Verizon’s Supercookie’
How a banner ad for H&R Block appeared on apple.com—without Apple’s OK

Someone, somewhere is injecting banner ads into webpages on the sly.
HTTPS as a ranking signal
August 6, 2014

REQUIRING HTTPS FOR...

• Service Workers
• HTTP/2
• Geo Location
• All new “powerful” web features
• Chrome and Firefox
TLS handshake 101

Client

Send me your certificate

Here it is

Looks good, I want to use X cipher

Yep, let's go.

(application data)

Server

Certificate verification (asymmetric crypto)

Symmetric key negotiation

2 RTTs

Client

encrypted app data

Server
1 RTT HANDSHAKE
TLS resumption 101

- Eliminates asymmetric crypto on the server via reuse of previously used parameters
- Eliminates full roundtrip, allowing 1-RTT connection establishment

Re-use negotiated parameters for the symmetric cipher
TLS resumption

**Session identifiers**
- Server assigns session ID
- Server caches parameters
- Client sends session ID
- Session is “resumed”

**Session tickets**
- Server encrypts parameters
- Server sets opaque ticket
- Client sends opaque ticket
- Server decrypts ticket and resumes session

Shared state is on the server

Shared state is on the client
A few things to think about…

1. **Session identifiers**
   a. Require a shared cache between servers for best results
   b. Sessions must be expired and rotated in a secure manner

1. **Session tickets**
   a. Require a shared ticket encryption key
   b. Shared encryption key must be rotated in a secure manner

1. **Perfect Forward Secrecy (PFS)**
   “Session ticket keys have to be distributed to all the frontend machines, without being written to any kind of persistent storage, and frequently rotated.”

Terminate TLS at the CDN edge...

**CDNs are not just for static content.**

Edge termination can significantly reduce TCP and TLS handshake costs!

*Before you hand over the keys to your kingdom, make sure your CDN has their TLS stack optimized! You may be surprised...*
Online Certificate Status Protocol (OCSP)

Stop the world and query the OCSP server

- DNS lookup
- TCP connect
- Wait for server response

What if the OCSP check times out, gets blocked, etc? See “Revocation still doesn’t work.”
Eliminating OCSP latency

- Chrome blocks on EV certs only
- Other browsers may block on all (FF)

Use OCSP stapling!

1. Server retrieves the OCSP response
2. Server “staples” response to certificate
3. Client verifies stapled response

OCSP endpoint
TLS handshake with stapled OCSP response...

```
$> openssl s_client -connect example.com:443 -tls1 -tlsextdebug -status

OCSP Response Data:
  OCSP Response Status: successful (0x0)
  Response Type: Basic OCSP Response
  Version: 1 (0x0)
  Responder Id: C = IL, O = StartCom Ltd., CN = StartCom Class 1 Server OCSP Signer
  Produced At: Feb 18 17:53:53 2014 GMT
  Responses:
  Certificate ID:
    Hash Algorithm: sha1
    Issuer Name Hash: 6568874F40750F016A3475625E1F5C93E5A26D58
    Issuer Key Hash: EB4234D098B0AB9FF41B6B08F7CC642EEF0E2C45
    Serial Number: 0B60D5
  Cert Status: good
```

Stapled OCSP means no blocking!

OCSP stapling increases certificate size! Is this a problem for your site? Better check.
How many RTTs does your certificate incur?

- Average certificate chain depth: 2-3 certificates
- Average certificate size: ~1~1.5KB
- Plus OCSP response…
- Many cert chains overflow the old TCP (4 packet) CWND
  - Upgrade your servers to use IW10!

3+ RTT TLS handshake due to 2 RTT cert?
Check your server, you may be surprised...

- Capture a tcpdump of your handshake and check the exchange
- Some servers will pause on “large certificates” until they get an ACK for the first 4KB of the certificate (doh!)
1-RTT non-resumed handshake with **TLS False Start**

Client sends application data immediately after “Finished”.

- **Eliminates 1RTT**
- **No protocol changes...**
- **Only timing is affected**

In practice...
- **Some servers break (ugh)**
- **Hence, opt-in behavior...**
Deploying False Start...

**Chrome and Firefox**
- ALPN advertisement - e.g. "http/1.1"
- Forward secrecy ciphersuite - e.g. ECDHE

**Safari**
- Forward secrecy ciphersuite

**Internet Explorer**
- Blacklist + timeout
- If handshake fails, retry without False Start

**TL;DR:** enable ALPN advertisement and forward secrecy to get 1RTT handshakes.
Ingredients for a 1-RTT TLS experience…

1. **False Start** = 1-RTT handshake for new visitors
   a. New users have to perform public-key crypto handshake

1. **Session resumption** = 1-RTT handshake for returning visitors
   a. Plus, we can skip public-key crypto by reusing previous parameters

1. **OCSP Stapling**
   a. No OCSP blocking to verify certificate status

1. **False Start + Session Resumption + OCSP stapling**
   a. 1-RTT handshake for new and returning visitors
   b. Returning visitors can skip the public-key crypto
What’s wrong with this picture?

