Computational Soundness for Formal Adversaries

Jonathan Herzog
CIS, TOC, LCS, MIT
(Joint work w/ Ron Rivest, Nancy Lynch)

Needham–Schroeder Public Key Protocol

- First published public-key protocol (1978)

   1. $A \longrightarrow B$: $\{\!|N_a\, A|\!\}_{K_B}$
   2. $B \longrightarrow A$: $\{\!|N_a\, N_b|\!\}_{K_A}$
   3. $A \longrightarrow B$: $\{\!|N_b|\!\}_{K_B}$

- A, B are principals
   - K_A: public key of A
- $\{\!|M|\!\}_K$ is the encryption of M in key K
- N_a, N_b are fresh random values ("nonces")
- Designed to provide mutual authentication and secrecy of N_a, N_b

How to Analyze?

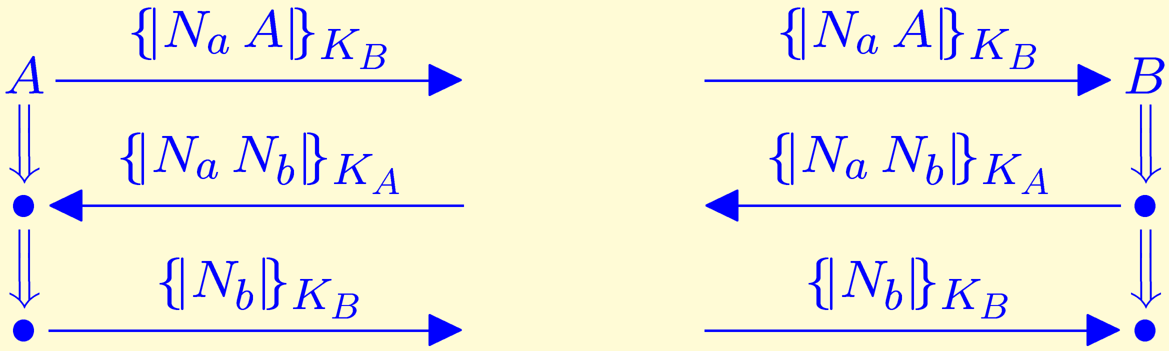
- Cryptographic protocols can be analyzed using the same tools used for primitives (Bellare & Rogaway 1994, 1995)
- However, "computational" model not the only framework for protocol analysis
- Dolev-Yao ("formal") model developed in formal methods community
  - Reflects character of that field
- Both have advantages; how do they relate?
- This talk:
  - Explain formal model
  - Examine relationship between computational and formal models in one key area: the adversary

Overview

- Motivation
- A security definition
- Implementation and proof

NS Protocol: Initiator's view

- Initiator's authentication goal: If A sees view on left, then B sees view on right

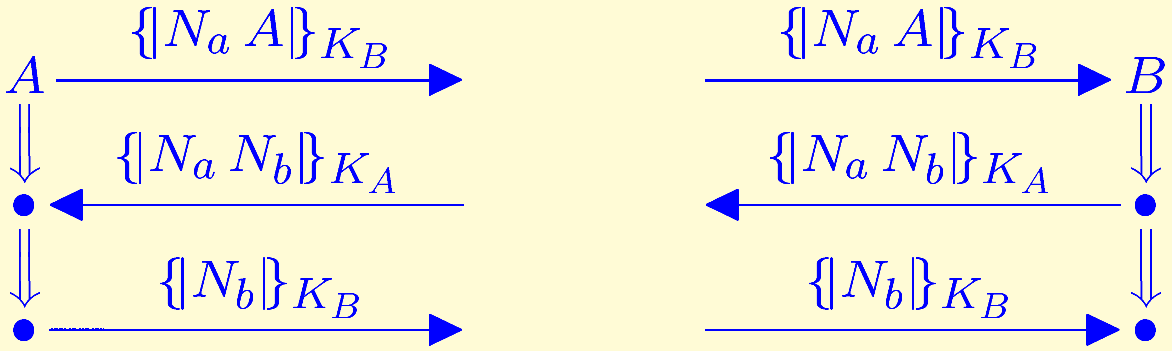


  (except possibly last step)

- How to see this:
  - Fresh nonce N_a sent out encrypted in B's public key
  - Received back
  - Someone must have decrypted it
  - B must be out there, with corresponding view

NS Protocol: Responder's View

- Responder's authentication goal: If B sees view on right, then A sees view on left



