Security Protocols
CS 236
On-Line MS Program
Networks and Systems Security
Peter Reiher

Outline

- Designing secure protocols
- Key exchange protocols
- Common security problems in protocols

Basics of Security Protocols

- Assume (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
- Many of the concepts are important in many areas of computer/network security

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

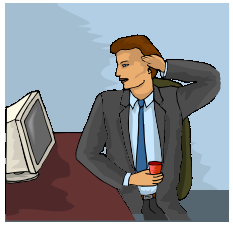
Alice

Bob

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 and may limit applicability

Goals of Security Protocols

- Each protocol is intended to achieve some very particular goal
  - Like setting up a key between two parties
- Protocols may only be suitable for that particular purpose
- Important secondary goal is minimalism
  - Fewest possible messages
  - Least possible data
  - Least possible encryption

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

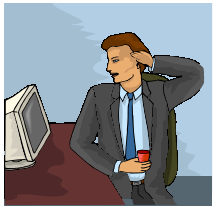
Key Exchange With Symmetric Encryption and an Arbitrator

- 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

K_A

K_B



Alice

Bob

Alice Requests Session Key for Bob

Who knows what at this point?

K_A Trent K_B

Step Two

K_A

K_B

Alice

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

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

K_S

K_A

$E_{K_B}(K_S)$

K_B

K_S

Alice

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

Bob

Who knows what at this point?

K_A

Trent

K_S

K_B

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

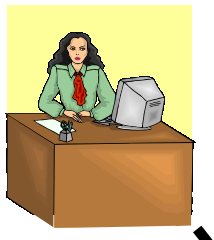
Problems With the Protocol

- 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

K_A

Mallory

K_M

K_B

Bob

Alice Requests Session Key for Mallory

K_A

Trent

K_M

K_B

More precisely, what do they think they know?

Who knows what at this point?

Trent Does His Job

K_A

Alice

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

K_M

Mallory

K_B

Bob

Trent

K_A

K_M

K_B

Alice Gets Ready to Talk to Bob

$E_{K_M}(K_S)$

K_A

Alice

K_S

K_M

Mallory

K_B

Bob

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

Alice

K_M

K_B

Bob K_{S1}

K_S

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

Mallory 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

Mallory

Bob

K_S

Alice's big secret

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

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

Bob's big secret

K_S

K_{S1}

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

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

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

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

Alice's big secret

Bob's big secret

K_{S1}

Alice's big secret

Bob's big secret

$E_{K_{S1}}$(**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 $\boldsymbol{K_A}$
  2). Include identity of other participant in response - $E_{\boldsymbol{K_A}}(K_S, \text{Bob})$

Applying the First Fix

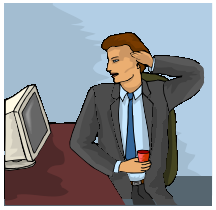
Alice

K_A

Mallory

K_M

K_B



Bob

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

Mallory can't read the request

Trent

K_A K_B K_M

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

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

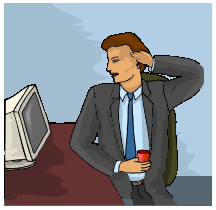
Alice

K_A

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

Mallory

K_B



Bob

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

K_A Trent K_B

Step Two

K_A

Alice

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

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

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

Mallory

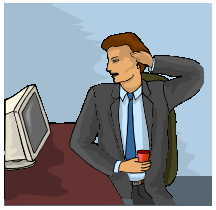
Trent

K_S

K_A

K_B

K_B



Bob

Step Three

K_S

K_A

$E_{K_B}(K_S)$

K_B

K_S

Alice

Bob

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

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

$E_{K_B}(K_S)$

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

Mallory

What can Mallory do with his saved messages?

K_A

Trent

K_S

K_B

Mallory Waits for His Opportunity

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

K_A

K_B

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

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

$E_{K_B}(K_S)$

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

Mallory

K_A

K_B

What Will Happen Next?

K_S

K_A

$E_{K_B}(K_S)$

K_B

K_S

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

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

K_S

Mallory

$E_{K_B}(K_S)$

What's so bad about that?

What if Mallory has cracked K_S?

K_A

K_B