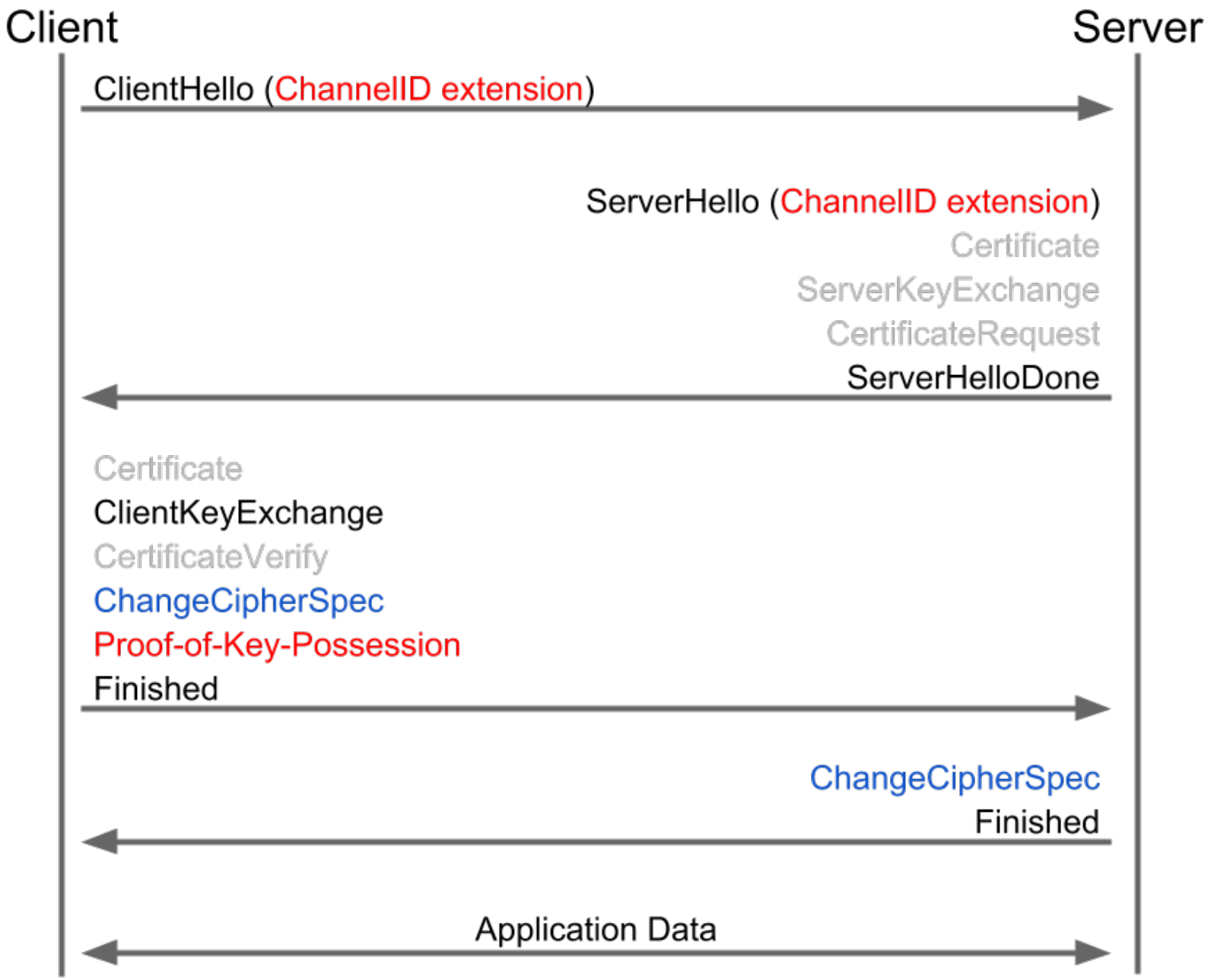
TLS ChannelID

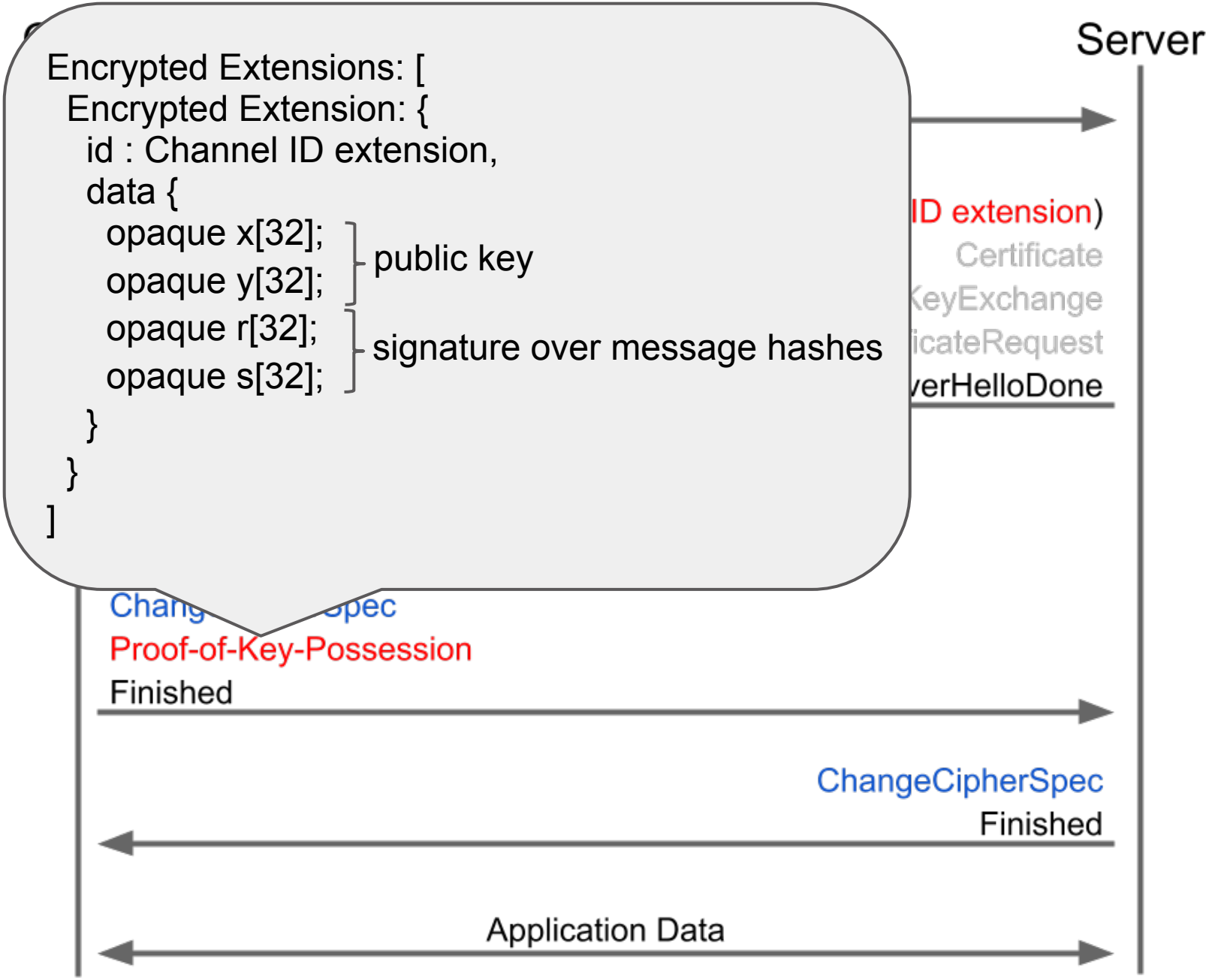
IETF 87 (2013-08-02 Berlin)

Dirk Balfanz, Google

Overview

1. IPR status
2. IETF control of standard
3. What is it? Why do we need it? Why in TLS?
4. Implementation status
5. Desired results


Client
Server
ClientHello (ChannelID extension)
ServerHello (ChannelID extension)
Certificate
ServerKeyExchange
CertificateRequest
ServerHelloDone
Certificate
ClientKeyExchange
CertificateVerify
ChangeCipherSpec
Proof-of-Key-Possession
Finished
ChangeCipherSpec
Finished
Application Data


Server
Encrypted Extensions: [
Encrypted Extension: {
id : Channel ID extension,
data {
opaque x[32];
opaque y[32];
public key
opaque r[32];
opaque s[32];
signature over message hashes
}
}
]
ID extension)
Certificate
KeyExchange
icateRequest
verHelloDone
Chang
Spec
Proof-of-Key-Possession
Finished
ChangeCipherSpec
Finished
Application Data

What is it?

- Gives persistent[1] cryptographic identifier for client.

[1]can be reset by user.

Why do we need it?

- Lets us build on transport layer security for application-level protocols through *channel binding*.
  Example: channel-bound cookies (TOFU)
  Example: channel-bound login assertions
  Example: channel-bound SAML (or OIC, or …) assertions

- Detects cookie theft, MITM during login, etc.

Why in TLS?

