More Security Protocols
CS 239
Computer Security
February 6, 2006

Outline

- Combining key distribution and authentication
- Verifying security protocols

Combined Key Distribution and Authentication

- Usually the first requires the second
  - Not much good to be sure the key is a secret if you don't know who you're sharing it with
- How can we achieve both goals?
  - In a single protocol
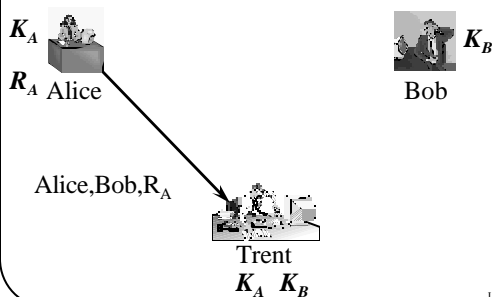 - With relatively few messages

Needham-Schroeder Key Exchange

- Uses symmetric cryptography
- Requires a trusted authority
  - Who takes care of generating the new key
- More complicated than some protocols we've seen

Needham-Schroeder, Step 1

K_A

R_A Alice

K_B

Bob

Alice,Bob,R_A

Trent
K_A K_B

What's the Point of R_A?

- R_A is random number chosen by Alice for this invocation of the protocol
  - Not used as a key, so quality of Alice's random number generator not too important
- Helps defend against replay attacks
- This kind of random number is sometimes called a *nonce*

Needham-Schroeder, Step 2

K_A

R_A Alice

Including R_A prevents replay
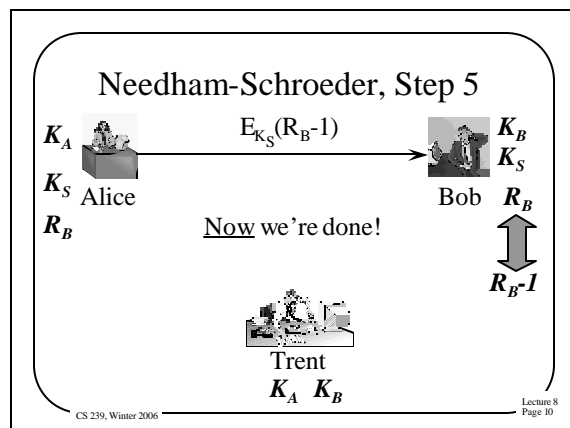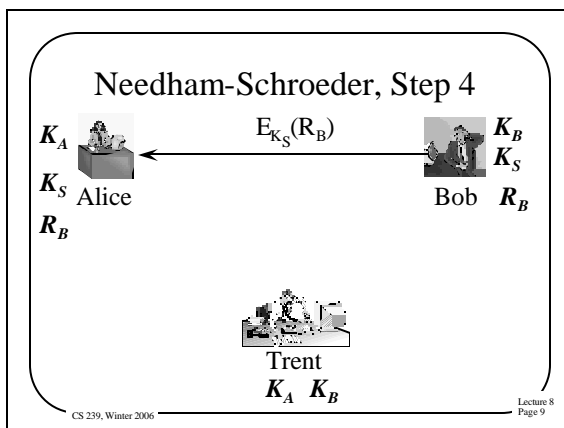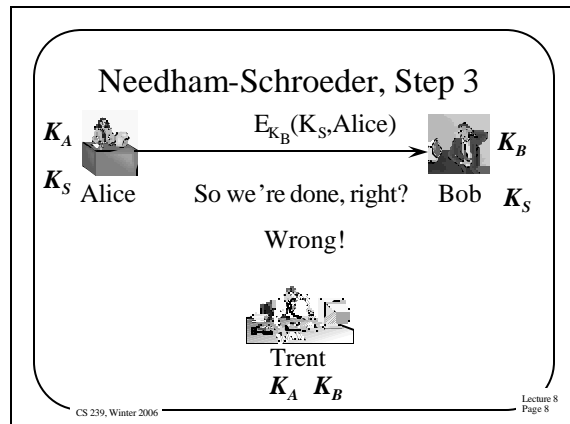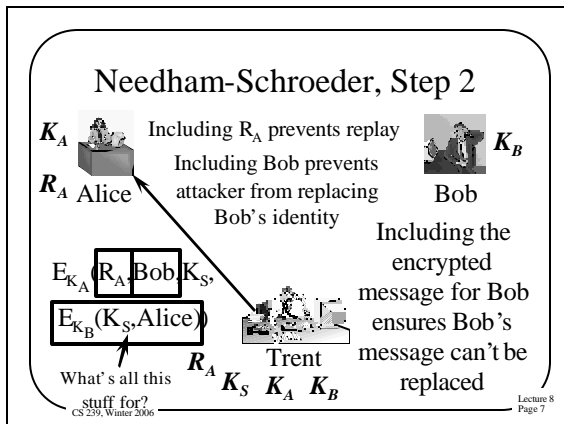
Including Bob prevents attacker from replacing Bob's identity

K_B

Bob

Including the encrypted message for Bob ensures Bob's message can't be replaced

$E_{K_A}(R_A, Bob, K_S,$

$E_{K_B}(K_S, Alice))$

What's all this stuff for?

