Signatures, Certificates, and Web

Spring 2024

David Kohlbrenner
dkohlbre@cs

Thanks to Franzi Roesner, Dan Boneh, Dieter Gollmann, Dan Halperin, David Kohlbrenner, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
Logistics

• Lab 1b grades will be delayed

• Lab 2 will go out later this week
  • We’ll adjust content or duration due to starting it later
Review: RSA Cryptosystem [Rivest, Shamir, Adleman 1977]

- **Key generation:**
  - Generate large primes $p$, $q$
    - Say, 2048 bits each (need primality testing, too)
  - Compute $n=pq$ and $\varphi(n)=(p-1)(q-1)$
  - Choose small $e$, relatively prime to $\varphi(n)$
    - Typically, $e=3$ or $e=2^{16}+1=65537$
  - Compute unique $d$ such that $ed \equiv 1 \mod \varphi(n)$
    - Modular inverse: $d \equiv e^{-1} \mod \varphi(n)$
    - Public key = $(e,n)$; private key = $(d,n)$

- **Encryption of $m$:** $c = m^e \mod n$
- **Decryption of $c$:** $c^d \mod n = (m^e)^d \mod n = m$
Actually, RSA is bad and stop using it

• Math is OK, implementation isn’t
  • Yes, all the implementations

• https://blog.trailofbits.com/2019/07/08/fuck-rsa/

• Sorry I just spent time teaching it to you
  • Maybe you would’ve preferred projected coordinate math on elliptic curves?
Digital Signatures: Basic Idea

Given: Everybody knows Bob’s public key
Only Bob knows the corresponding private key

Goal: Bob sends a “digitally signed” message
1. To compute a signature, must know the private key
2. To verify a signature, only the public key is needed
RSA Signatures

• Public key is \((n,e)\), private key is \((n,d)\)

• To sign message \(m\): 
  \[ s = m^d \mod n \]
  - Signing & decryption are same underlying operation in RSA
  - It’s infeasible to compute \(s\) on \(m\) if you don’t know \(d\)

• To verify signature \(s\) on message \(m\):
  verify that \(s^e \mod n = (m^d)^e \mod n = m\)
  - Just like encryption (for RSA primitive)
  - Anyone who knows \(n\) and \(e\) (public key) can verify signatures produced with \(d\) (private key)

• In practice, also need padding & hashing
  - Without padding and hashing: Consider multiplying two signatures together
  - Standard padding/hashing schemes exist for RSA signatures
DSS Signatures

• Digital Signature Standard (DSS)

• Public key: \((p, q, g, y=g^x \mod p)\), private key: \(x\)

• Each signing operation picks a new random value, to use during signing. Security breaks if two messages are signed with that same value.

• Security of DSS requires hardness of discrete log
  • If could solve discrete logarithm problem, would extract \(x\) (private key) from \(g^x \mod p\) (public key)
  • Again: We’ve discussed discrete logs modulo integers; significant advantages to using **elliptic curve groups** instead.
Post-Quantum

• If quantum computer become a reality
  • It becomes much more efficient to break conventional asymmetric encryption schemes (e.g., factoring becomes “easy”)

• There exists efforts to make quantum-resilient asymmetric encryption schemes
  • (Check out NIST’s PQC competition!)
Authenticity of Public Keys

Problem: How does Alice know that the public key they received is really Bob’s public key?
Threat: Person-in-the Middle
Distribution of Public Keys

• Public announcement or public directory
  • Risks: forgery and tampering

• Public-key certificate
  • Signed statement specifying the key and identity
    • $\text{sig}_{\text{CA}}(\text{“Bob”, } \text{PK}_B)$
    • Additional information often signed as well (e.g., expiration date)

• Common approach: certificate authority (CA)
  • Single agency responsible for certifying public keys
  • After generating a private/public key pair, user proves their identity and knowledge of the private key to obtain CA’s certificate for the public key (offline)
  • Every computer is pre-configured with CA’s public key
You encounter this every day...

