CSCI-1680
The End (of lectures)
TLS, Wrapup

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, Scott Shenker and John Jannotti
• TCP: Due Friday, April 29 by 11:59pm
• Office hours: 3-5pm today (Zoom)
  – I will be in the building otherwise, ping me
• Late days: use them on TCP

• Look for grade reports today/tomorrow

• Course feedback:
  – Critical Review
  – I will also send you a form
Today’s Lecture

- PKI and TLS
- Wrapup
- Final project overview
PKIs, TLS, and HTTPS
The story so far

- Asymmetric crypto: each entity gets a key in two parts
  - $K_{\text{priv}}$: Private key, kept secret
  - $K_{\text{pub}}$: Public key, shared with everyone
The story so far

• Asymmetric crypto: each entity gets a key in two parts
  – $K_{\text{priv}}$: Private key, kept secret
  – $K_{\text{pub}}$: Public key, shared with everyone

• Can provide important security properties
  – Authentication/Integrity: A signs message with $K_{\text{priv},A}$, anyone with $K_{\text{pub},A}$ can verify message came from A
  – Confidentiality: A encrypts message to B with $K_{\text{pub},B}$, B can decrypt with $K_{\text{priv},B}$

THE ENTITY THAT HOLDS A'S PRIVATE KEY IS A
The story so far

- Asymmetric crypto: each entity gets a key in two parts
  - $K_{\text{priv}}$: Private key, kept secret
  - $K_{\text{pub}}$: Public key, shared with everyone

- Can provide important security properties
  - Authentication/Integrity: A signs message with $K_{\text{priv},A}$, anyone with $K_{\text{pub},A}$ can verify message came from A
  - Confidentiality: A encrypts message to B with $K_{\text{pub},B}$, B can decrypt with $K_{\text{priv},B}$

- But: how do we know if we can trust a public key?
Public key crypto is very powerful …

- … but the **realities** of tying public keys to real world identities turn out to be quite hard

- **PKI**: Trust distribution mechanism
  - Authentication via **Digital Certificates**
Public Key Infrastructure (PKI)

Public key crypto is very powerful …
- … but the **realities** of tying public keys to real world identities turn out to be quite hard

- PKI: **Trust distribution** mechanism
  - Authentication via **Digital Certificates**

- Note: Trust doesn’t mean someone is honest, just that they are who they say they are…
Managing Trust

• The most solid level of trust is rooted in our direct personal experience
  – E.g., Alice’s trust that Bob is who they say they are
  – Clearly doesn’t scale to a global network!
Managing Trust

• The most solid level of trust is rooted in our direct personal experience
  – E.g., Alice’s trust that Bob is who they say they are
  – Clearly doesn’t scale to a global network!

• In its absence, we rely on delegation
  – Alice trusts Bob’s identity because Charlie attests to it ....
  – .... and Alice trusts Charlie
Managing Trust, con’t

- Trust is not particularly transitive
  - Should Alice trust Bob because she trusts Charlie …
  - … and Charlie vouches for Donna …
  - … and Donna says Eve is trustworthy …
  - … and Eve vouches for Bob’s identity?

- Two models of delegating trust
  - Rely on your set of friends and their friends
    - “Web of trust” -- e.g., PGP
  - Rely on trusted, well-known authorities (and those they trust…)
    - “Trusted root” -- e.g., HTTPS
PKI Conceptual framework

Public keys managed by Certificate Authorities (CAs)

• Everyone knows public key for some root CAs
PKI Conceptual framework

Public keys managed by Certificate Authorities (CAs)
- Everyone knows public key for some root CAs
- To publish a public key for entity X, root CA R signs X’s public key
  - What this means: CA agrees that this is X’s public key
  - Creates a Certificate: \( \{K_{pub, X}, \text{signature}, \text{metadata}\} \)
PKI Conceptual framework

Public keys managed by Certificate Authorities (CAs)

- Everyone knows public key for some root CAs
- To publish a public key for entity X, root CA R signs X’s public key
  - What this means: CA agrees that this is X’s public key
  - Creates a Certificate: \( \{ K_{pub,X}, \text{signature}, \text{metadata} \} \)

- Given signature, anyone who knows the root can verify
  - Delegates trust of \( K_{pub,X} \) to CA
  - If you trust the CA, you now trust X