300ms RTT, 1.5Mbps...

- It’s a 2-RTT handshake... we know better!
  - At least there is no OSCP overhead!

- It’s a 2-RTT time to first byte!
  Large records are buffered, which delays processing!
TLS record size + latency gotchas...

TLS allows up to 16KB of application per record

- New connection + 16KB record = CWND overflow and an extra RTT
- Lost or delayed packet delays processing of entire record

[8 Reassembled TCP Segments (11221 bytes): #169(1460), #170(1460), #172(1460), #174(1460), #175(1460), #177(1460), #179(1460), #180(1001)]

This record is split across 8 TCP packets
Optimizing record size...

1. **Google servers implement dynamic record sizing**
   a. New connections start with 1400 byte records (aka, single MTU)
   b. After ~1MB is sent, switch to 16K records
   c. After ~1s of inactivity, reset to 1400 byte records

1. **Most servers don’t optimize this case at all...**
   a. HAProxy recently landed dynamic sizing patch - yay!
   b. Nginx landed ssl_buffer_size: static override - better, but meh...
   c. Cloudflare released patch for Nginx dynamic record sizing

**TL;DR:** there is no “perfect record size”. Adjust dynamically.
Quick sanity check...

theory is great, but does this all work in practice?
Tuning Nginx TLS Time To First Byte (TTTFB)

- Pre 1.5.7: bug for 4KB+ certs, resulting in 3RTT+ handshakes
- 1.7.1 added ssl_buffer_size: 4KB record size remove an RTT
- 1.7.1 with NPN and forward secrecy → **1RTT handshake**

https://www.igvita.com/2013/12/16/optimizing-nginx-tls-time-to-first-byte/
<table>
<thead>
<tr>
<th>Server</th>
<th>Session identifiers</th>
<th>Session tickets</th>
<th>OCSP stapling</th>
<th>Dynamic record sizing</th>
<th>ALPN</th>
<th>Forward secrecy</th>
<th>HTTP/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ATS</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>dynamic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>bud</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>static</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Brocade vTM</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>F5 BIG-IP</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>H2O</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>static (1.4k)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>HAPROXY</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>dynamic</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Hitch</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>IIS</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>NetScaler</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>NGINX</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>static (16k)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>node.js</td>
<td>yes</td>
<td>yes</td>
<td>optional</td>
<td>optional</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Go</td>
<td>yes</td>
<td>yes</td>
<td>optional</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>nghttpx</td>
<td>yes</td>
<td>yes</td>
<td>dynamic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Provider</td>
<td>Session identifiers</td>
<td>Session tickets</td>
<td>OCSP stapling</td>
<td>Dynamic record sizing</td>
<td>ALPN</td>
<td>Forward secrecy</td>
<td>HTTP/2</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>-----------------------</td>
<td>------</td>
<td>-----------------</td>
<td>--------</td>
</tr>
<tr>
<td>Akamai</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>configurable (static)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>AWS ELB</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>AWS CloudFront</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>CDN77</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CloudFlare</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>dynamic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>ChinaNetCenter</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>EdgeCast</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Fastly</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>dynamic</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Google App Engine</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>dynamic</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Heroku</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Imperva Incapsula</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Instant Logic</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>configurable (static)</td>
<td>yes</td>
<td>yes</td>
<td>spdy/3.1</td>
</tr>
<tr>
<td>Limelight</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>MaxCDN</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>KeyCDN</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>configurable (static)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>QUANTIL</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
TLS has exactly one performance problem: it is not used widely enough.

*Everything else can be optimized.*

Data delivered over an unencrypted channel is insecure, untrustworthy, and trivially intercepted. We owe it to our users to protect the security, privacy, and integrity of their data — all data must be encrypted while in flight and at rest. Historically, concerns over performance have been the common excuse to avoid these obligations, but today that is a false dichotomy. Let's dispel some myths.
MORE HTTPS DEBUGGING
CSP VIOLATION REPORTS

- HTTP Response header

Content-Security-Policy-Report-Only:
  default-src https:;
  report-uri http://reportcollector.example.com/collector.cgi
**STRICT TRANSPORT SECURITY - HSTS**

- Server response header
  - Strict-Transport-Security: max-age=15768000; includeSubDomains; preload

- Chrome/Firefox preload list
KEY PINNING - HPKP

• Reducing/Reporting MITM attacks

• Server response header
  o Public-Key-Pins:
  o report-uri=https://example.net/pkp-report

• Doesn’t apply to “local trust anchors” 😞
• Chrome, Firefox, Opera
THANK YOU!

Tim Kadlec (@tkadlec) & Pat Meenan (@patmeenan)
June 20-21, 2016 at Velocity Conference (#velocityconf)

Icons courtesy of The Noun Project