  – How to see this:
    - B sends out fresh random value N_b in A's public key
    - N_b comes back
    - Someone must have decrypted it
    - A must be out there, with corresponding view

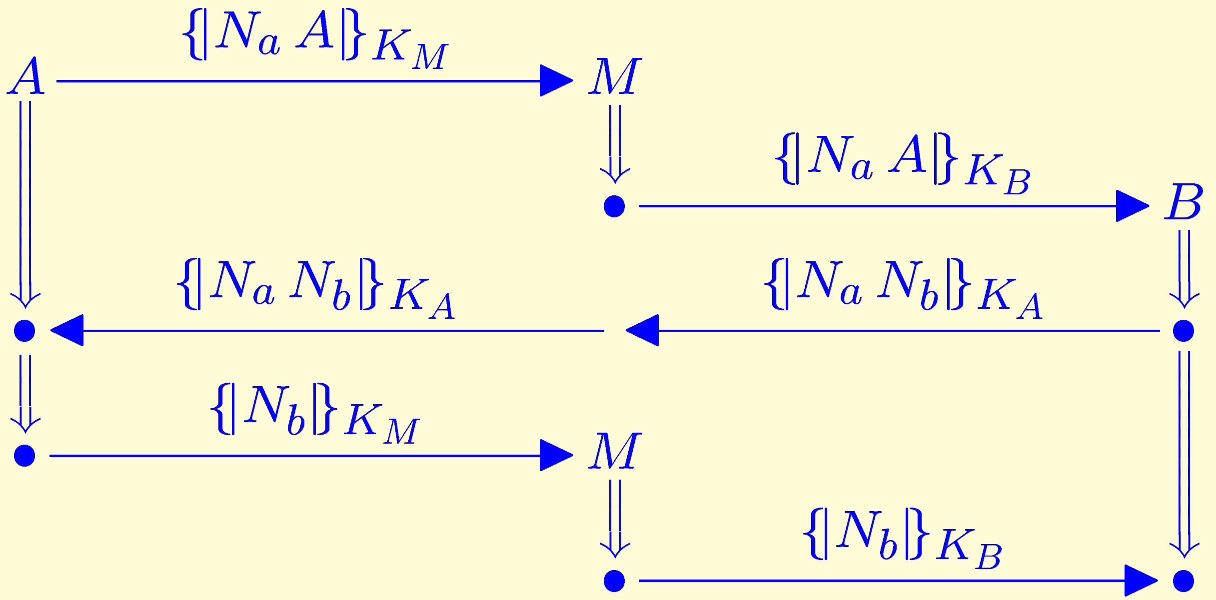
Very Simple Adversary

- Consider simple adversary:
- Presumed to start knowing some initial set of messages
  - Can guess predictable values
- Intercepts every message sent by honest participant
- Can perform *only* the following operations:
  - Encrypt known message with public key
  - Decrypt known message with known private key
  - Separate a known pair
  - Pair two known messages
  - Generate new nonces, keys
- Can perform finite number of such operations before sending known message to any honest participant

Demo 1

- Such an adversary can be specified as simple state machine
- Honest participants also simple state machines
- Behavior individual state machines can be composed into larger machine
- Security goals expressable as properties of the larger system
- Many tools exist to check system against goals
  - This example: Murϕ

Needham-Schroeder: Flawed!



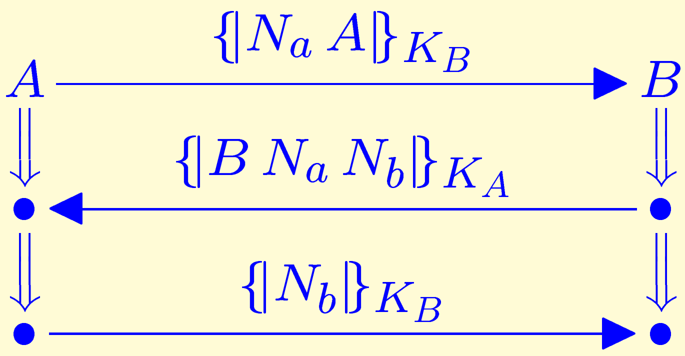
- Due to Gavin Lowe (1995)
  - Found using automated tool!