- Root CAs: pre-installed in your system/browser
### DigiCert Assured ID Root CA

- **Certificate Authority**: DigiCert Assured ID Root CA
- **Country or Region**: US
- **Organization**: DigiCert Inc
- **Organizational Unit**: www.digicert.com
- **Common Name**: DigiCert Assured ID Root CA

#### Details

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Issuer Name</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Country or Region</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 Assured ID Root CA</td>
</tr>
<tr>
<td><strong>Serial Number</strong></td>
<td>0C E7 E0 E5 17 D8 46 FE 8F E5 60 FC 1B F0 30 39</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>Signature Algorithm</strong></td>
<td>SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Not Valid Before</strong></td>
<td>Thursday, November 9, 2006 at 19:00:00 Eastern Standard Time</td>
</tr>
<tr>
<td><strong>Not Valid After</strong></td>
<td>Sunday, November 9, 2031 at 19:00:00 Eastern Standard Time</td>
</tr>
<tr>
<td><strong>Public Key Info</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Algorithm</strong></td>
<td>RSA Encryption (1.2.840.113549.1.1.1)</td>
</tr>
<tr>
<td><strong>Parameters</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Public Key</strong></td>
<td>256 bytes : AD OE 15 CE E4 43 80 5C ...</td>
</tr>
<tr>
<td><strong>Exponent</strong></td>
<td>65537</td>
</tr>
<tr>
<td><strong>Key Size</strong></td>
<td>2,048 bits</td>
</tr>
<tr>
<td><strong>Key Usage</strong></td>
<td>Verify</td>
</tr>
</tbody>
</table>
## Amazon Root CA
Root certificate authority
Expires: Saturday, January 16, 2038 at 19:00:00 Eastern Standard Time
This certificate is valid

<table>
<thead>
<tr>
<th>Name</th>
<th>Kind</th>
<th>Date Modified</th>
<th>Expires</th>
<th>Keychain</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA Certificate Services</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2028 at 18:59:59</td>
<td>System Roots</td>
</tr>
<tr>
<td>AC RAIZ FNMT-RCM</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2028 at 19:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Actalis Authentication Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Sep 22, 2030 at 07:22:02</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Commercial</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2030 at 09:06:06</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Networking</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2030 at 09:08:24</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Premium</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2040 at 09:10:36</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Premium ECC</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2040 at 09:20:24</td>
<td>System Roots</td>
</tr>
<tr>
<td>Amazon Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Jan 16, 2038 at 19:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Amazon Root CA 2</td>
<td>certificate</td>
<td>--</td>
<td>May 25, 2040 at 20:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Amazon Root CA 3</td>
<td>certificate</td>
<td>--</td>
<td>May 25, 2040 at 20:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Amazon Root CA 4</td>
<td>certificate</td>
<td>--</td>
<td>May 25, 2040 at 20:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>ANF Global Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Jun 8, 2033 at 13:45:38</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Feb 9, 2036 at 16:40:36</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root CA - G2</td>
<td>certificate</td>
<td>--</td>
<td>Apr 30, 2039 at 14:10:09</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root CA - G3</td>
<td>certificate</td>
<td>--</td>
<td>Apr 30, 2039 at 14:19:06</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root Certificate Authority</td>
<td>certificate</td>
<td>--</td>
<td>Feb 9, 2025 at 19:18:14</td>
<td>System Roots</td>
</tr>
<tr>
<td>Atos TrustedRoot 2011</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2030 at 18:59:59</td>
<td>System Roots</td>
</tr>
<tr>
<td>Autoridad de Certificación Firmaprofesional CIF A62634068</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2030 at 03:38:15</td>
<td>System Roots</td>
</tr>
<tr>
<td>Autoridad de Certificación Raiz del Estado Venezolano</td>
<td>certificate</td>
<td>--</td>
<td>Dec 17, 2030 at 18:59:59</td>
<td>System Roots</td>
</tr>
<tr>
<td>Baltimore CyberTrust Root</td>
<td>certificate</td>
<td>--</td>
<td>May 12, 2025 at 19:59:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Bypass Class 2 Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Oct 26, 2040 at 04:38:03</td>
<td>System Roots</td>
</tr>
<tr>
<td>Bypass Class 3 Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Oct 26, 2040 at 04:28:58</td>
<td>System Roots</td>
</tr>
<tr>
<td>CA Disig Root R1</td>
<td>certificate</td>
<td>--</td>
<td>Jul 19, 2042 at 05:06:56</td>
<td>System Roots</td>
</tr>
<tr>
<td>CA Disig Root R2</td>
<td>certificate</td>
<td>--</td>
<td>Jul 19, 2042 at 05:15:30</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certigna</td>
<td>certificate</td>
<td>--</td>
<td>Jun 29, 2027 at 11:13:05</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certinomis - Autorité Racine</td>
<td>certificate</td>
<td>--</td>
<td>Sep 17, 2028 at 04:28:59</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certinomis - Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Oct 21, 2033 at 05:17:18</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certplus Root CA G1</td>
<td>certificate</td>
<td>--</td>
<td>Jan 14, 2038 at 19:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certplus Root CA G2</td>
<td>certificate</td>
<td>--</td>
<td>Jan 14, 2038 at 19:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>certSIGN ROOT CA</td>
<td>certificate</td>
<td>--</td>
<td>Jul 4, 2031 at 13:20:04</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certum CA</td>
<td>certificate</td>
<td>--</td>
<td>Jun 11, 2027 at 06:46:39</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certum Trusted Network CA</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2029 at 07:07:37</td>
<td>System Roots</td>
</tr>
</tbody>
</table>
PKI hierarchy

