15-441 Computer Networking

Lecture 4 – Applications
DNS, SSL

Outline

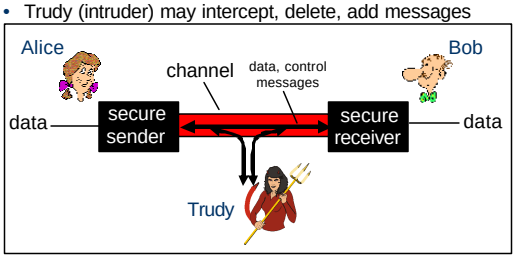
- Security: Cryptography Introduction
- Security: Authentication
- Security: Key Distribution
- DNS

What is Network Security?

- Confidentiality: only sender, intended receiver should "understand" message contents
  - Sender encrypts message
  - Receiver decrypts message
- Authentication: sender, receiver want to confirm identity of each other
- Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- Access and Availability: services must be accessible and available to users

Friends and Enemies: Alice, Bob, Trudy

- Well-known in network security world
- Bob & Alice want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



Who might Bob, Alice be?

- … well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- On-line banking client/server
- DNS servers
- Routers exchanging routing table updates
- Other examples?

There are bad Guys (and Girls) Out There!

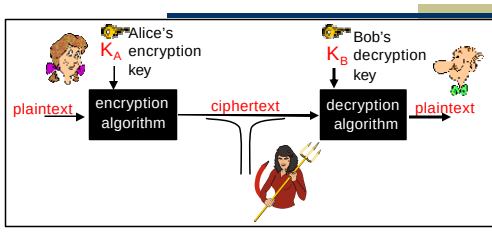
Q: What can a "bad guy" do?

A: A lot!

- *Eavesdrop:* intercept messages
- Actively *insert* messages into connection
- *Impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *Hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *Denial of service*: prevent service from being used by others (e.g., by overloading resources)

more on this later ……

The Language of Cryptography



Symmetric key crypto: sender, receiver keys *identical*

Public-key crypto: encryption key *public*, decryption key *secret* (private)

Symmetric Key Cryptography

Substitution cipher: substituting one thing for another

- monoalphabetic cipher: substitute one letter for another

plaintext: abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfghjklpoiuytrewq

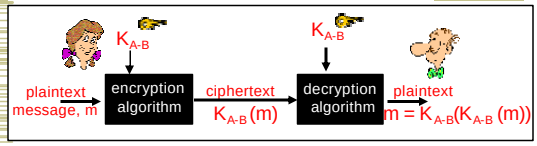
E.g.: **Plaintext: bob. i love you. alice**

ciphertext: nkn. s gktc wky. mgsbc

Q: How hard to break this simple cipher?:

- Brute force (how hard?)
- Other?

Symmetric Key Cryptography



Symmetric key crypto: Bob and Alice both know same (symmetric) key: K_{A-B}

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher
- Q: How do Bob and Alice agree on key value?

Public Key Cryptography

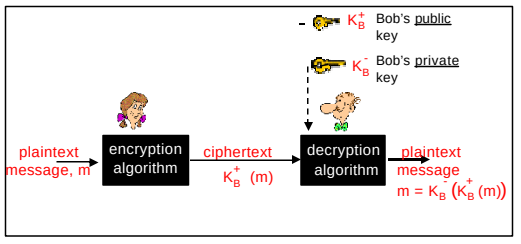
Symmetric Key Crypto

- Requires sender, receiver know shared secret key
- Q: How to agree on key in first place (particularly if never "met")?

Public Key Cryptography

- Radically different approach [Diffie-Hellman76, RSA78]
- Sender, receiver do *not* share secret key
- *Public* encryption key known to *all*
- *Private* decryption key known only to receiver

Public Key Cryptography



Public Key Encryption Algorithms

Requirements:

1. Need $K_B^+(\,)$ and $K_B^-(\,)$ such that

$$K_B^-(K_B^+(m)) = m$$

2. Given public key K_B^+, it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

RSA: Another Important Property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

Use public key first, followed by private key

Use private key first, followed by public key

Result is the same!

