Automated Analysis of Cryptographic Protocols Using Murφ

John Mitchell Mark Mitchell
Ulrich Stern

Stanford University

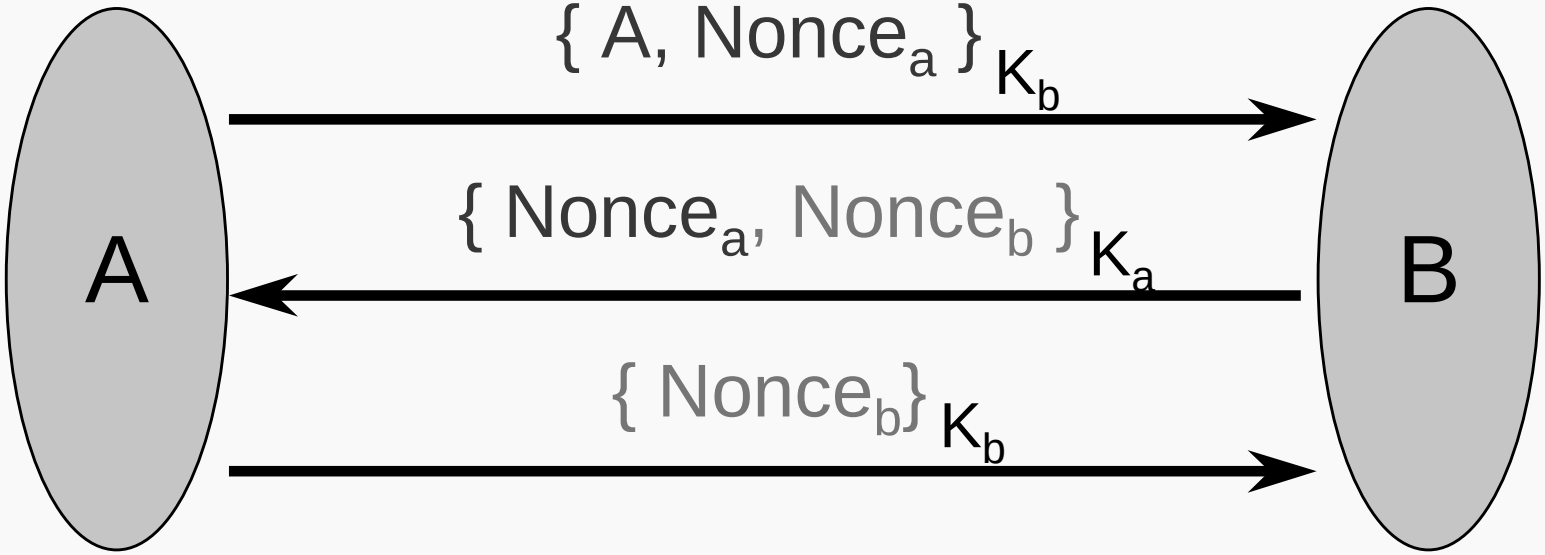
Cryptographic Protocols

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

Automated Analysis

- Define finite-state system
  - Bound on number of steps
  - Finite number of participants
  - Nondeterministic adversary with finite options
- Pose correctness condition
- Exhaustive search using “verification” tool

