HTTP, Security, and You

Eric Rescorla
ekr@rtfm.com

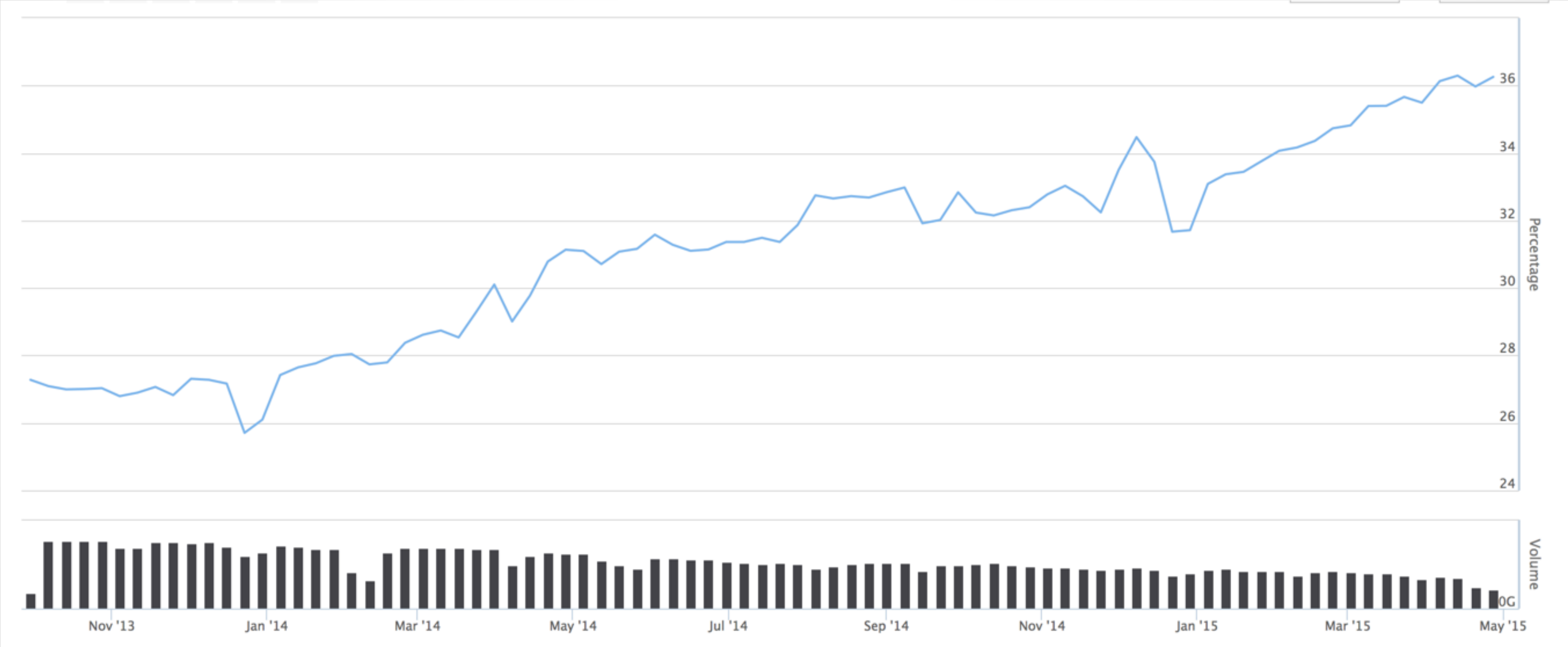
Agenda

- The state of the world
- Overview of TLS 1.3
- Stuff encryption gets in the way of
- What better security features can we offer?

