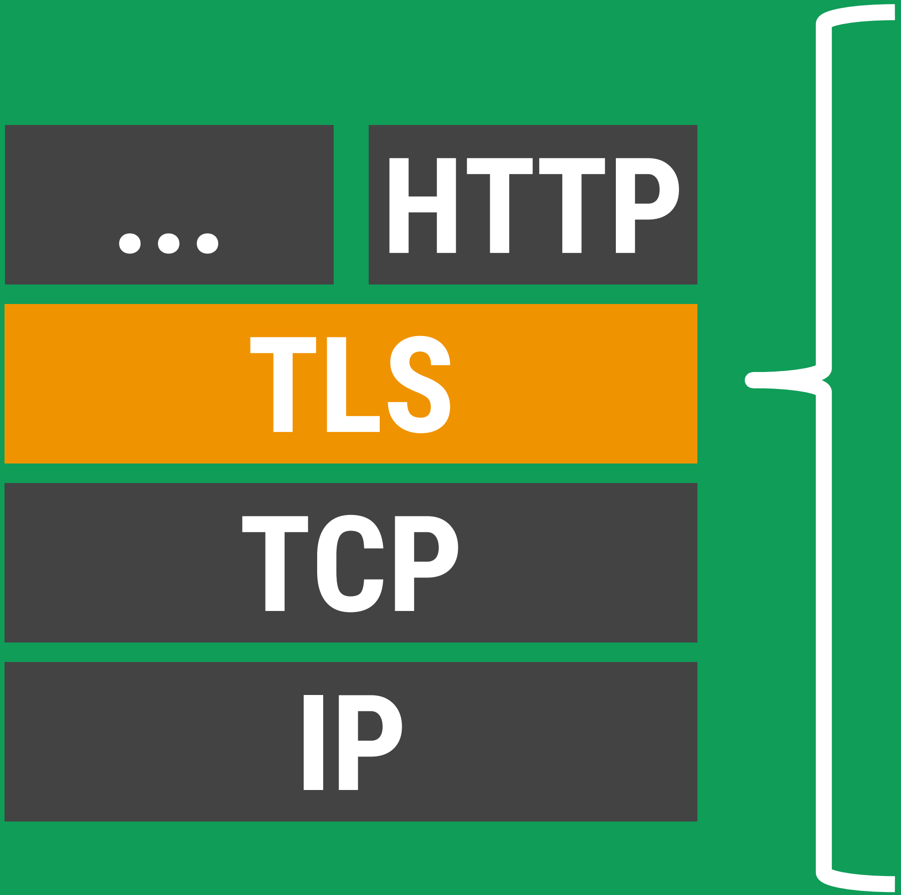
Node.js TLS

Deirdre Connolly
github.com/dconnolly
@durumcrustulum



...