SSL/TLS: Encryption & authentication for connections
SSL/TLS High Level

• SSL/TLS consists of **two** protocols
  • Familiar pattern for key exchange protocols

• Handshake protocol
  • Use public-key cryptography to establish a shared secret key between the client and the server

• Record protocol
  • Use the secret symmetric key established in the handshake protocol to protect communication between the client and the server
This certificate is intended for the following purpose(s):
- All issuance policies

Issued to:  UW Services CA

Issued by:  UW Services CA

Valid from  2/25/2003  to  9/3/2030

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

Issued to:  *.cs.washington.edu

Issued by:  InCommon RSA Server CA

Valid from  3/19/2020  to  3/20/2022
Hierarchical Approach

• Single CA certifying every public key is impractical
• Instead, use a trusted root authority (e.g., Verisign)
  • Everybody must know the root’s public key
  • Instead of single cert, use a certificate chain
    • $\text{sig}_{\text{Verisign}}(\text{"AnotherCA"}, \text{PK}_{\text{AnotherCA}})$,
      $\text{sig}_{\text{AnotherCA}}(\text{“Alice”, PK}_A)$
  • Not shown in figure but important:
    • Signed as part of each cert is whether party is a CA or not

• What happens if root authority is ever compromised?
Trusted(?) Certificate Authorities
Turtles All The Way Down...

The saying holds that the world is supported by a chain of increasingly large turtles. Beneath each turtle is yet another: it is "turtles all the way down".

[Image from Wikipedia]
Corporate CAs? -- Gradescope

• Many corporations require that all company machines have an additional **Root Certificate** installed, owned and controlled by the company IT.

• This would allow the company to create a certificate for any website, service, etc. they want and have it trusted by any company machine. (But not by anyone else’s).

• What does this let corporate IT do?
• Why might they want to do that?
Many Challenges...

• Hash collisions
• Weak security at CAs
  • Allows attackers to issue rogue certificates
• Users don’t notice when attacks happen
  • We’ll talk more about this later in the course
• How do you revoke certificates?
Attacking CAs

Security of DigiNotar servers:

• All core certificate servers controlled by a single admin password (Pr0d@dm1n)
• Software on public-facing servers out of date, unpatched
• No anti-virus (could have detected attack)

DigiNotar is a Dutch Certificate Authority. They sell SSL certificates.

Somehow, somebody managed to get a rogue SSL certificate from them on July 10th, 2011. This certificate was issued for domain name .google.com.

What can you do with such a certificate? Well, you can impersonate Google — assuming you can first reroute Internet traffic for google.com to you. This is something that can be done by a government or by a rogue ISP. Such a reroute would only affect users within that country or under that ISP.
More Rogue Certs

• In Jan 2013, a rogue *.google.com certificate was issued by an intermediate CA that gained its authority from the Turkish root CA TurkTrust
  • TurkTrust accidentally issued intermediate CA certs to customers who requested regular certificates
  • Ankara transit authority used its certificate to issue a fake *.google.com certificate in order to filter SSL traffic from its network

• This rogue *.google.com certificate was trusted by every browser in the world
Bad CAs

• **DarkMatter** ([https://groups.google.com/g/mozilla.dev.security.policy/c/nnlVNfqgz7g/m/TseYqDzaDAAJ](https://groups.google.com/g/mozilla.dev.security.policy/c/nnlVNfqgz7g/m/TseYqDzaDAAJ) and [https://bugzilla.mozilla.org/show_bug.cgi?id=1427262](https://bugzilla.mozilla.org/show_bug.cgi?id=1427262))
  • Security company wanted to get CA status
  • Questionable practices

  • Major company, regular participant in standards
  • Poor practices, mismanagement 2013-2017
  • CA distrusted in Oct 2018

• Recall: Turtles all the way down. How can we trust the CAs? What happens if we can’t?
Certificate Revocation

• Revocation is very important
• Many valid reasons to revoke a certificate
  • Private key corresponding to the certified public key has been compromised
  • User stopped paying their certification fee to this CA and CA no longer wishes to certify them
  • CA’s private key has been compromised!
•Expiration is a form of revocation, too
  • Many deployed systems don’t bother with revocation
  • Re-issuance of certificates is a big revenue source for certificate authorities
Certificate Revocation Mechanisms

• Certificate revocation list (CRL)
  • CA periodically issues a signed list of revoked certificates
    • Credit card companies used to issue thick books of canceled credit card numbers
    • Can issue a “delta CRL” containing only updates

• Online revocation service
  • When a certificate is presented, recipient goes to a special online service to verify whether it is still valid
    • Like a merchant dialing up the credit card processor
Attempt to Fix CA Problems:

Certificate Transparency

- **Problem:** browsers will think nothing is wrong with a rogue certificate until revoked
- **Goal:** make it impossible for a CA to issue a bad certificate for a domain *without the owner of that domain knowing*
- **Approach:** auditable certificate logs
  - Certificates published in public logs
  - Public logs checked for unexpected certificates

[www.certificate-transparency.org](http://www.certificate-transparency.org)
Next Major Topic!
Web+Browser Security
Big Picture: Browser and Network

Browser

OS

Hardware

request

reply

website

Network
Where Does the Attacker Live?