• In reality, hierarchy of trust
• Root CAs sign certificates for Intermediate CAs
• Intermediate CAs sign certificates for general users/sites
- Kpub, CS.Brown, Edu, SCAI
- Kpub, CA1, E2OOT
- From Your System
  - Verify
- Kpub, Root -> Trust

TLS?

STACK OF CERTS
PKI hierarchy

- In reality, hierarchy of trust
- Root CAs sign certificates for Intermediate CAs
- Intermediate CAs sign certificates for general users/sites

The further up the hierarchy, the more protections it needs
- CA's often use Hardware Security Modules (HSMs), other physical protections...
- What happens if a CA is compromised?
**How do you get a cert?**

- Every CA has a policy
- As a general user/small org
  - Pay a hosting company
  - Let's encrypt

Hosting company

1. Key pub into $$
2. CERT
HIERARCHY OF CERTIFICATES → SELF-SIGNED

1. ROOT CAS
   - Most protected - Physical and digital security measures
   - Least accessible (signing might be offline)
   - Longest validity
   - Hardest to change
   - "Less" protected, easier to use
   - Signs "end user" certs
   - Root could revoke if compromised

2. INTERMEDIATE CAS
   - 2.5-3
   - Wildcard certs
   - Least trusted
   - Generally can't sign others
   - Most "used"
   - Easiest to be compromised

3. END-USER CERTS
   - EG: CS.BROWN.EDU
   - Least
   - Most "used"
   - Leaked

- Allocated for whole domain
- Might be stored on servers used by man
- Least impactful if leaked
LETS ENCRYPT (≈ 2014)

Tell LE: Go to my site (nonce)

Webserver 

VTY, SH

LE visits VTY, SH/challenge, verify (challenge)

If verification is successful,

Is this more/less secure than paying?

It proves that requester is in control of VTY, SH (at the time of update)

It does not prove that it's owned by me.

(Which should happen when paying for it)
If you connect to a site w/ a TLS error, likely errors are:

- Expired
- Name is wrong
- Self-signed
  (Not trusted/signed by anyone)

- Common for servers when first set up not properly configured
Inside the Server’s Certificate

• **Common name:** Domain name for cert (e.g., amazon.com)
• Amazon’s **public key**
• A bunch of auxiliary info (physical address, type of cert, expiration time)
• URL to revocation center to check for revoked keys
• Name of certificate’s **signatory** (who signed it)
• A public-key **signature** of a hash of all this
  – Constructed using the signatory’s private RSA key
Putting It All Together: HTTPS
Putting It All Together: HTTPS

- Steps after clicking on https://www.amazon.com
Putting It All Together: HTTPS

- Steps after clicking on https://www.amazon.com
- https = “Use HTTP over TLS”
  - SSL = Secure Socket Layer (older version)
  - TLS = Transport Layer Security
    - Successor to SSL, and compatible with it
    - RFC 4346, and many others
Putting It All Together: HTTPS