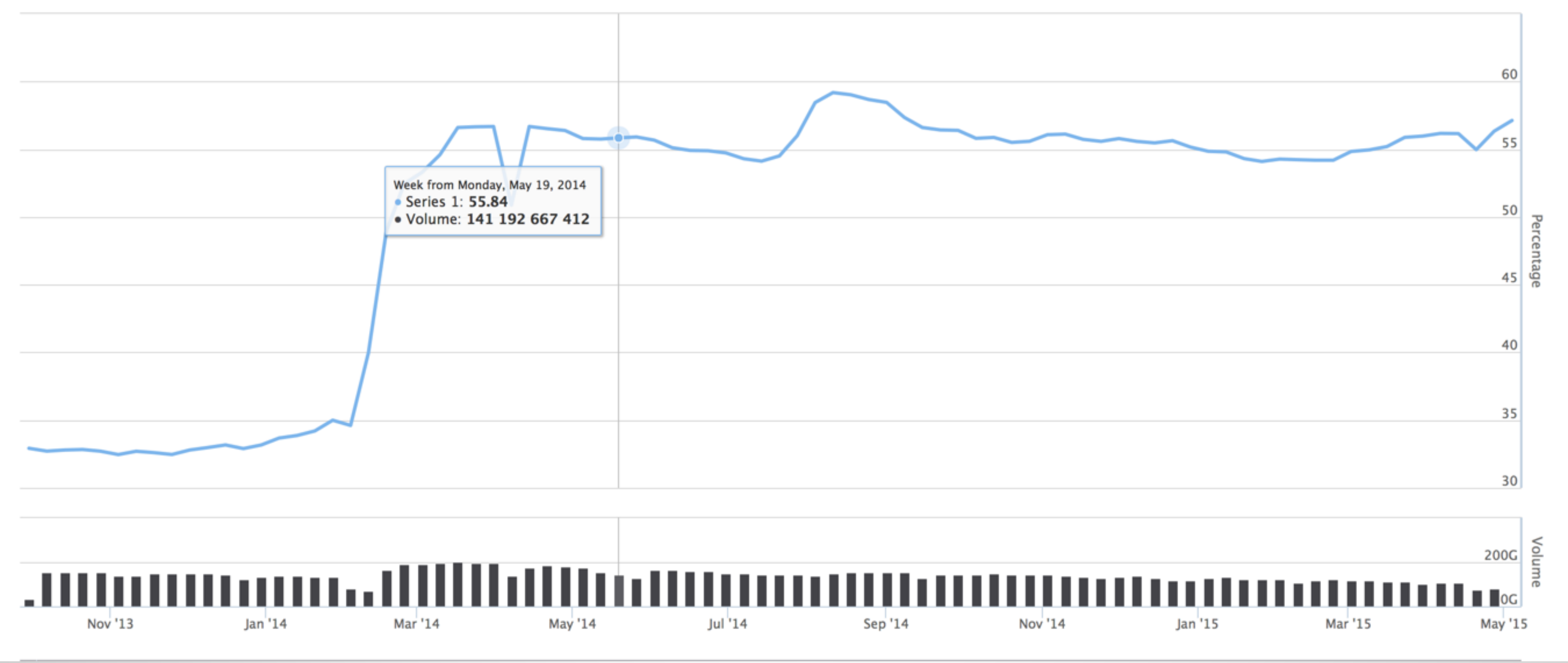
The future will be encrypted

- Big sites are rapidly moving to all-TLS
  - Google, Facebook, Twitter
- Encryption is table stakes for new designs
  - WebRTC, QUIC, WhatsApp (now)
- Let’s Encrypt will make this easier
- Browser pressure to transition to HTTPS

HTTPS by Page (Firefox)



HTTPS by Transactions (Firefox)



TLS 1.3 Goals

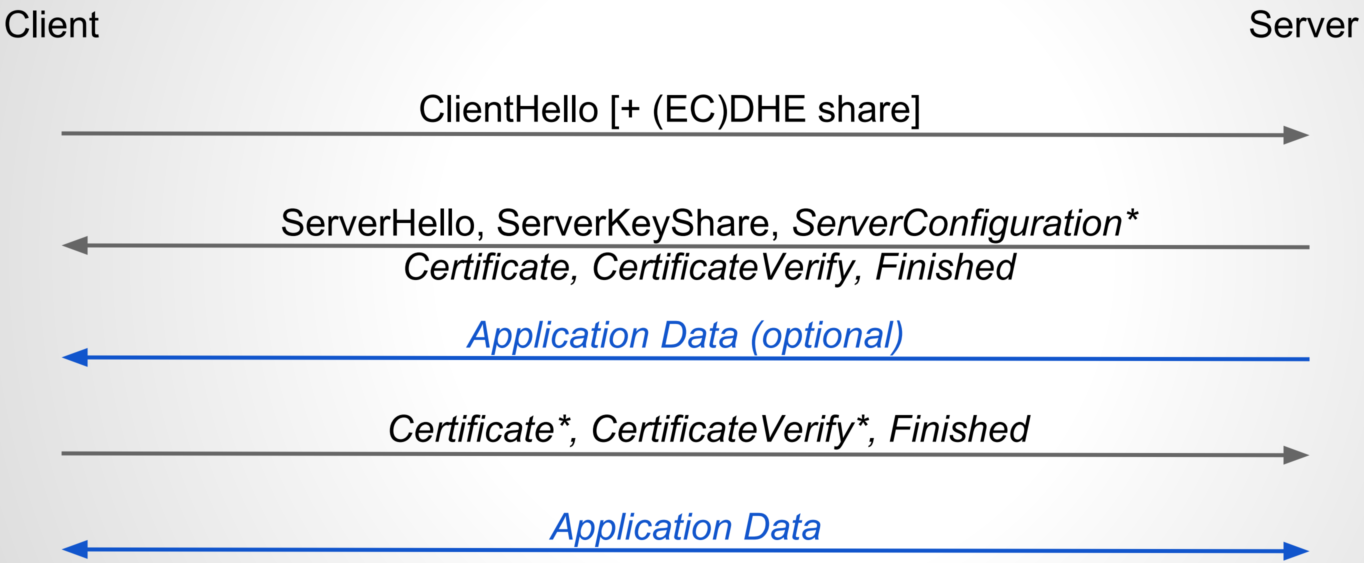
- *Clean up*: Remove unused or unsafe features
- *Improve privacy*: Encrypt more of the handshake
- *Improve latency*: Target: 1-RTT handshake for naive clients; 0-RTT handshake for repeat connections
- *Continuity*: Maintain existing important use cases

