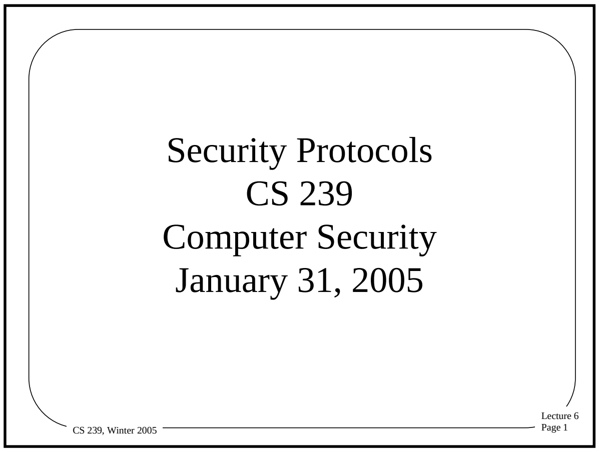
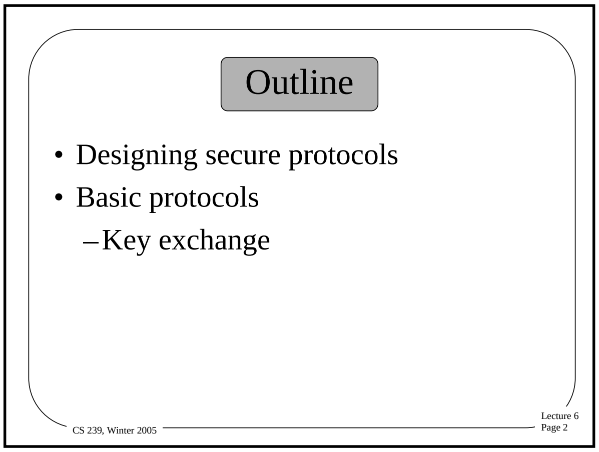
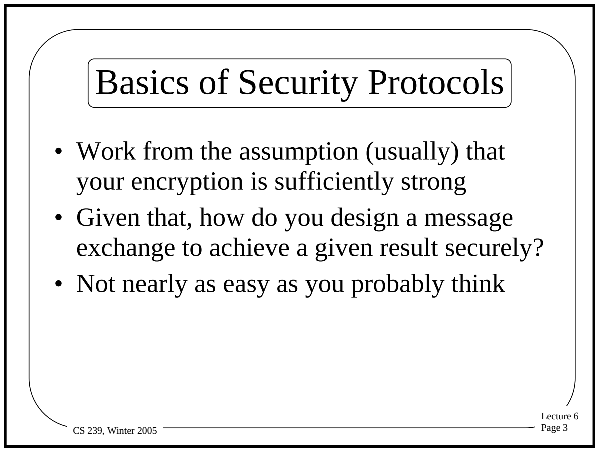


