## Networking Roadmap

<table>
<thead>
<tr>
<th>Layer</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. Application</td>
<td>Web security</td>
</tr>
<tr>
<td>4.5. Secure transport</td>
<td>TLS</td>
</tr>
<tr>
<td>4. Transport</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>3. Internet</td>
<td>IP</td>
</tr>
<tr>
<td>2. Link</td>
<td></td>
</tr>
<tr>
<td>1. Physical</td>
<td></td>
</tr>
</tbody>
</table>

### Extra protocols

<table>
<thead>
<tr>
<th>Connect for the first time</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DHCP</td>
</tr>
<tr>
<td>Convert hostname to IP address</td>
<td>DNS, DNSSEC</td>
</tr>
</tbody>
</table>
Secure Channels
Applying crypto technology in practice

• Two simple abstractions cover 80% of the use cases for crypto:
  – “Sealed blob”: Data that is encrypted and authenticated under a particular key ("object security")
  – Secure channel: Communication channel that can’t be eavesdropped on or tampered with ("channel security")

• TLS – a secure channel
Building Secure End-to-End Channels

- End-to-end = communication protections achieved all the way from originating client to intended server
  - With no need to trust intermediaries

- Dealing with threats:
  - Eavesdropping: Encryption (including session keys)
  - Manipulation (injection, MITM): Integrity (use of a MAC); replay protection
  - Impersonation: Signatures

What’s missing?
(Availability …)
Building A Secure End-to-End Channel: SSL/TLS

- SSL = Secure Sockets Layer (predecessor)
- TLS = Transport Layer Security (standard)
  - Both terms used interchangeably
- Security for any application that uses TCP
  - Secure = encryption/confidentiality + integrity + authentication (of server, but not of client)
- Multiple uses
  - Puts the ‘s’ in “https”
  - Secures mail sent between servers (STARTTLS)
  - Virtual Private Networks
Building A Secure End-to-End Channel: TLS

- TLS = *Transport Layer Security*
- Secure channel for applications that use TCP
  - Secure = encryption/confidentiality + integrity + authentication (of server, but *not* of client)
  - E.g., puts the ‘s’ in “https”
Regular web surfing - http:// URL

But if we click here …
Note: Amazon makes sure that all of these images, etc., are now also fetched via https: URLs.

Doing so gives the web page full integrity, in keeping with end-to-end security.

(Browsers do not provide this “promotion” automatically.)
RSA Encryption

We saw RSA in class as a digital signature scheme, but it can also be used as a public-key encryption algorithm:

• The encrypt algorithm is similar to the verify algorithm, and the decrypt similar to the sign algorithm

• Small differences: encrypt the message with special padding, instead of signing a hash of the message
1. Suppose a browser (client) connects to a server which returns a certificate from a trusted CA.
2. Client browser and server will exchange symmetric keys using TLS.
3. Then, they will send encrypted & authenticated traffic to each other.
**HTTPS Connection (SSL / TLS)**

- Browser (client) connects to Amazon’s HTTPS server
- Client picks 256-bit random number $R_B$, sends over list of crypto algorithms it supports
- Server picks 256-bit random number $R_S$, selects algorithms to use for this session
- Server sends over its certificate with its $PK_{Amazon}$

(all of this is in the clear)

- **Client now validates cert**
HTTPS Connection (SSL / TLS), cont.

- For RSA, browser constructs “Premaster Secret” \( PS \)
- Browser sends \( PS \) encrypted using Amazon’s public RSA key \( PK_{Amazon} \)
- Using \( PS \), \( R_B \), and \( R_S \), browser & server derive symm. \textit{cipher keys} (\( C_B \), \( C_S \)) & MAC \textit{integrity keys} (\( I_B \), \( I_S \))
  - One pair to use in each direction

\[ \{PS\}_{PK_{Amazon}} \]

Here’s my cert

\( \approx 2-3 \text{ KB of data} \)
HTTPS Connection (SSL / TLS), cont.

