

# CSE508      Network Security

10/12/2016    **TLS**

Michalis Polychronakis  
*Stony Brook University*

# **TLS** (Transport Layer Security)

Predecessor: **SSL** (Secure Socket Layer)

Most widely used protocol for encrypted data transmission

Same basic design, different crypto algorithms

Designed to provide secure communication over the insecure internet

Authentication, confidentiality, and integrity

Used in many services and secure versions of protocols

HTTP, POP, IMAP, SMTP, OpenVPN, CalDAV, CardDAV, LDAP, NNTP, FTP, IRC, SIP, ...

Separate port number: HTTPS: 443, FTPS: 990, IMAPS: 993, ...

# History



## SSL developed at Netscape

v1: never released

v2 (1994): serious weaknesses

v3 (1995): re-design, basis of what we use today

## TLS working group was formed to migrate SSL to IETF

TLS 1.0 (1999): minor differences but incompatible with SSL 3  
(different crypto algorithms)

TLS 1.1 (2006): mostly security fixes, TLS extensions

TLS 1.2 (2008): authenticated encryption, more flexible

TLS 1.3 (WIP): removal of legacy/weak algorithms, lower latency

## Endless cycle of vulnerabilities and improvements

Insecure renegotiation, RC4 weaknesses, compression side channels, padding oracle attacks, buggy implementations, PKI attacks, ...

Lately with fancy names, too: *BEAST*, *CRIME*, *TIME*, *Lucky 13*, *BREACH*, *POODLE*, *FREAK*, *Heartbleed*, *DROWN*, ...

## **Handshake protocol**

Negotiate session keys and crypto algorithms to be used

Authentication (server and optionally client)

Takes 6–10 messages, depending on features used

## **Record Protocol**

*Message transport:* [header | data] records (16K)

*Encryption and integrity:* after handshake completion

*Compression:* before encryption... not a good idea

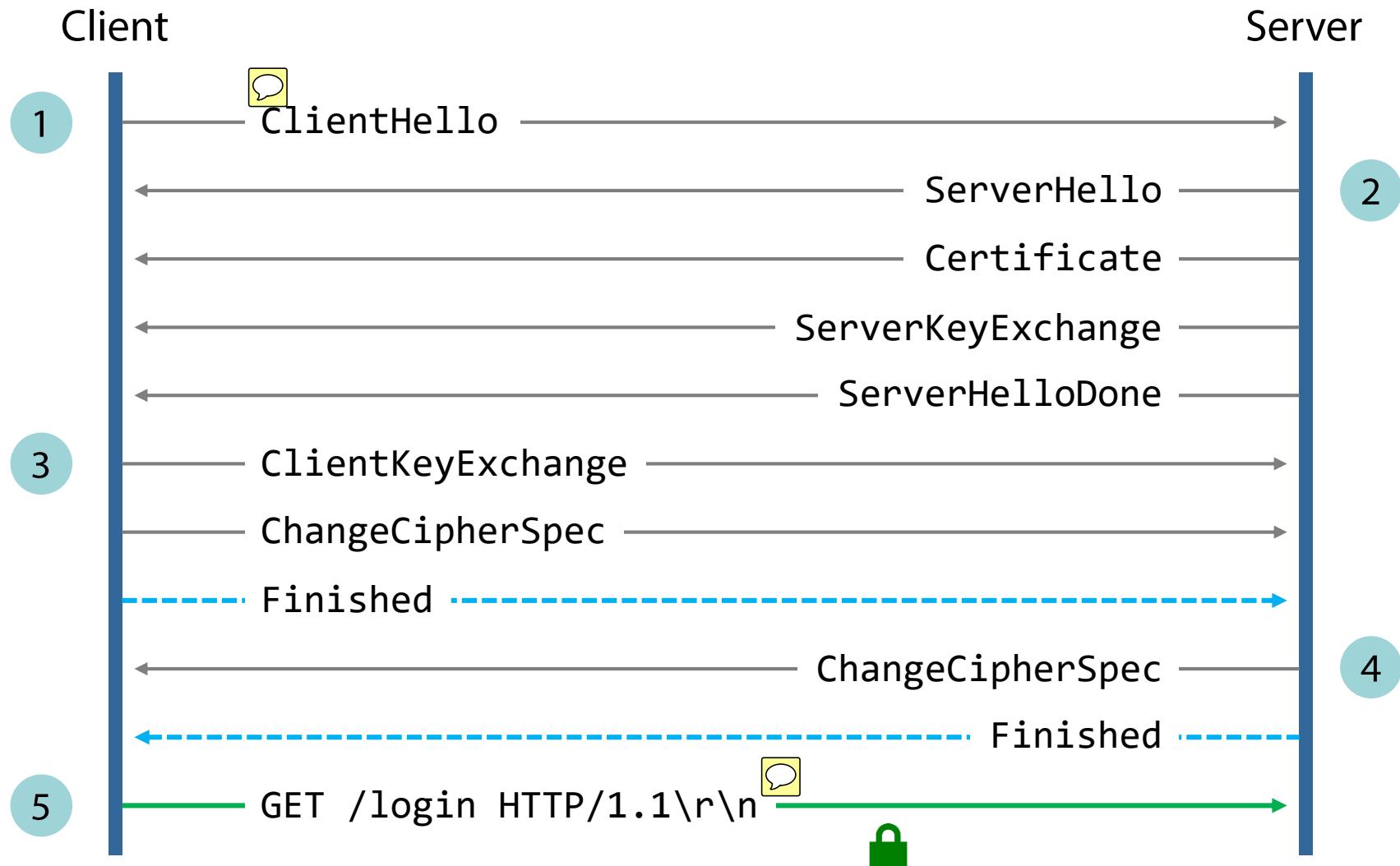
Side-channel attacks (e.g., CRIME) 

*Subprotocols:* allow for extensibility

TLS defines four core subprotocols:

*handshake, change cipher spec, application data, alert*

# TLS 1.2 Handshake (Ephemeral DH)



# Cipher Suite Negotiation

ClientHello: *here are the cipher suites I support*

```
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256  
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256  
TLS_RSA_WITH_AES_128_GCM_SHA256  
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA  
TLS_DHE_RSA_WITH_AES_128_CBC_SHA  
TLS_RSA_WITH_AES_128_CBC_SHA  
TLS_RSA_WITH_3DES_EDE_CBC_SHA  
TLS_RSA_WITH_RC4_128_SHA  
...
```

ServerHello: *let's use this one* 

```
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
```

The server might not support the best of the client's suites

Offers some other version hoping that the client will accept it

# Downgrade Attacks

Force a weaker cipher suite selection through MitM

SSL 2: no handshake integrity



SSL 3: protocol rollback protection (still breakable)

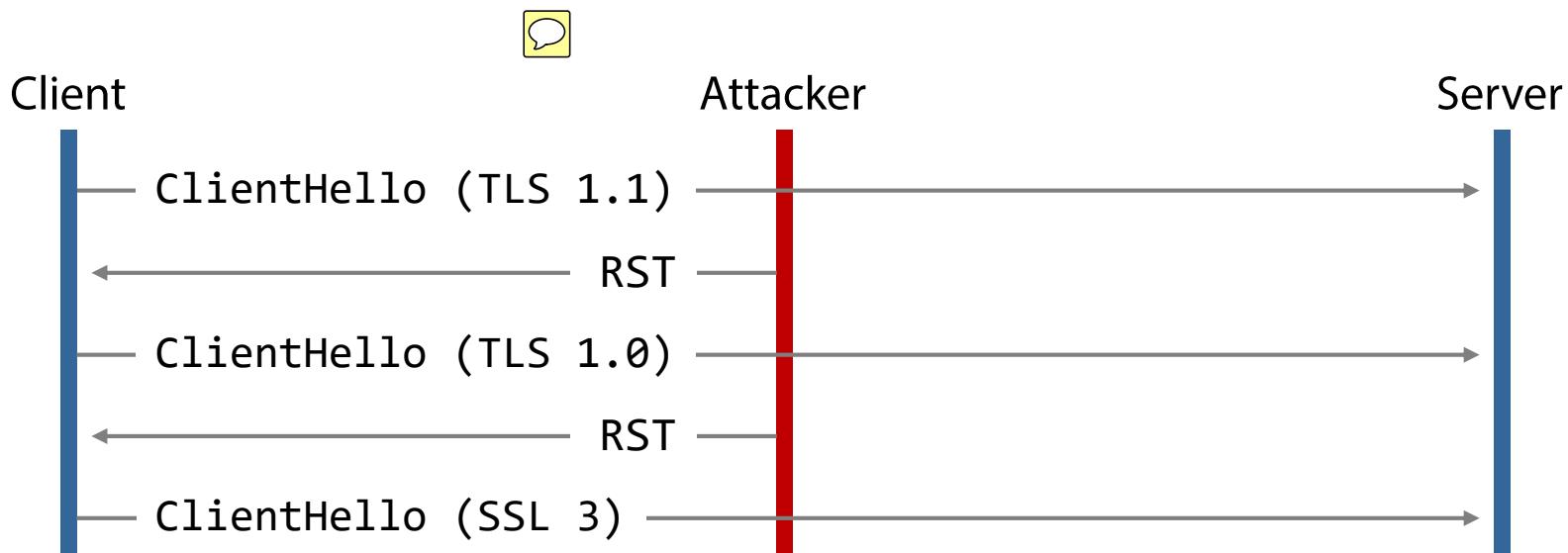
TLS 1.0 and on: additional protections

Due to server bugs and interoperability issues, browsers responded by voluntarily downgrading the protocol upon handshake failure