Security Protocols
CS 239
Computer Security
January 31, 2005



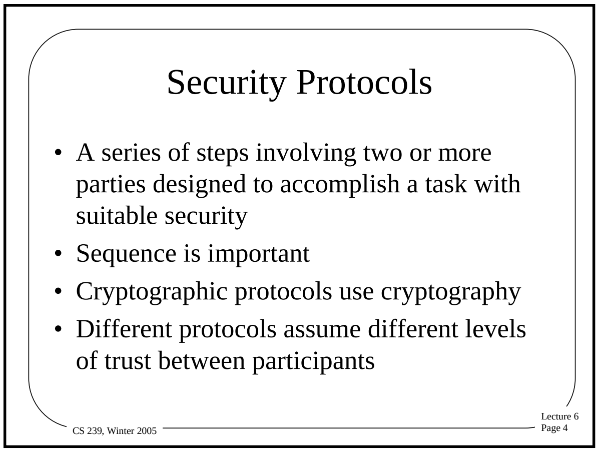
Outline

- Designing secure protocols
- Basic protocols
  - Key exchange



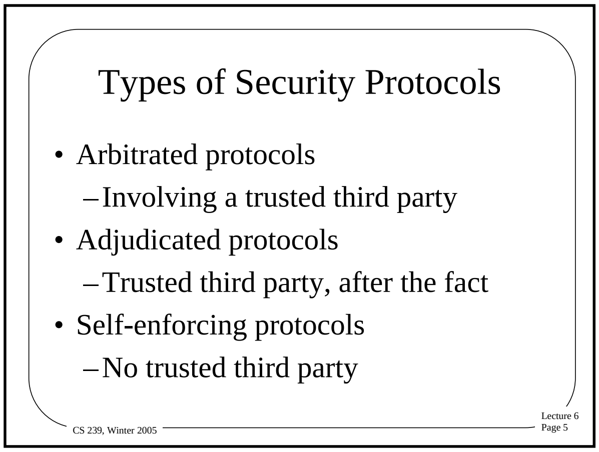
Basics of Security Protocols

- Work from the assumption (usually) that your encryption is sufficiently strong
- Given that, how do you design a message exchange to achieve a given result securely?
- Not nearly as easy as you probably think



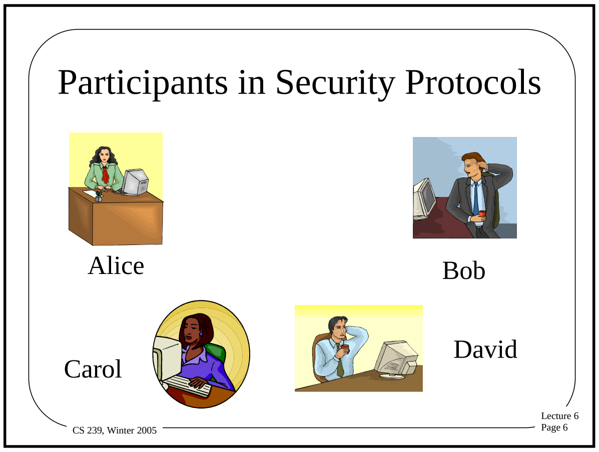
Security Protocols

- A series of steps involving two or more parties designed to accomplish a task with suitable security
- Sequence is important
- Cryptographic protocols use cryptography
- Different protocols assume different levels of trust between participants



Types of Security Protocols

- Arbitrated protocols
  - Involving a trusted third party
- Adjudicated protocols
  - Trusted third party, after the fact
- Self-enforcing protocols
  - No trusted third party



Participants in Security Protocols

And the Bad Guys

Eve

Who only listens passively

And sometimes Alice or Bob might cheat

Mallory

Who is actively malicious

Trusted Arbitrator

Trent

A disinterested third party trusted by all legitimate participants

Arbitrators often simplify protocols, but add overhead

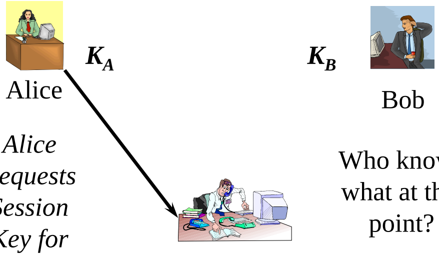
Key Exchange Protocols

- Often we want a different encryption key for each communication session
- How do we get those keys to the participants?
  - Securely
  - Quickly
  - Even if they’ve never communicated before

Key Exchange With Symmetric Encryption and a Arbitrator

- Alice and Bob want to talk securely with a new key
- They both trust Trent
  - Assume Alice & Bob each share a key with Trent
- How do Alice and Bob get a shared key?

Step One



Alice K_A

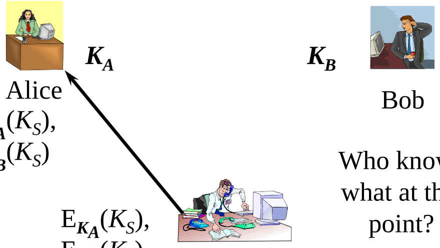
K_B Bob

Alice Requests Session Key for Bob

Who knows what at this point?

K_A Trent K_B

Step Two



Alice K_A

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

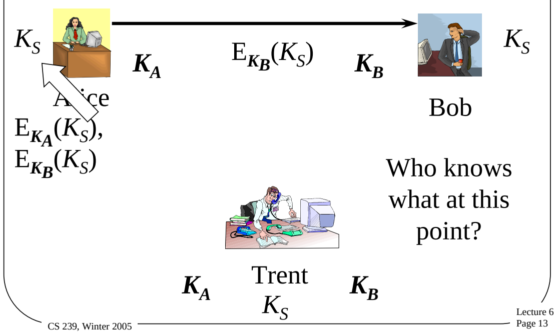
K_B Bob

Who knows what at this point?

