Crypto and modern authentication on the web
• Certificates, TLS, HTTPS, revocation
• Trusting the trusted third party?
• Web of trust
• Passwords etc.
Digital certificates

TLS, HTTPS, Revocation

https://s-media-cache-ak0.pinimg.com/originals/ed/af/55/edaf5554d92824ef3555d8b9fbff60c5.jpg
• For convenience, we will use PK_A and SK_A to denote public and secret keys for Alice
Trusted third party, revisited (1)

- TTP is a bottleneck for every conversation
- TTP must be online to start a new conversation
- TTP can read every message
- TTP must be trusted to tell the truth!
- Does not solve bootstrapping problem
Trusted 3rd party, revisited (2)

Alice

Bob: Verify cert with PK_T, verify message with PK_A

S(SK_A, E(PK_B, m)) + cert

Alice owns PK_A. Signed, PK_T

PK_T plus verification
With certificates

- TTP is a bottleneck for every conversation
- TTP must be online to start a new conversation
- TTP can read every message
- TTP must be trusted to tell the truth!
- Does not solve bootstrapping problem
Certificates in practice

- TTP = Certificate Authority
  - Verisign, Comodo, Thawt, etc.
- Alice = web server
- Bob = user who visits alice.com
  - Validate talking to the real alice.com
  - Set up encrypted session for HTTPS
- This is a hierarchical public key infrastructure (PKI)
This certificate has been verified for the following uses:

<table>
<thead>
<tr>
<th>SSL Client Certificate</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>SSL Server Certificate</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Issued To</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name (CN)</td>
</tr>
<tr>
<td>Organization (O)</td>
</tr>
<tr>
<td>Organizational Unit (OU)</td>
</tr>
<tr>
<td>Serial Number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issued By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name (CN)</td>
</tr>
<tr>
<td>Organization (O)</td>
</tr>
<tr>
<td>Organizational Unit (OU)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Period of Validity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begins On</td>
</tr>
<tr>
<td>Expires On</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fingerprints</th>
</tr>
</thead>
</table>
Certificate types

Why are these different?

This is an EV (extended validation) certificate; browsers show the full name for these kinds of certs.
Transport layer security (TLS)

- Runs on top of TCP/IP
- Protocols for secure comms
  - Confidentiality with block and stream ciphers
  - Integrity with MACs
  - Authenticity with certificates
- Replacement for SSL (secure sockets layer)
  - Several problems including padding attacks
TLS protocol overview

browser (initiates connection)

server (authenticates itself)

Client hello
Version, crypto options, \texttt{nonce}

Server hello + server cert (PKs)
Version, crypto options, \texttt{nonce}, signed PK certificate

Server key exchange (if using DH)
PreMaster secret encrypted with server’s PK

Compute \( K \) based on nonces & PreMaster

Client key exchange

\texttt{Switch to negotiated cipher}\texttt{Data transmission}

Compute \( K \) based on nonces & PreMaster
HTTPS

- HTTP “on top of” TLS

- Pros: **Avoid MITM**
  - Includes e.g. reducing video quality, inserting ads

- Cons
  - Takes more time
  - Network service/ISP can’t compress or cache it
  - Network service/ISP wants to insert ads

https://www.eff.org/https-everywhere
Revoking certificates

- When you detect compromise or change keys, you have to notify the CA

- CA then *revokes* the certificate
  - Revocation list
  - Online cert status protocol
  - Short expiry times
Revocation list

- CA publishes list of revoked certs
- User (in practice, browser) must periodically download the newest list
  - Check when validating a certificate
- Vulnerability window since last list update
  - Or until certificate expires
- Can be beaten via DOS (why?)
Online certificate status

- During validation, ask CA whether cert is revoked
  - Transfers less excess data

- Gets rid of vulnerability window
  - But can’t accept any cert if CA is not online!

