Few foils based on Kurose & Ross ©, see: http://www.aw.com/kurose-ross/

My site: http://amir.herzberg.name

Course site: http://hl2.biu.ac.il
Network Security Goals

Confidentiality: only sender, intended receiver should “understand” message contents
- sender encrypts message
- receiver decrypts message
- Privacy: hide `who is doing what with whom`

Authentication: sender, receiver want to confirm identity of each other

Integrity: sender, receiver want to ensure messages are not altered (in transit, or afterwards) without detection

Access and Availability: services must be accessible and available to users
Shared Media (Broadcast) Networks

- Shared media net: all traffic passes thru all computers
  - Mostly Local Area Networks (LAN)
  - E.g. Ethernet, token-ring, Wireless LANs, Cellular...
  - Usually: promiscuous mode listends to all messages on Net

- Shared Media Attack Model:
  - Easy: eavesdropping (sniffing) - passive attack
    - Unless cryptographically protected: encryption
  - Harder (but possible): spoofing - active attack

### Diagram

- **Ring**: Network topology with computers connected in a circle.
- **Bus**: Network topology with computers connected in a line.
- **Star (Hub)**: Network topology with central hub and computers connected to it.
Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice (lovers!) want to communicate “securely”
- Eve (intruder/eavesdropper) may intercept, delete, add messages
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......*
Network Security: Agenda

- Shared Media: Eavesdropping (sniffing), other threats
- Main weapon: cryptography
  - Confidentiality (encryption)
  - Message authentication
  - Signatures and Certificates
- Internet Threats, Attacks and Defenses
  - Secure request/response protocols
  - Secure connection `tunnels`: IP-Sec, SSL/TLS
  - Denial of Service Attacks
  - Firewalls
- Conclusions
Encryption Protects Confidentiality

symmetric key crypto: shared secret key ($K_{E,B} = K_{D,B}$)

public-key crypto: Bob has a public encryption key $K_{E,B}$ and a matching private decryption key $K_{D,B}$
Symmetric key cryptosystem

Symmetric (shared) key crypto: Bob and Alice share key $k$
- Example: One-Time Pad (bitwise XOR): $E_k(m) = k \oplus m$, $D_k(c) = k \oplus m$
  - Attacker can't learn anything new on $m$ (regardless of his speed/time)
  - But: key is as long as total length of messages sent
  - Too long for most scenarios
- Other schemes use shorter keys but are `computationally secure`
- US (NIST) standards:
  - 1977-2000: DES (56 bit key)
  - 2001-????: AES (128 bit key)
Public key cryptography

\[ c = E_{BPub}(m) \]

\[ m = D_{BPriv}(c) = D_{BPriv}(E_{BPub}(m)) \]

**Asymmetric, Public Key Cryptosystem (PKCS):**
Alice knows only Bob’s public key BPub, Bob knows private key BPriv

- Most common PKCS: **RSA**: [Rivest, Shamir, Adelman, 1978]
- Slower than symmetric (shared) key cryptosystems
  - Longer keys (e.g. 1024b) for same level of security (e.g. 128b AES)
  - Slow encryption, decryption operations
  - Use RSA only to encrypt an shared key, AES to encrypt message
Encryption and Integrity

- Encryption hides the message from Eve
- Question: can Eve change/forge message?
  - Or: does encryption ensure message integrity?
- With Public Key Encryption:
  - Eve can replace $E_{B_{Pub}}(m)$ with fake: $E_{B_{Pub}}(m')$
- With Symmetric (Shared) Key Encryption:
  - This seems harder to do
  - But given $c = m \oplus k$, attacker can send $c \oplus \text{mask}$, to invert any bit in decrypted message (use mask)
- $\Rightarrow$ Encryption does not ensure integrity!
**Message Authentication Code (MAC)**

- Shared key message authentication (integrity)
- Message sent together with $Tag=\text{MAC}_k(m)$
- Received message, tag are valid iff $Tag=\text{MAC}_k(m)$
- Efficient (even more than shared-key encryption)
- But: Alice can later deny having sent $m$ to Bob (why?)

![Diagram of message authentication code](image)
Public Key Digital Signatures: concept

- Sign using a private, secret signature key
- Everybody knows the public validation key
- Everybody can validate signatures at any time
  - Provides non-repudiation - signer is committed

\[ \text{Sign}_{S_A}(m) + P_A \]

Verify using \( P_A \) that \( \sigma \)
is Alice’s signature on \( m \)
Metaphor to Public Key Signature: Cylinder Seal (ancient Iran)

- Private key: cylinder seal
- Public key: impression of seal
- Document: printed on clay tablet
- Signed document: clay tablet with impression of seal
- Hard to create impression without seal
- Hard to change clay tablet (with impression)
- Hard to copy impression
- Clay tablets (w/ impressions) last long time
- RSA can also be used for digital signature scheme
Key Distribution Problem

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

Solution:
- trusted key distribution center (KDC) acting as intermediary between entities
- KDC needs shared key with each entity, work online