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

K_A Trent K_S K_B

Step Three



Who knows what at this point?

What Has the Protocol Achieved?

- Alice and Bob both have a new session key
- The session key was transmitted using keys known only to Alice and Bob
- Both Alice and Bob know that Trent participated
- But there are vulnerabilities

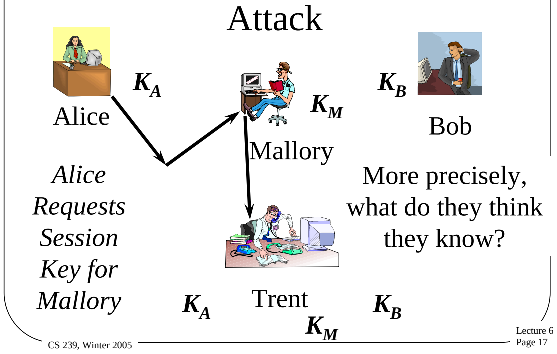
Problems With the Protocol

- What if the initial request was grabbed by Mallory?
- Could he do something bad that ends up causing us problems?
- Yes!

The Man-in-the-Middle Attack

- A class of attacks where an active attacker interposes himself secretly in a protocol
- Allowing alteration of the effects of the protocol
- Without necessarily attacking the encryption

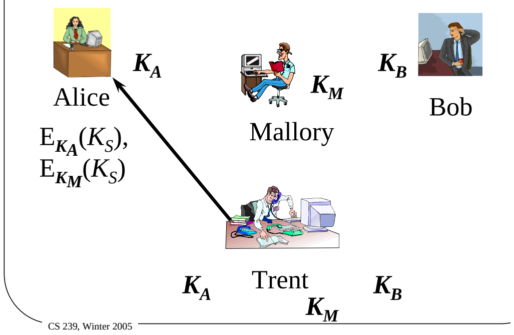
Applying the Man-in-the-Middle Attack

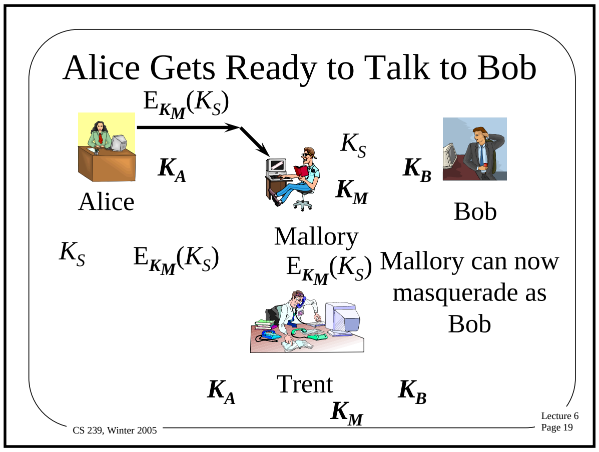


Alice Requests Session Key for Mallory

More precisely, what do they think they know?

