Outline: Crypto Part 3

- Recap: Secure communication channels
- Authenticating endpoints: Certificates (Certs)
- Issuing Certs and Certificate Infrastructure (PKI)
- Attacks, Countermeasures
- Real World Secure Channels: SSL / TLS
Template For Secure Channels (TLS, SSH, IPSec, …)

Key Exchange (“Handshake”)

A

B

Symmetric Encryption (“Record Protocol”)

<encrypted data>

<encrypted data>

<encrypted data>

...

• Last lecture: Naïve strategy can be secure against passive adversaries.
• But the above template does not provide authentication & integrity.
Securing Key Exchange against Active Attackers

**Key Challenge:** Authenticity: How do we know that PK is really Bob’s?

1. Alice picks a random AES key $K$.
2. Alice encrypts $PK'$ with $K$: $C = Enc(PK', K)$.
3. Alice encrypts $PK$ with $K$: $C = Enc(PK, K)$.
4. Alice sends $PK$ and $PK'$ to Bob.
5. Bob decrypts $PK'$ with $SK$: $K ← Dec(SK, C)$.
7. Bob computes $AES-GCM(K, M_1)$.
8. Bob sends $K$ to Alice.
9. Alice decrypts $PK$ with $SK$: $K ← Dec(SK, C)$.
10. Alice computes $AES-GCM(K, M_1)$.
Authentication with Certificates ("Certs")

Suppose we had a globally trusted entity, BlaséInc.

BlaséInc could issue certificates ("certs") that state what other organizations’ public keys are.

Cert = a document that says:
1. An Entity (e.g., Bob) has a public key that is:
2. pk=0x7b5532..., where the document is
3. signed using the BlaséInc’s private signing key

Trusted entity, BlaséInc, known as a Certificate Authority (CA)
Authentication with Certificates ("Certs")

VK* pre-installed on every machine by manufacturer or built into OS code.
Securing Key Exchange against Active Attackers

- Is cert for Bob?
- Does the cert have correct signature (check w/ VK*)?

C = Enc(PK, K)

Verify Same Key (MAC(K, Dialogue))

C = Dec(SK, C)

AES-GCM(K, M₁)

Pick random AES key K

Alice

VK*

Bob

PK, SK

Keygen

(Active Attacker)
Recap: Secure communication channels
Authenticating endpoints: Certificates (Certs)
Issuing Certs and Certificate Infrastructure (PKI)
Attacks, Countermeasures
Real World Secure Channels: SSL / TLS
Issuing Certificates: Validation

\[(PK^*, SK^*)\]

uchicago.edu

\[(PK_1, SK_1)\]

cert_1 = [PK_1, “uchicago.edu”, σ_1]

- CA must check that key \(PK_1\) really does belong to “uchicago.edu”

Domain Validation (DV): Check that party with that key can control domain.

Org. Validation (OV) and Extended Validation (EV): Also check company name, location etc via public records.
ACME Protocol by Let’s Encrypt

1. Requestor submits public key and request to CA
2. CA gives a challenge to requestor
3. Requestor places challenge on server or DNS records
4. CA checks challenge and then issues cert if challenge matches

\[
\text{cert}_1 = [PK_1, "uchicago.edu", \sigma_1]
\]
Scaling Certificates to the Internet

Having one CA works fine if the Internet has just a few entities and everyone agrees that the CA is trustworthy.
Scaling Certificates to the Internet

But the Internet has billions of devices...
And not everyone agrees on a trusted party (CA)...
To handle scaling:
- Allow a trusted Root CA to delegate their trust to multiple intermediate CA’s
- Any of these intermediate CA’s can then create a certificate for someone
  - 100’s of intermediate CA’s on the Internet
Scaling: Intermediate CAs and Cert Chains

To check PK₂ recursive validation:
1) Check cert₂ to make sure PK₂ for uchicago.edu
2) Get PK₁ and cert₁ to check sig of cert₂
3) If cert₁ issued by root CA, use PK* to its check sig.

