Security I: Concepts and Applications

Lecture 20

COS 461: Computer Networks

Kyle Jamieson
Internet’s Design: Insecure

- Designed for simplicity
- “On by default” design

- Readily available zombie machines
- Attacks look like normal traffic
- Internet’s federated operation obstructs cooperation for diagnosis/mitigation
Eavesdropping - Message Interception (Attack on Confidentiality)

- Unauthorized access to information
- Packet sniffers and wiretappers (e.g. tcpdump)
- Illicit copying of files and programs
Integrity Attack - Tampering

- Stop the flow of the message
- Delay and optionally modify the message
- Release the message again

A

Perpetrator

B
Authenticity Attack - Fabrication

- Unauthorized assumption of other’s identity
- Generate and distribute objects under identity

Masquerader: from A
Attack on Availability

- Destroy hardware (cutting fiber) or software
- Modify software in a subtle way
- Corrupt packets in transit

- Blatant denial of service (DoS):
  - Crashing the server
  - Overwhelm the server (use up its resource)
Basic Security Properties

- **Confidentiality**: Concealment of information or resources
- **Authenticity**: Identification & assurance of origin of info
- **Integrity**: Trustworthiness of data/resources; preventing improper/unauthorized changes
- **Availability**: Ability to use desired information/resource
- **Non-repudiation**: Offer of evidence that a party indeed is sender or a receiver of certain information
- **Access control**: Facilities to determine and enforce who is allowed access to what resources (host, software, network, ...)
Security protocols at many layers

• Application layer
  – E-mail: PGP, using a web-of-trust
  – Web: HTTP-S, using a certificate hierarchy

• Transport layer
  – Transport Layer Security/ Secure Socket Layer

• Network layer
  – IP Sec

• Network infrastructure
  – DNS-Sec and BGP-Sec
Introduction to Cryptography
Cryptographic Algorithms: Goal

• One-way functions: cryptographic hash
  – Easy to compute hash
  – Hard to invert

• “Trapdoor” functions: encryption/signatures
  – Given ciphertext alone, hard to compute plaintext (invert)
  – Given ciphertext and key (the “trapdoor”), relatively easy to compute plaintext
  – “Level” of security often based on “length” of key
Encryption and MAC/Signatures

Confidentiality (Encryption)

Sender:
• Compute $C = \text{Enc}_K(M)$
• Send $C$

Receiver:
• Recover $M = \text{Dec}_K(C)$

Auth/Integrity (MAC / Signature)

Sender:
• Compute $s = \text{Sig}_K(\text{Hash}(M))$
• Send $<M, s>$

Receiver:
• Compute $s' = \text{Ver}_K(\text{Hash}(M))$
• Check $s' == s$

These are simplified forms of the actual algorithms
Symmetric vs. Asymmetric Crypto
\textit{a.k.a.} Secret vs. Public Key Crypto

- Symmetric crypto (all crypto pre 1970s)
  - Sender and recipient share a common key
  - All classical encryption algorithms are private-key
  - Dual use: confidentiality or authentication/integrity
    - Encryption vs. msg authentication code (MAC)

- Public-key crypto
  - (Public, private) key associated w/ea. entity ("Alice")
  - Anybody can encrypt to Alice, anybody can verify Alice’s message
  - Only Alice can decrypt, only Alice can “sign”
  - Developed to address “key distribution” problem and “digital signatures” (w/o prior establishment)
Why still both?

• **Symmetric Pros and Cons**
  – Simple and very fast (1000-10000x faster than asymmetric)
  – Must agree/distribute the key beforehand
  – AES/CBC (256-bit) → 80 MB/s (for 2048 bits, .003 ms)

• **Public Key Pros and Cons**
  – Easier key pre-distro.: “Public Key Infrastructure” (PKI)
  – Much slower
  – 2048-RSA → 6.1ms Decrypt, 0.16ms Encrypt

• **Common “engineering” approach:**
  – Best of both worlds via “hybrid” scheme: Use public key to distribute a new random “session” key b/w sender and recipient, then symmetric crypto for remainder of session
Email Security:
Pretty Good Privacy (PGP)
Sender and Receiver Keys

• If the receiver knows the sender’s public key
  – Sender authentication
  – Sender non-repudiation

• If the sender knows the receiver’s public key
  – Confidentiality
  – Receiver authentication
Sending an E-Mail Securely

• **Sender digitally signs the message**
  - Using the sender’s private key

• **Sender encrypts the data**
  - Using a one-time session key
  - Sending the session key, encrypted with the receiver’s public key

• **Sender converts to an ASCII format**
  - Converting the message to base64 encoding
  - (Email messages must be sent in ASCII)
Public Key Certificate