Retrying connection with lower SSL/TLS version

Attackers can exploit this by blocking the initial handshake attempts

Or modify the client's list of supported clients (and include only weak ones)



# SSL 3.0 is now completely removed by most browsers

Browser	Version	Platforms	TLS/SSL support history of web browsers										Protocol selection by user [n 2]					
			SSL protocols		TLS protocols			Certificate Support			Vulnerabilities fixed[n 1]							
			SSL 2.0 (insecure)	SSL 3.0 (insecure)	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3 (proposed)	EV [n 3][48]	SHA-2 [49]	ECDSA [50]	BEAST [n 4]	CRIME [n 5]	POODLE (SSLv3) [n 6]	RC4 [n 7]	FREAK [51][52]	Logjam	
Google Chrome <small>(Chrome for Android [n 8] [n 9])</small>	1–9	Windows (7+) OS X (10.4+) Linux Android (4.1+) iOS (9.0+) Chrome OS	Disabled by default	Enabled by default	Yes	No	No	No	Yes (only desktop)	needs SHA-2 compatible OS [49]	needs ECC compatible OS [50]	Not affected [57]	Vulnerable (HTTPS)	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Yes[n 10]
	10–20		No [58]	Enabled by default	Yes	No	No	No	Yes (only desktop)	needs SHA-2 compatible OS [49]	needs ECC compatible OS [50]	Not affected	Vulnerable (HTTPS/SPDY)	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Yes[n 10]
	21		No	Enabled by default	Yes	No	No	No	Yes (only desktop)	needs SHA-2 compatible OS [49]	needs ECC compatible OS [50]	Not affected	Mitigated [59]	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Yes[n 10]
	22–25		No	Enabled by default	Yes	Yes [50]	No [50][51][52][53]	No	Yes (only desktop)	needs SHA-2 compatible OS [49]	needs ECC compatible OS [50]	Not affected	Mitigated	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	26–29		No	Enabled by default	Yes	Yes	No	No	Yes (only desktop)	needs SHA-2 compatible OS [49]	needs ECC compatible OS [50]	Not affected	Mitigated	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	30–32		No	Enabled by default	Yes	Yes	Yes [51][52][53]	No	Yes (only desktop)	needs SHA-2 compatible OS [49]	needs ECC compatible OS [50]	Not affected	Mitigated	Vulnerable	Vulnerable	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	33–37		No	Enabled by default	Yes	Yes	Yes	No	Yes (only desktop)	needs SHA-2 compatible OS [49]	needs ECC compatible OS [50]	Not affected	Mitigated	Partly mitigated [n 12]	Lowest priority [56][67][68]	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	38, 39		No	Enabled by default	Yes	Yes	Yes	No	Yes (only desktop)	needs ECC compatible OS [50]	Needs ECC compatible OS [50]	Not affected	Mitigated	Partly mitigated	Lowest priority	Vulnerable (except Windows)	Vulnerable	Temporary [n 11]
	40		No	Disabled by default [65][69]	Yes	Yes	Yes	No	Yes (only desktop)	Needs ECC compatible OS [50]	Needs ECC compatible OS [50]	Not affected	Mitigated	Mitigated [n 13]	Lowest priority	Vulnerable (except Windows)	Vulnerable	Yes[n 14]
	41, 42		No	Disabled by default	Yes	Yes	Yes	No	Yes (only desktop)	Needs ECC compatible OS [50]	Needs ECC compatible OS [50]	Not affected	Mitigated	Mitigated	Lowest priority	Mitigated	Vulnerable	Yes[n 14]
	43		No	Disabled by default	Yes	Yes	Yes	No	Yes (only desktop)	Needs ECC compatible OS [50]	Needs ECC compatible OS [50]	Not affected	Mitigated	Mitigated	Only as fallback [n 15][70]	Mitigated	Vulnerable	Yes[n 14]
	44–47		No	No [71]	Yes	Yes	Yes	No	Yes (only desktop)	Needs ECC compatible OS [50]	Needs ECC compatible OS [50]	Not affected	Mitigated	Not affected	Only as fallback [n 15]	Mitigated	Mitigated [72]	Temporary [n 11]
	48, 49		No	No	Yes	Yes	Yes	No	Yes (only desktop)	Needs ECC compatible OS [50]	Needs ECC compatible OS [50]	Not affected	Mitigated	Not affected	Disabled by default [n 16][73][74]	Mitigated	Mitigated	Temporary [n 11]
	50–53		No	No	Yes	Yes	Yes	No	Yes (only desktop)	Needs ECC compatible OS [50]	Needs ECC compatible OS [50]	Not affected	Mitigated	Not affected	Disabled by default [n 16][73][74]	Mitigated	Mitigated	Temporary [n 11]
54–60	61		No	No	Yes	Yes	Yes	Disabled by default (Experimental)	Yes (only desktop)	Yes	Yes	Not affected	Mitigated	Not affected	Disabled by default [n 16][73][74]	Mitigated	Mitigated	Temporary [n 11]

## TLS 1.2 Session Resumption



Full handshake: 6-10 messages and two network round-trips

Along with CPU-intensive crypto operations, cert validation, ...



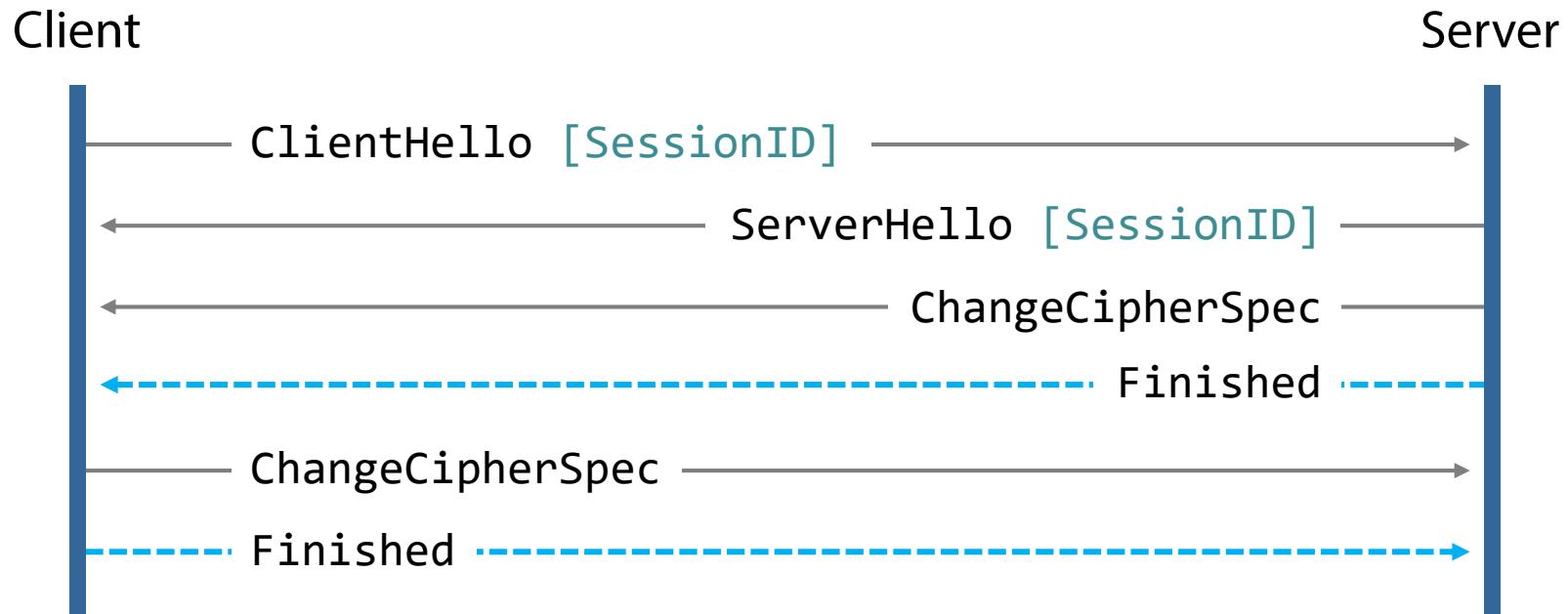
Avoid re-negotiation by remembering security parameters