HTTP

TLS

TCP

IP

Authentication

identity verification of server + client

Data integrity

protection against malicious middlemen

Encryption

privacy of exchanged communication

Transport Layer Security

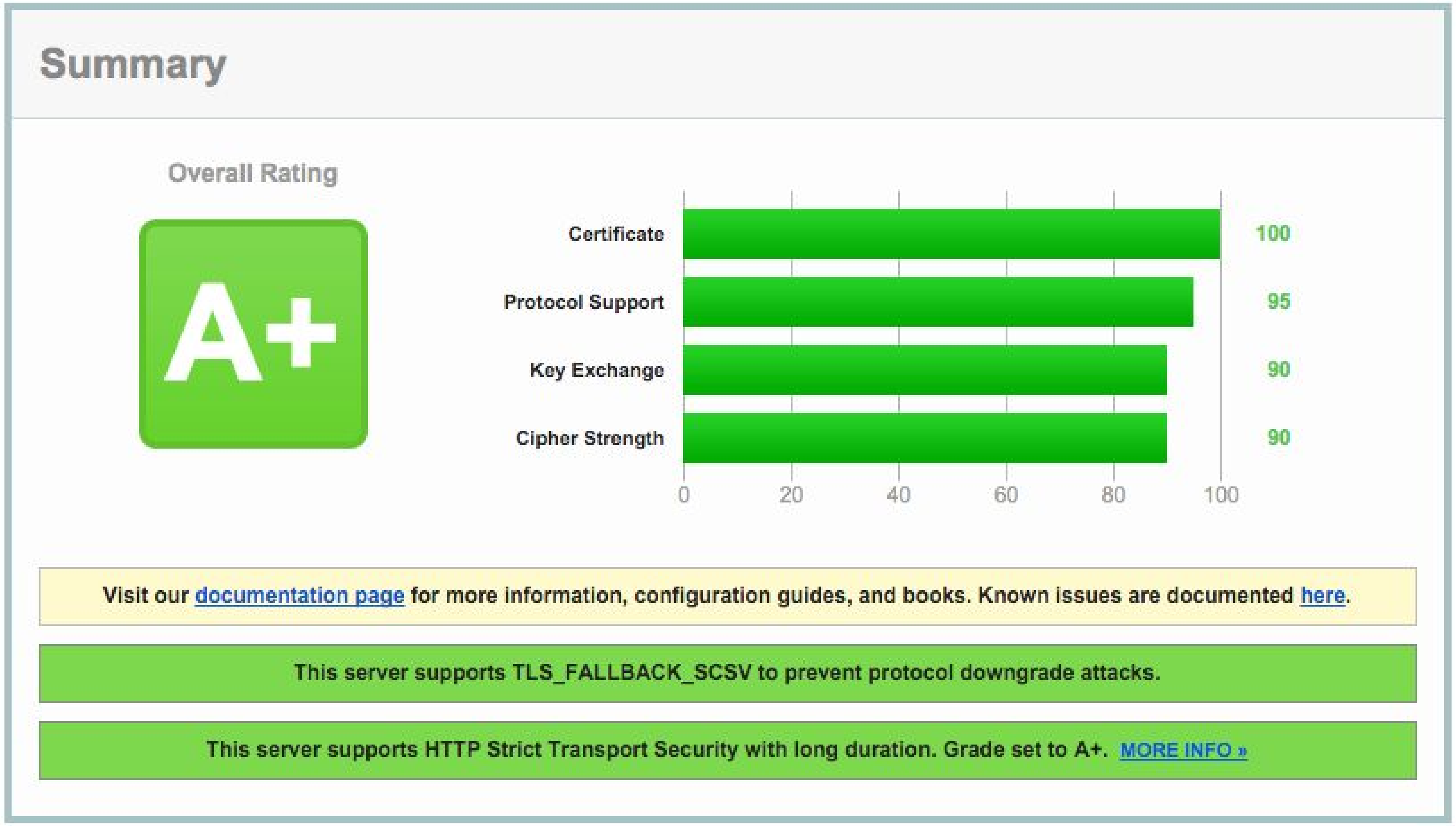
```
var https = require(‘https’);
var server = https.createServer({
    key: privateKey,
    cert: certificate,
    ca: certificateAuthorityCertificate
}, app);
```

**Node 4.0+*

```
var https = require(‘https’);
var server = https.createServer({
    key: privateKey,
    cert: certificate,
    ca: certificateAuthorityCertificate
}, app);
```

That’s it.

SSLabs



Summary

Overall Rating

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server supports TLS_FALLBACK_SCSV to prevent protocol downgrade attacks.

This server supports HTTP Strict Transport Security with long duration. Grade set to A+. MORE INFO »

Tests security of TLS, HTTP and X.509 certs

Free!