Removed features

- Static RSA
- Custom (EC)DHE groups
- Compression
- Renegotiation*
- Non-AEAD ciphers
- Simplified resumption

* Special accommodation for inline client authentication

TLS 1.3 1-RTT Handshake



Client

Server

ClientHello [+ (EC)DHE share]

ServerHello, ServerKeyShare, *ServerConfiguration**

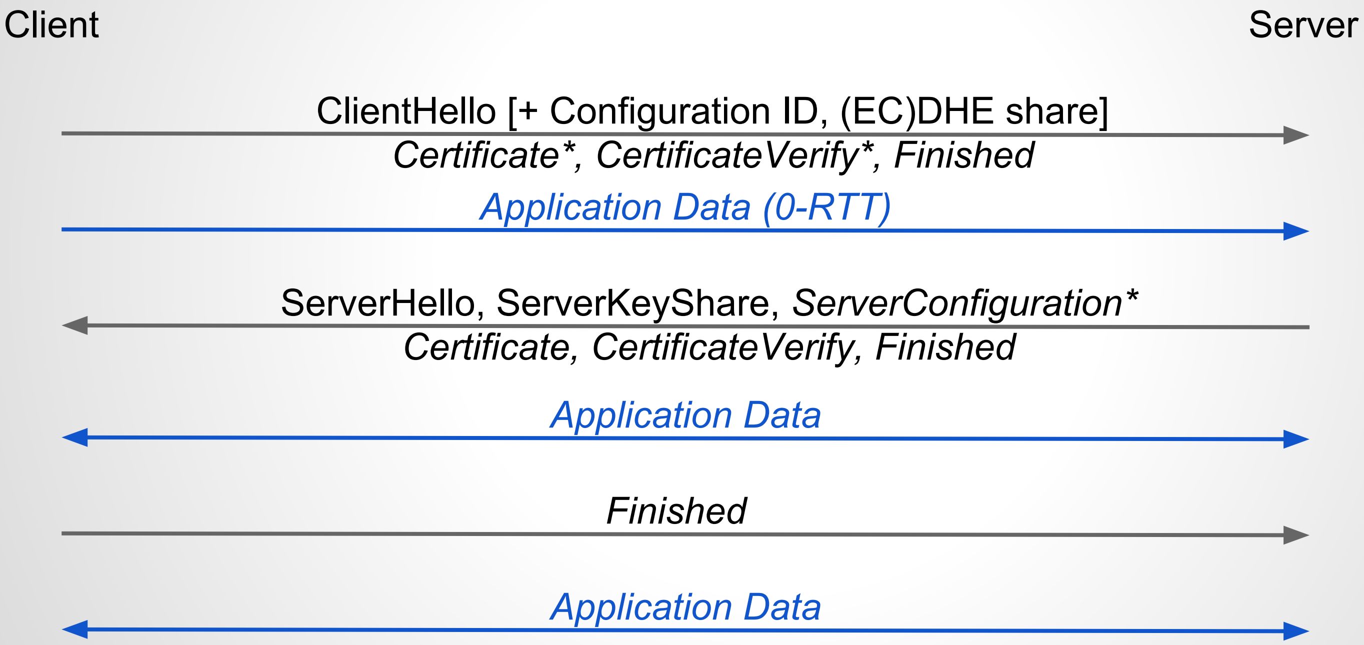
Certificate, CertificateVerify, Finished

Application Data (optional)

Certificate, CertificateVerify*, Finished*

Application Data

TLS 1.3 0-RTT Handshake



Client

Server

ClientHello [+ Configuration ID, (EC)DHE share]

Certificate, CertificateVerify*, Finished*

Application Data (0-RTT)