Server assigns and sends a unique session ID as part of ServerHello

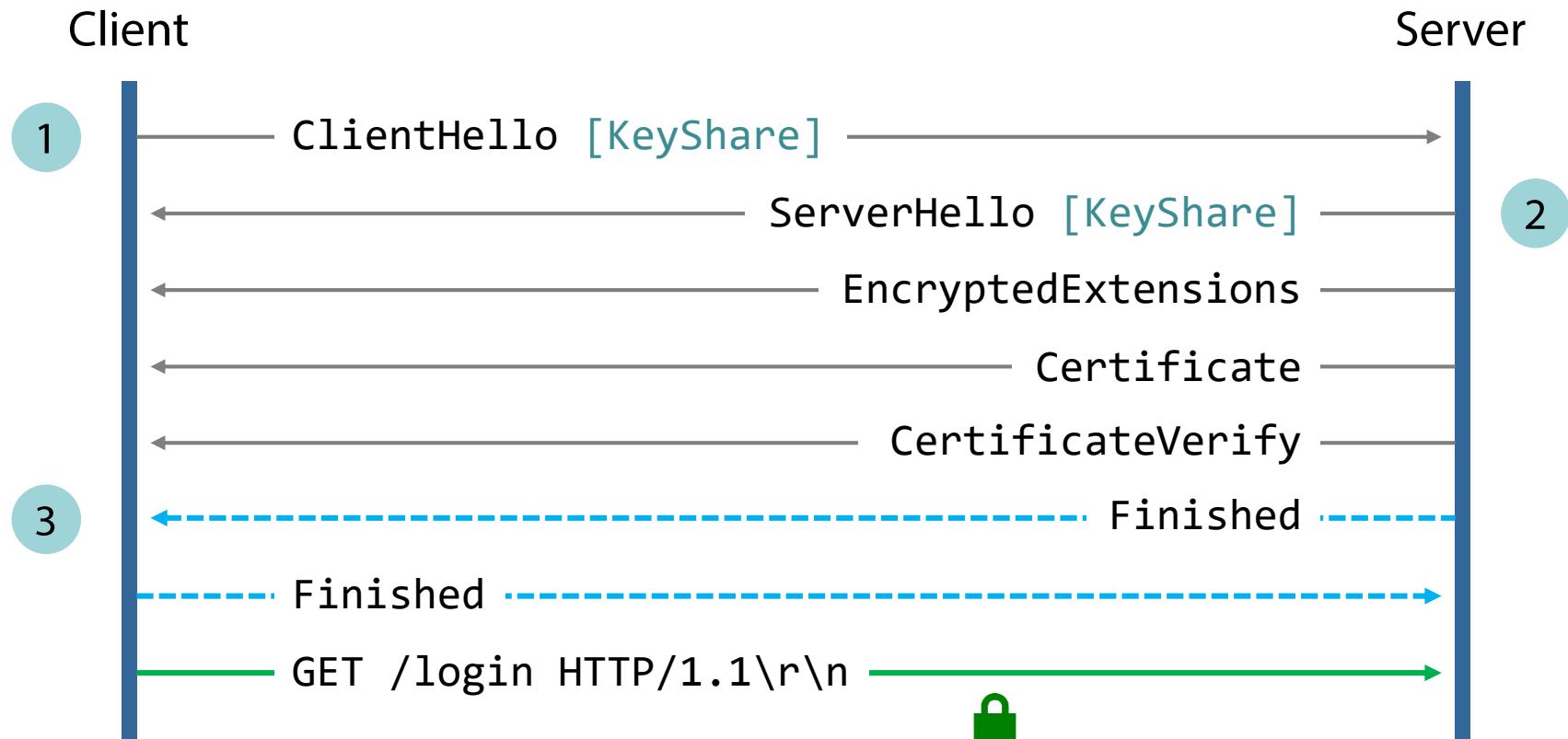
In future connections, the client sends the session ID to resume the session



Alternative: *session tickets* (all state is kept at client)



# TLS 1.3 Handshake (Ephemeral DH)



Latest draft supports even zero-RTT handshakes

Clients include encrypted data in the initial handshake messages  
Based on configuration identifier previously sent by the server

# **Server (and Client) Authentication**

After handshake completion, the client knows it can “trust” the information in the server’s certificate

Assuming it trusts the issuing CA

SSL/TLS certs are based on the X.509 PKI standard

How is the certificate associated with the server?

*Common Name (CN): server’s hostname*

Same process is supported for authenticating clients

Highly-secure web services, some VPN services, ...

Not used often in practice

Common alternative: username + password over TLS connection

## Certificate Fields

**Version:** v1 (basic), v2 (additional fields), v3 (extensions)

**Serial Number:** high-entropy integer

**Signature Algorithm:** encryption and hash algorithm used to sign the certificate

**Issuer:** complex field that contains the *distinguished name (DN)* of the certificate issuer

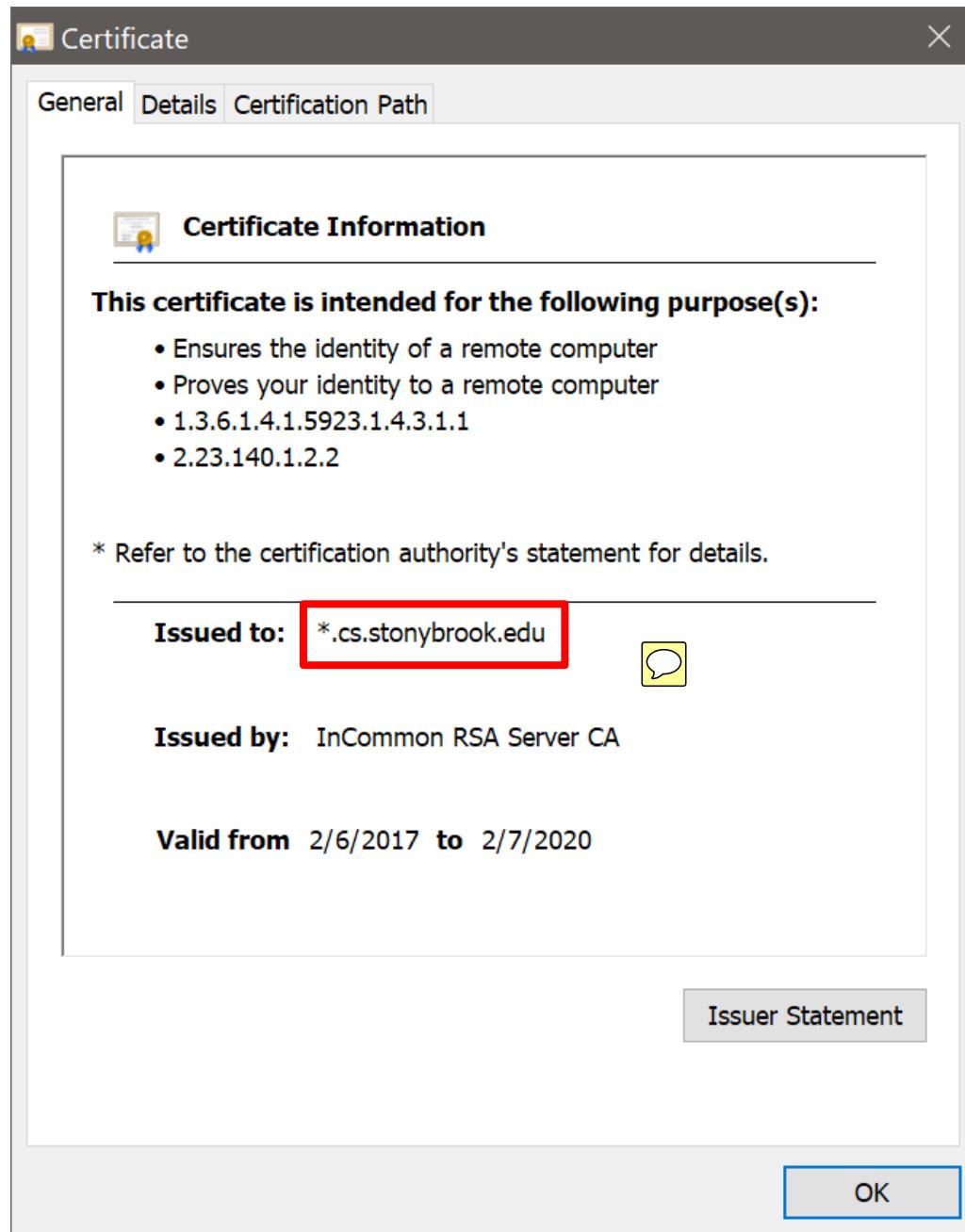
**Validity:** starting and ending date of validity period