Score evolves over time as new vulnerabilities/bugs are discovered.

https://www.ssllabs.com/ssltest

TLS Handshake

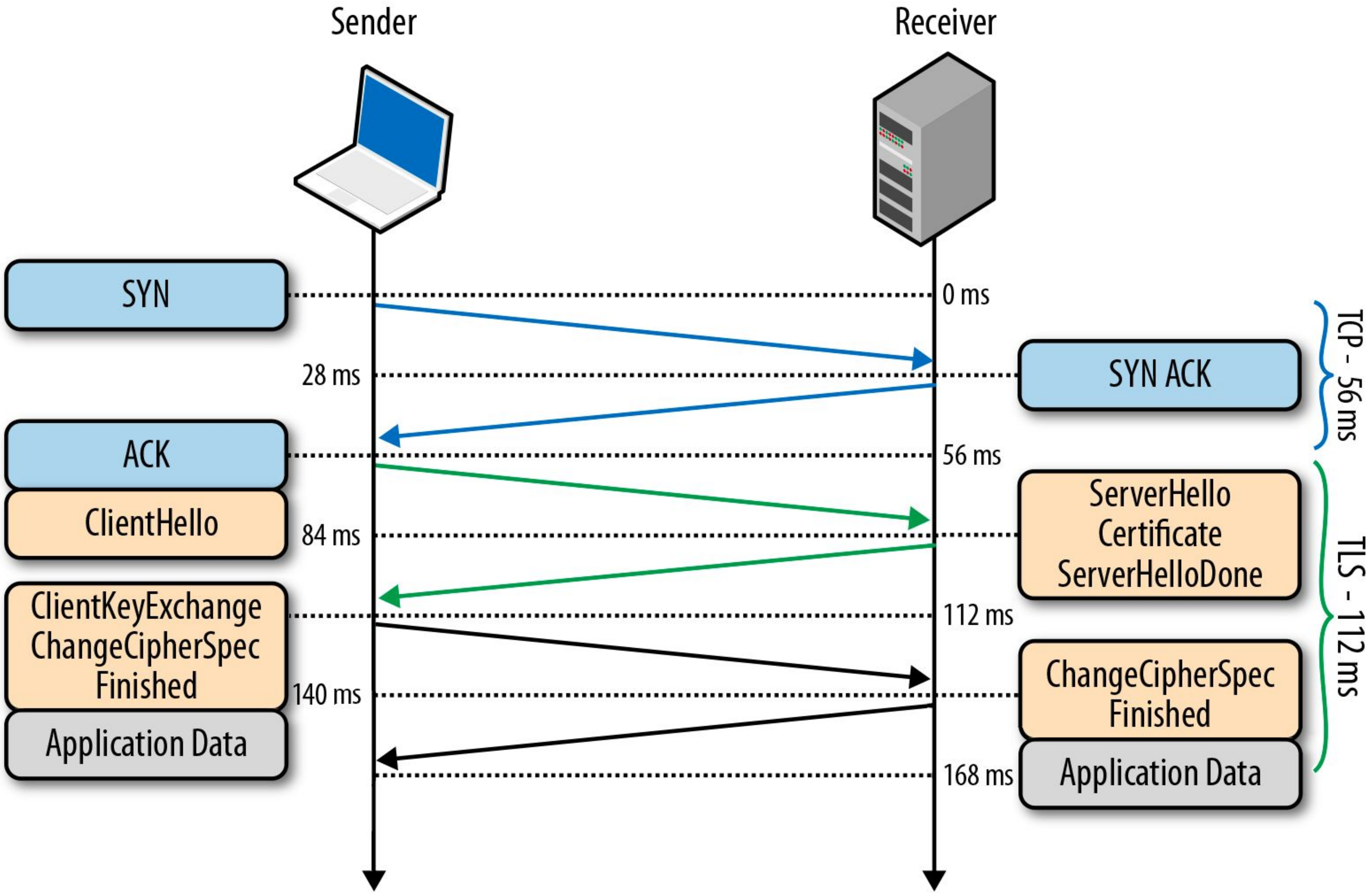


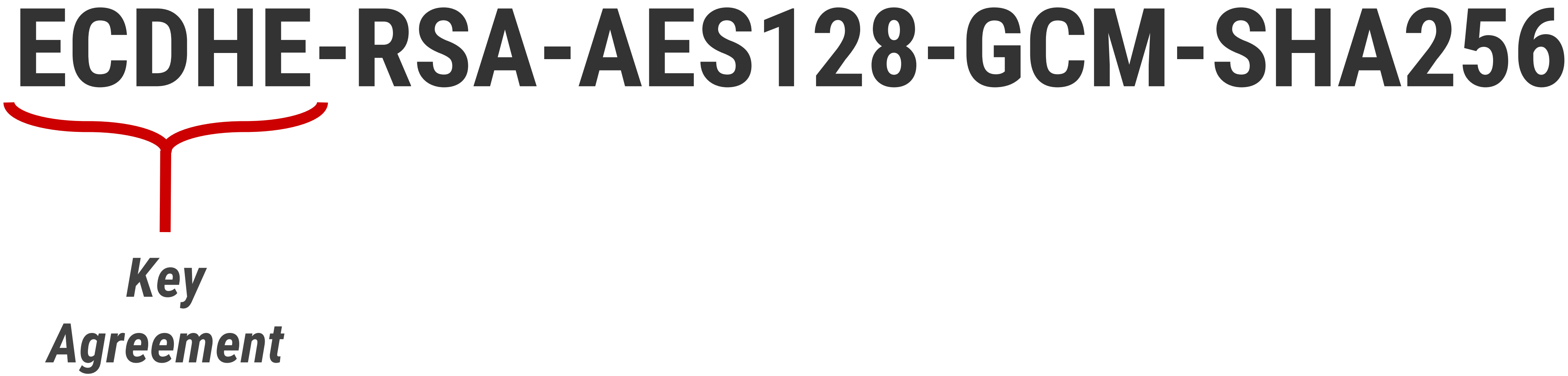
Diagram: High Performance Browser Networking