PK₁ bound to Root ⇒ PK₁ bound to CA ⇒ PK₂ bound to uchicago.edu
X.509 Certificates

Cert Content Includes:

• Cert’s Serial number
• Cert’s Expiration date
• Common name of subject (e.g., Bob [google.com])
• Public key of subject
• Extensions (possibly many)
• CA info (name of CA that is issuing the cert, etc.)
• CA’s Signature (on hash of cert)
<table>
<thead>
<tr>
<th>Public Key Info</th>
<th>RSA Encryption (1.2.840.113549.1.1.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
<td>None</td>
</tr>
<tr>
<td>Public Key</td>
<td>CA E9 01 25 77 E9 74 B8 C8 F7 99 DA D6 87 79 35 D7 31 CA D7 83 11 83 32 FA F1 43 CC C8 85 7B 76 EF 79 B8 4B B8 E0 35 87 EE A4 34 17 DC 5A 0D 5A 04 D3 F1 BA E7 98 9F 49 FC D5 B9 2C FB C8 DD 36 47 4D 07 FE 41 11 75 B0 42 F7 6D 40 4C BF F5 B6 C7 FE 05 0D DE 3B 7C E9 9F 6A 1C 1C 89 2E AA E8 F5 E3 5B 04 55 16 B0 48 92 C7 F9 37 11 99 F8 C5 85 C1 24 96 71 6F 7B B6 6B 35 39 92 8C EF 17 91 D1 97 D7 EF 93 6E 96 F1 EE C6 0D 5A EA 39 C6 4E 33 E2 CA F2 9A 41 F4 A2 41 9C E8 EA 46 FB EF 71 C0 A6 D3 C6 A5 94 81 4B 12 5E 80 63 87 7C 2F A6 BA A5 9A 31 9E 81 63 7F 0F 26 25 B6 6D 62 C2 AD B4 E7 68 FD C9 F8 86 2C 3F F8 E1 59 F3 3E 73 08 DF 6C 92 98 21 D2 AD EF 23 E7 33 A2 D4 5E 67 74 E3 AB 08 DF 15 31 9A 9D 3B 36 7D 6B 77 48 60 17 A4 10 F3 17 77 53 E0 29 D9 F9 A4 12 DF 39 DA D1</td>
</tr>
<tr>
<td>Exponent</td>
<td>65537</td>
</tr>
<tr>
<td>Key Size</td>
<td>2,048 bits</td>
</tr>
<tr>
<td>Key Usage</td>
<td>Encrypt, Verify, Wrap, Derive</td>
</tr>
<tr>
<td>Signature</td>
<td>11 F9 F9 6D C6 92 D1 B9 E7 13 E6 0D BA E6 19 65 BB 16 4B DE E1 C2 3A 62 55 D1 61 80 93 F0 2A B2 7D 9E 76 CE 10 4A D6 96 4E 5C 00 5D BD 8C 83 74 CF C1 14 91 2B 15 4B 2D 67 4A 84 A2 A4 54 7A B1 C9 8E F5 A7 93 8D 30 BF OC 9B EF 98 36 D4 BD B6 11 63 C2 51 23 71 7B 8D 4C 9B B7 AD A9 FE A8 4E 48 B2 83 A1 36 75 97 2B 36 4A 72 C4 AA C6 B6 A8 4A C0 F4 37 BD 0E B1 A8 FB EC B6 B5 A8 CB A2 BB C7 47 D7 D4 DB 05 80 72 BA CB C7 79 B1 63 CC 55 D7 68 9C 41 2B E7 D9 F0 C2 8F 11 15 7D C5 D5 34 27 5C 7C B5 D9 A8 3F 3C DF C5 1D AA 52 03 19 AE 5B FC FF 42 68 15 A3 01 CB F8 0E FE 9B A1 76 B8 43 1C 6B 9C 57 38 87 B1 3B 4A 33 98 09 CF 25 F4 75 34 AE 1E 7B CD 0F EF A0 4C 5B 92 B7 F1 FD 66 1B 49 67 B0 65 5A 90 1D 1D 54 D2 CF FF FD 07 DC 7A 88 59 51 55 16 7F 83 D4 FC 19 F4 28</td>
</tr>
</tbody>
</table>
Root CA’s & Root Certificates
Outline: Crypto Part 3