Trent Does His Job





Alice Gets Ready to Talk to Bob

$E_{K_M}(K_S)$

K_S

K_A

K_M

K_B

Alice

Bob

Mallory

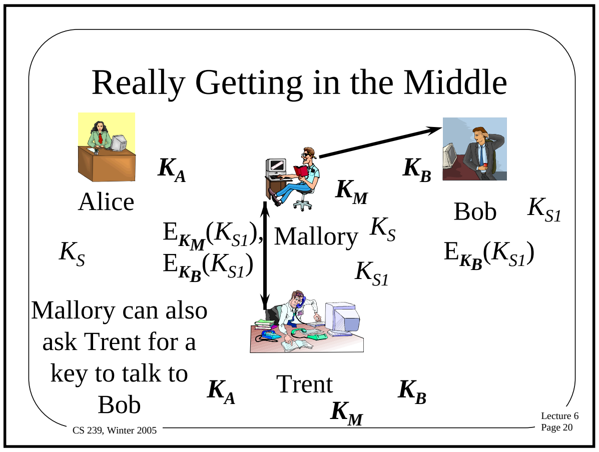
K_S $E_{K_M}(K_S)$

$E_{K_M}(K_S)$

Mallory can now masquerade as Bob

K_A Trent K_B

K_M



Really Getting in the Middle

K_A

K_M

K_B

Alice

Bob K_{S1}

$E_{K_M}(K_{S1})$,
$E_{K_B}(K_{S1})$

Mallory K_S

K_S

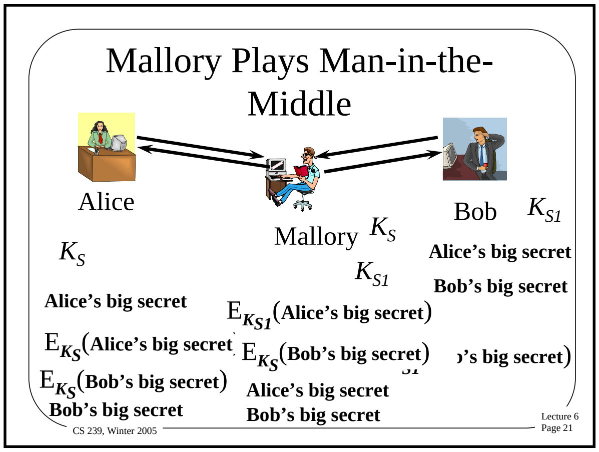
K_{S1}

$E_{K_B}(K_{S1})$

Mallory can also ask Trent for a key to talk to Bob

K_A Trent K_B

K_M



Mallory Plays Man-in-the-Middle

Alice

Bob K_{S1}

Mallory K_S

K_S

Alice's big secret

K_{S1}

Bob's big secret

Alice's big secret

$E_{K_{S1}}$(**Alice's big secret**)

E_{K_S}(**Alice's big secret**

E_{K_S}(**Bob's big secret**)

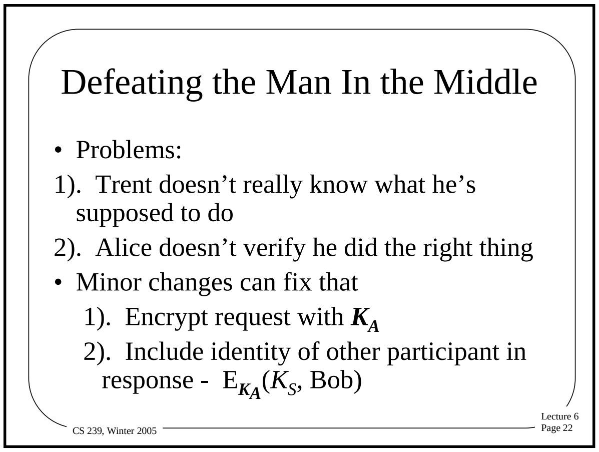
's big secret)

E_{K_S}(**Bob's big secret**)

Bob's big secret

Alice's big secret

Bob's big secret



Defeating the Man In the Middle

- Problems:

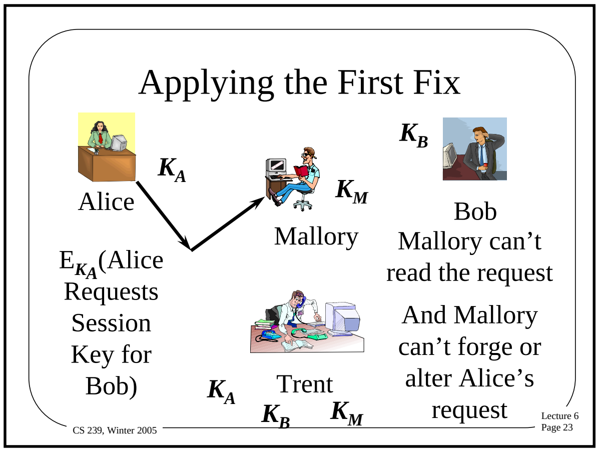
1). Trent doesn't really know what he's supposed to do

2). Alice doesn't verify he did the right thing

- Minor changes can fix that

  1). Encrypt request with K_A

  2). Include identity of other participant in response - $E_{K_A}(K_S$, Bob)



Applying the First Fix

K_B

K_A

K_M

Alice

Bob

Mallory

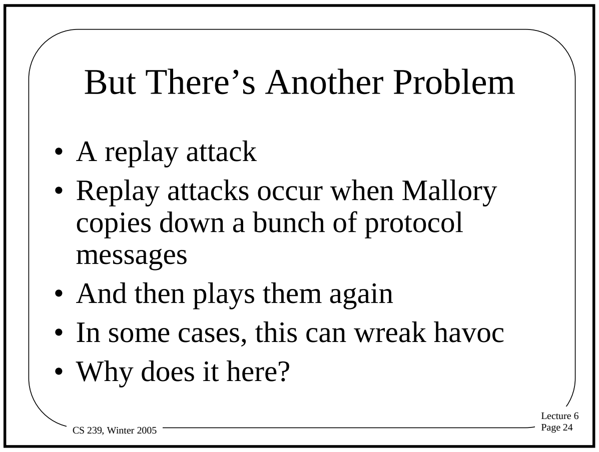
Mallory can't read the request

E_{K_A}(Alice Requests Session Key for Bob)