Dolev–Yao Model

- Dolev-Yao model: Adversary is simple
- Pros:
  - Very easy to work in
  - Need only consider *specific* adversary
  - Short proofs
    - Re-usable general theorems
  - Automation
    - Find flaws automatically
    - Develop proofs automatically
  - Good for examining *structure* of protocols
- Cons:
  - Well...best explained by example

Needham–Schroeder–Lowe Protocol

- Lowe “fixed” NS protocol by adding B’s name to second message:



- Murϕ finds no flaws (Demo 2)
  - Murϕ doesn’t generate proofs, but...
  - Can also be proven no flaw exists in Dolev-Yao model
  - Hence, secure against formal adversary
- What can we conclude, really?

Tale of Two Models

- Viewing the computational model as a "semantics" for the formal model:
- Formal model is *complete*
  - Any flaw found in formal model will exist in computational model
- Would like formal model to be *sound*
  - Proof of correctness in formal model implies correctness in computational model
  - Much less work for us
- Any demonstration of soundness will involve many components
- Here, we focus entirely on adversary

Point of this talk: show cryptography can limit computational adversary to formal adversary

Related Work

- This work greatly inspired by Abadi and Rogaway (2000)
  - Their work considered weaker computational adversary
- Incorporating poly-time indistinguishability into process calculus (Mitchel et al. 1998,1999,2001)

Overview

- Motivation
- A security definition
- Implementation and proof

Formal Adversary Revisited

- Want to limit computational adversary to formal adversary
- One way to represent a formal adversary is by its possible *attacks*
  - Sequence of (adversary) queries, (participant) responses

$$\mathtt{R}_0 \quad \mathtt{Q}_1 \quad \mathtt{R}_1 \quad \ldots \quad \mathtt{Q}_{n-1} \quad \mathtt{R}_{n-1} \quad \mathtt{Q}_n \quad \mathtt{R}_n$$

- Each query $\mathtt{Q}_i$ must be *derivable* from initial adversary knowledge (S_0), all previous responses
- If S is a set of messages, $C\,[S]$ is the smallest set so that
  - $S \subseteq C\,[S]$,
  - S is closed under pairing, separation
  - S is closed under encryption, decryption with keys in S
- Want that

$$\mathtt{Q}_i \in C\left[S_0 \cup \{\mathtt{R}_0, \mathtt{R}_1, \ldots \mathtt{R}_{i-1}\}\right]$$

Generalizing Formal Adversary

- Two reasons to generalize constraint slightly
  1. Protocol independence
     - Don't know what $\mathtt{R}_0$, $\mathtt{R}_1 \ldots \mathtt{R}_{i-1}$ will look like
     - Wouldn't want a protocol-specific result anyway
  2. Initial knowledge
     - Don't know what will be in S_0
     - *Will* contain public values, its own nonces, etc.
     - Might contain arbitrary other things
- More general constraint on adversary:

If formal adversary "knows" messages in set S, can only make messages in $C[S]$

- Assumption: S contains all public values (names, public keys, etc.)

Ideal Encryption — Take 1

- How do we enforce this constraint? Through cryptography!
- An encryption operation is *ideal* if

$$\begin{array}{l} \forall\, \mathtt{A}_{PPT}, \\ \forall\, \text{sets of messages } S \\ \forall\, M \notin C\left[S\right], \\ \forall\, c > 0,\ \forall\ \text{s.l. } k, \\ \Pr[m \leftarrow \mathtt{A}(S) : m = M] \leq \frac{1}{k^c} \end{array}$$

- Wait a second here... This makes no sense!

Message Encoding

- Must bring formal messages down into the world of bit-strings
- Let $[\![M]\!]_k$ be the encoding of M
  - Function from M to distributions on bit-strings
- Defined recursively:
  - $[\![N]\!]_k$ is a random k-bit string
  - $[\![K]\!]_k$ generated by key generation algorithm (run on 1^k)
  - $[\![A]\!]_k$ can be defined any way you like
  - $[\![M_1\ M_2]\!]_k$ is the (tagged) concatenation of $[\![M_1]\!]_k$ and $[\![M_2]\!]_k$
  - $[\![\{\!|M|\!\}_K]\!]_k$ made from $[\![M]\!]_k$ and $[\![K]\!]_k$ using arbitrary public-key encryption algorithm
    - Can think of encryption algorithm as parameter of conversion operation