Mitigation: SSL/TLS (not covered further)

Mitigation: Browser security model + web app security (this/next week)
Two Sides of Web Security

(1) Web browser
   • Responsible for securely confining content presented by visited websites

(2) Web applications
   • Online merchants, banks, blogs, Google Apps ...
   • Mix of server-side and client-side code
     • Server-side code written in PHP, JavaScript, C++ etc.
     • Client-side code written in JavaScript (... sort of)
   • Many potential bugs: XSS, XSRF, SQL injection
But at least 3 actors!
Browser: All of These Should Be Safe

- Safe to visit an evil website

- Safe to visit two pages
  - Simultaneously
  - Sequentially

- Safe delegation
Browser Security Model

Goal 1: Protect local system from web attacker
→ Browser Sandbox

Goal 2: Protect/isolate web content from other web content
→ Same Origin Policy
Browser Sandbox

Goals: Protect local system from web attacker; protect websites from each other

• E.g., safely execute JavaScript provided by a website
• No direct file access, limited access to OS, network, browser data, content from other websites
• Tabs and iframes in their own processes
• Implementation is browser and OS specific*

*For example, see: https://chromium.googlesource.com/chromium/src/+/master/docs/design/sandbox.md

From Chrome Bug Bounty Program

<table>
<thead>
<tr>
<th>High-quality report with functional exploit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandbox escape / Memory corruption in a non-sandboxed process</td>
</tr>
</tbody>
</table>
## Same Origin Policy

**Goal:** Protect/isolate web content from other web content

Website origin = (scheme, domain, port)

<table>
<thead>
<tr>
<th>Compared URL</th>
<th>Outcome</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>http://www.example.com/dir/page.html</code></td>
<td>Success</td>
<td>Same protocol and host</td>
</tr>
<tr>
<td><code>http://www.example.com/dir2/other.html</code></td>
<td>Success</td>
<td>Same protocol and host</td>
</tr>
<tr>
<td><code>http://www.example.com:81/dir/other.html</code></td>
<td>Failure</td>
<td>Same protocol and host but different port</td>
</tr>
<tr>
<td><code>https://www.example.com/dir/other.html</code></td>
<td>Failure</td>
<td>Different protocol</td>
</tr>
<tr>
<td><code>http://en.example.com/dir/other.html</code></td>
<td>Failure</td>
<td>Different host</td>
</tr>
<tr>
<td><code>http://example.com/dir/other.html</code></td>
<td>Failure</td>
<td>Different host (exact match required)</td>
</tr>
<tr>
<td><code>http://v2.www.example.com/dir/other.html</code></td>
<td>Failure</td>
<td>Different host (exact match required)</td>
</tr>
</tbody>
</table>

[Example from Wikipedia]
Same Origin Policy is Subtle!

• Browsers didn’t always get it right...
  • In 2023 we’re pretty good though

• Lots of cases to worry about it:
  • DOM / HTML Elements
  • Navigation
  • Cookie Reading
  • Cookie Writing
  • Iframes vs. Scripts
HTML + DOM + JavaScript

This is the title

This is a sample page.

alert(“Hello world”);

<iframe src=“http://example.com”> </iframe>
Same-Origin Policy: DOM

Only code from same origin can access HTML elements on another site (or in an iframe).

- **www.bank.com** (the parent) can access HTML elements in the iframe (and vice versa).
- **www.evil.com** (the parent) cannot access HTML elements in the iframe (and vice versa).

```html
<html>
<body>
<iframe src="http://www.bank.com/iframe.html">
</iframe>
</body>
</html>
```
Browser Cookies

- HTTP is stateless protocol
- **Browser cookies are used to introduce state**
  - Websites can store small amount of info in browser
  - Used for authentication, personalization, tracking...
  - Cookies are often secrets

```
POST login.php
username and pwd

HTTP Header: Set-cookie:
  login_token=13579;
  domain = (who can read);
  expires = (when expires)

GET restricted.html

Cookie: login_token=13579
```
Same Origin Policy: Cookie Writing

Which cookies can be set by login.site.com?

- **allowed domains**
  - ✓ login.site.com
  - ✓ .site.com

- **disallowed domains**
  - ✗ othersite.com
  - ✗ .com
  - ✗ user.site.com

login.site.com can set cookies for all of .site.com (domain suffix), but not for another site or top-level domain (TLD)