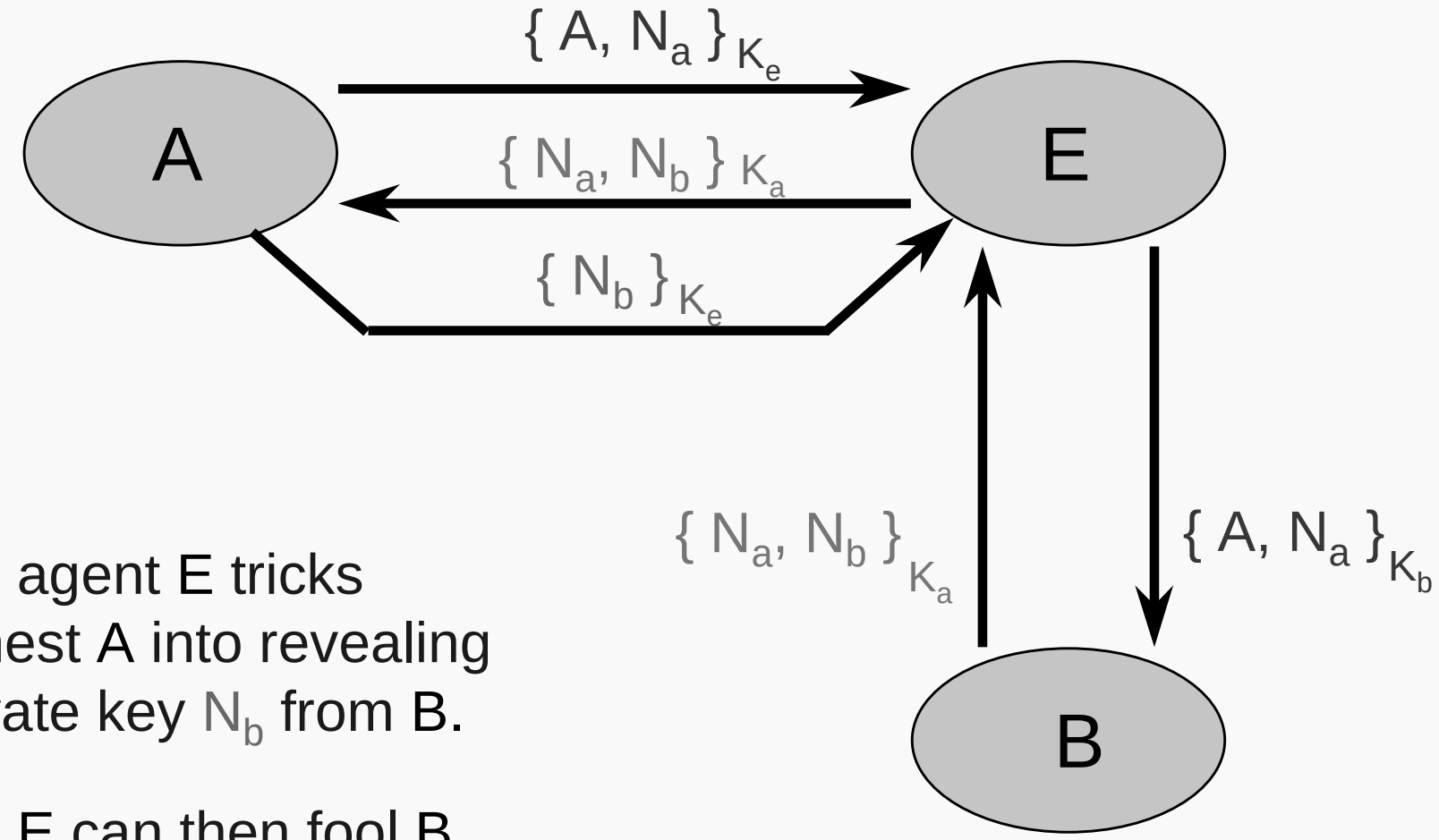
Needham-Schroeder Key Exchange



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.

Analyzing Security Protocols

- Think long and hard
- BAN, GNY Logic ...
- Logic-based tools
- Exhaustive state-enumeration tools
  - Similar work done using CSP

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)

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
- Future goal: automatic generation

Limitations

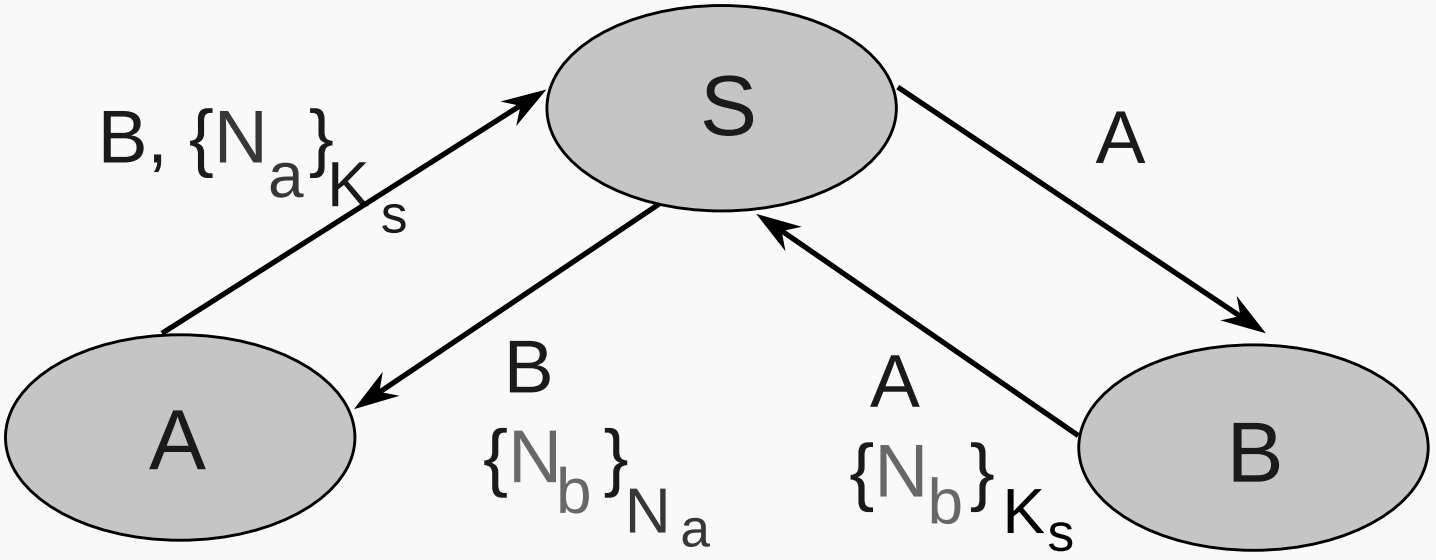
- System size with current methods
  - 2-6 participants

    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

Analysis Results

- Analyze common protocols
  - Needham-Schroeder
  - Kerberos
    - Found bug in documented algorithm (*not* in RFC)
    - one client, two servers
  - TMN cellular phone protocol
    - Found all known bugs automatically
    - Model algebraic properties of encryption function
- Work in progress: SSL protocol

TMN Protocol



- A initiates and B sends session key
- Several bugs:
  - replay step 3 for chosen N_a' $\quad I \rightarrow S : I, \{N_b\}_{Ks}$

Modeling Challenge

- Avoid repeated keys by storing list
  - Do not allow new session with old key
- But RSA allows “blinding”:
  - Adversary sends multiple of old key
  - Divides later message by multiplier
- Need to model multiplication in Murφ
  - Model message by pair: datum, blinding bit

Secure Sockets Layer (*SSL*)

- Three goals
  - Negotiate specific encryption scheme
    - Possible “version attack”
  - Authenticate client and server
    - Appeal to “signature authority”
  - Use public key to transmit secret key

Several underlying primitives: public key, signature scheme, hash function, private key

Conclusions

- Murφ equal or better than other tools
  - Programming-language-style input
  - Fast execution: symmetry reduction, …
- Understand protocol, find unintuitive errors
- Limitations of the approach
  - Manual formulation adversary
    - Time consuming, potential for errors
  - Paradigm ignores probability, complexity