- For RSA, browser constructs “Premaster Secret” **PS**

- Browser sends **PS** encrypted using Amazon’s public RSA key **PK_{Amazon}**

- Using **PS**, **R_B**, and **R_S**, browser & server derive symmetric cipher keys (**C_B, C_S**) & MAC integrity keys (**I_B, I_S**)
  - One pair to use in each direction

These seed a cryptographically strong pseudo-random number generator (PRNG).

Q: why **R_B** and **R_S**?

A: prevents a replay attack, attacker captures handshake from either the client or server and replays it. Why not only one of them? You don't need to check for reuse by the other side... just make sure you don't reuse it on your side!
HTTPS Connection (SSL / TLS), cont.

- For RSA, browser constructs “Premaster Secret” PS
- Browser sends PS encrypted using Amazon’s public RSA key $PK_{Amazon}$
- Using PS, $R_B$, and $R_S$, browser & server derive symm. cipher keys ($C_B$, $C_S$) & MAC integrity keys ($I_B$, $I_S$)
  - One pair to use in each direction
- Browser & server exchange MACs computed over entire dialog so far

Q: Why?
A: So they know they have the same ($C_B$, $C_S$), ($I_B$, $I_S$)
For RSA, browser constructs “Premaster Secret” \( PS \)

- Browser sends PS encrypted using Amazon’s public RSA key \( PK_{Amazon} \)

- Using PS, \( R_B \), and \( R_S \), browser & server derive symm. cipher keys \( (C_B, C_S) \) & MAC integrity keys \( (I_B, I_S) \)
  - One pair to use in each direction

- Browser & server exchange MACs computed over entire dialog so far

- If good MAC, browser displays 

---

**HTTPS Connection (SSL / TLS), cont.**

Here’s my cert

\( \{PS\}_{PK_{Amazon}} \)

~2-3 KB of data

MAC(\( \text{dialog}, I_B \))

MAC(\( \text{dialog}, I_S \))
The communication before the green lock is called the SSL handshake; its purpose is to establish shared symmetric keys for secure communication.
HTTPS Connection (SSL / TLS), cont.

- For RSA, browser constructs “Premaster Secret” \( PS \)
- Browser sends \( PS \) encrypted using Amazon’s public RSA key \( PK_{\text{Amazon}} \)
- Using \( PS \), \( R_B \), and \( R_S \), browser & server derive symm. cipher keys \( (C_B, C_S) \) & MAC integrity keys \( (I_B, I_S) \)
  - One pair to use in each direction
- Browser & server exchange MACs computed over entire dialog so far
- If good MAC, Browser displays
- All subsequent communication encrypted w/ symmetric cipher (e.g., AES128) cipher keys in some agreed upon chaining mode, MACs
  - Sequence #'s included with every message to thwart replay attacks
Alternative: Key Exchange via Diffie-Hellman

- For Diffie-Hellman, server generates random $a$, sends public params and $g^a \mod p$

Q: How can we prevent MITM?
A: Server signs $g^a \mod p$ using $SK_{Amazon}$, browser verifies using $PK_{Amazon}$ from server certificate
Alternative: Key Exchange via Diffie-Hellman

- For Diffie-Hellman, server generates random $a$, sends public params and $g^a \mod p$
  - Signed with server’s private key
- Browser verifies signature using PK from certificate
- Browser generates random $b$, computes $PS = g^{ab} \mod p$, sends to server its public key
- Server also computes $PS = g^{ab} \mod p$
- Remainder is as before: from $PS$, $R_B$, and $R_S$, browser & server derive symm. cipher keys ($C_B$, $C_S$) and MAC integrity keys ($I_B$, $I_S$), etc...
RSA versus Diffie-Hellman

• **Forward secrecy**: If attacker steals long term secret key of server, $SK_{Amazon}$, should not be able to read past conversations (cannot compromise past session keys ($C_B, C_S$) & ($I_B, I_S$))

• Why matters?
  – Attackers log traffic now. Compromise key in future and try to decrypt the traffic.