- And, the CA gets to know where you browse

- What happens if you don’t get an answer?
OCSP stapling

- Certificate presenter must append time-stamped OCSP validation ("stapled" on)
- Easier to abort if you don’t get one
Short expiration

- Make all certificates have **very short** expirations (e.g. 10 min or less)
  - For the most part, renew automatically
  - But, some browsers accept expired certs
- Revocation == decline to renew
- Stapling more or less adds this property
Tusting the Trusted Third Party
Where do CAs come from?

- CA public keys shipped with browsers, OS
  - iOS9 shipped with >50 that start with A-C
    - see [here](#) for full list
  - I had >100 in my browser as of last night
CA compromise

- 2001: Verisign issued two code-signing certificates for Microsoft Corporation
  - To someone who *didn’t actually* work at MS
  - No functional revocation paradigm
- 2011: Signing keys compromised at Comodo and DigiNotar
  - Bad certs for Google, Yahoo!, Tor, others
  - Seem to have been used mostly in Iran
- Some CAs are less picky than others
Case study: Superfish (Feb 2015)

- Lenovo laptops shipped with “Superfish” adware
- Installs self-signed root cert into browsers
  - MITM on every HTTPS site to inject ads
- Worse: Same private key for every laptop
  - Password = “komodia” (company
- **Lenovo** “did not find any evidence to substantiate security concerns”

Case study: Symantec

• Sept. 2015: Google: Symantec not properly validating certs
  • Thawte issued an EV cert for google.com
  • Issued during ”internal testing”

• Oct 2015: Symantec finds 2500 such errors

• January 2017: at least 30k problem certs!

• March 2017: Chrome stops trusting Symantec EV!
Fixing rogue CA problems

- Limit which CAs can issue for which domains
- Certificate pinning
  - Browser, apps fix certain CA or cert for a server
  - Shipped with product, or on first use
  - Not always appropriate, hard to maintain
Fixing rogue CA problems (2)

- Broad surveillance
  - People on many networks report certs to Notaries
  - Check that others saw the same cert you did
  - Privacy implications

- Cert. transparency: Public unforgeable audit log
  - Uses crypto, Merkle hash trees
    - Only accept certs published in log
  - Same idea: \textit{Non-equivocation}

- How Google caught Symantec

\url{https://www.eff.org/observatory} \url{https://www.eff.org/sovereign-keys}
Rogue verification

• Many certs are “bad”
  • Self-signed, expired, hostname mismatch …

• Most of these are false positives!
  • Then what?

• End user warnings

• “Custom” validation

• Ignore expiration dates
Web of trust
Web of trust

- Alternative PKI — not hierarchical
  - Pioneered by PGP

- Don’t rely on centralized authorities

- Everyone issues certificates for people they know
Trust chains in web of trust

Alice trusts Bob
Bob trusts Cookie
Cookie vouches for Donald
Donald sends message to Alice

vouches for

sends message to
A matter of trust

- Context:
  - Alice trusts Bob to diligently check identity
  - But Bob is only signing identity, not necessarily belief that Cookie is equally vigilant

- Transitivity: Alice trusts Bob, and Bob trusts Cookie.
  - But does that mean Alice should trust Cookie?
  - Trust for honesty == trust for good judgment?
Web-of-trust in practice

- Automatically find many such paths
  - More, shorter paths = higher confidence?

- Difficult to use
  - Still have bootstrapping problems
  - When should I agree to sign what?
  - Historically, serious UX problems as well
Passwords, etc.

http://lorrie.cranor.org/quilts/
Threats to passwords

- Online guessing
- Offline guessing
- Phishing
- Keyloggers
- Shoulder surfing
Online guessing

• Guessing directly in the UI

• Defenses:
  • Password blacklists
  • Throttling (time, captcha)
  • Lockout after k attempts
Offline guessing

- Attacker obtains password file / database
- Attempts to use it to gain access
  - Key observation: Don’t store in plaintext!
Offline guessing: Defenses

- Hashing
  - Not encryption! (Why?)
  - Salting
- Honeywords