Outline

- Security: Cryptography Introduction
- Security: Authentication
- Security: Key Distribution
- DNS

Authentication

"The Doors of Durin, Lord of Moria. Speak, friend, and enter."

- Words on the gate to Moria

Password? → Mellon (Elvish for friend)

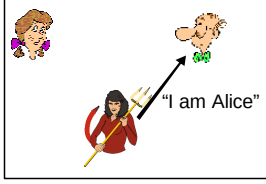
"Too simple for a learned lore master in these suspicious days. Those were happier times."

- Gandalf

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"



In a network, Bob can not "see" Alice, so Trudy simply declares herself to be Alice

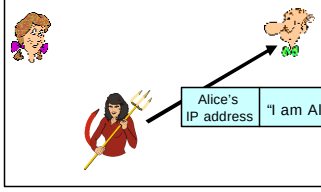
Authentication: Another Try

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

Failure scenario??

Authentication: Another Try

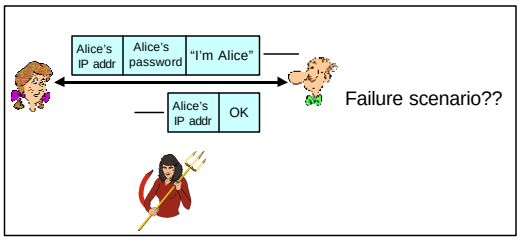
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Trudy can create a packet "spoofing" Alice's address

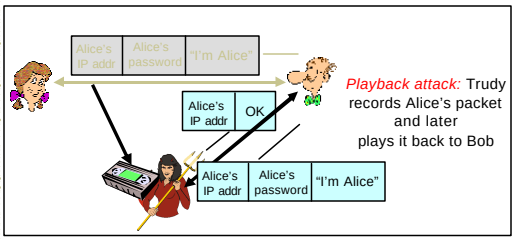
Authentication: Another Try

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



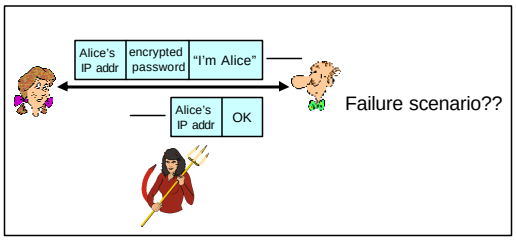
Authentication: Another Try

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



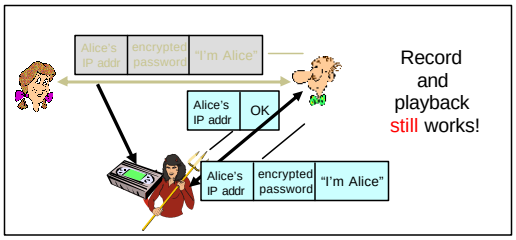
Authentication: Yet Another Try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Authentication: Another Try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.

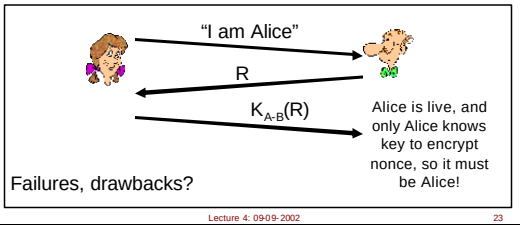


Authentication: Yet Another Try

Goal: avoid playback attack

Nonce: number (R) used only *once-in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key