• TLS with RSA does not have forward secrecy

• TLS DH has forward secrecy
Exchange with RSA

- For RSA, browser constructs “Premaster Secret” PS
- Browser sends PS encrypted using Amazon’s public RSA key $PK_{Amazon}$
- Using PS, $R_B$, and $R_S$, browser & server derive symm. cipher keys ($C_B$, $C_S$) & MAC integrity keys ($I_B$, $I_S$)
  - One pair to use in each direction
- Browser & server exchange MACs computed over entire dialog so far
- If good MAC, Browser displays 💭
- All subsequent communication encrypted w/ symmetric cipher (e.g., AES128) cipher keys in some chaining mode, MACs
  - Sequence #’s thwart replay attacks

---

Q: Forward secrecy?
A: No forward secrecy because attacker can decrypt PS and knows $R_B$, and $R_S$ and computes secrets
Exchange via Diffie-Hellman

- For Diffie-Hellman, server generates random $a$, sends public params and $g^a \mod p$
  - Signed with server’s private key
- Browser verifies signature using PK from certificate
- Browser generates random $b$, computes $PS = g^{ab} \mod p$, sends to server
- Server also computes $PS = g^{ab} \mod p$
- Remainder is as before: from $PS$, $R_B$, and $R_S$, browser & server derive symm. cipher keys ($C_B$, $C_S$) and MAC integrity keys ($I_B$, $I_S$), etc…

Q: Forward secrecy?
A: Has forward secrecy because shared secret never sent over the network! If attacker as $SK_{Amazon}$, cannot decrypt $a.$
HTTPS Connection (SSL / TLS)

- Browser (client) connects via TCP to Amazon’s HTTPS server
- Client picks 256-bit random number $R_B$, sends over list of crypto protocols it supports
- Server picks 256-bit random number $R_S$, selects protocols to use for this session
- Server sends over its certificate
- (all of this is in the clear)
- **Client now validates cert**
Certificates

• Browser compares domain name in cert w/ URL
  – Note: this provides an end-to-end property
    (as opposed to say a cert associated with an IP address)

• Browser accesses separate cert belonging to issuer or CA
  – These are hardwired into the browser – and trusted!

• Browser applies CA’s public key to verify signature S, obtaining hash of what CA signed
  – Compares with its own SHA-256 hash of Amazon’s cert

• Assuming hashes match, now have high confidence it’s indeed Amazon with that PK…
  – assuming signatory is trustworthy

= assuming didn’t lose private key; assuming didn’t sign thoughtlessly
End-to-End ⇒ Powerful Protections

• Attacker runs a sniffer to capture our WiFi session?
  – (maybe by breaking crummy WEP security)
  – **But:** encrypted communication is unreadable
    • No problem!

• DNS cache poisoning gives client wrong IP address
  – Client goes to wrong server
  – **But:** certificate won’t match
    • No problem!

• Attacker hijacks our connection, injects new traffic
  – **But:** data receiver rejects it due to failed integrity check
    • No problem!
Powerful Protections, cont.

• Attacker manipulates routing to run us by an eavesdropper or take us to the wrong server?
  – But: they can’t read; we detect impersonation
    • No problem!

• Attacker slips in as a Man In The Middle?
  – But: they can’t read, they can’t inject
  – They can’t even replay previous encrypted traffic
  – No problem!
Validating Amazon’s Identity, cont.

- Browser retrieves cert belonging to the **issuer**
  - These are hardwired into the browser – and **trusted**!

- What if browser can’t find a cert for the issuer?
This Connection is Untrusted

You have asked Firefox to connect securely to www.mikestoolbox.org, but we can't confirm that your connection is secure.

Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.

What Should I Do?

If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.

Get me out of here!

Technical Details

www.mikestoolbox.org uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is not trusted.

(Error code: sec_error_untrusted_issuer)

I Understand the Risks

Safari can't verify the identity of the website “www.mikestoolbox.org”.

The certificate for this website was signed by an unknown certifying authority. You might be connecting to a website that is pretending to be “www.mikestoolbox.org”, which could put your confidential information at risk. Would you like to connect to the website anyway?

Show Certificate  Cancel  Continue
Validating Amazon’s Identity, cont.

• Browser retrieves cert belonging to the issuer – These are hardwired into the browser – and trusted!

• What if browser can’t find a cert for the issuer?

• If it can’t find the cert, then warns the user that site has not been verified – Can still proceed, just without authentication

• Q: Which end-to-end security properties do we lose if we incorrectly trust that the site is whom we think?

• A: All of them! – Goodbye confidentiality, integrity, authentication – Man in the middle attacker can read everything, modify, impersonate
SSL / TLS Limitations

• Properly used, SSL / TLS provides powerful end-to-end protections

• Used by many sites, reasons why not all sites:
  – Cost of public-key crypto (fairly minor)
    o Takes non-trivial CPU processing (but today a minor issue)
    o Note: symmetric key crypto on modern hardware is non-issue
  – Hassle of buying/maintaining certs (Let’s Encrypt addresses it)
  – Integrating with other sites that don’t use HTTPS
  – Latency: extra round trips ⇒ 1st page slower to load
  – Cannot cache encrypted pages
SSL / TLS Limitations, cont.

• Problems that SSL / TLS does not take care of?

• TCP-level denial of service
  – SYN flooding
  – RST injection
    o (but does protect against data injection!)

• server-side coding/logic flaws

• Vulnerabilities introduced by server inconsistencies
Regular web surfing: http://URL

So *no integrity* - a MITM attacker can alter pages returned by server ...

And when we click here …
… attacker has changed the corresponding link so that it’s ordinary http rather than https!

We never get a chance to use TLS’s protections! :-(

“sslstrip” attack
SSL / TLS Limitations, cont.

- Problems that SSL / TLS does not take care of?

- server-side coding/logic flaws
- Vulnerabilities introduced by server inconsistencies
- Browser coding/logic flaws
- User flaws
  - Weak passwords
  - Phishing
- Issues of trust …
TLS/SSL Trust Issues

• User has to make correct trust decisions …
Welcome to eBay

Ready to bid and buy? Register here

Join the millions of people who are already a part of the eBay family. Don't worry, we have room for one more.

Register as an eBay Member and enjoy privileges including:

- Bid, buy and find bargains from all over the world
- Shop with confidence with PayPal Buyer Protection
- Connect with the eBay community and more!

Sign in to your account

Back for more fun? Sign in now to buy, bid and sell, or to manage your account.

User ID: bieber
Password: ************

Keep me signed in for today. Don't check this box if you're at a public or shared computer.

Sign in

Having problems with signing in? Get help.

Protect your account: Create a unique password by using a combination of letters and numbers that are not
Welcome to eBay

Ready to bid and buy? Register here

Join the millions of people who are already a part of the eBay community.

Register as an eBay Member and enjoy privileges including:

- Bid, buy and find bargains from all over the world
- Shop with confidence with PayPal Buyer Protection
- Connect with the eBay community and more!

Register

Internet Explorer

When you send information to the Internet, it might be possible for others to see that information. Do you want to continue?

- In the future, do not show this message.

Yes  No

Sign in

Having problems with signing in? Get help.

Protect your account: Create a unique password by using a combination of letters and numbers that are not
Please confirm your identity.

Please answer security question

Select your secret question...

Answer the secret question you provided.

What is your other eBay user ID or another email?

What email used to be associated with this account?