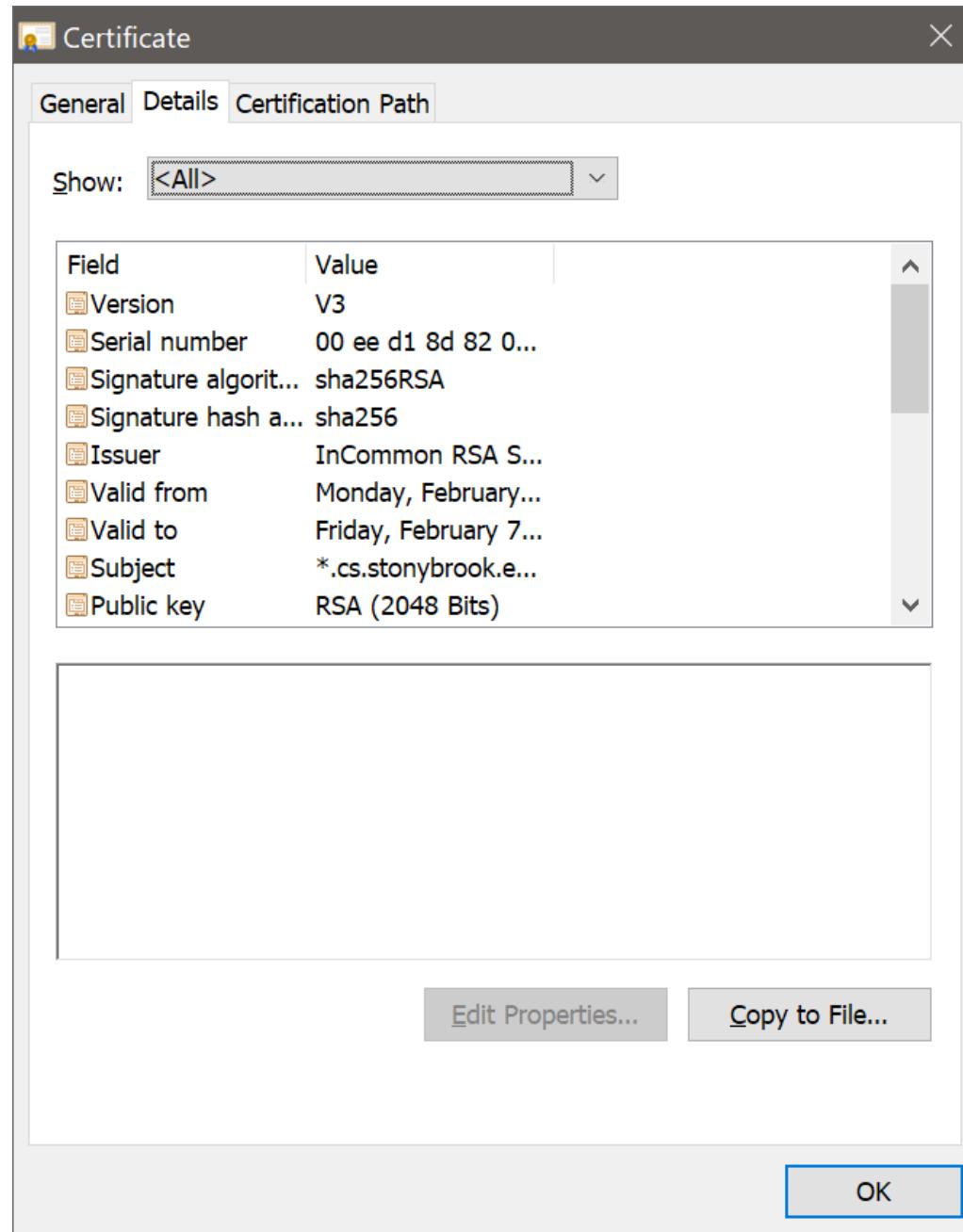
**Subject:** DN of the entity associated with the public key for which the certificate is issued

Deprecated in favor of the Subject Alternative Name (SAN) extension:  
DNS name, IP address, or URI (also supports binding to multiple identities)

**Public Key:** The subject's public key

**Signature**





# Certificate Chains

Trust anchors: systems are pre-configured with ~200 trusted root certificates 

System/public store: used by OS, browsers, ...

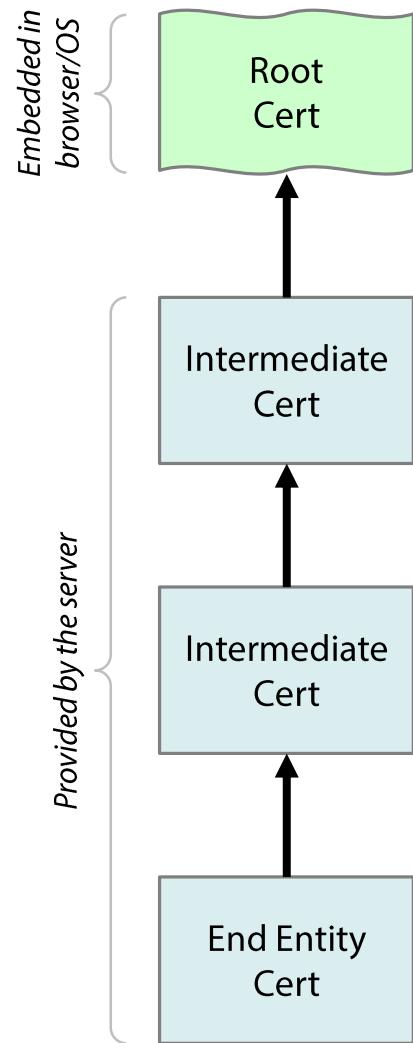
More can be added in the local/private cert store: vendor-specific certs, MitM certs for content inspection filters/AVs, ...

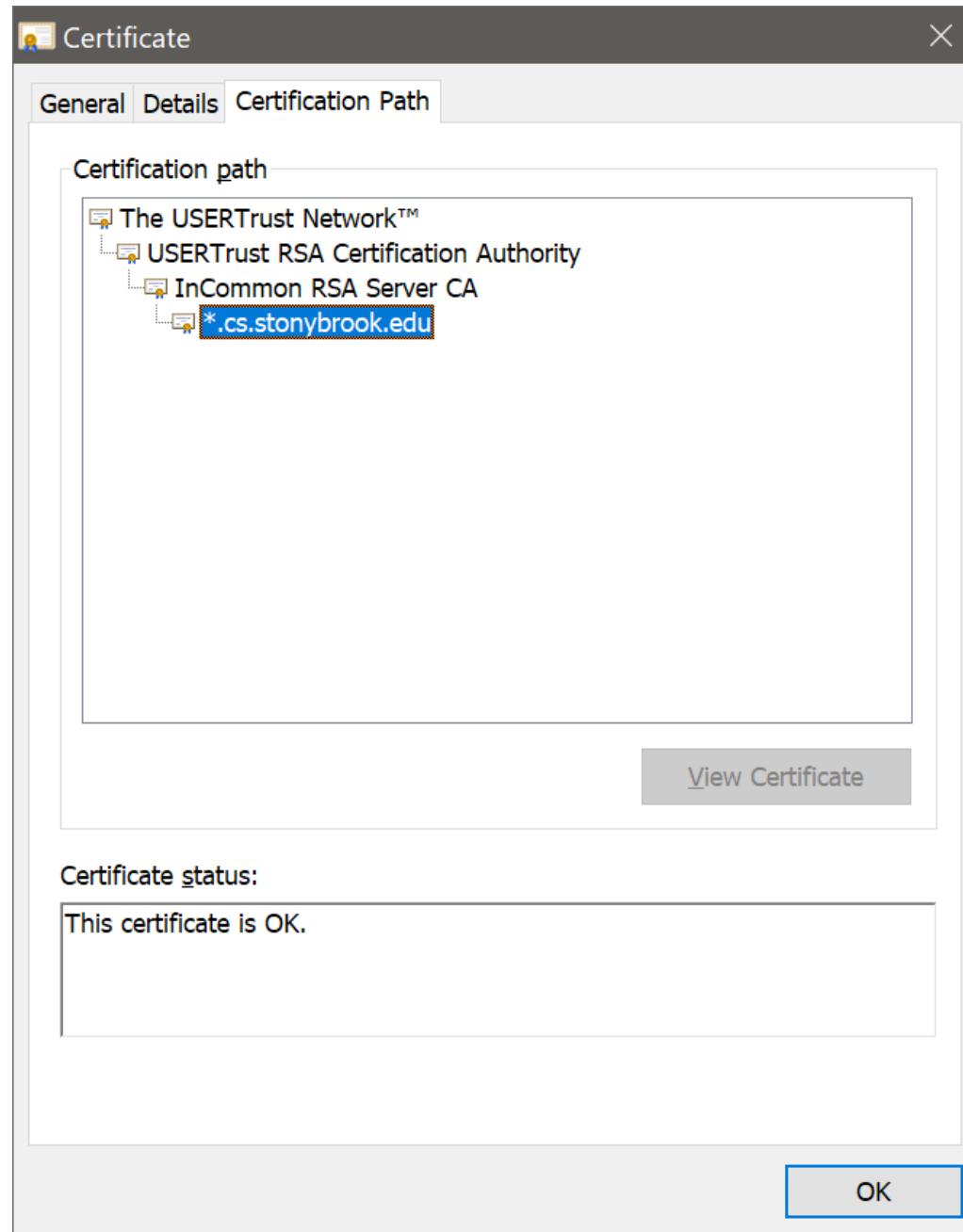
Server provides a *chain of certificates*