- Steps after clicking on https://www.amazon.com
- **https** = "Use HTTP over TLS"
  - **SSL** = Secure Socket Layer (older version)
  - **TLS** = Transport Layer Security
    - Successor to SSL, and compatible with it
    - RFC 4346, and many others
- Provides security layer (authentication, encryption) on top of transport layer
  - Fairly transparent to the app (once set up)
HTTPS Connection (SSL/TLS), con’t
HTTPS Connection (SSL/TLS), con’t
HTTPS Connection (SSL/TLS), con’t
HTTPS Connection (SSL/TLS), con’t

Browser

Amazon

SYN

SYN ACK

ACK
HTTPS Connection (SSL/TLS), con’t

- **ASYMMETRIC CRYPTO**
  - **SYMMETRIC CRYPTO** (AES)
  - HASHING (AUTH/INTEGRITY)

Browser

Amazon

- SYN
- SYN ACK
- ACK

Hello. I support (TLS+RSA+AES128+SHA1) or (SSL+RSA+3DES+MD5) or …
HTTPS Connection (SSL/TLS), con’t

Browser

Amazon

SYN

SYN ACK

ACK

Hello. I support (TLS+RSA+AES128+SHA1) or (SSL+RSA+3DES+MD5) or …

Let’s use TLS+RSA+AES128+SHA1
HTTPS Connection (SSL/TLS), con’t

- Browser (client) connects via TCP to Amazon’s HTTPS server
- Client sends over list of crypto protocols it supports
- Server picks protocols to use for this session
- Server sends over its certificate
- (all of this is in the clear)

**Diagram:**

Browser (client) sends SYN, SYN ACK, and ACK to Amazon's server.
Amazon responds with a message indicating supported protocols:
```
Hello. I support (TLS+RSA+AES128+SHA1) or (SSL+RSA+3DES+MD5) or ...
```
Amazon sends its certificate.
Browser chooses TLS+RSA+AES128+SHA1 as the protocol to use.

**Note:**
- (~1 KB of data)
Validating Amazon’s Identity
Validating Amazon’s Identity

• Browser retrieves cert belonging to the signatory

\[ \text{VALIDATE CHAIN OF CERTS} \]
Validating Amazon’s Identity

- Browser retrieves cert belonging to the **signatory**
- If it can’t find the cert, then warns the user that site has not been verified
  - And may ask whether to continue
  - Could still proceed, just **without authentication**

- **Browser Error**

- **Also checks**
  - **Expiration Date**
  - If cert was revoked.
Validating Amazon’s Identity

- Browser retrieves cert belonging to the **signatory**

- If it can’t find the cert, then warns the user that site has not been verified
  - And may ask whether to continue
  - Could still proceed, just **without authentication**

- Browser uses public key in signatory’s cert to decrypt signature
  - Compares with its own hash of Amazon’s cert
Validating Amazon’s Identity

- Browser retrieves cert belonging to the **signatory**

- If it can’t find the cert, then warns the user that site has not been verified
  - And may ask whether to continue
  - Could still proceed, just without authentication

- Browser uses public key in signatory’s cert to decrypt signature
  - Compares with its own hash of Amazon’s cert

- Assuming signature matches, now have high confidence it’s indeed Amazon
  - … assuming signatory is trustworthy
HTTPS Connection (SSL/TLS), con’t

Browser

Amazon

Here’s my cert

~1 KB of data
HTTPS Connection (SSL/TLS), con’t

• Browser constructs a random **session key** $K$
• Browser constructs a random session key $K$
• Browser encrypts $K$ using Amazon’s public key
HTTPS Connection (SSL/TLS), con’t

- Browser constructs a random **session key** $K$
- Browser encrypts $K$ using Amazon’s public key
- Browser sends $E(K, KA_{public})$ to server
• Browser constructs a random session key $K$
• Browser encrypts $K$ using Amazon’s public key
• Browser sends $E(K, KA_{public})$ to server
HTTPS Connection (SSL/TLS), con't

- Browser constructs a random session key $K$
- Browser encrypts $K$ using Amazon’s public key
- Browser sends $E(K, KA_{public})$ to server
- Browser displays 🏠
HTTPS Connection (SSL/TLS), con’t

- Browser constructs a random *session key* $K$
- Browser encrypts $K$ using Amazon’s public key
- Browser sends $E(K, KA_{\text{public}})$ to server
- Browser displays !
- All subsequent communication encrypted w/ symmetric cipher using key $K$
  - E.g., client can authenticate using a password
When does this break down?