R_A Trent

K_S K_A K_B

Needham-Schroeder, Step 3

K_A

K_S Alice

$E_{K_B}(K_S, Alice)$

K_B

So we're done, right? Bob K_S

Wrong!

Trent

K_A K_B

Needham-Schroeder, Step 4

K_A

K_S Alice

R_B

$E_{K_S}(R_B)$

K_B
K_S

Bob R_B

Trent

K_A K_B

Needham-Schroeder, Step 5

K_A

K_S Alice

R_B

$E_{K_S}(R_B-1)$

K_B
K_S

Bob R_B

Now we're done!

R_B-1

Trent

K_A K_B

What's All This Extra Stuff For?

K_A

Alice

Alice knows she's talking to Bob

Trent said she was

K_B

Bob

Can Mallory jump in later?

$E_{K_A}(R_A, Bob, K_S,$

$E_{K_B}(K_S, Alice))$

No, only Bob could read the key package Trent created

Trent

K_S K_A K_B

What's All This Extra Stuff For?

K_A

K_S Alice

$E_{K_B}(K_S, Alice)$

K_B

Can Mallory jump in later?

What about those random numbers?

messages will use K_S, which Mallory doesn't know

Bob knows he's talking to Alice
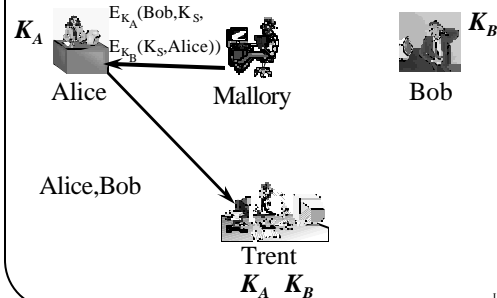
Trent

K_A K_B

Mallory Causes Problems

- Alice and Bob do something Mallory likes
- Mallory watches the messages they send to do so
- Mallory wants to make them do it again
- Can Mallory replay the conversation?
  - Let's try it without the random numbers

Mallory Waits For His Chance

K_A E_{K_A}(Bob,K_S, E_{K_B}(K_S,Alice)) K_B

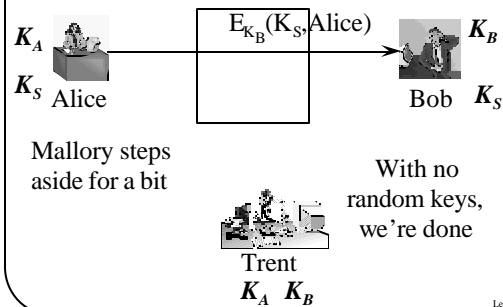Alice   Mallory   Bob

Alice,Bob

Trent
K_A K_B

What Will Alice Do Now?

- The message could only have been created by Trent
- It properly indicates she wants to talk to Bob
- It contains a perfectly plausible key
- Alice will probably go ahead with the protocol

The Protocol Continues

K_A E_{K_B}(K_S,Alice) K_B

K_S Alice   Bob K_S

Mallory steps aside for a bit

With no random keys, we're done
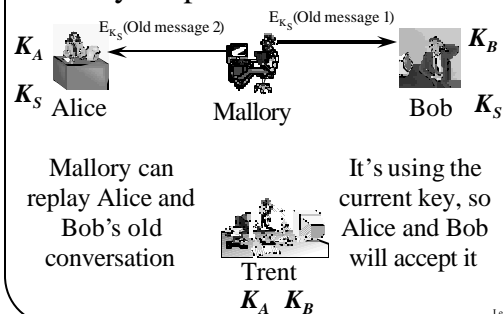
Trent
K_A K_B

So What's the Problem

- Alice and Bob agree K_S is their key
  - They both know the key
  - Trent definitely created the key for them
  - Nobody else has the key
- But . . .