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)
- Works offline, knows only public keys
Certification Authorities

- **Certification authority (CA):** binds public key (e.g. BPub) to identifier (e.g. name: `Bob`).
- **Bob (person, server) registers BPub with CA.**
  - Bob convinces the CA that his name is Bob, sends BPub.
  - **CA creates certificate binding “Bob” to Bob’s public key.**
  - Certificate is digitally signed by CA - CA says “BPub is `Bob’s public key`”

![Diagram of Certification Authorities]

Network Security  7-15
Using Public Key Certificates

- When Alice wants Bob’s public key (to encrypt message to Bob or validate Bob’s signature):
  - gets Bob’s certificate (Bob or elsewhere).
  - apply CA’s public key to Bob’s certificate, get Bob’s public key (validated)
Certificates

- Similar to passport or driver’s license
- Binds a public key to a name and/or other attributes of keyholder, e.g. DNS name for web site
- Signed by a trusted party (Issuer / Certification Authority)
- Allows relying party (Bob, client) to validate name, attributes of key owner (Alice, web site)
Network Security: Agenda

- Shared Media: Eavesdropping (sniffing), other threats
- Main weapon: cryptography
  - Confidentiality (encryption)
  - Message authentication
  - Signatures and Certificates
- Internet Threats, Attacks and Defenses
  - Secure request/response protocols
  - Secure connection `tunnels`: IP-Sec, SSL/TLS
  - Denial of Service Attacks
  - Firewalls
- Conclusions
IP Source Address Spoofing

- **Adversary is in host 3 in net 1.2.3**
- **Sends packet with source IP addr=3.1.5.4**
  - Application can generate IP packets with any IP source address
- **Ingress filtering:** 1st router detects spoofing
  - Many routers, ISPs do not enforce
Internet Attack Model

- Easy: inject messages, spoof (misrepresent)
  - Source address spoofing (IP, e-mail)
  - Spoofing by deceitful content, address (web, e-mail)
- Harder: intercept (eavesdrop/modify) message
  - Except if in same LAN as attacker or broken router
  - Hijacking attacks: intercept message by...
    - Route hijack: force routing via LAN / router
    - Address hijack: source sends to attacker’s IP addr
    - Exercise: show such attacks with protocols we learned!

- Compare to shared-media attack model:
  - Easy: passive (eavesdropping)
  - Harder: active (modify, inject messages)

- Motivates: request-response protocols
Request/Response Protocols

- Client sends request, server sends response
- Reliable pairing of response to request
  - Random ID (nonce) in request
- Weak authentication of response
  - Since it is hard to intercept request
- Server is often stateless
  - Do not keep state (e.g. connection) for each request
  - Efficiency and resiliency to DOS (Denial Of Service)
- Preferable design for security services
  - Due to simplicity, efficiency, resiliency to DOS
- Secure (strong) authentication of response ...
Secure request-response matching

- Attach random nonce $N$ to request
- Attach $MAC_k(response, N)$ to response to validate
- Attach $MAC_k(request, N)$ to validate nonce, request
  - Does not prevent request re-play / reordering
  - To prevent replay: add time, $MAC_k(request, N, time)$
    - Server remembers nonces during `acceptable time window`
  - But this requires (some) state in server, and clocks

Request, $N$, $MAC_k(Request, N, time)$

Client

Response, $MAC_k(Response, N)$

Server

Or: request-response over reliable, secure connection
Network Security: Agenda

- Shared Media: Eavesdropping (sniffing), other threats
- Main weapon: cryptography
  - Confidentiality (encryption)
  - Message authentication
  - Signatures and Certificates
- Internet Threats, Attacks and Defenses
  - Secure request/response protocols
  - Secure connection `tunnels`: IP-Sec, SSL/TLS
  - Denial of Service Attacks
  - Firewalls
- Conclusions
Secure Connection (Tunnel): End-to-End vs. Hop-by-Hop

- Crypto protects traffic over insecure link/Net
- Link layer: one `hop` (e.g. wireless link)
- IP Layer (IP-Sec): transparent to application
- Transport Layer (SSL/TLS): easy, widely used
- Application Layer (PGP, S/MIME)
Secure Tunnels:

- Crypto protects traffic over insecure link/Net
- Hop-by-Hop (link layer) or End-to-End (higher layers)
- IP-Sec: also Gateway to Gateway or End-to-Gateway
SSL / TLS in a Nutshell

- A `secure TCP tunnel from client to server`:  
  - Confidentiality  
  - Message and connection integrity  
  - Authentication of server, optionally also of client
- Original goal and main use: secure credit card number
- Implemented in almost all web clients, servers
- Many implementations, libraries, e.g. Open-SSL
- SSL: Secure Socket Layer  
  - Since SSL (& TLS) operate on top of `standard` Sockets API
- TLS: Transport Layer Security  
  - Since TLS (& SSL) secure TCP (the transport layer)  
  - IETF standard version of SSL  
  - We usually say just SSL but refer to both
SSL’s Server Authentication