Have you ever sold something on eBay?
  ○ No
  ○ Yes

Certificate

Certificate Information

This certificate is intended for the following purpose(s):
  • Ensures the identity of a remote computer

* Refer to the certification authority’s statement for details.

Issued to: rover.ebay.com

Issued by: VeriSign Class 3 Secure Server CA - G3

Valid from 10/22/2010 to 12/1/2012

[Install Certificate] [Issuer Statement]

OK
Please confirm your identity.

Please answer security question:

Select your secret question...

Answer the secret question you provided.

What is your other eBay user ID or another email used to be associated with this account?

Have you ever sold something on eBay?

- No
- Yes
Please confirm your identity.

Please answer security question

Select your secret question...

Answer the secret question you provided.

What is your other eBay user ID or another

What email used to be associated with this account?

Have you ever sold something on eBay?

- No
- Yes

Certificate status:

This certificate is OK.
The equivalent as seen by most Internet users:

Internet Explorer

This page has an unspecified potential security flaw. Would you like to continue?

Yes  No

(note: an actual Windows error message!)
TLS/SSL Trust Issues, cont.

• “Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.”
  – Matt Blaze, circa 2001

• So how many CAs do we have to worry about, anyway?
### Keychain Access

Click to lock the System Roots keychain.

#### Keychains

- **login**
- **Micr...certificates**
- **System**
- **System Roots**

#### A-Trust-Qual-02

**Root certificate authority**

Expires: Tuesday, December 2, 2014 3:00:00 PM PT

- This certificate is valid

<table>
<thead>
<tr>
<th>Name</th>
<th>Kind</th>
<th>Expires</th>
<th>Keychain</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-Trust-nQual-01</td>
<td>certificate</td>
<td>Nov 30, 2014 3:00:00 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>A-Trust-nQual-03</td>
<td>certificate</td>
<td>Aug 17, 2015 3:00:00 PM</td>
<td>System Roots</td>
</tr>
<tr>
<td>A-Trust-Qual-01</td>
<td>certificate</td>
<td>Nov 30, 2014 3:00:00 PM</td>
<td>System Roots</td>
</tr>
<tr>
<td>A-Trust-Qual-02</td>
<td><strong>certificate</strong></td>
<td><strong>Dec 2, 2014 3:00:00 PM</strong></td>
<td><strong>System Roots</strong></td>
</tr>
<tr>
<td>AAA Certificate Services</td>
<td>certificate</td>
<td>Dec 31, 2028 3:59:59 PM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AC Raíz Certicámara S.A.</td>
<td>certificate</td>
<td>Apr 2, 2030 2:42:02 PM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AddTrust Class 1 CA Root</td>
<td>certificate</td>
<td>May 30, 2020 3:38:31 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AddTrust External CA Root</td>
<td>certificate</td>
<td>May 30, 2020 3:48:38 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AddTrust Public CA Root</td>
<td>certificate</td>
<td>May 30, 2020 3:41:50 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AddTrust Qualified CA Root</td>
<td>certificate</td>
<td>May 30, 2020 3:44:50 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>Admin-Root-CA</td>
<td>certificate</td>
<td>Nov 9, 2021 11:51:07 PM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AdminCA-CD-T01</td>
<td>certificate</td>
<td>Jan 25, 2016 4:36:19 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Commercial</td>
<td>certificate</td>
<td>Dec 31, 2030 6:06:06 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Networking</td>
<td>certificate</td>
<td>Dec 31, 2030 6:08:24 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Premium</td>
<td>certificate</td>
<td>Dec 31, 2040 6:10:36 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AffirmTrust Premium ECC</td>
<td>certificate</td>
<td>Dec 31, 2040 6:20:24 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>America Onli...nation Authority 1</td>
<td>certificate</td>
<td>Nov 19, 2037 12:43:00 PM</td>
<td>System Roots</td>
</tr>
<tr>
<td>America Onli...nation Authority 2</td>
<td>certificate</td>
<td>Sep 29, 2037 7:08:00 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AOL Time W...cation Authority 1</td>
<td>certificate</td>
<td>Nov 20, 2037 7:03:00 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>AOL Time W...cation Authority 2</td>
<td>certificate</td>
<td>Sep 28, 2037 4:43:00 PM</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root CA</td>
<td>certificate</td>
<td>Feb 9, 2035 1:40:36 PM</td>
<td>System Roots</td>
</tr>
<tr>
<td>Apple Root Certificate Authority</td>
<td>certificate</td>
<td>Feb 9, 2025 4:18:14 PM</td>
<td>System Roots</td>
</tr>
<tr>
<td>Application CA G2</td>
<td>certificate</td>
<td>Mar 31, 2016 7:59:59 AM</td>
<td>System Roots</td>
</tr>
<tr>
<td>ApplicationCA</td>
<td>certificate</td>
<td>Dec 12, 2017 7:00:00 AM</td>
<td>System Roots</td>
</tr>
</tbody>
</table>