Mallory Steps Back Into the Picture

K_A E_{K_S}(Old message 2) E_{K_S}(Old message 1) K_B

K_S Alice   Mallory   Bob K_S

Mallory can replay Alice and Bob's old conversation

It's using the current key, so Alice and Bob will accept it
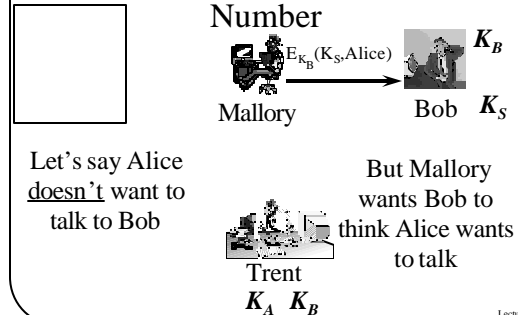
Trent
K_A K_B

How Do the Random Numbers Help?

- Alice's random number assures her that the reply from Trent is fresh
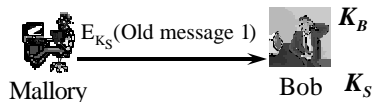- But why does Bob need another random number?

Why Bob Also Needs a Random Number

$E_{K_B}(K_S, Alice)$

K_B

Mallory

Bob K_S

Let's say Alice doesn't want to talk to Bob

But Mallory wants Bob to think Alice wants to talk

Trent

K_A K_B

So What?

$E_{K_S}(Old\ message\ 1)$

K_B

Mallory

Bob K_S

Mallory can now play back an old message from Alice to Bob
And Bob will have no reason to be suspicious

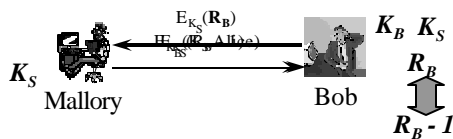Bob's random number exchange assures him that Alice really wanted to talk

So, Everything's Fine, Right?

- Not if any key K_S ever gets divulged
- Once K_S is divulged, Mallory can forge Alice's response to Bob's challenge
- And convince Bob that he's talking to Alice when he's really talking to Mallory

Mallory Cracks an Old Key

$E_{K_S}(R_B)$

$E_{K_S}((R_B, Alice)$

K_B K_S

K_S

Mallory

Bob

R_B

$R_B - 1$

Mallory enlists 10,000 computers belonging to 10,000 grandmothers to crack K_S

Unfortunately, Mallory knows K_S

So Mallory can answer Bob's challenge

Timestamps in Security Protocols

- One method of handling this kind of problem is timestamps
- Proper use of timestamps can limit the time during which an exposed key is dangerous
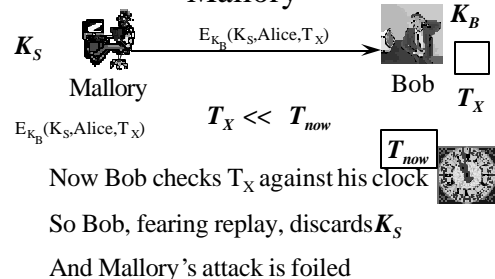- But timestamps have their own problems

Using Timestamps in the Needham-Schroeder Protocol

- The trusted authority includes timestamps in his encrypted messages to Alice and Bob
- Based on a global clock
- When Alice or Bob decrypts, if the timestamp is too old, abort the protocol

Using Timestamps to Defeat Mallory

K_S Mallory $E_{K_B}(K_S, Alice, T_X)$ Bob K_B

T_X

$E_{K_B}(K_S, Alice, T_X)$ $T_X \ll T_{now}$

T_{now}

Now Bob checks T_X against his clock

So Bob, fearing replay, discards K_S

And Mallory's attack is foiled

Problems With Using Timestamps

- They require a globally synchronized set of clocks
  - Hard to obtain, often
  - Attacks on clocks become important
- They leave a window of vulnerability

The Suppress-Replay Attack

- Assume two participants in a security protocol
  - Using timestamps to avoid replay problems
- If the sender's clock is ahead of the receiver's, attacker can intercept message
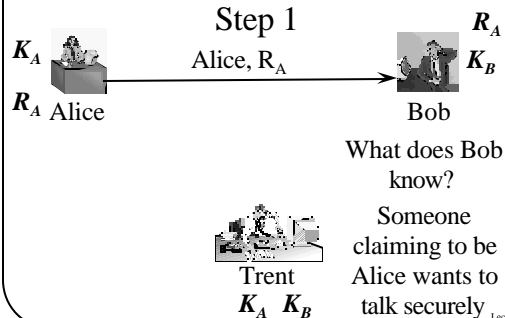 - And replay later, when receiver's clock still allows it