Ideal Encryption — Take 2

- An encryption operation is *ideal* if, when used in $[\![\cdot]\!]_k$:

$$\begin{array}{l} \forall\, \mathtt{A}_{PPT}, \\ \forall\, \text{sets of messages } S, \\ \forall\, M \notin C\,[S]\,, \\ \forall\, c > 0,\ \forall\ \text{s.l. } k, \\ \Pr[s \leftarrow [\![S]\!]_k\,;\, m \leftarrow \mathtt{A}(1^k, s) : m \in [\![M]\!]_k] \leq \frac{1}{k^c} \end{array}$$

- If M is atomic, $[\![M]\!]_k$ has one element
- If M is compound, $[\![M]\!]_k$ may be non-trivial distribution

Adversary Reduction

- Definition discusses a single query of the adversary
  - *If* computational adversary creates valid encoding, *then* only likely messages are in closure
- Adversary gets to make n queries!
- But n (# of queries) is constant with respect to k
- Union bound demonstrates closure respected over all n queries
- Thus: if computational adversary only creates valid encodings, must mimic formal adversary

- Says nothing about garbage queries
  - Would require discussion of honest participants
  - Subject of another (later) talk

Overview

- Motivation
- A security definition
- Implementation and proof

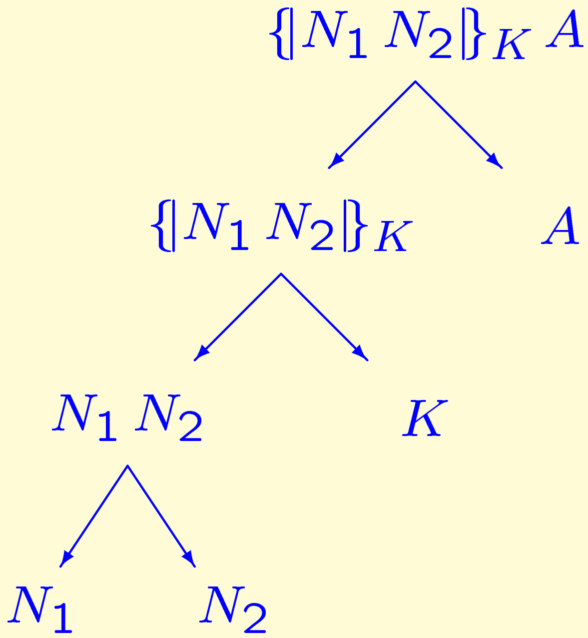
Implementing Ideal Encryption

- Ideal cryptography no good unless it can be achieved
- Thm: Let $\mathsf{E}_e(m)$ produce two things:
  - $c \leftarrow \mathsf{E}'_e(m)$, where E' is a (one-pass) GM-secure encryption scheme, and
  - $p \leftarrow \mathtt{P}(c, \langle m, r, e \rangle, O(c))$, where
    - $\mathtt{P}$ is the prover for a NIZK PK system,
    - r is the randomness used to make c, and
    - $O(\cdot)$ is the random oracle,

  Then E is ideal.

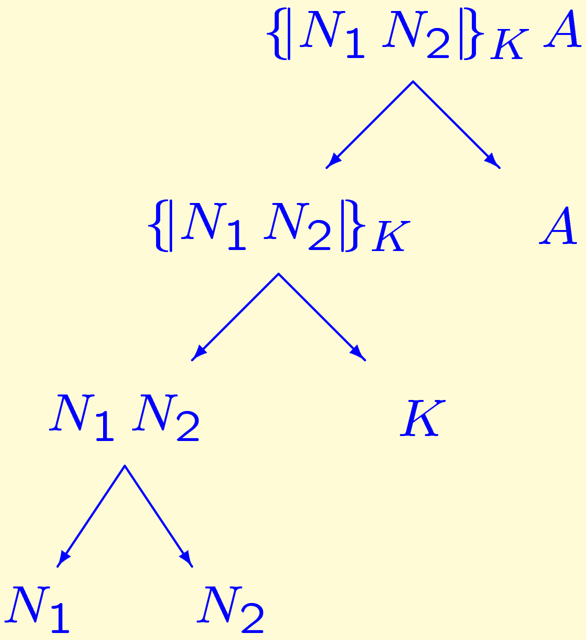
Proof (1/4)

Look at parse tree of M.
If all paths from root to leaf go through element in $C\,[S]$, root is in closure.
Why? If both children in $C\,[S]$, then parent in $C\,[S]$.