- Recap: Secure communication channels
- Authenticating endpoints: Certificates (Certs)
- Issuing Certs and Certificate Infrastructure (PKI)
- Attacks, Countermeasures
- Real World Secure Channels: SSL / TLS
What if attacker got a “valid” cert for uchicago.edu that has their malicious key?

- “Machine-in-the-middle” can read/change all traffic undetected

Google warns of man-in-the-middle attacks on Iranian users

Intrusion into Dutch SSL provider led to cyber snooping
CA Security

Some common attacks to get rogue certificate:
• Fool or bypass a CA’s validation process
• Compromise a CA organization and generate malicious cert’s

Figure 3: Boulder architecture. Let’s Encrypt developed and operates a Go-based open-source CA software platform named Boulder, which is composed of single-purpose components that communicate over gRPC, as illustrated here. The certificate lifecycle unfolds roughly from left to right in the diagram.

“Let’s Encrypt: An Automated Certificate Authority to Encrypt the Entire Web”, CCS 2019
Sample of CA Security Incidents


• 2011, Root CA DigiNotar: Hacker issues rogue cert for *.google.com, others. Used to MitM by Iranian government.

• 2013, Root CA TurkTrust: Accidentally issues intermediate CA cert, used to issue gmail.com cert.

• …

• 2019, Root CA Comodo: Pushes email login credentials to public GitHub repo…

(Slide inspiration: Dan Boneh)
Countermeasure: Public-Key Pinning

• **Goal:** Eliminate Root / Intermediate CA’s with bad hygiene or who you don’t trust

• Server (e.g., website) can tell client (e.g., browser) to only accept certs signed by certain CA’s
  • Code trusted CA keys into client app (e.g., Chrome only trusts certs signed by Google’s CA), or
  • Send special application message telling client what to pin (More common)

• Helped discover some rogue certs from previous slide

• What are some problems with this defense?
  • If server hacked… attacker can pin a malicious key/cert: will only connect w/ attacker cert!
  • Website error: pin wrong or broken key… website inaccessible!

Now deprecated because of these issues
Countermeasure: Revocation

Publicly list bad (revoked) certificates so they are no longer accepted

- CA or Server (that was issued cert) can revoke

**Mass Revocation: Millions of certificates revoked by Apple, Google & GoDaddy**

The DarkMatter debate is already having industry-wide ramifications

Millions of SSL/TLS certificates – among other digital certificates – are being revoked right now as a result of an operational error that caused the generation of non-compliant serial numbers.

March 3, 2020

**Let’s Encrypt to Revoke 3 Million SSL Certificates on March 4**

The world’s leading free SSL provider announces that millions of certificates are being revoked due to a bug they discovered days ago – giving subscribers potentially only hours to respond
Cert Revocation Lists (CRLs)

CA's CRL Server

(\(PK^*, SK^*\))

- Each CA provides a list of revoked cert's
- Clients can download CRL and check cert's they receive against the list
- Problems:
  - List will get too large
  - Difficult to keep current

Revoked serial numbers:
09823342365
23423482349
98072344456
...

Client's computer
Revocation: Online Certificate Status Protocol (OCSP)

- Add another server to connect to, slowing connection
- What if OCSP server times out?
- Privacy problem?
Revocation: OCSP Stapling

- TLS Extension that allows for OCSP response to be included with cert
- Client checks CA signature and time-stamp on response (~hours old).
- Certs can have “must staple” extension.
Revocation: OCSP Stapling

- OCSP server goes down => uchicago.edu goes down (no OCSP response to attach to cert)
Certificate Transparency (CT): How do we find rogue certs?

Scenario: Attackers compromise a CA and create rogue certs for google.com that have (1) attacker’s public keys and (2) valid CA signature.