• Binding between identity and a public key
  – “Identity” is, for example, an e-mail address
  – “Binding” ensured using a digital signature

• Contents of a certificate
  – Identity of the entity being certified
  – Public key of the entity being certified
  – Identity of the signer
  – Digital signature
  – Digital signature algorithm id
Web of Trust for PGP

• **Decentralized solution**
  – Protection against state actor intrusion
  – No central certificate authorities

• **Customized solution**
  – Individual decides whom to trust, and how much
  – Multiple certificates with different confidence levels

• **Key-signing parties!**
  – Collect and provide public keys in person
  – Sign other’s keys, and get your key signed by others
HTTP Security
HTTP Threat Model

• **Eavesdropper**
  - Listening on conversation (confidentiality)

• **Man-in-the-middle**
  - Modifying content (integrity)

• **Impersonation**
  - Bogus website (authentication, confidentiality)
HTTP-S: Securing HTTP

- HTTP sits on top of secure channel (SSL/TLS)
  - TCP port 443 vs. 80

- All (HTTP) bytes encrypted and authenticated
  - No change to HTTP itself!

- Where to get the key???
Learning a Valid Public Key

- **What is that lock?**
  - Securely binds domain name to public key (PK)
    - If PK is authenticated, then any message signed by that PK cannot be forged by non-authorized party
  - Believable only if you trust the attesting body
    - Bootstrapping problem: Who to trust, and how to tell if this message is actually from them?
Hierarchical Public Key Infrastructure

• Public key certificate
  – Binding between identity and a public key
  – “Identity” is, for example, a domain name
  – Digital signature to ensure integrity

• Certificate authority
  – Issues public key certificates and verifies identities
  – Trusted parties (e.g., VeriSign, GoDaddy, Comodo)
  – Preconfigured certificates in Web browsers
<table>
<thead>
<tr>
<th>Subject Name</th>
<th><a href="http://www.wellsfargo.com">www.wellsfargo.com</a></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business Category</strong></td>
<td>Private Organization</td>
</tr>
<tr>
<td><strong>Inc. Country</strong></td>
<td>US</td>
</tr>
<tr>
<td><strong>Inc. State/Province</strong></td>
<td>Delaware</td>
</tr>
<tr>
<td><strong>Serial Number</strong></td>
<td>251212</td>
</tr>
<tr>
<td><strong>Country</strong></td>
<td>US</td>
</tr>
<tr>
<td><strong>State/Province</strong></td>
<td>California</td>
</tr>
<tr>
<td><strong>Locality</strong></td>
<td>San Francisco</td>
</tr>
<tr>
<td><strong>Organization</strong></td>
<td>Wells Fargo &amp; Company</td>
</tr>
<tr>
<td><strong>Organizational Unit</strong></td>
<td>DCG–PSG</td>
</tr>
<tr>
<td><strong>Common Name</strong></td>
<td><a href="http://www.wellsfargo.com">www.wellsfargo.com</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issuer Name</th>
<th>DigiCert Global CA G2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Country</strong></td>
<td>US</td>
</tr>
<tr>
<td><strong>Organization</strong></td>
<td>DigiCert Inc</td>
</tr>
<tr>
<td><strong>Common Name</strong></td>
<td>DigiCert Global CA G2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Validity</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Not Before</strong></td>
<td>2/7/2019, 7:00:00 PM (Eastern Daylight Time)</td>
</tr>
<tr>
<td><strong>Not After</strong></td>
<td>2/8/2021, 7:00:00 AM (Eastern Daylight Time)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subject Alt Names</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DNS Name</strong></td>
<td><a href="http://www.wellsfargo.com">www.wellsfargo.com</a></td>
</tr>
</tbody>
</table>
# Certificate