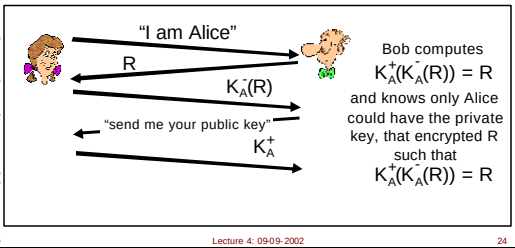


Authentication: ap5.0

ap4.0 requires shared symmetric key

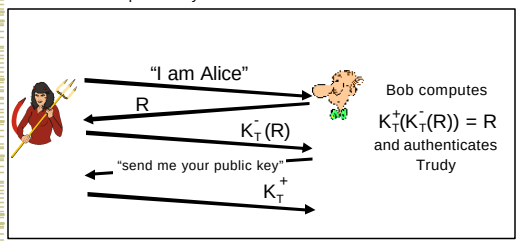
- Can we authenticate using public key techniques?

ap5.0: use nonce, public key cryptography



ap5.0: Security Hole

ap5.0 only as "secure" as the distribution of public keys



Outline

- Security: Cryptography Introduction
- Security: Authentication
- Security: Key Distribution
- DNS

Trusted Intermediaries

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- Trusted key distribution center (KDC) acting as intermediary between entities

Public key problem:

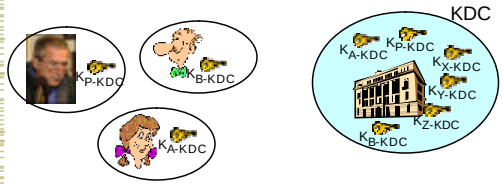
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

- Trusted certification authority (CA)

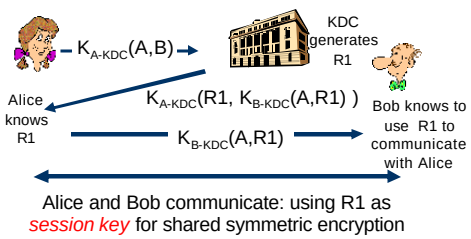
Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.



Key Distribution Center (KDC)

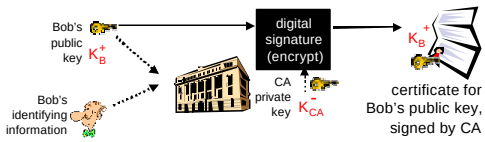
Q: How does KDC allow Bob, Alice to determine shared symmetric secret key to communicate with each other?



Alice and Bob communicate: using R1 as *session key* for shared symmetric encryption

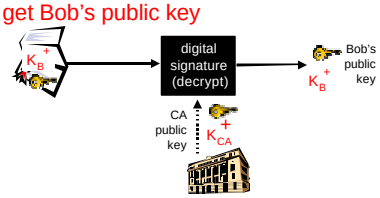
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - Certificate containing E's public key digitally signed by CA – CA says "this is E's public key"



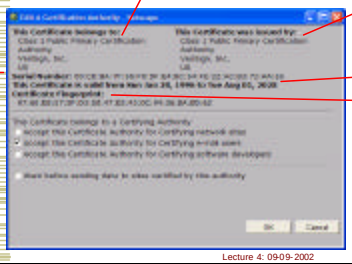
Certification Authorities

- When Alice wants Bob's public key:
  - Gets Bob's certificate (Bob or elsewhere).
  - Apply CA's public key to Bob's certificate, get Bob's public key



Certificate Contents

- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)



- Info about certificate issuer
- Valid dates
- Digital signature by issuer

Secure Sockets Layer (SSL)

- Transport layer security to any TCP-based app using SSL services.
- Used between Web browsers, servers for e-commerce (shttp).
- Security services:
  - Server authentication
  - Data encryption
  - Client authentication (optional)
- Server authentication:
  - SSL-enabled browser includes public keys for trusted CAs.
  - Browser requests server certificate, issued by trusted CA.
  - Browser uses CA's public key to extract server's public key from certificate.
- Check your browser's security menu to see its trusted CAs.

SSL (continued)

Encrypted SSL session:

- Browser generates *symmetric session key*, encrypts it with server's public key, sends encrypted key to server.
- Using private key, server decrypts session key.
- Browser, server know session key
  - All data sent into TCP socket (by client or server) encrypted with session key.
- SSL: basis of IETF Transport Layer Security (TLS).
- SSL can be used for non-Web applications, e.g., IMAP.
- Client authentication can be done with client certificates.