How does Google or the CA discover these rogue certs were issued or in use?

Cert Transparency:
- Require all cert’s added to public audit logs
- Domains & CA’s can check audit logs for rogue certs & revoke them
Certificate Transparency (CT)

Simplified strategy to find certificates we should revoke:

• An auditor maintains a list (log) of every certificate ever issued
• Whenever a CA issues a cert, they submit (add) cert to this log
• Clients only accept a server’s cert if it appears on the log
• Each server (domain) can now monitor the logs to see if anyone (and who) issued a rogue certificate for them
  • If so, add the rogue cert to revocation lists
  • If CA has pattern of issuing rogue cert’s, ban them
Certificate Transparency (CT)

- CT Log server maintains a list of all certs issued by CA(s).
- “Monitors” check for improper certs; help domains & CA(s) find bad cert’s
- Clients only accept certs if server also has valid SCT’s for certs
- In practice: multiple CT log servers

SCT: Signed Proof that cert was logged

CA

(PK*, SK*)

CA

CT Log server

Hello

google.com

(PK, Cert, SCT)

Cert + SCT

Cert

Cert + SCT

Cert

cert1
cert2
cert3

...
Challenges with CT

- List is huuuuge (every issued cert… solution: temporal sharding)
- Trust the CT Log?
- (Monitors) Who checks the logs?
- Privacy (e.g., enterprise has private servers)?
Cert Transparency & OCSP

How do CT and OCSP compare?

- OCSP: Allows clients to determine if a cert is valid
- CT: Allows domains (cert owners) and CA’s to find malicious cert’s
Outline: Crypto Part 3

- Recap: Secure communication channels
- Authenticating endpoints: Certificates (Certs)
- Issuing Certs and Certificate Infrastructure (PKI)
- Attacks, Countermeasures
- Real World Secure Channels: SSL / TLS
**TLS in the Protocol Stack**

- **Goal**: Allow any application using TCP to transmit data with E2E security

- TLS takes requests from applications (e.g. browser speaking HTTP) and transmits them securely to another host on the Internet
History: SSL/TLS

• SSL = “Secure Sockets Layer”
• TLS = “Transport Layer Security” (renaming of SSL)
TLS Adoption (HTTPS)

Percentage of pages loaded over HTTPS in Chrome by platform

(Source: transparencyreport.google.com, via Matt Green)
Attacks on TLS

- SLOTH
- POODLE
- BEAST
- Cross-protocol DH/ECDH attack
- Bleichenbacher
- BLEACH
- BREACH
- HEIST
- CCS injection
- Lucky microseconds
- Virtual host confusion
- SSL stripping
- STARTTLS injection
- Janner et al
- DROWN
- SMACK
- BREACH
- HEIST
- FREAK
- Logjam
- RC4 biases, rc4nomore, Bar Mitzvah
- Collisions
- Type
- IME Implementation
- Triple handshake attack
- Cross-protocol DH/ECDH attack
- OpenSSL
- LibreSSL
- BoringSSL
- NSS
- GnuTLS
- SCG
- OpenSSL
- LibreSSL
- BoringSSL
- NSS
- GnuTLS
- SCG
- Web browsers: Chrome, Firefox, IE/Edge, Safari
- Web servers: Apache, IIS, nginx, node, ...
- Application SDKs
- Certificates
- Protocols:
- HTTP, IMAP, ...
TLS Protocol: Very Similar to Our Template

- Is cert for Bob?
- Is cert in CT logs and has it been revoked?
- Does the certificate chain have valid signatures?

Hello [Protocols & Init]

cert=[PK,"Bob",σ]

C = Enc(PK, K)

Verify Integrity & Keys (MAC(K, Dialogue))

AES-GCM(K,M₁)

PK, SK ← Keygen

Alice

Bob

Pick random key K

K

K→Dec(SK,C)

K
Registrar has set Final Exam Schedule

Wed, Mar 6 from 8-10 PM

(For BOTH sections)
The End