**167 items**
TLS/SSL Trust Issues

• “Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.”  
  – Matt Blaze, circa 2001

• So how many CAs do we have to worry about, anyway?

• Of course, it’s not just their greed that matters …
Solo Iranian hacker takes credit for Comodo certificate attack

Security researchers split on whether 'ComodoHacker' is the real deal

By Gregg Keizer
March 27, 2011 08:39 PM ET

Computerworld - A solo Iranian hacker on Saturday claimed responsibility for stealing multiple SSL certificates belonging to some of the Web's biggest sites, including Google, Microsoft, Skype and Yahoo.

Early reaction from security experts was mixed, with some believing the hacker's claim, while others were dubious.

Last week, conjecture had focused on a state-sponsored attack, perhaps funded or conducted by the Iranian government, that hacked a certificate reseller affiliated with U.S.-based Comodo.

On March 23, Comodo acknowledged the attack, saying that eight days earlier, hackers had obtained nine bogus certificates for the log-on sites of Microsoft's Hotmail, Google's Gmail, the Internet phone and chat service Skype and Yahoo Mail. A certificate for Mozilla's Firefox add-on site was also acquired.
A Dutch company appears to have issued a digital certificate for Google.com to someone other than Google, who may be using it to try to re-direct traffic of users based in Iran.

Yesterday, someone reported on a Google support site that when attempting to log in to Gmail the browser issued a warning for the digital certificate used as proof that the site is legitimate, according to this thread on a Google support forum site.
Final Report on DigiNotar Hack Shows Total Compromise of CA Servers

The attacker who penetrated the Dutch CA DigiNotar last year had complete control of all eight of the company’s certificate-issuing servers during the operation and he may also have issued some rogue certificates that have not yet been identified. The final report from a

Evidence Suggests DigiNotar, Who Issued Fraudulent Google Certificate, Was Hacked Years Ago

from the diginot dept

The big news in the security world, obviously, is the fact that a fraudulent Google certificate made its way out into the wild, apparently targeting internet users in Iran. The Dutch company DigiNotar has put out a statement saying that it discovered a breach back on July 19th during a security audit, and that fraudulent certificates were generated for "several dozen" websites. The only one known to have gotten out into the wild is the Google one.
TLS/SSL Trust Issues

• “Commercial certificate authorities protect you from anyone from whom they are unwilling to take money.”
  – Matt Blaze, circa 2001

• So how many CAs do we have to worry about, anyway?

• Of course, it’s not just their greed that matters …

• … and it’s not just their diligence & security that matters …
  – “A decade ago, I observed that commercial certificate authorities protect you from anyone from whom they are unwilling to take money. That turns out to be wrong; they don't even do that much.” - Matt Blaze, circa 2010
Conclusion

• Use SSL/TLS to secure communications end-to-end
• Relies on trustworthiness of certificates, which does not always hold