More Security Protocols
CS 239
Computer Security
February 4, 2004

Outline

- Combining key distribution and authentication
- Verifying security protocols

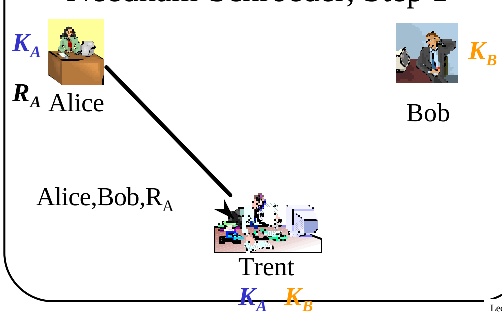
Combined Key Distribution and Authentication

- 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



What's the Point of R_A?

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

Needham-Schroeder, Step 2

K_A R_A Alice

Including R_A prevents replay

Including Bob prevents attacker from replacing Bob's identity

Bob K_B

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

What's all this stuff for?

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

Trent K_S K_A K_B

Needham-Schroeder, Step 3

K_A K_S Alice

$E_{K_B}(K_S, Alice)$

Bob K_B K_S

So we're done, right?

Wrong!

Trent K_A K_B

Needham-Schroeder, Step 4

K_A K_S R_B Alice

$E_{K_S}(R_B)$

Bob K_B K_S R_B

Trent K_A K_B

Needham-Schroeder, Step 5

K_A K_S R_B Alice

$E_{K_S}(R_B\text{-}1)$

Bob K_B K_S R_B R_B-1

Now we're done!

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

Bob K_B

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

$E_{K_B}(K_S, Alice)$

Bob K_B

Bob knows he's talking to Alice

Trent said she was

Can Mallory jump in later?

No, all later messages will use K_S, which Mallory doesn't know

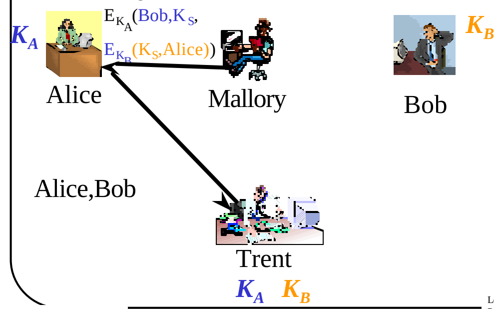
What about those random numbers?

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

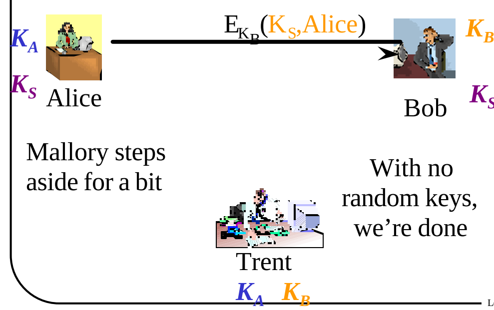
Mallory Waits For His Chance



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

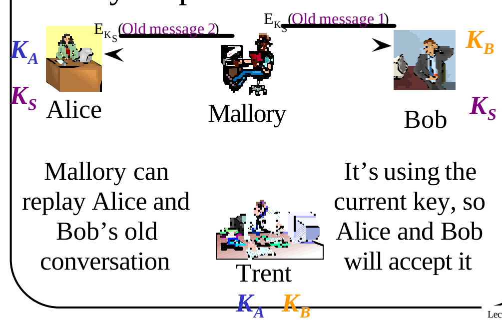
The Protocol Continues



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 . . .

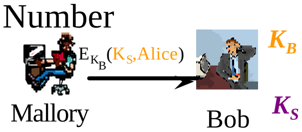
Mallory Steps Back Into the Picture



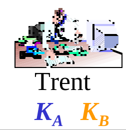
How Do the Random Numbers Help?

- 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



Let's say Alice <u>doesn't</u> want to talk to Bob



But Mallory wants Bob to think Alice wants to talk

So What?

Mallory can now play back an old message from Alice to Bob

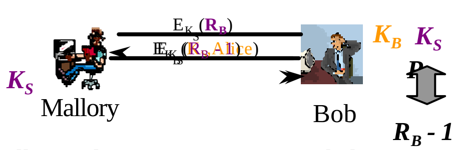
And Bob will have no reason to be suspicious

Bob's random number exchange assured 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



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

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

K_B

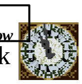
Mallory

Bob T_X

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

$T_X << T_{now}$

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

R_A Alice

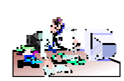
Alice, R_A

R_A

K_B

Bob

What does Bob know?