• TLS is hard to implement
• Need to trust the CAs
• Users need to understand warnings

NEVER BUILD YOUR OWN
LOTS OF FAMOUS BUGS.

ALSO A UX PROBLEM!
"TLS DECRYPTION"

GOOGLE → TLS → PROXY → TLS → CORPORATE SYSTEM

- FILTERING
- INSPECTION
As of July 2021, the Trustworthy Internet Movement estimated the ratio of websites that are vulnerable to TLS attacks.[71]

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Insecure</th>
<th>Security</th>
<th>Secure</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Renegotiation attack</td>
<td>0.1%</td>
<td>Depends</td>
<td>99.2%</td>
<td>0.7%</td>
</tr>
<tr>
<td></td>
<td>support insecure renegotiation</td>
<td>&lt;0.1% support both</td>
<td>support secure renegotiation</td>
<td>no support</td>
</tr>
<tr>
<td>RC4 attacks</td>
<td>0.4%</td>
<td>6.5%</td>
<td>93.1%</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>support RC4 suites used with modern browsers</td>
<td>support some RC4 suites</td>
<td>no support</td>
<td></td>
</tr>
<tr>
<td>TLS Compression (CRIME attack)</td>
<td>&gt;0.0%</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>vulnerable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Heartbleed</td>
<td>&gt;0.0%</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>vulnerable</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ChangeCipherSpec injection attack</td>
<td>0.1%</td>
<td>0.2%</td>
<td>98.5%</td>
<td>1.2%</td>
</tr>
<tr>
<td></td>
<td>vulnerable and exploitable</td>
<td>vulnerable, not exploitable</td>
<td>not vulnerable</td>
<td>unknown</td>
</tr>
<tr>
<td>Poodle attack against TLS</td>
<td>0.1%</td>
<td>0.1%</td>
<td>99.8%</td>
<td>0.2%</td>
</tr>
<tr>
<td>(Original Poodle against SSL 3.0 is not included)</td>
<td>vulnerable and exploitable</td>
<td>vulnerable, not exploitable</td>
<td>not vulnerable</td>
<td>unknown</td>
</tr>
<tr>
<td>Protocol downgrade</td>
<td>6.6%</td>
<td>N/A</td>
<td>72.3%</td>
<td>21.0%</td>
</tr>
<tr>
<td></td>
<td>Downgrade defence not supported</td>
<td></td>
<td>Downgrade defence supported</td>
<td>unknown</td>
</tr>
</tbody>
</table>

Wikipedia table, source: https://www.ssllabs.com/ssl-pulse/
### Keychain Access

#### Amazon Root CA

Root certificate authority
Expires: Saturday, January 16, 2038 at 19:00:00 Eastern Standard Time