Handling Clock Problems

1). Rely on clocks that are fairly synchronized and hard to tamper
   - Perhaps GPS signals
2). Make all comparisons against the same clock
   - So no two clocks need to be synchronized

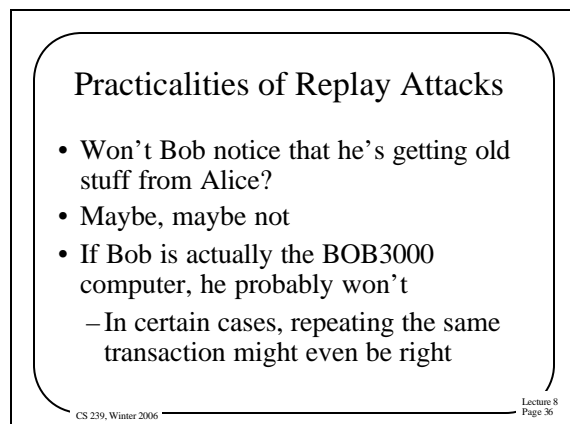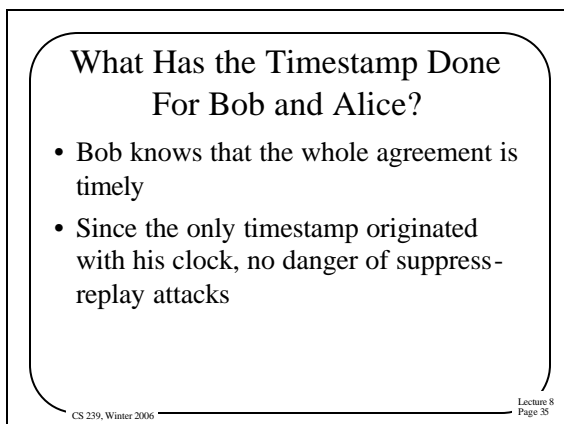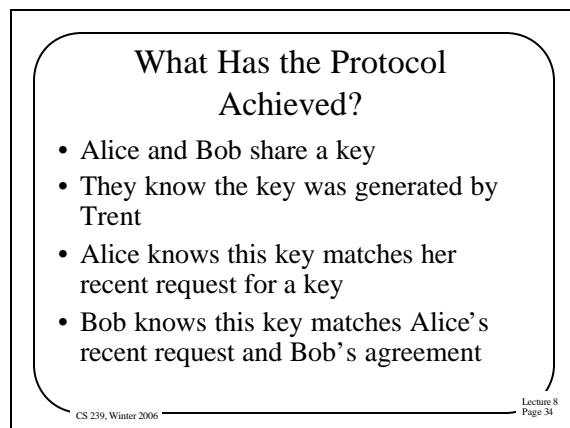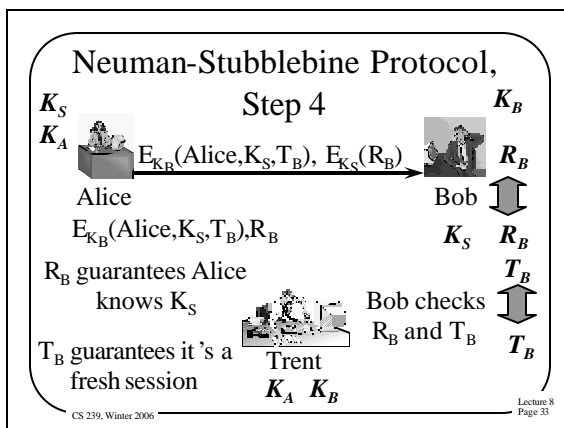## Neuman-Stubblebine Protocol, Step 1

K_A Alice, R_A R_A K_B

R_A Alice   Bob

What does Bob know?

Someone claiming to be Alice wants to talk securely

Trent K_A K_B

5

Neuman-Stubblebine Protocol, Step 2

K_A

R_A Alice

R_A
K_B
Bob,R_B, Bob R_B
T_B
$E_{K_B}(Alice,R_A,T_B)$

Trent knows Bob thinks Alice wants to talk to him But does she really?

Trent
K_A K_B

Alice,R_A,T_B

Neuman-Stubblebine Protocol, Step 3

K_A

Bob,R_A,K_S,T_B

R_A Alice

K_B
R_B
$E_{K_A}(Bob,R_A,K_S,T_B)$, Bob T_B
$E_{K_B}(Alice,K_S,T_B)$,R_B

Alice knows:
1. Bob heard her message
2. Trent created a new key

Trent
K_A K_B

K_S

Alice,R_A,T_B

Neuman-Stubblebine Protocol, Step 4

K_S
K_A

$E_{K_B}(Alice,K_S,T_B)$, $E_{K_S}(R_B)$

K_B
R_B

Alice Bob

$E_{K_B}(Alice,K_S,T_B)$,R_B

K_S R_B
T_B

R_B guarantees Alice knows K_S

Bob checks R_B and T_B

T_B guarantees it's a fresh session

Trent
K_A K_B

T_B

What Has the Protocol Achieved?