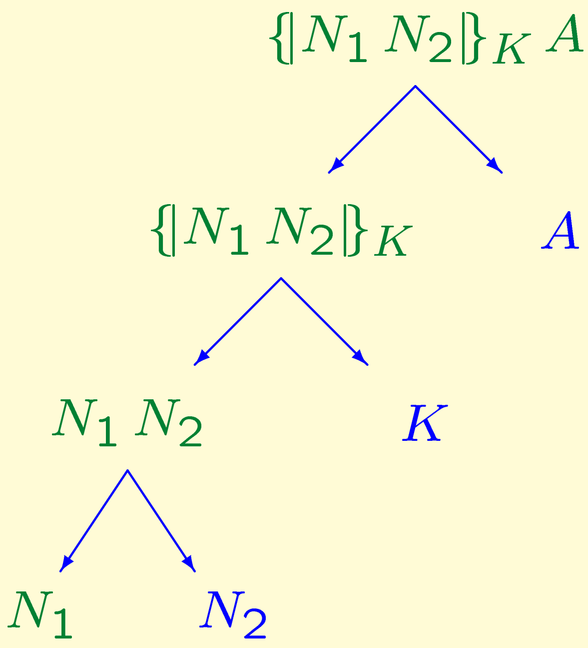
Proof (2/4)

If all paths go through $C[S]$, then root is in $C[S]$. Know this is not the case. Hence, at least one path does not go through $C[S]$



Proof (3/4)

Along this path: If adversary can create parent, can create both children. Easy to undo pair operation. If adversary can create ciphertext, can create plaintext. (Thanks to NIZK PK in encryption.) Hence, adversary can create some leaf not in $C\,[S]$.



Proof (4/4)

- Adversary can create encoding of some atomic element M' not in $C[S]$
- M' cannot be public information
  - Must be nonce or private key
- Two cases
- Case 1: M' is component of something in $C[S]$
  - Example: $\{\!|N_1|\!\}_K \in C[S]$ even though $N_1 \notin C[S]$
  - Then M' must be only plaintext of encryptions
  - Hence, adversary able to break encryption scheme
- Case 2: M' *not* component of something in $C[S]$
  - Then adversary able to guess random nonce/key from independent values

Conclusions

- Ideal encryption can be instantiated. This means...
- Possible to limit computational adversary to attacks available to formal adversary. This means...
- The Dolev-Yao model *might* have computational semantics, which would mean...
- Automatic verification of cryptographic protocols (to our satisfaction) might be possible

Open Questions

- Other cryptographic operations
  - Signatures, symmetric encryption, etc.
- How to deal with honest participants?
  - Behavior in error cases usually unspecified
  - Error codes might leak information
- How does definition of ideal cryptography relate to other definitions?
  - Definition seems similar to "plaintext-aware" encryption
  - Construction seems similar to constructions of schemes secure against CCA

Miscellany

- Written in LaTeX. Presented using **ActiveDvi**, the "presentation tool of choice for the discriminating hacker"
- Housewarming party! Saturday night!

Proofs of Knowledge

- Let $L \in NP$ have witness relation W. Then $(\mathtt{P}, \mathtt{V}, \mathtt{S} = (\mathtt{S}_1, \mathtt{S}_2), \mathtt{E})$ is a NIZK PK system for L if:
  1. Everyone has access to common random ("reference") string
  2. Completeness: All true statements can be proved
     - If $x \in L$, $\mathtt{P}$ has access to $(x, w) \in W$, $\mathtt{P}$ can send a p to $\mathtt{V}$ such that $\mathtt{V}(x, w)$ accepts.
  3. Soundness: hard to "prove" false statements
     - If $x \notin L$, adversary $\mathtt{A}$ generates p, chances of $\mathtt{V}(x, w)$ accepting is small.
  4. Simulatability: Adversary gains nothing from $\mathtt{P}$
     - Adversaries act the same whether they access (random string, $\mathtt{P}$) or $(\mathtt{S}_1,\ \mathtt{S}_2)$
     - Hence, no adversary *needs* access to prover
     - Mostly useful for proofs via reductions

Proofs of Knowledge

5. Extractability: If adversary can create fresh proof, then related adversary can produce witness

- If $\mathtt{A}$ can, with probability P, produce a pair (x, p) so that $\mathtt{V}(x, p)$ accepts, then $\mathtt{E}$ (with access to $\mathtt{A}$) can create w so that $(x, w) \in W$.
- For our talk:
  - (x, p) is a ciphertext
  - w is the plaintext
- Hence, if $\mathtt{A}$ can create "new" ciphertext, a related adversary $\mathtt{E}^{\mathtt{A}}$ can create the plaintext