**www.wellsfargo.com**

<table>
<thead>
<tr>
<th>Certificate</th>
<th>DigiCert Global CA G2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Name</td>
<td></td>
</tr>
<tr>
<td><strong>Country</strong></td>
<td>US</td>
</tr>
<tr>
<td><strong>Organization</strong></td>
<td>DigiCert Inc</td>
</tr>
<tr>
<td><strong>Common Name</strong></td>
<td>DigiCert Global CA G2</td>
</tr>
<tr>
<td>Issuer Name</td>
<td></td>
</tr>
<tr>
<td><strong>Country</strong></td>
<td>US</td>
</tr>
<tr>
<td><strong>Organization</strong></td>
<td>DigiCert Inc</td>
</tr>
<tr>
<td><strong>Organizational Unit</strong></td>
<td><a href="http://www.digicert.com">www.digicert.com</a></td>
</tr>
<tr>
<td><strong>Common Name</strong></td>
<td>DigiCert Global Root G2</td>
</tr>
<tr>
<td>Validity</td>
<td></td>
</tr>
<tr>
<td><strong>Not Before</strong></td>
<td>8/1/2013, 8:00:00 AM (Eastern Daylight Time)</td>
</tr>
<tr>
<td><strong>Not After</strong></td>
<td>8/1/2028, 8:00:00 AM (Eastern Daylight Time)</td>
</tr>
<tr>
<td>Public Key Info</td>
<td></td>
</tr>
<tr>
<td><strong>Algorithm</strong></td>
<td>RSA</td>
</tr>
<tr>
<td><strong>Key Size</strong></td>
<td>2048</td>
</tr>
<tr>
<td><strong>Exponent</strong></td>
<td>65537</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td></td>
</tr>
<tr>
<td><strong>Serial Number</strong></td>
<td>0C:8E:0E:C9:0D:6A:89:15:88:04:06:1E:E2:41:F9:AF</td>
</tr>
<tr>
<td><strong>Signature Algorithm</strong></td>
<td>SHA-256 with RSA Encryption</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>Download</strong></td>
<td>PEM (cert) PEM (chain)</td>
</tr>
</tbody>
</table>
Transport Layer Security (TLS)

Based on the earlier Secure Socket Layer (SSL) originally developed by Netscape
TLS Handshake Protocol

- Send new random value, list of supported ciphers
- Send pre-secret, encrypted under PK
- Create shared secret key from pre-secret and random
- Switch to new symmetric-key cipher using shared key
- Send new random value, digital certificate with PK
- Create shared secret key from pre-secret and random
- Switch to new symmetric-key cipher using shared key
TLS Record Protocol

• **Messages from application layer are:**
  – Fragmented or coalesced into blocks
  – Optionally compressed
  – Integrity-protected using an HMAC
  – Encrypted using symmetric-key cipher
  – Passed to the transport layer (usually TCP)

• **Sequence #s on record-protocol messages**
  – Prevents replays and reorderings of messages
Comments on HTTPS

- **HTTPS authenticates server, not content**
  - If CDN (Akamai) serves content over HTTPS, customer must trust Akamai not to change content

- **Symmetric-key crypto after public-key ops**
  - Handshake protocol using public key crypto
  - Symmetric-key crypto much faster (100-1000x)

- **HTTPS on top of TCP, so reliable byte stream**
  - Can leverage fact that transmission is reliable to ensure: each data segment received exactly once
  - Adversary can’t successfully drop or replay packets
IP Security
IP Security

• There are range of app-specific security mechanisms
  – eg. TLS/HTTPS, S/MIME, PGP, Kerberos, ...

• But security concerns that cut across protocol layers

• Implement by the network for all applications?

Enter IPSec!
IPSec

- General IP Security framework

- Allows one to provide
  - Access control, integrity, authentication, originality, and confidentiality

- Applicable to different settings
  - Narrow streams: Specific TCP connections
  - Wide streams: All packets between two gateways
IPSec Uses
Benefits of IPSec

• If in a firewall/router:
  – Strong security to all traffic crossing perimeter
  – Resistant to bypass

• Below transport layer
  – Transparent to applications
  – Can be transparent to end users

• Can provide security for individual users
IP Security Architecture

• Specification quite complex
  – Mandatory in IPv6, optional in IPv4

• Two security header extensions:
  – Authentication Header (AH)
    • Connectionless integrity, origin authentication
      – MAC over most header fields and packet body
    • Anti-replay protection
  – Encapsulating Security Payload (ESP)
    • These properties, plus confidentiality
Encapsulating Security Payload (ESP)

- **Transport mode:** Data encrypted, but not header
  - After all, network headers needed for routing!
  - Can still do traffic analysis, but is efficient
  - Good for host-to-host traffic

- **Tunnel mode ("IP-in-IP")**
  - Encrypts entire IP packet
  - Add new header for next hop
  - Good for VPNs, gateway-to-gateway security
Replay Protection is Hard

• **Goal:** Eavesdropper can’t capture encrypted packet and duplicate later
  – Easy with TLS/HTTP on TCP: Reliable byte stream
  – But IP Sec at packet layer; transport may not be reliable

• **IPSec solution:** Sliding window on sequence #’s
  – All IPSec packets have a 64-bit sequence number
  – Receiver keeps track of which seqno’s seen before
    • \([\text{latest} - \text{window} + 1, \text{latest}]\); window ~64 packets
  – Accept packet if
    • seqno > latest (and update latest)
    • Within window but has not been seen before
  – If reliable, could remember last, and accept iff last + 1
Conclusions

• Security at many layers
  – Application, transport, and network layers
  – Customized to the properties and requirements

• Exchanging keys
  – Public key certificates
  – Certificate authorities vs. Web of trust

• Next time
  – Network security: DNS, BGP