This certificate is valid

<table>
<thead>
<tr>
<th>Name</th>
<th>Kind</th>
<th>Date Modified</th>
<th>Expires</th>
<th>Keychain</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA Certificate Services</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2028 at 18:59:59</td>
<td>System Roots</td>
</tr>
<tr>
<td>AC RAIZ FNMT-RCM</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2029 at 19:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Actalis Authentication Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Sep 22, 2030 at 07:22:02</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Commercial</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2030 at 09:06:06</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Networking</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2030 at 09:08:24</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Premium</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2040 at 09:10:36</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Premium ECC</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2040 at 09:20:24</td>
<td>System Roots</td>
</tr>
<tr>
<td>Amazon Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Jan 16, 2038 at 19:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Amazon Root CA 2</td>
<td>certificate</td>
<td>--</td>
<td>May 25, 2040 at 20:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Amazon Root CA 3</td>
<td>certificate</td>
<td>--</td>
<td>May 25, 2040 at 20:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Amazon Root CA 4</td>
<td>certificate</td>
<td>--</td>
<td>May 25, 2040 at 20:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>ANF Global Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Jun 5, 2033 at 13:45:38</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Feb 9, 2035 at 16:40:36</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root CA - G2</td>
<td>certificate</td>
<td>--</td>
<td>Apr 30, 2039 at 14:10:09</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root CA - G3</td>
<td>certificate</td>
<td>--</td>
<td>Apr 30, 2039 at 14:19:06</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root Certificate Authority</td>
<td>certificate</td>
<td>--</td>
<td>Feb 9, 2025 at 19:18:14</td>
<td>System Roots</td>
</tr>
<tr>
<td>Atos TrustedRoot 2011</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2030 at 18:59:59</td>
<td>System Roots</td>
</tr>
<tr>
<td>Autoridad de Certificacion Firmaprofesional CIF A62634068</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2030 at 03:38:15</td>
<td>System Roots</td>
</tr>
<tr>
<td>Autoridad de Certificacion Raiz del Estado Venezolano</td>
<td>certificate</td>
<td>--</td>
<td>Dec 17, 2030 at 18:59:59</td>
<td>System Roots</td>
</tr>
<tr>
<td>Baltimore CyberTrust Root</td>
<td>certificate</td>
<td>--</td>
<td>May 12, 2025 at 19:59:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Bypass Class 2 Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Oct 26, 2040 at 04:38:03</td>
<td>System Roots</td>
</tr>
<tr>
<td>Bypass Class 3 Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Oct 26, 2040 at 04:28:58</td>
<td>System Roots</td>
</tr>
<tr>
<td>CA Disig Root R1</td>
<td>certificate</td>
<td>--</td>
<td>Jul 19, 2042 at 05:06:56</td>
<td>System Roots</td>
</tr>
<tr>
<td>CA Disig Root R2</td>
<td>certificate</td>
<td>--</td>
<td>Jul 19, 2042 at 05:15:30</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certigna</td>
<td>certificate</td>
<td>--</td>
<td>Jun 29, 2027 at 11:13:05</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certinomis – Autorité Racine</td>
<td>certificate</td>
<td>--</td>
<td>Sep 17, 2028 at 04:28:59</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certinomis – Root CA</td>
<td>certificate</td>
<td>--</td>
<td>Oct 21, 2033 at 05:17:18</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certplus Root CA G1</td>
<td>certificate</td>
<td>--</td>
<td>Jan 14, 2038 at 19:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certplus Root CA G2</td>
<td>certificate</td>
<td>--</td>
<td>Jan 14, 2038 at 19:00:00</td>
<td>System Roots</td>
</tr>
<tr>
<td>certSIGN ROOT CA</td>
<td>certificate</td>
<td>--</td>
<td>Jul 4, 2031 at 13:20:04</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certum CA</td>
<td>certificate</td>
<td>--</td>
<td>Jun 11, 2027 at 06:46:39</td>
<td>System Roots</td>
</tr>
<tr>
<td>Certum Trusted Network CA</td>
<td>certificate</td>
<td>--</td>
<td>Dec 31, 2029 at 07:07:37</td>
<td>System Roots</td>
</tr>
</tbody>
</table>
Wrapping up

- This is our last formal lecture
- From here: work on final project
What (I hope) you have learned

• Skill: network programming (and soft. eng)
  Socket programming
    – Server programming/robustness
    – Implementing protocols
What (I hope) you have learned

• **Skill: network programming (and soft. eng)**
  - Socket programming
  - Server programming/robustness
  - Implementing protocols

• **Knowledge: How the Internet Works**
  - Internet architecture and design
  - Key Internet protocols
  - Some applications (Web, DNS, …)
What (I hope) you have learned

• **Skill:** network programming (and soft. eng)
  - Socket programming
  - Server programming/robustness
  - Implementing protocols

• **Knowledge:** How the Internet Works
  - Internet architecture and design
  - Key Internet protocols
  - Some applications (Web, DNS, …)

• **Insights:** key concepts / abstractions
  - Protocols, layering, naming, security challenges …
What (I hope) you have learned

• Skill: network programming (and soft. eng)
  Socket programming
    – Server programming/robustness
    – Implementing protocols
• Knowledge: How the Internet Works
  – Internet architecture and design
  – Key Internet protocols
  – Some applications (Web, DNS, …)
• Insights: key concepts / abstractions
  – Protocols, layering, naming, security challenges …

My goal: give you tools to understand new networking challenges you encounter
Networking principles

• Some general CS concepts
  – Hierarchy (IP addressing, DNS, PKI, …)
  – Indirection (ARP, DNS, …)
  – Caching