And Mallory can't forge or alter Alice's request

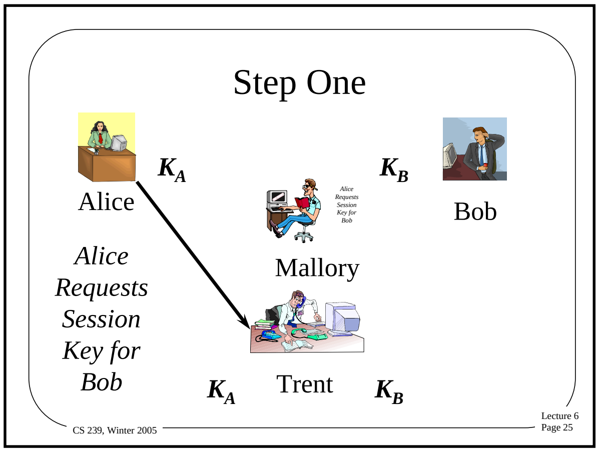
K_A Trent

K_B K_M



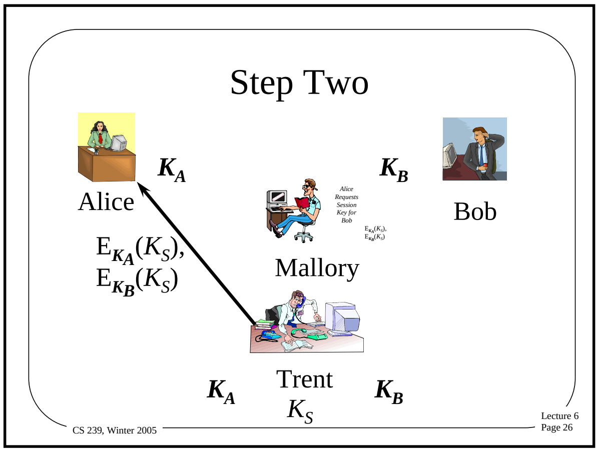
But There's Another Problem

- A replay attack
- Replay attacks occur when Mallory copies down a bunch of protocol messages
- And then plays them again
- In some cases, this can wreak havoc
- Why does it here?

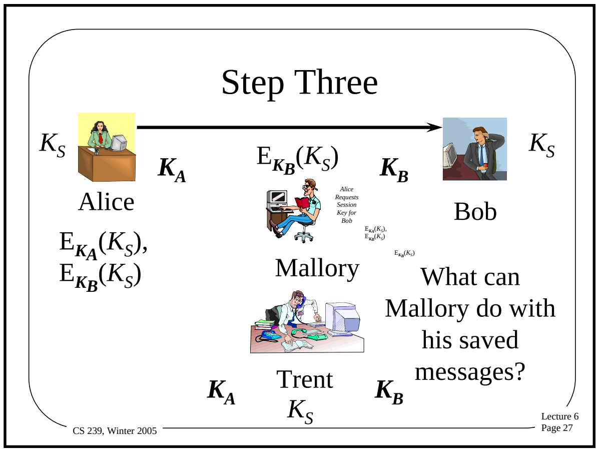


Step One

Alice
K_A
K_B
Bob
Alice Requests Session Key for Bob
Mallory
Alice Requests Session Key for Bob
K_A
Trent
K_B