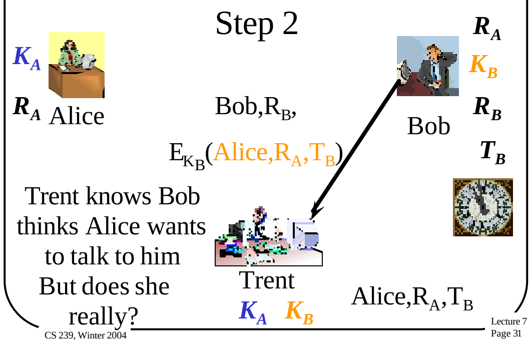


Trent

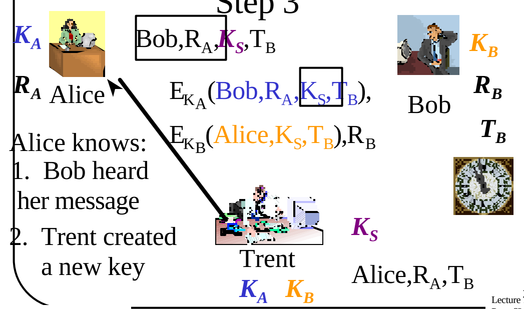
K_A K_B

Someone claiming to be Alice wants to talk securely

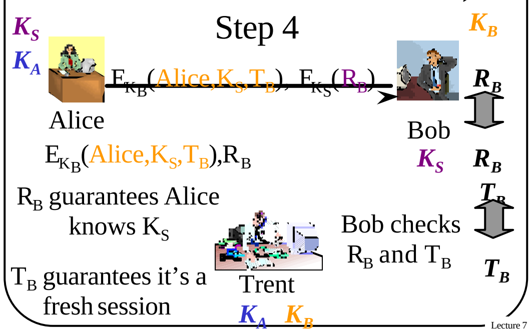
Neuman-Stubblebine Protocol, Step 2



Neuman-Stubblebine Protocol, Step 3



Neuman-Stubblebine Protocol, Step 4



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

What Else Can You Do With Security Protocols?

- Secret splitting and secret sharing
- Fair coin flips and other games
- Simultaneous contract signing
- Secure elections
- Zero knowledge proofs off-line
- Lots of other neat stuff

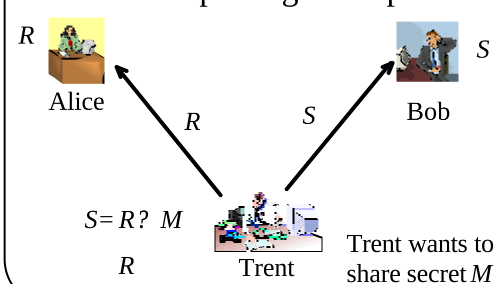
Secret Splitting and Secret Sharing

- What if we have a secret that we need to recover later?
- We need to have it in other people’s hands
- But we don’t want anyone to be able to tell the secret

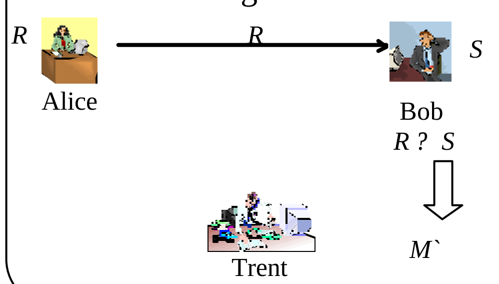
Secret Splitting

- Divide the secret among two or more people
- They can combine to retrieve the secret
- But neither can guess the secret themselves

Secret Splitting Example



Recovering the Secret



What If We Want To Do This Securely?

- What cryptographic steps would we perform to ensure security?
- That only Alice and Bob have secret components
- That they have components of the real secret
- What about ensuring that Alice and Bob both learn the secret if either does?

Secret Sharing

- Say we have three participants
  - Alice, Bob, Carol
- Can we arrange that:
  - None of them know the secret alone
  - Any pair of them can produce the secret
- Yes, using various secret sharing protocols

Bit Commitment

- Alice wants to make a choice now
- And prove to Bob what that choice was
- Without telling him the choice now
- How can Bob be sure that Alice isn't cheating?

Basic Bit Commitment

E_{K_S}
b
R

Alice

R

$E_{K_S}(R,b)$


R
Bob

$E_{K_S}(R,b)$ $E_{K_S}(R,b)$

Bob can't tell yet what bit Alice chose

Since Bob doesn't have E_{K_S}

Now Alice Claims the Bit Was 1

E_{K_S}
b
R

Alice

b == 1

E_{K_S}


R
E_{K_S}
Bob

$E_{K_S}(R,b)$ $E_{K_S}(R,b)$

How does Alice prove it?

If *b == 1* , Alice told the truth *R,b*

Why Does This Work?

- Bob can't learn what *b* was until Alice tells him K_S
- Alice gives Bob a cryptographic package that she can't change
- Since the package includes *R*, Alice can't generate two keys, one for 0 and the other for 1

Making This Work Over the Network

- What would we have to do if Mallory was hanging around trying to screw things up?
- What if we wanted to keep the value of *b* secret from Mallory?
- What if we wanted to ensure that Mallory couldn't replace Alice's choice?

Fair Coin Flips

- Two participants cryptographically "flip a coin"
- Based on clever use of bit commitment
  - "Cut and choose"
- Basic version assumes no interfering third party
- And no need for secrecy
- Similar approaches can work for other games of chance

Simultaneous Contract Signing

- Alice and Bob want to sign a contract
  - But only if each is sure the other also signs
- Basic method uses an arbitrator
- Non-arbitrated cryptographic method uses probabilistic outcome

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