Service: user-facing application. Application-defined messages
Service: multiplexing applications
Reliable byte stream to other node (TCP)
Unreliable datagram (UDP)
Service: move packets to any other node
Internet Protocol (IP)
Service: move frames to other node across link
May add reliability, medium access control
Service: move bits to other node across link
Networking principles

- Some general CS concepts
  - Hierarchy (IP addressing, DNS, PKI, …)
  - Indirection (ARP, DNS, …)
  - Caching

- Some concepts (a bit) networking-specific
  - Layering
  - Multiplexing
  - End-to-end argument
  - Robustness principles

Service: user-facing application.
Application-defined messages

Service: multiplexing applications
Reliable byte stream to other node (TCP)
Unreliable datagram (UDP)

Service: move packets to any other node
Internet Protocol (IP)

Service: move frames to other node across a link
May add reliability, medium access control

Service: move bits to other node across a link
Lots of challenges out there

Our Internet architecture was designed in the 1980s, where modern scale and complexity was unimaginable

Now...

- No one knows how big the Internet is
- No one is in charge
- Anyone can add any application
- Packets traverse many paths, countries, regulatory domains
Your final project

Basically: pick networking topic we’ve learned, or something we haven’t and build something to help you understand how it works

• I will give you some sample topics, or you can pick your own
• Work with your current group

• You don’t have to finish—just try out something, write up what you did and what you learned
Your final project

Basically: pick networking topic we’ve learned, or something we haven’t and build something to help you understand how it works

• I will give you some sample topics, or you can pick your own
• Work with your current group

• You don’t have to finish—just try out something, write up what you did and what you learned

A lot like research!
Some ideas

- Exploring network performance
  - Measure, e.g., TCP congestion control under various conditions
  - Measure DNS/ping times for some site with a CDN
  - Deliverable: test out some conditions, graph results
Some ideas

- Exploring network performance
  - Measure, e.g., TCP congestion control under various conditions
  - Measure DNS/ping times for some site with a CDN
  - Deliverable: test out some conditions, graph results

- Measuring/characterizing traffic
  - How much of my daily traffic is unencrypted vs. encrypted?
  - How does Zoom work? How much bandwidth does it need?
  - Deliverable: capture packets, report on some data
Some ideas

- **Exploring network performance**
  - Measure, e.g., TCP congestion control under various conditions
  - Measure DNS/ping times for some site with a CDN
  - Deliverable: test out some conditions, graph results

- **Measuring/characterizing traffic**
  - How much of my daily traffic is unencrypted vs. encrypted?
  - How does Zoom work? How much bandwidth does it need?
  - Deliverable: capture packets, report on some data

- **Software Defined Networking (SDN)**
  - Build a network-wide application in a single program
  - Deliverable: implement MAC learning, ECMP, ...
Some ideas

• Exploring network performance
  – Measure, e.g., TCP congestion control under various conditions
  – Measure DNS/ping times for some site with a CDN
  – Deliverable: test out some conditions, graph results

• Measuring/characterizing traffic
  – How much of my daily traffic is unencrypted vs. encrypted?
  – How does Zoom work? How much bandwidth does it need?
  – Deliverable: capture packets, report on some data

• Software Defined Networking (SDN)
  – Build a network-wide application in a single program
  – Deliverable: implement MAC learning, ECMP, …

• Defining modern network protocols
  – Frameworks like gRPC => specify messages and generate code
  – Deliverable: build something like snowcast in gRPC
How it works

• I’ll post some sample topics and links to get started
  – Gearup session with some demos: next week
How it works

• I’ll post some sample topics and links to get started
  – Gearup session with some demos: next week
• On/before May 6: submit a short description of what you want to do, get feedback
How it works

• I’ll post some sample topics and links to get started
  – Gearup session with some demos: next week

• On/before May 6: submit a short description of what you want to do, get feedback

• On/before May 19: submit any code + a short writeup explaining what you did, what you learned
How it works

• I’ll post some sample topics and links to get started
  – Gearup session with some demos: next week
• On/before May 6: submit a short description of what you want to do, get feedback
• On/before May 19: submit any code + a short writeup explaining what you did, what you learned
• Meanwhile: office hours/meeting slots available
Thank you!!

I know it’s been rocky, but I hope it’s been fun!