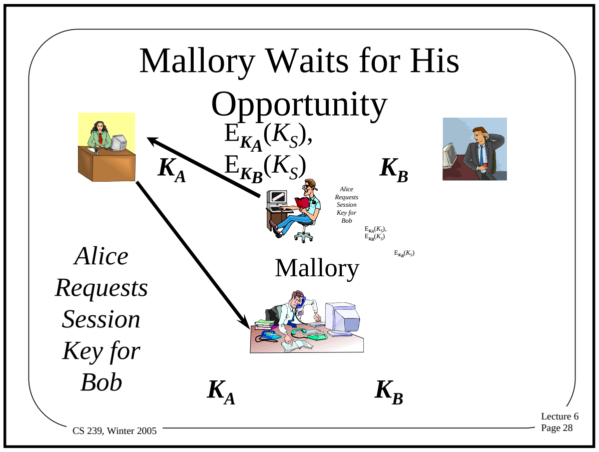


Step Two



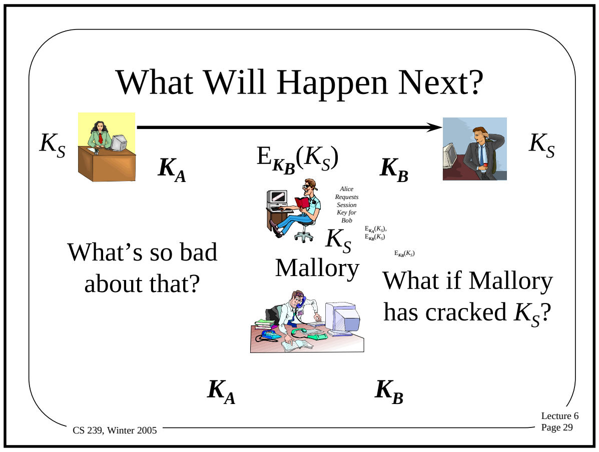
Step Three

What can Mallory do with his saved messages?



Mallory Waits for His Opportunity

$E_{K_A}(K_S)$,
$E_{K_B}(K_S)$
K_A
K_B
Alice Requests Session Key for Bob
$E_{K_A}(K_S)$, $E_{K_B}(K_S)$
$E_{K_B}(K_S)$
Mallory
Alice Requests Session Key for Bob
K_A
K_B



What Will Happen Next?

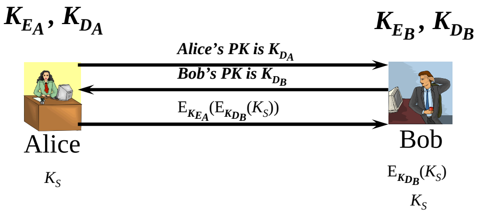
What's so bad about that?

What if Mallory has cracked K_S?

Key Exchange With Public Key Cryptography

- With no trusted arbitrator
- Alice sends Bob her public key
- Bob sends Alice his public key
- Alice generates a session key and sends it to Bob encrypted with his public key, signed with her private key
- Bob decrypts Alice's message with his private key
- Encrypt session with shared session key

