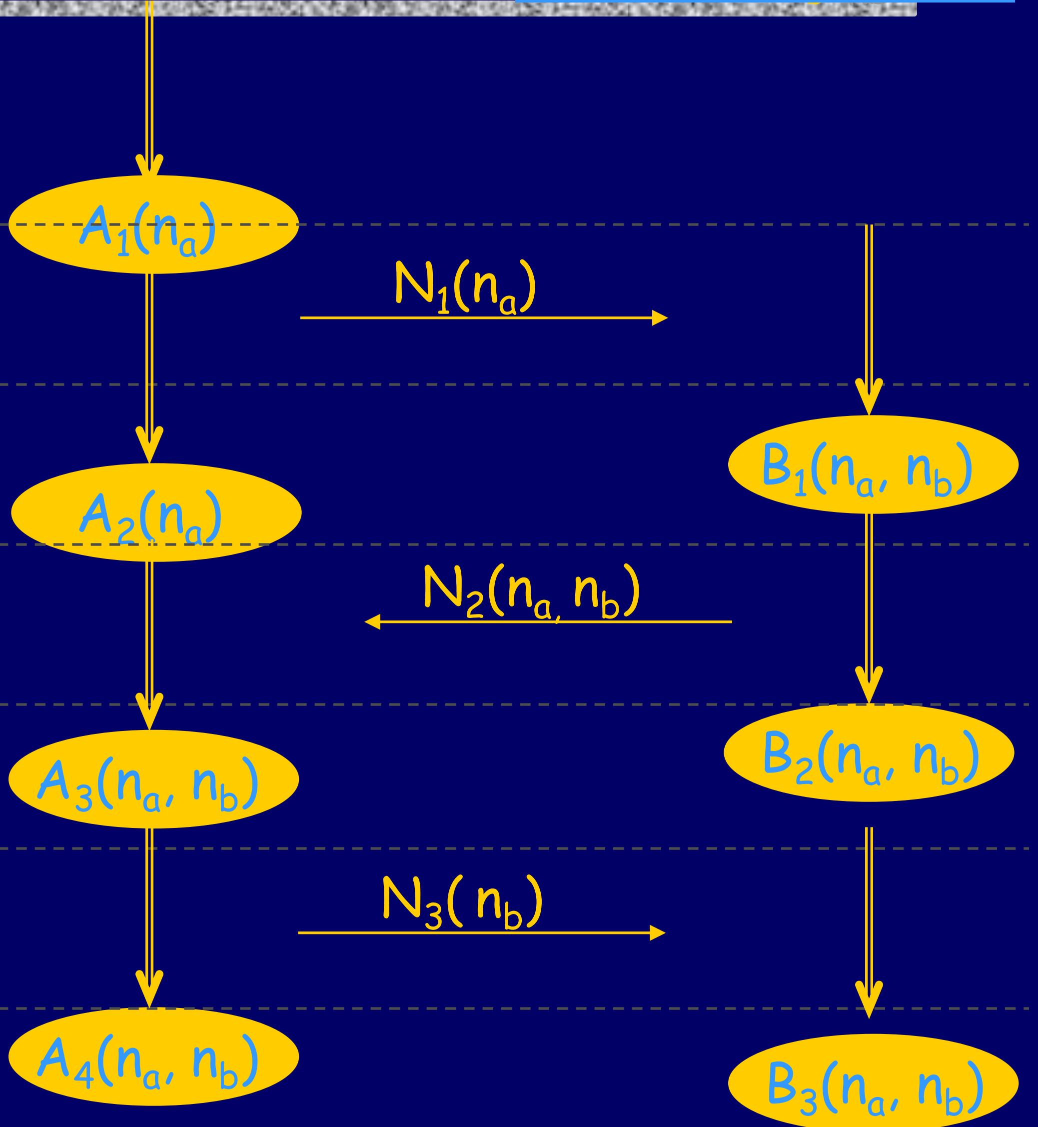
$A_1(x) \to A_2(x), N_1(x)$

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

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

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

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



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 $n_2\langle x,y\rangle \mid b_2\langle x,y\rangle$ end

Spi-Calculus [Abadi Gordon 97]

- 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., ...