- Alice and Bob share a key
- They know the key was generated by Trent
- Alice knows this key matches her recent request for a key
- Bob knows this key matches Alice's recent request and Bob's agreement

---

What Has the Timestamp Done For Bob and Alice?

- Bob knows that the whole agreement is timely
- Since the only timestamp originated with his clock, no danger of suppress-replay attacks

---

Practicalities of Replay Attacks

- Won't Bob notice that he's getting old stuff from Alice?
- Maybe, maybe not
- If Bob is actually the BOB3000 computer, he probably won't
  - In certain cases, repeating the same transaction might even be right

What Else Can You Do With Security Protocols?

- Secret sharing
- Fair coin flips and other games
- Simultaneous contract signing
- Secure elections
- Lots of other neat stuff

Verifying Security Protocols

- Security protocols are obviously very complicated
- And any flaw in the protocol can be very expensive
- Thus, verifying their correctness is of great value
- How to do it?

Basic Approaches to Verifying Protocols

- Use standard specification and verification languages and tools
- Use expert systems
- Use logics for the analysis of knowledge and beliefs
- Use formal methods based on algebraic term-rewriting properties of cryptography

Using Standard Specification and Verification Tools

- Treat protocol as a computer program and prove its correctness
- The oldest approach
- Using
  - Finite state machines
  - First-order predicate calculus
  - Specification languages

Problems With the Approach

- Very laborious
- Worse, correctness isn't the same as security
  - The correctness you prove may not have even considered the possibility of certain attacks
- Too many protocols that have been "proven" have had security problems

Using Expert Systems

- Develop an expert system that knows a lot about security protocols
- Run it against proposed protocols
- In particular, use the expert system to determine if the protocol can reach an undesirable state
  - Such as exposing a secret key

Problems With the Expert System Approach

- Good at identifying flaws
  - Provided they are based on already known problems
- Not so good at proving correctness or security
- Or at uncovering flaws based on new attacks

Using Belief and Knowledge Logics

- An increasingly popular approach
- Describe certain properties that a security protocol should have
- Use logic to demonstrate the presence (or absence) of those properties

BAN Logic

- Named for its creators (Burrows, Abadi, and Needham)
- The most popular method of this kind
- Used to reason about authentication
  - Not other aspects of security
- Allows reasoning about beliefs in protocols

Sample BAN Logic Statements

- Alice believes X.
- Alice sees X.
- Alice said X.
- X is fresh.

Steps in Applying BAN Logic

- Convert protocol to an idealized form
- Add all assumptions about initial state
- Attach logical formulae to the statements
- Apply logical postulates to the assertions and assumptions to discover the beliefs of protocol parties

What Can BAN Logic Do?

- Discover flaws in protocols
  - Found flaws in Needham-Schroeder
- Discover redundancies
  - In Needham-Schroeder, Kerberos, etc.

Critiques of BAN Logic

- Translations into idealized protocols may not reflect the real protocol
- Doesn't address all important security issues for protocols
- Some feel that BAN logic can deduce characteristics that are obviously false

Using Algebraic Term-Rewriting Modeling Methods

- Model the protocol as an algebraic system
- Express the state of the participants' knowledge about the protocol
- Analyze the attainability of certain states

Use of These Methods

- NRL Protocol Analyzer
  - Has discovered flaws in several protocols
- A relatively new method
- Weakest link seems to be formalizing protocol into an algebraic system

Specialized Approaches

- Stubblebine & Gligor's method of modeling weak crypto checksums
  - Found problems in Kerberos and Privacy-Enhanced Mail
  - Not useful for other types of analysis
- Woo-Lam's approach for key distribution protocols
- Pfitzmann's method for digital signatures
- There are others

An Entirely Different Approach

- Instead of using formal methods to verify security protocols,
- Use them to develop such protocols
- Some early work done using this approach
- Not clear if it will be fruitful

Bottom Line on Security Protocol Analysis

- Has been successful in finding some problems
- No one believes existing methods can find all problems
- Some knowledgeable observers think no method will ever be able to find all problems
- So, a useful tool, but not a panacea
- Research in this area continues