- Critical to authenticate (identify) the server
  - To protect secrets sent to server by the user (passwd, cc#, ...)
  - To ensure validity of information from the server

- SSL authenticates using server certificate
  - Containing DNS-name and public key of server
  - SSL handshake confirms the server has matching private key
  - Certificate signed by a Certificate Authority (CA)
  - Browser (or other application) knows to validate CA’s signature

- So is it safe to use SSL-protected web sites?
  - Over 115 CA’s in IEv6, weak validation of DNS-name ownership
  - Users don’t validate DNS-name of site (complexity, spoofing)
  - Users may use non-SSL server without noticing (ditto...)
  - Solutions: known but not yet deployed (see SSL lecture in site)
SSL’s Trust & Security Services

- Confidentiality & authenticity of messages
  - Done by SSL’s `secure data transfer`
- Server (site) authentication:
  - Customer needs to identify bank, merchant, etc.
  - Main use of SSL in browsers... but insecurely!!
- Client authentication
  - Usually done with passwords, cookies - not SSL
- Not supported by SSL
  - Validate rating, certification, other credentials
  - Non-repudiation (requires signatures)
  - Prevent clogging / denial-of-service (DOS)
Network Security: Agenda

- Shared Media: Eavesdropping (sniffing), other threats
- Main weapon: cryptography
  - Confidentiality (encryption)
  - Message authentication
  - Signatures and Certificates
- Internet Threats, Attacks and Defenses
  - Secure request/response protocols
  - Secure connection `tunnels`: IP-Sec, SSL/TLS
  - Denial of Service Attacks
  - Firewalls
- Conclusions
Denial Of Service (Clogging) Attack

- Attacker tries to exhaust resources of host / server / router / user

- Resources include:
  - Computations (CPU time)
  - Storage (e.g. for state of requests/connections)
  - Open TCP connections
    - Limited (10s to several thousand connections - depending on hardware, operating system)
    - So server `never` keeps open connections! Always request-response (and server closes connection, no state)
    - SYN flooding DOS attack: attacker sends `SYN` flow (open connection); server waits...
**SYN flooding DOS (clogging) attack**

- Recall TCP connection setup process...
- Attacker sends many SYN requests (using different spoofed client IP addresses), no ACK...
- Uses up server’s capacity for open connections
- Possible solution: request must contain `cookie` (next)
  - More solutions, details - see Network Security lecture

```
Server

Sends SYN-ACK and waits…

SYN
with fake IP
source address

Hacker
```
Cookies and DDOS Attack

- Cookies: client `pays` for server resources
  - Client must send `cookie` with request
    - E.g. with every SYN (TCP connection) request
    - `Cookie` is `expensive` (takes time to compute)
  - Server spends resources only if Cookie is Ok
    - It is easy (fast) to validate cookies
  - Example: cookie=x s.t. 
    \[ h(x, \text{req}, \text{time}) = *00000 \]

- But: attacker can use many Zombies - broken machines...

- Distributed Denial of Service (DDOS) Attack

- Why is it easy to capture Zombies?
Why Computers are Insecure?

- Most PCs use insecure OSs
  - Most designed for `home` - security not a goal
  - Others support separation btw users
  - Few/none restrict capabilities of applications
  - ➔ malicious/vulnerable/buggy app can harm all!!

- PCs run buggy, vulnerable, even malicious code
  - Many sources (libraries, shareware, ...)
  - Limited awareness & tools (e.g. bounds checking)
  - Limited product liability and consequent damages
  - ➔ most computers don’t fix known vulnerabilities
  - ➔ Easy `fix`: prevent access to (insecure) PCs...
Firewalls

Firewall
A secure/trusted machine, isolating organization’s internal net from larger Internet (or another net/internet), allowing some packets to pass, blocking others.
Packet Filtering

- Internal network connected to Internet via router firewall (packet filtering)
- Router filters packet-by-packet, decision to forward/drop packet based on:
  - Source IP address, destination IP address
  - TCP/UDP source and destination port numbers
  - TCP SYN and ACK bits (identify client vs. server)
Firewall cannot Isolate Insiders!

- A corrupted internal PC can bypass firewall:
  - Initiate communication from inside to outside
  - Overcome restrictions on protocols by encapsulating
  - Firewall scans for known viruses $\rightarrow$ encrypt viruses

- More on firewalls - functions, usage, limitations... see network security lecture or many books/sites on firewalls
Network Security: Summary

Basic tool: cryptography
- Cryptosystems for confidentiality
- Signatures, MAC for authentication
- More... `Applied introduction to crypto` next year

Internet Threat model
- Injection and DOS often easier then Eavesdropping
- Attackers: Hackers, Insiders, Malware

Internet Security Mechanisms, Standards:
- `Secure tunnels` - IP-Sec, TLS/SSL, ...
- Firewalls, intrusion detection, vulnerability scanners
- More... `secure communic. & commerce` next term