1. TLS is good at what it does - let's not reinvent the wheel at other layers.
   cipher negotiations, session renegotiations (while payloads continue to flow!), various attack mitigations, etc.

2. There are:
   ~3 TLS implementations, but
   ~100 HTTP implementations, and
   ~10000000 applications.

Why in TLS?

1. TLS is good at what it does - let's not reinvent the wheel at other layers.
   cipher negotiations, session renegotiations (while payloads continue to flow!), various attack mitigations, etc.

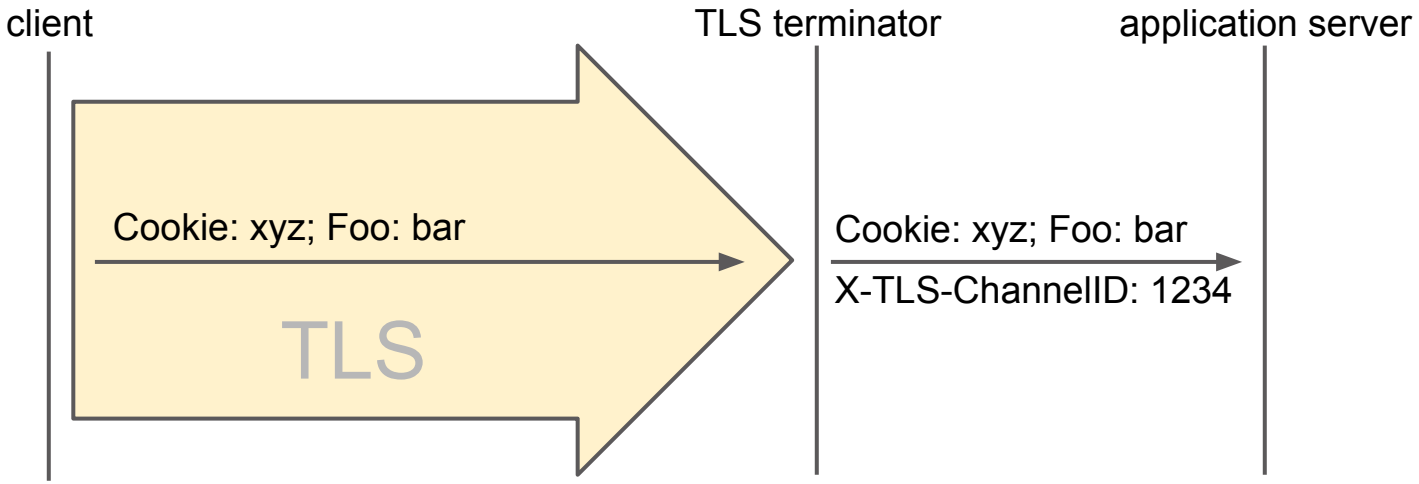
2. There are:
   ~3 TLS i
   ~100 HTTP imp , and
   ~10000000 app

Let's do the heavy lifting here

Common Concerns

- *ChannelID means that TLS terminators now must get involved in application-layer auth (e.g., check cookie-binding)!*

- No! TLS terminators simply add X-CGI-TLS-ChannelID header to the HTTP request, the rest is done by application backend.



Common Concerns

- *This belongs in the application/HTTP layer!*

- Different things belong in different layers:
  - user authentication at the application layer
  - session management at the HTTP layer
  - confidentiality/integrity (creating a “secure channel”) at the TLS layer

Common Concerns

- *It's a layer violation!*
- It's a narrow and well-defined mechanism that allows upper layers to benefit from transport layer security.

Common Concerns

- *Changing TLS means boiling the ocean!*

- We boiled it for you: patches to openssl and NSS exist.
- Changing 100s of HTTP implementation is way more risky.

Common Concerns

- *I want to protect my cookies, but I won't/can't run TLS!*
- Go for it! :-)

Common Concerns

- *I can’t change my TLS implementation!*
- Yes you can.
  (Surely you’re patching security holes, right?)
- *I really can’t.*
- It’s ok to just change the latest version - this change is backwards-compatible.

Implementation Status

- Implemented in Chrome & Google's server fleet.
  - 2nd generation implementation (after OBC)
- Open-source patches exist for openssl, NSS
  - AOSP has picked up openssl patch
  - openssl main branch waiting for standardization
- No performance issues in server, client
  - contrast with older OBC implementation
- Some false positives
  - browser bugs
  - enterprise "proxies" and client fallbacks

Desired Result

- Adoption as WG Item for Standards-Track RFC

Open Questions

- Uses EncryptedExtensions
  - draft-agl-tls-nextprotoneg
- Cipher negotiations vs. extension versioning