A certificate from an intermediate CA is trusted if there is a valid chain of trust all the way back to a trusted root CA

Any certificate authority can issue and sign certificates for any subject

The system is only as secure as the weakest certificate authority... 





# HTTPS

Most common use of SSL/TLS

More than half of web traffic is now encrypted

Crypto is expensive, needs more CPU cycles

**Not** a big deal these days (native hardware support)

Mixed content: Ad networks, mashups, ...

**Stop** using them! ...easier said than done (lost revenue, increased development time) 

Incentives: Google rewards HTTPS sites  with higher ranking

Virtual Hosting: initially incompatible

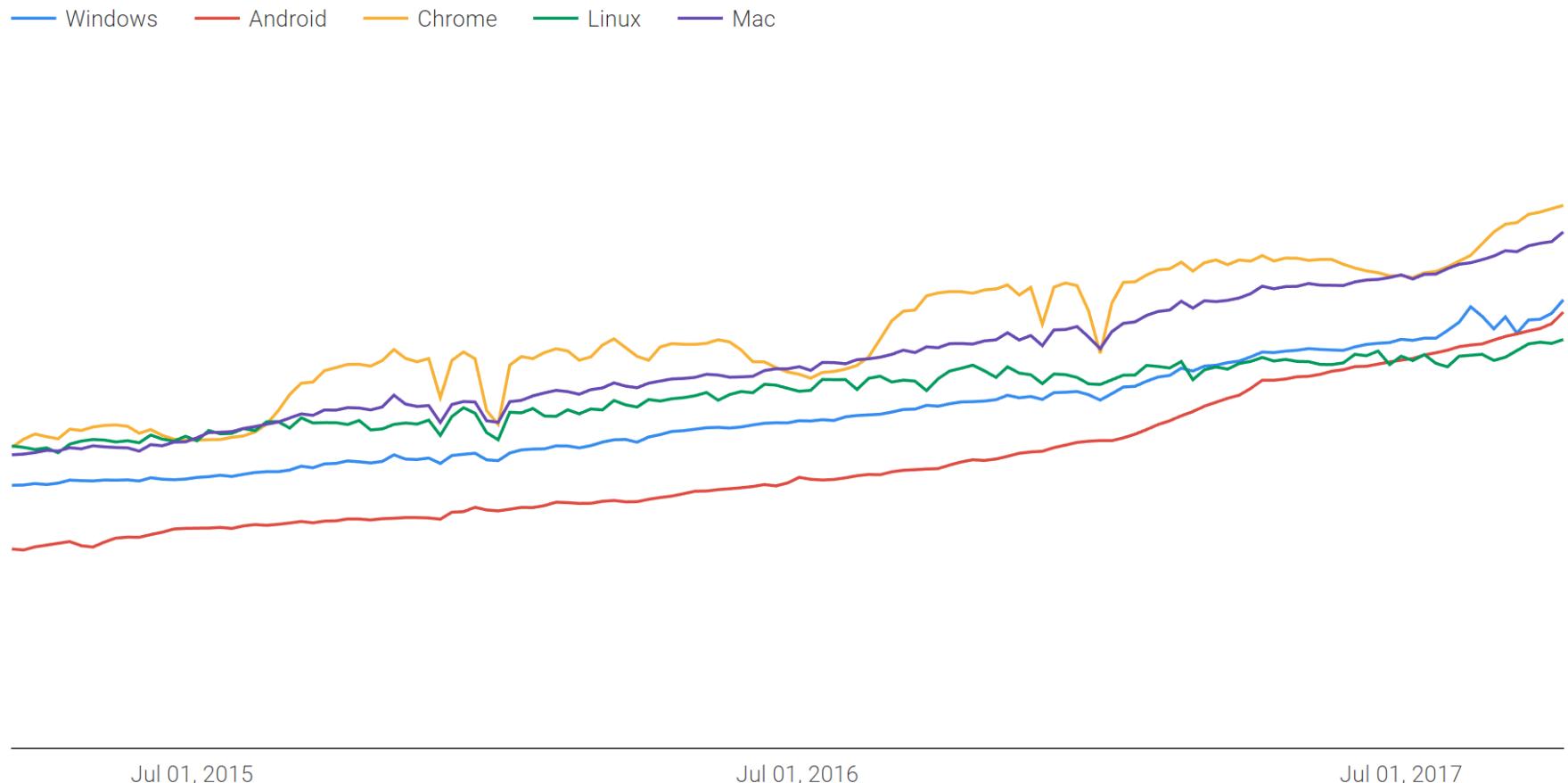
**Not** anymore: solved as of TLS 1.1 through the *Server Name Indication (SNI)* extension → what about IE6/WinXP users?

Needs expertise and certs cost \$\$\$\$

**Not** anymore: letsencrypt.org 



# HTTPS Encryption in Chrome



# Browser Security Indicators

Convey information about the security of a page

Locks, shields, keys, green bars...

*"This page was fetched using SSL"*



Page content was not viewed or altered by a network adversary

Certificate is valid (e.g. not expired), issued by a CA trusted by the browser, and the subject name matches the URL's domain

*"This page uses an invalid certificate"* A small icon consisting of a red triangle symbol followed by the text "Not secure" and then "https://" in a red color.

*"Parts of the page are not encrypted"* A small icon consisting of a grey circle with an 'i' symbol inside, followed by "https://" and a yellow speech bubble icon.

*"The legal entity operating this web site is known"*

Extended Validation (EV) certificates





# very.badssl.com



Mixed Form

This form submits to HTTP.

Submit



This page is trying to load scripts from unauthenticated sources.

[Load unsafe scripts](#)

[Learn more](#)