Basic Key Exchange Using PK



K_{E_A}, K_{D_A}

K_{E_B}, K_{D_B}

Alice's PK is K_{D_A}

Bob's PK is K_{D_B}

$E_{K_{E_A}}(E_{K_{D_B}}(K_S))$

Alice

K_S

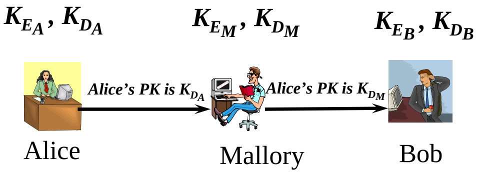
Bob

$E_{K_{D_B}}(K_S)$

K_S

Bob verifies the message came from Alice
Bob extracts the key from the message

Man-in-the-Middle With Public Keys



K_{E_A}, K_{D_A}

K_{E_M}, K_{D_M}

K_{E_B}, K_{D_B}

Alice's PK is K_{D_A}

Alice's PK is K_{D_M}

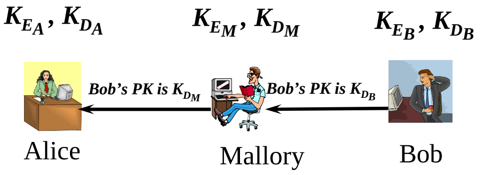
Alice

Mallory

Bob

Now Mallory can pose as Alice to Bob

And Bob Sends His Public Key



K_{E_A}, K_{D_A}

K_{E_M}, K_{D_M}

K_{E_B}, K_{D_B}

Bob's PK is K_{D_M}

Bob's PK is K_{D_B}

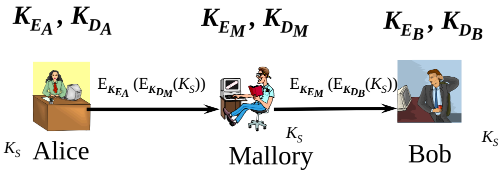
Alice

Mallory

Bob

Now Mallory can pose as Bob to Alice

Alice Chooses a Session Key



K_{E_A}, K_{D_A}

K_{E_M}, K_{D_M}

K_{E_B}, K_{D_B}

$E_{K_{E_A}}(E_{K_{D_M}}(K_S))$

$E_{K_{E_M}}(E_{K_{D_B}}(K_S))$

K_S Alice

K_S

Mallory

Bob

K_S

Bob and Alice are sharing a session key

Unfortunately, they're also sharing it with Mallory

Defeating This Man-in-the-Middle Attack

- Use Rivest and Shamir's *interlock protocol*
- Doesn't require any authorities
- Essentially, send stuff in pieces of an encrypted whole
- The man in the middle has little chance of correctly dealing with pieces

Using the Interlock Protocol

- Alice sends Bob her public key
- Bob sends Alice his public key
- Alice encrypts her message in Bob's public key and sends half of it to Bob
- Bob encrypts his message in Alice's public key and sends half of it to Alice
- Alice sends her other half to Bob

Continuing the Interlock Protocol

- Bob puts Alice's two halves together and decrypts
- Bob sends the other half of his encrypted message to Alice
- Alice puts Bob's halves together and decrypts

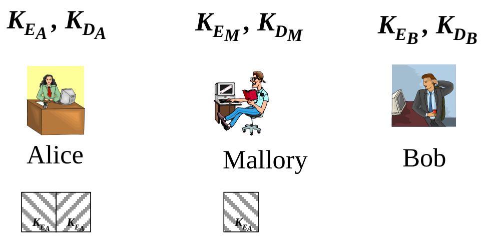
Why Does This Protocol Help?

- Because the man in the middle must provide half of an encrypted message before he gets all of it
- Consider one part of the attack -
  - Mallory wants to translate the message in Alice's public key into Mallory's public key

What Does Mallory Do?

- Mallory has deceptively sent out her public key to Bob and Alice
  - Claiming it's theirs
  - And Mallory knows their public keys
- Alice send Mallory half of an encrypted message
- Now Mallory must send Bob half an encrypted message

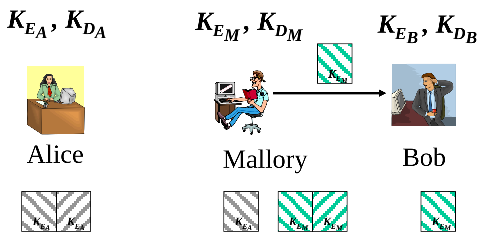
Mallory's Situation



Mallory's Problem

- Mallory can't yet decrypt Alice's message
  - Since he only has half of it
- Mallory must provide Bob two matching halves eventually
  - And one right now
- Mallory's only choice is to generate a new message before he knows the real message

Mallory's Only Option



Why Is This A Problem For Mallory?

- Mallory must now spoof proper contents of Bob and Alice's conversation
- Without knowing the real contents until later
- Bob and Alice are likely to notice problems quickly

Is This Generally Feasible?

- Not really
- Assumes Bob has a useful, unguessable message before Alice's message arrives
- Not really the way the world works
- If Mallory can guess Bob's message, he can play the standard man-in-the-middle game

Diffie/Hellman Key Exchange

- Securely exchange a key
  - Without previously sharing any secrets
- Alice and Bob agree on a large prime n and a number g
  - g should be primitive mod n
- n and g don't need to be secrets

Exchanging a Key in Diffie/Hellman

- Alice and Bob want to set up a session key
  - How can they learn the key without anyone else knowing it?
- Protocol assumes authentication
- Alice chooses a large random integer x and sends Bob $X = g^{x} mod\ n$

Exchanging the Key, Con't

- Bob chooses a random large integer y and sends Alice $Y = g^{y}\ mod\ n$
- Alice computes $k = Y^{x}\ mod\ n$
- Bob computes $k' = X^{y}\ mod\ n$
- k and k' are both equal to $g^{xy} mod\ n$
- But nobody else can compute k or k'

Why Can't Others Get the Secret?

- What do they know?
  - n, g, X, and Y
  - Not x or y
- Knowing X and y gets you k
- Knowing Y and x gets you k'
- Knowing X and Y gets you nothing
  - Unless you compute the discrete logarithm to obtain x or y