Network Security (Summary)

- Cryptography (symmetric and public)
  - Basic techniques & tradeoffs
- Authentication
  - Common styles of attack
- Key distribution
  - Why needed

…. used in many different security scenarios

- secure email, secure transport (SSL), IP sec, 802.11 WEP

Outline

- Security: Cryptography Introduction
- Security: Authentication
- Security: Key Distribution
- DNS

Naming

- How do we efficiently locate resources?
  - DNS: name → IP address
  - Service location: description → host
- Other issues
  - How do we scale these to the wide area?
  - How to choose among similar services?

Obvious Solutions (1)

Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Single point of update
- Doesn't *scale!*

Obvious Solutions (2)

Why not use /etc/hosts?

- Original Name to Address Mapping
  - Flat namespace
  - /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: machine per domain → machine per user
  - Many more downloads
  - Many more updates

Domain Name System Goals

- Basically building a wide area distributed database
- Scalability
- Decentralized maintenance
- Robustness
- Global scope
  - Names mean the same thing everywhere
- Don't need
  - Atomicity
  - Strong consistency

DNS Records

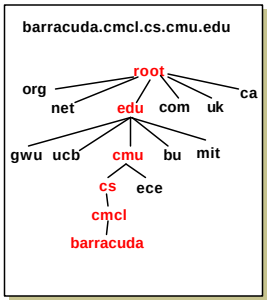
RR format: **(class, name, value, type, ttl)**

- DB contains tuples called resource records (RRs)
  - Classes = Internet (IN), Chaosnet (CH), etc.
  - Each class defines value associated with type

FOR IN class:

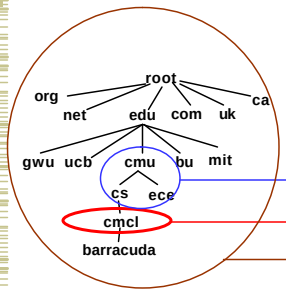
- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is name of authoritative name server for this domain
- Type=CNAME
  - **name** is an alias name for some "canonical" (the real) name
  - **value** is canonical name
- Type=MX
  - **value** is hostname of mailserver associated with **name**

Hierarchical Name Space



- Administrative hierarchy
  - "." as separator
- Host name to address section
  - Top-level domains → edu, gov, ca, us, etc.
  - Sub-domains = subtrees
  - Human readable name = leaf → root path

DNS Design: Zone Definitions



- Zone = contiguous section of name space
- E.g., Complete tree, single node or subtree
- A zone has an associated set of name servers

DNS Design: Cont.

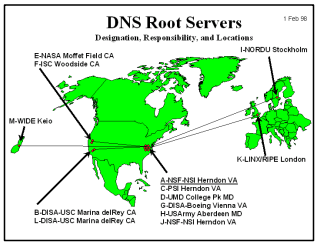
- Zones are created by convincing owner node to create/delegate a subzone
  - Records within zone stored multiple redundant name servers
  - Primary/master name server updated manually
  - Secondary/redundant servers updated by zone transfer of name space
    - Zone transfer is a bulk transfer of the "configuration" of a DNS server – uses TCP to ensure reliability
- Example:
  - CS.CMU.EDU created by CMU.EDU administrators

Servers/Resolvers

- Each host has a resolver
  - Typically a library that applications can link to
  - Local name servers hand-configured (e.g. /etc/resolv.conf)
- Name servers
  - Either responsible for some zone or…
  - Local servers
    - Do lookup of distant host names for local hosts
    - Typically answer queries about local zone

DNS: Root Name Servers

- Responsible for "root" zone
- Approx. dozen root name servers worldwide
  - Currently {a-m}.root-servers.net
- Local name servers contact root servers when they cannot resolve a name
  - Configured with well-known root servers



Next Lecture

- How DNS resolves names
- How well does DNS work today
- HTTP intro and details