Done

# very.badss

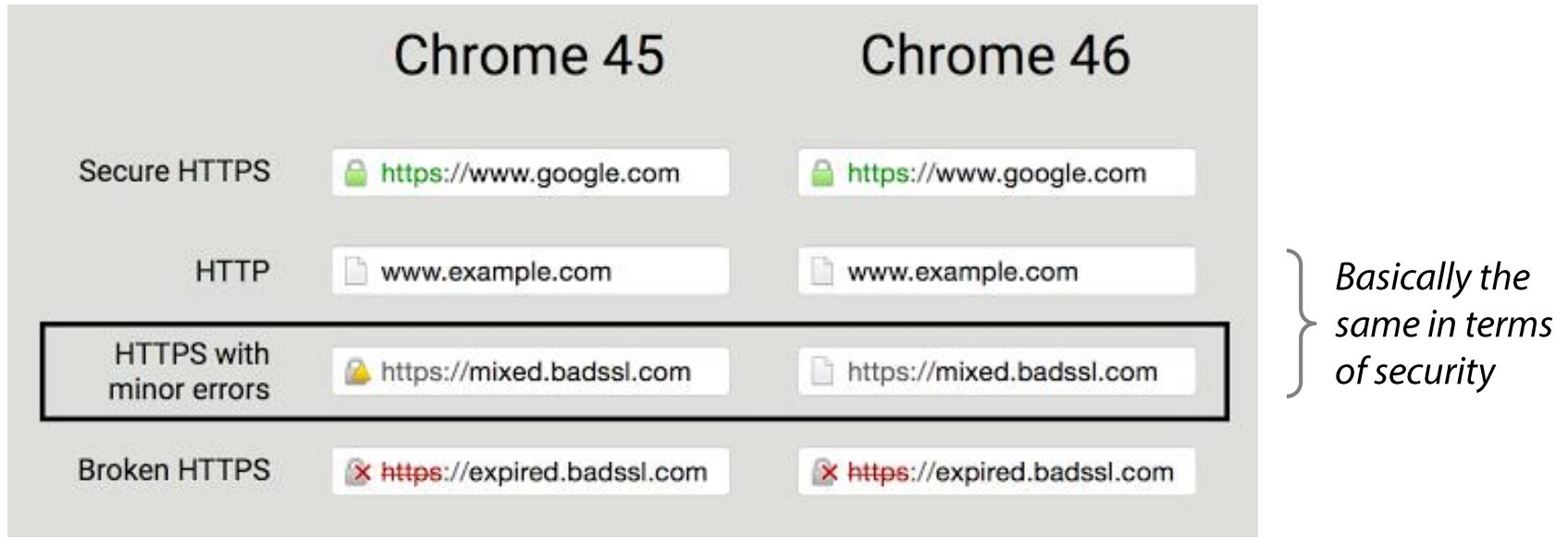


Mixed Form

This form submits to HTTP.

Submit

# Mixed Content Warning



Fewer security states for users to remember

Reflects better the security state of the page

**Non-HTTPS traffic is a vulnerability!**

MitM/MotS attacks on the HTTP part are trivial...

# Marking HTTP as Non-Secure

Phase 1: page is marked “Not Secure” when

The page *contains a password field*

The user *interacts with a credit card field*

Treatment of HTTP pages with  
password or credit card form fields:

Current (Chrome 53)

ⓘ login.example.com

Jan. 2017 (Chrome 56)

ⓘ Not secure | login.example.com

# Marking HTTP as Non-Secure

Phase 2: page is marked “Not Secure” when

The page *contains a password field*

The user *interacts with any input field*

The user is browsing in Chrome *incognito mode*

	Treatment of HTTP pages outside Incognito mode:	Treatment of HTTP pages in Chrome Incognito mode:
Current (Chrome 58)	 example.com	 Not secure   example.com
Oct. 2017 (Chrome 62) at page load	 example.com	 Not secure   example.com
Oct. 2017 (Chrome 62) when entering data	 Not secure   example.com	 Not secure   example.com

# Marking HTTP as Non-Secure

In the future: all HTTP pages are marked “Not Secure”

Eventual treatment of all  
HTTP pages in Chrome:



Not secure

example.com



# SSL stripping

Browsing sessions often start with a plain HTTP page

Web sites switch to HTTPS only for login or checkout

Example: Facebook in 2010 (optional full HTTPS in 2011, on by default in 2013)

Users type addresses without specifying “https://”

Browser connects over HTTP → site may redirect to HTTPS

## SSLstrip [Moxie Marlinspike, Black Hat DC 2009]

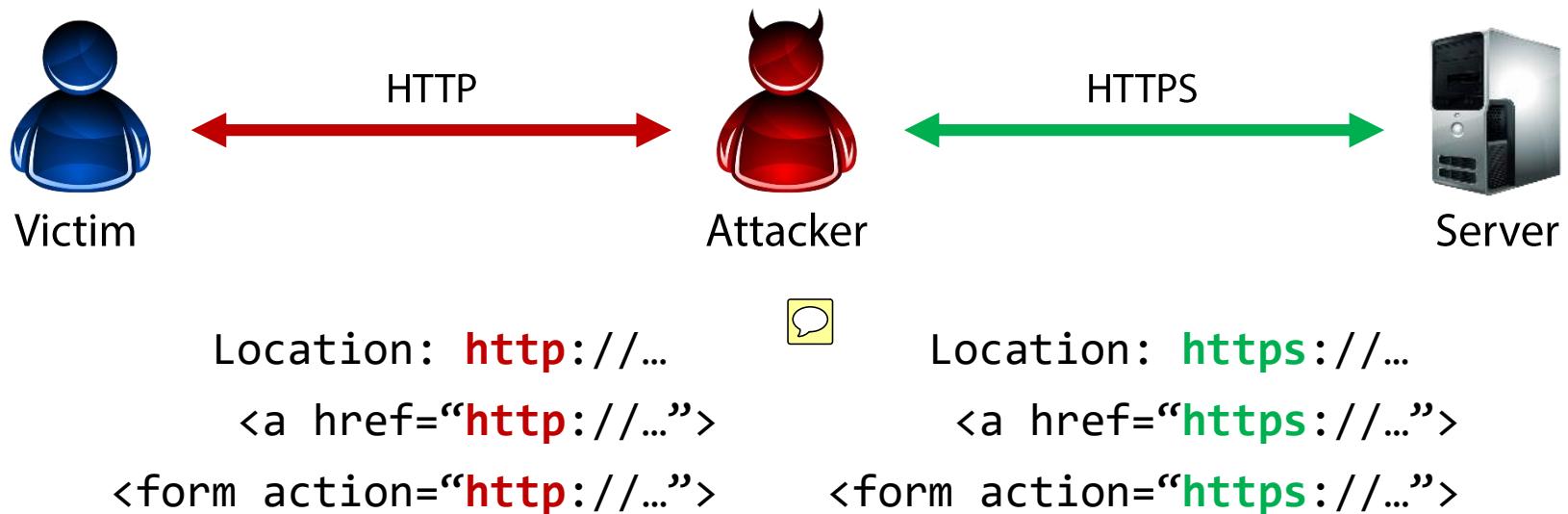
MitM attack to prevent redirection to HTTPS

Watch for **HTTPS** redirects and links, and map them to **HTTP** links

...or homograph-similar *valid* HTTPS links:

<https://www.bank.com.attacker.com>

# SSL stripping



*Missing lock icon, but who is going to notice?*

# HSTS (HTTP Strict Transport Security)

## Defense against SSL stripping and other issues

Convert any insecure links (`http://`) into secure links (`https://`)  
*before* accessing a resource

Treat all errors (e.g., invalid certificate, mixed content) as fatal:  
do not allow users to access the web application

A server implements an HSTS policy by supplying an extra HTTP header

**Strict-Transport-Security: max-age=31536000**

*“Use only HTTPS for future requests to this domain for the next year”*

An instance of *trust on first use (TOFU)*

The initial request *remains unprotected* if sent over HTTP

HSTS preloading: browser comes with a list of known HSTS sites

# HSTS Preloading



The screenshot shows a browser window displaying the file `transport_security_state_static.json` from the Chromium source code. The file content is a JSON array of objects, each representing a domain and its HSTS configuration. The first object in the array is highlighted with a red box:

```
// Force HTTPS for sites that have requested it.
[{"name": "www.paypal.com", "mode": "force-https"}, {"name": "paypal.com", "mode": "force-https"}, {"name": "www.elanex.biz", "mode": "force-https"}, {"name": "jottit.com", "include_subdomains": true, "mode": "force-https"}, {"name": "sunshinepress.org", "include_subdomains": true, "mode": "force-https"}, {"name": "www.noisebridge.net", "mode": "force-https"}, {"name": "neg9.org", "mode": "force-https"}, {"name": "riseup.net", "include_subdomains": true, "mode": "force-https"}, {"name": "factor.cc", "mode": "force-https"}, {"name": "members.mayfirst.org", "include_subdomains": true, "mode": "force-https"}, {"name": "support.mayfirst.org", "include_subdomains": true, "mode": "force-https"}, {"name": "id.mayfirst.org", "include_subdomains": true, "mode": "force-https"}, {"name": "lists.mayfirst.org", "include_subdomains": true, "mode": "force-https"}, {"name": "webmail.mayfirst.org", "include_subdomains": true, "mode": "force-https"}, {"name": "roundcube.mayfirst.org", "include_subdomains": true, "mode": "force-https"}, {"name": "aladdinschools.appspot.com", "mode": "force-https"}, {"name": "ottospora.nl", "include_subdomains": true, "mode": "force-https"}, {"name": "www.paycheckrecords.com", "mode": "force-https"}, {"name": "lastpass.com", "mode": "force-https"}, {"name": "www.lastpass.com", "mode": "force-https"}, {"name": "keyerror.com", "include_subdomains": true, "mode": "force-https"}, {"name": "entropia.de", "mode": "force-https"}, {"name": "www.entropia.de", "mode": "force-https"}, {"name": "romab.com", "include_subdomains": true, "mode": "force-https"}, {"name": "logentries.com", "mode": "force-https"}, {"name": "www.logentries.com", "mode": "force-https"}, {"name": "stripe.com", "include_subdomains": true, "mode": "force-https"}, {"name": "cloudsecurityalliance.org", "include_subdomains": true, "mode": "force-https"}, {"name": "login.care.pt", "include_subdomains": true, "mode": "force-https"}]
```

# MitM is Still Possible...

## Rogue certificates

Most governments have a trusted root CA planted in our systems

Attackers may break into CAs and forge certificates

## Pre-planted/generated certificates

Default static keys: Lenovo, Dell, anti-malware software, ...