ServerHello, ServerKeyShare, *ServerConfiguration**

Certificate, CertificateVerify, Finished

Application Data

Finished

Application Data

Anti-replay (oops)

- Anti-replay turns out to be hard in 0-RTT
  - This is a distributed state problem
  - It’s broken in QUIC too
- Resolution: don’t even try
  - Only use 0-RTT client data for idempotent requests (GETs)
  - Difficult application integration issue
    - But too big a win not to do

Encryption makes some stuff harder

- Caching
- Network state management (w/ UDP)
- Network policy enforcement
  - DLP, Virus scanning, Parental controls, Censorship
- Law enforcement access
- Traffic discrimination
- Ad injection
- Malware injection

Not all of these are desirable. Some of these are what encryption is designed to prevent.

Digression: Performance*

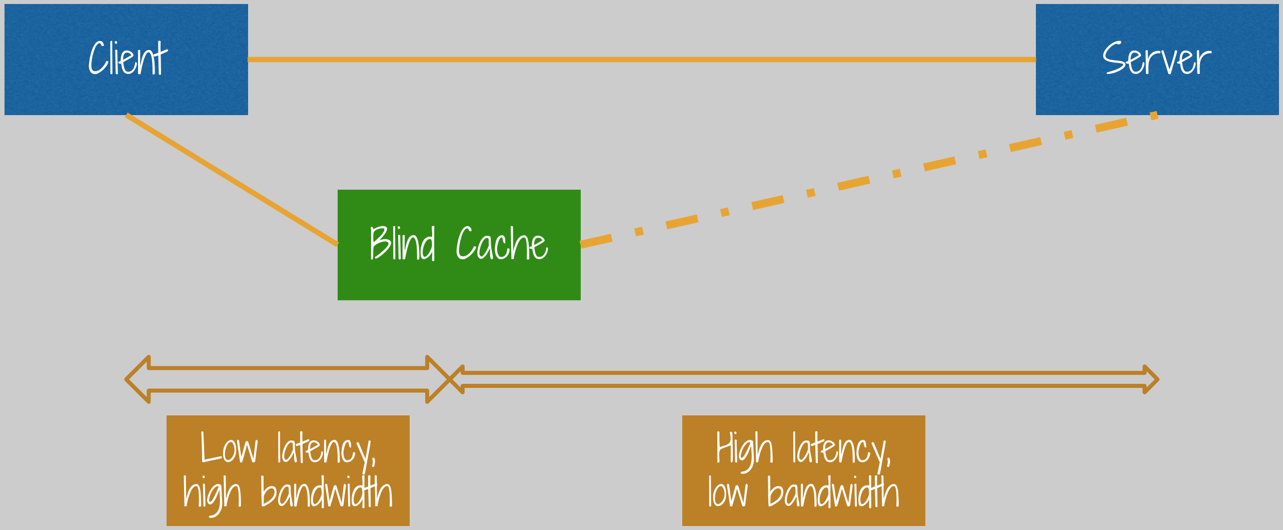
- AES-GCM: 2.53 C/B on Ivy Bridge, 1.03 C/B on Haswell
- OpenSSL P-256 signature: 17,111 ops/sec on Sandy Bridge 3.4 GHz
- OpenSSL P-256 ECDHE: 8,087 ops/sec on Sandy Bridge 3.4 GHz

* Results from Shay Gueron in various venues in 2013

Caching

- End-to-end encryption means caching doesn’t work
- This makes people sad
- But lots of people want to deploy untrusted caches anyway
- Let’s solve both at once

Blind cache architecture (preview)



Enterprise/Parental Content Inspection

- Already a problem in existing networks
  - There’s a lot of ciphertext anyway
- Two basic options
  - MITM proxies
  - Endpoint inspection hooks
- Assumption: *The inspector controls the endpoint!*

Transitioning to all-HTTPS (I)

- New content is moving toward HTTPS
  - “Powerful features”
  - Firefox HTTP Deprecation
- There’s a lot of legacy content
  - Comparatively easy to deploy a TLS server
  - Harder to deal with the URLs
    - These are of type “`http:`”
  - Don’t forget mixed content
  - Some combination of HSTS and Upgrade-Insecure

Transitioning to all-HTTPS (II)

- The world would be much safer if browsers couldn’t do non-secure HTTP at all
- How do we get the long tail?
- Some sort of universal HSTS?
- Whitelist of old servers?
  - Bloom filters, safe browsing, blah blah blah
- World’s biggest proxy?

Advanced stuff

- TLS is just a baseline
  - Makes it much easier to reason about
- What about advanced features?
  - SRI
  - SRI-based caching
  - Digital signing of requests/responses
  - HTTP content encryption
  - Strong client authentication