Low entropy during key generation: repeated or factorable keys

## Self-signed certificates

If desperate... will trigger browser warning

## Exploitation of certificate validation flaws

Programming errors while checking date, hostname, ...



Contact Us | Subscribe



Search Netcraft

Search

[Home](#) [News](#) [Anti-Phishing](#) [Security Testing](#) [Internet Data Mining](#) [Performance](#) [About Netcraft](#)

# StartSSL suspends services after security breach

StartSSL has suspended issuance of digital certificates and related services following a security breach on 15 June. A trademark of Eddy Nigg's StartCom, the StartSSL certificate authority is well known for offering free domain validated SSL certificates, but also sells organisation and extended validation certificates.

StartSSL™ - The Swiss Army Knife of Digital Certificates & PKI

**Maintenance**

Due to a security breach that occurred at the 15th of June, issuance of digital certificates and related services has been suspended. Our services will remain offline until further notice.

Subscribers and holders of valid certificates are not affected in any form.

Visitors to web sites and other parties relying on valid certificates are not affected.

We apologize for the temporary inconvenience and thank you for your understanding.

More than 25 thousand websites in Netcraft's SSL survey use certificates issued by StartSSL. These are recognised by Internet Explorer, Firefox, Chrome and other mainstream browsers.

StartSSL is not alone in offering free certificates. AffirmTrust recently trumped StartSSL's one-year

certificates with its own offer of free three-year domain validated SSL certificates. Coincidentally, AffirmTrust announced its launch [on the same day](#) as the StartSSL security breach.

StartSSL is also not the only certificate authority to come under attack this year. In March, Comodo came [under attack](#) through three of its resellers. By compromising a [GlobalTrust](#) website, the so-called *ComodoHacker* managed to fraudulently issue several valid certificates, including ones for the login pages of Yahoo and Skype. These certificates were subsequently revoked and browser software was updated to explicitly

## Most Popular

1. [January 2016 Web Server Survey](#)
2. [DigitalOcean becomes the second largest hosting company in the world](#)
3. [January 2015 Web Server Survey](#)
4. [eBay scripting flaws being actively exploited by fraudsters](#)
5. [Certificate revocation: Why browsers remain affected by Heartbleed](#)
6. [September 2015 Web Server Survey](#)
7. [February 2016 Web Server Survey](#)
8. [Fraudsters modify eBay listings with JavaScript redirects and proxies](#)
9. [March 2015 Web Server Survey](#)
10. [AlphaBay darknet phishing attack impersonates .onion domain](#)



Regis

[MAIN MENU](#)

MY STORIES: 25

FORUMS

SUBSCRIBE

JOBS

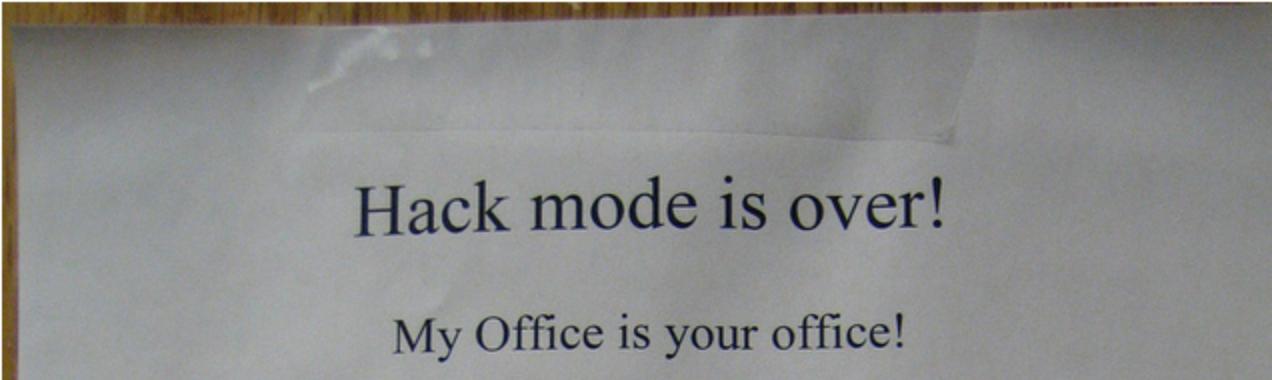
# RISK ASSESSMENT / SECURITY & HACKTIVISM

## Comodo hacker: I hacked DigiNotar too; other CAs breached

The hacker behind this year's Comodo hack has claimed responsibility for the ...

by Peter Bright - Sep 6, 2011 5:36pm EDT

[Share](#)  [Tweet](#)  [Email](#) 35



Photograph by Augie Schwer

### LATEST FEATURE STORY ▾



[FEATURE STORY \(2 PAGES\)](#)

### This could be the food future—if you can han

It's an evening of entomology—cooki and trying to understand an insect di

### WATCH ARS VIDEO ▾



**Security**

# Trustwave to escape 'death penalty' for SSL skeleton key

Moz likely to spare certificate-confession biz same fate as DigiNotar

14 Feb 2012 at 09:28, John Leyden



12



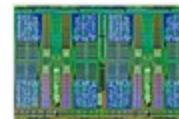
10

**Analysis** Trustwave's admission that it issued a digital "skeleton key" that allowed an unnamed private biz to spy on SSL-encrypted connections within its corporate network has sparked a fiery debate about trust on the internet.

Trustwave, an SSL certificate authority, confessed to supplying a subordinate root certificate as part of an information security product that allowed a customer to monitor employees' web communications - even if the staffers relied on HTTPS. Trustwave said the man-in-the-middle (MitM) gear was designed both to be tamper-proof and to work only within its unnamed client's compound. Despite these precautions, Trustwave now admits that the whole approach was misconceived and would not be repeated. In addition, it revoked the offending certificate.

Trustwave came clean without the need for pressure beforehand. Even so its action have split security experts and prompted calls on Mozilla's Bugzilla security list to remove the Trustwave root certificate

## Most read



AMD to fix slippery hypervisor-busting in its CPU microcode



First working Apple ransomware infects Transmission BitTo app downloads



Amazon douses flambé vows to restore Fire fondleslab encryption



MAME goes fully F



McAfee gaffe a quick kill for enterprise

# Self-signed Certificate Warning: One click away...

The screenshot shows a web browser window with the title "SSL Error". The address bar contains "https://www.twitter.com/". A large yellow warning box is displayed, containing the following text:

**The site's security certificate is not trusted!**

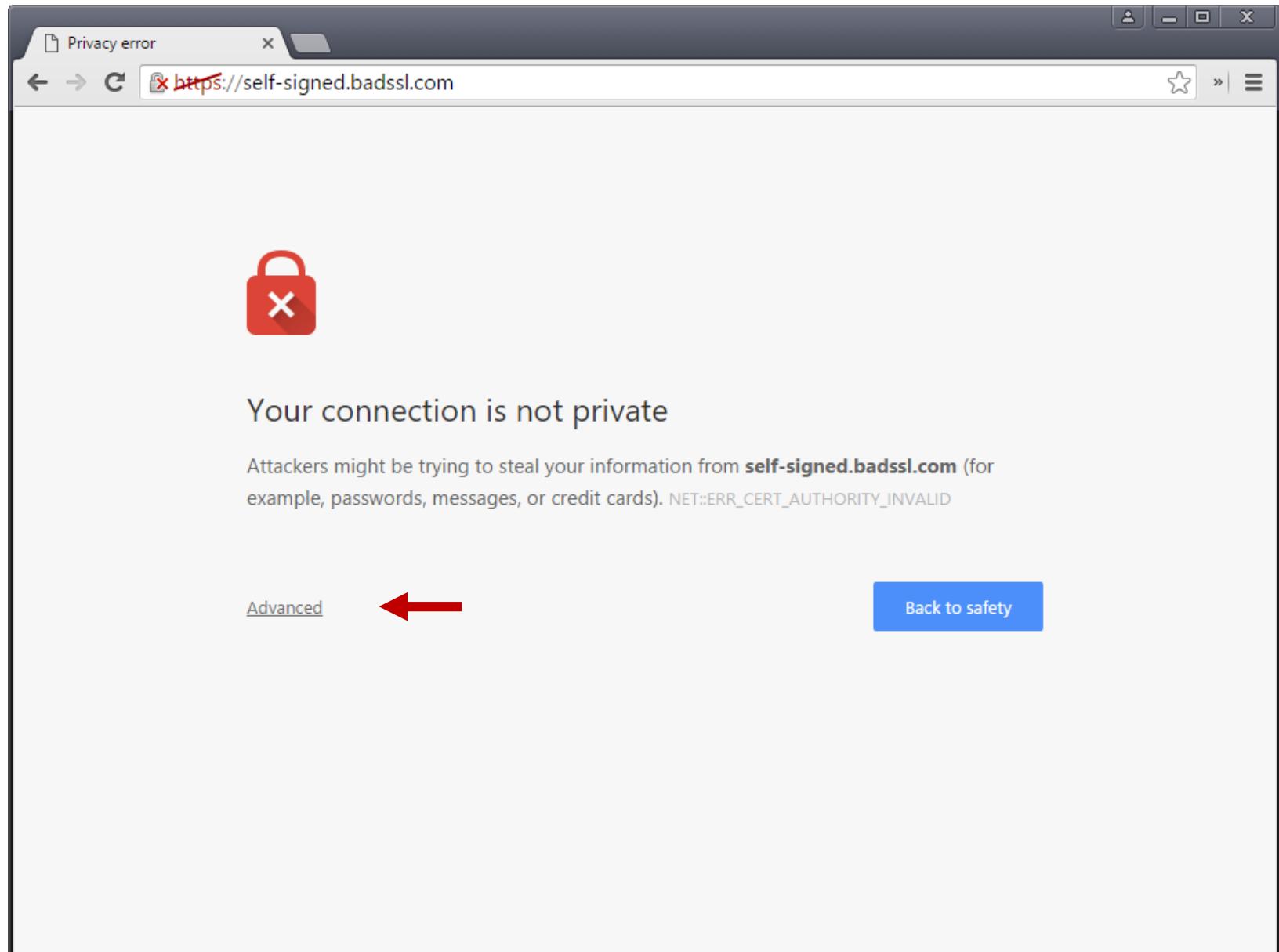
You attempted to reach [www.twitter.com](https://www.twitter.com), but the server presented a certificate issued by an entity that is not trusted by your computer's operating system. This may mean that the server has generated its own security credentials, which Chrome cannot rely on for identity information, or an attacker may be trying to intercept your communications.

You should not proceed, **especially** if you have never seen this warning before for this site.

[Proceed anyway](#) [Back to safety](#)

[Help me understand](#)

# Self-signed Certificate Warning: Two clicks away...



# Self-signed Certificate Warning: Two clicks away...

A screenshot of a web browser window titled "Privacy error". The address bar shows the URL <https://self-signed.badssl.com>. The page content displays a large red padlock icon with a white "X" inside. Below it, the text "Your connection is not private" is centered. A warning message follows: "Attackers might be trying to steal your information from **self-signed.badssl.com** (for example, passwords, messages, or credit cards). NET::ERR\_CERT\_AUTHORITY\_INVALID". At the bottom, there are two buttons: "Hide advanced" on the left and "Back to safety" on the right. A red arrow points to the "Hide advanced" link. Further down, another red arrow points to the link "Proceed to self-signed.badssl.com (unsafe)".

Privacy error

https://self-signed.badssl.com

Your connection is not private

Attackers might be trying to steal your information from **self-signed.badssl.com** (for example, passwords, messages, or credit cards). NET::ERR\_CERT\_AUTHORITY\_INVALID

[Hide advanced](#) ← [Back to safety](#)

This server could not prove that it is **self-signed.badssl.com**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

[Proceed to self-signed.badssl.com \(unsafe\)](#) ←

# GOTO FAIL

## iOS 7.0.6 signature verification error

Legitimate-looking TLS certificates with a mismatched private keys were unconditionally accepted...

```
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail; ← ?!!?!!?
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
...
fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

**Check never executed**

## HPKP (HTTP Public Key Pinning)

Prevent *certificate forgery*: strong form of web site authentication

Browser knows the *valid* public keys of a particular website

If a seemingly valid chain does not include at least one known pinned key, cert is rejected → not issued according to the site's expectations

Doesn't apply for *private root certificates* 

Would break preconfigured proxies, anti-malware, content filters, ...

Many incidents involving rogue certificates were discovered after browsers started rolling out pinning

Similar deployment as HSTS

TOFU: HTTP response header

Built-in pins in browsers

Must be used very carefully – things can go wrong

HPKP suicide: site can be bricked if keys are lost/stolen

RansomPKP: compromise the server and push a malicious HPKP key

## HSTS ▾ capturing events (70)

HSTS is HTTPS Strict Transport Security: a way for sites to elect to always use HTTPS. See <http://dev.chromium.org/sts>.

### Add domain

Input a domain name to add it to the HSTS set:



Domain:

Include subdomains for STS:

Include subdomains for PKP:

Public key fingerprints:

*(public key fingerprints are comma separated and consist of the hash function followed by a foreshash and the base64 encoded fingerprint, for example sha1/Guzek9LMwR3KeIS8wwS9gBvVtIg=)*

**Add**

### Delete domain

Input a domain name to delete it from the HSTS set (you cannot delete preloaded entries):

Domain:

**Delete**

### Query domain

Input a domain name to query the current HSTS set:

Domain:

**Query**

**Not found**



HSTS

capturing events (202)

### Delete domain

Input a domain name to delete it from the HSTS set (*you cannot delete preloaded entries*):

Domain: 

### Query domain

Input a domain name to query the current HSTS set:

Domain: **Found:**

static\_sts\_domain: google.com

static\_upgrade\_mode: OPPORTUNISTIC

static\_sts\_include\_subdomains: true

static\_sts\_observed: 1454980244

static\_pkp\_domain: google.com

static\_pkp\_include\_subdomains: true

static\_pkp\_observed: 1454980244

static\_spki\_hashes: sha1/vq70yjSnqOco9nyMCDGdy77ejM=,sha1/Q9rWM05T+KmAym79hfRqo3mQ40o=,sha1/wHqYaI2J+6sFZAwRfap9ZbjKzE4=

dynamic\_sts\_domain:

dynamic\_upgrade\_mode: UNKNOWN

dynamic\_sts\_include\_subdomains:

dynamic\_sts\_observed:

dynamic\_pkp\_domain:

dynamic\_pkp\_include\_subdomains:

dynamic\_pkp\_observed:

dynamic\_spki\_hashes:

# Google Security Blog

The latest news and insights from Google on security and safety on the Internet

## Enhancing digital certificate security

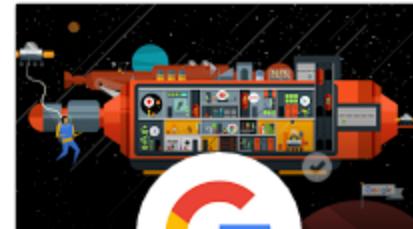
January 3, 2013

Posted by Adam Langley, Software Engineer

Late on December 24, Chrome detected and blocked an unauthorized digital certificate for the "`*.google.com`" domain. We investigated immediately and found the certificate was issued by an intermediate certificate authority (CA) linking back to TURKTRUST, a Turkish certificate authority. Intermediate CA certificates carry the full authority of the CA, so anyone who has one can use it to create a certificate for any website they wish to impersonate.

In response, we updated Chrome's certificate revocation metadata on

 Search blog ...



Google

[google.com/+google](http://google.com/+google)

News and updates on Google's products, technology and more

 Follow

+1

+ 11,007,947

# Certificate Revocation

Mechanism to allow revocation of compromised or no longer needed certificates

## Certificate revocation list (CRL)

Signed list of all serial numbers belonging to revoked certificates that have not yet expired

Main problem: lists tend to be large, making real-time lookups slow

Can the attacker block connectivity to the status server?

## Online Certificate Status Protocol (OCSP)

Obtain the revocation status of a *single* certificate → faster

But performance and privacy issues still remain

**OCSP stapling:** server embeds OCSP response directly into the TLS handshake

# Certificate Transparency

Public monitoring and auditing of certificates

Identify mistakenly or maliciously issued certs and rogue CAs

## *Certificate logs*

Network services maintaining cryptographically assured, publicly auditable, append-only records of certificates

## *Monitors*

Periodically contact all log servers and watch for suspicious certificates

## *Auditors*

Verify that logs are behaving correctly and are cryptographically consistent

Check that a particular certificate appears in a log