*Ciphersuite**

ECDHE-RSA-AES128-GCM-SHA256

OpenSSL-style

Ciphersuite*

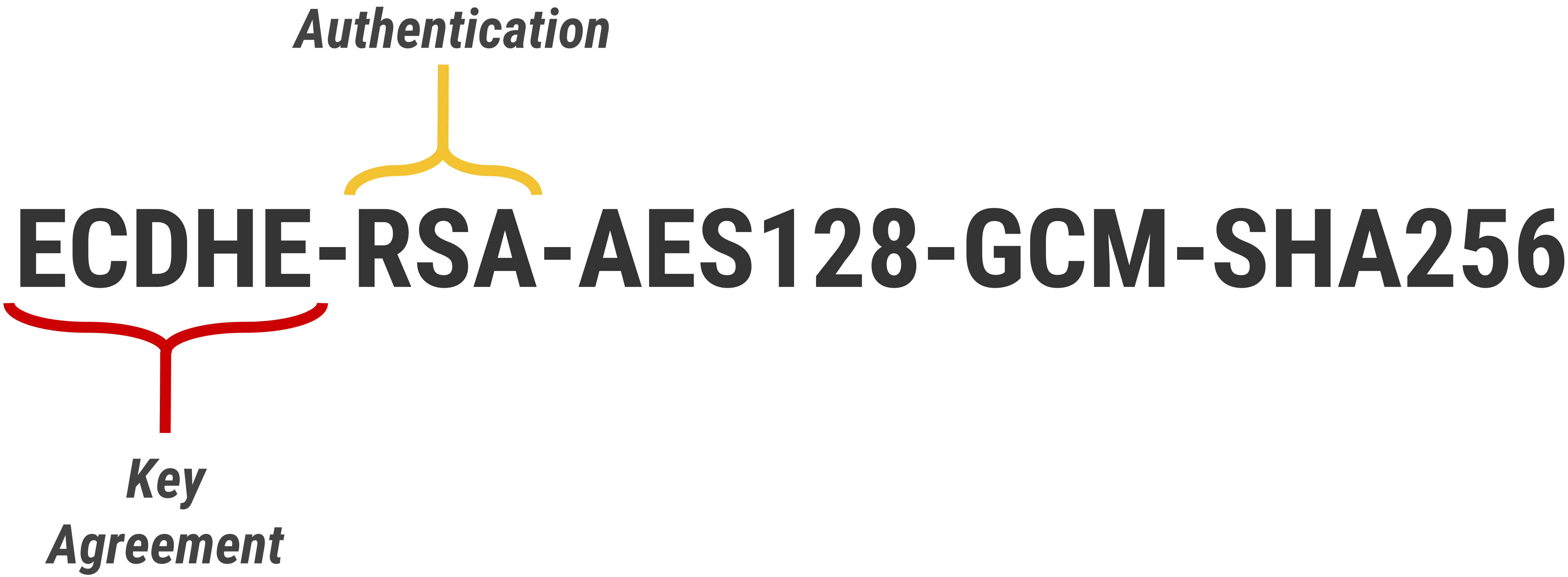


ECDHE-RSA-AES128-GCM-SHA256

Key Agreement

**OpenSSL-style*

*Ciphersuite**



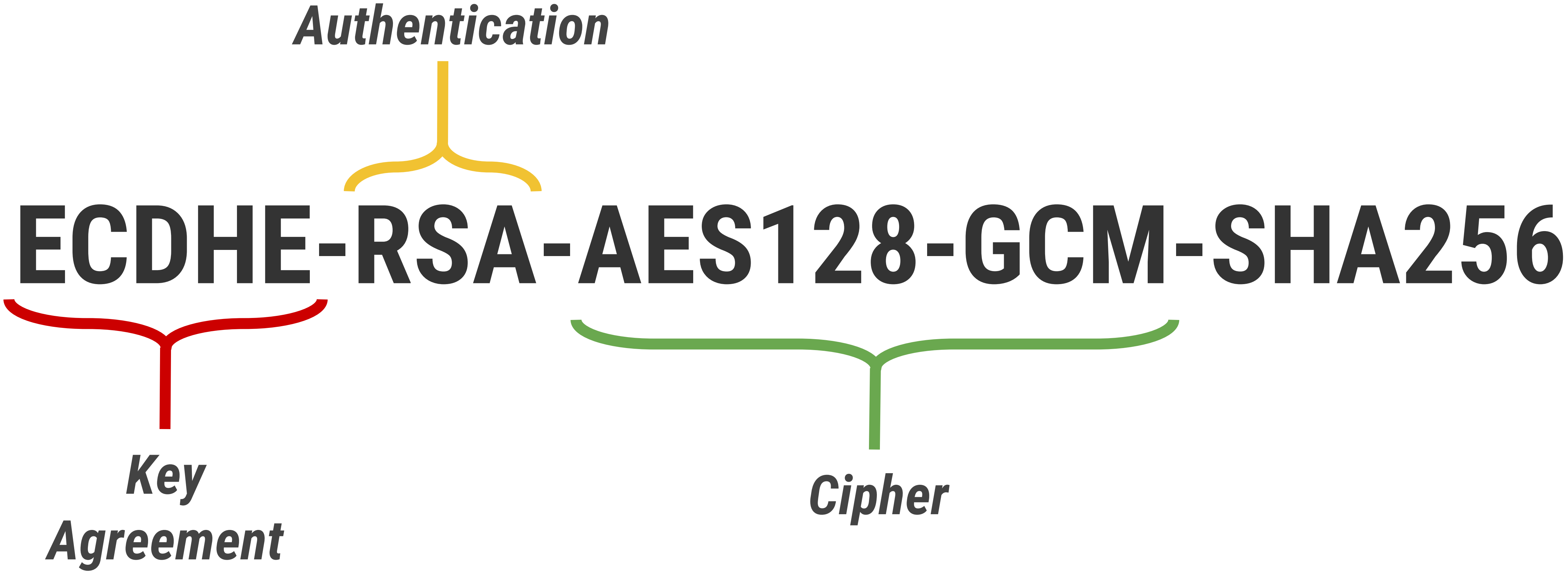
Authentication

ECDHE-RSA-AES128-GCM-SHA256

Key Agreement

**OpenSSL-style*

Ciphersuite*



Authentication

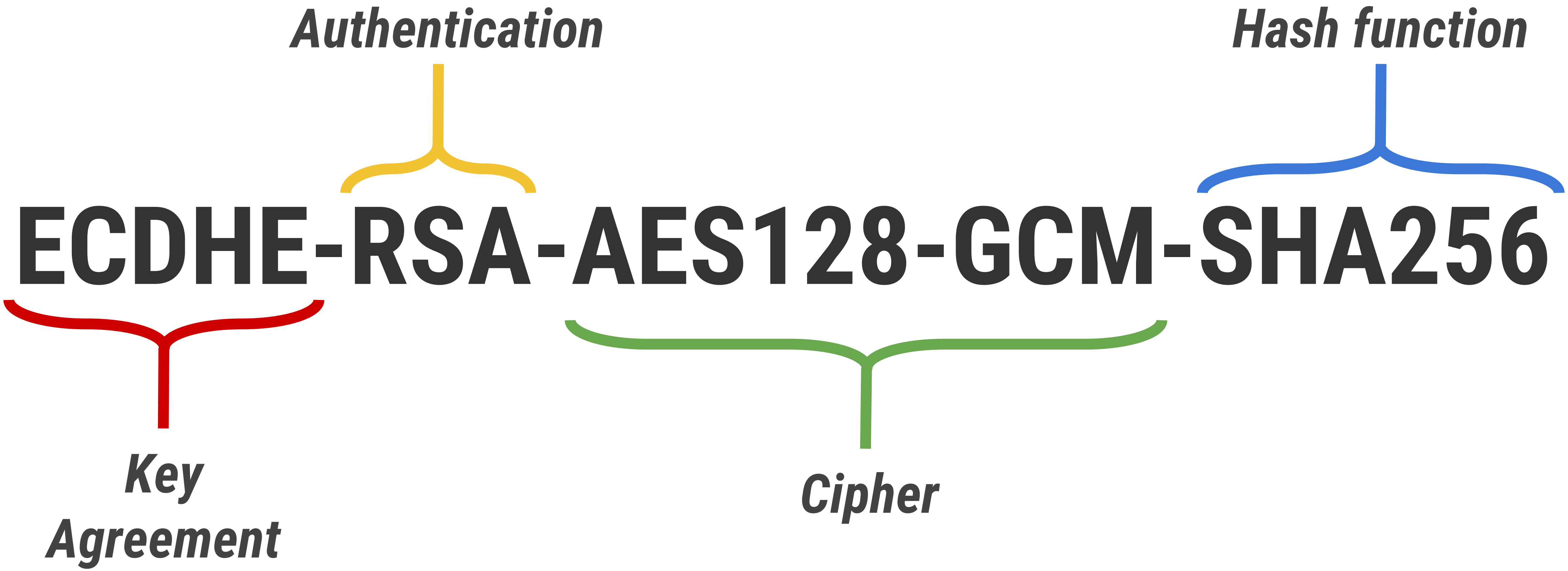
ECDHE-RSA-AES128-GCM-SHA256

Key Agreement

Cipher

*OpenSSL-style

Ciphersuite*



Authentication

Hash function

ECDHE-RSA-AES128-GCM-SHA256

Key Agreement

Cipher

*OpenSSL-style

node-tls ciphersuites

Prioritize forward-secure key exchanges (*DHE)

AES-128 is strong, fast, not susceptible to 256 attack

AEAD where possible (GCM), CBC if not

SHA-2! SHA-1 is ~broken, MD5 is hellabroken

No DES! No RC4! No EXPORT ciphers!

Always encrypt, always authenticate!

```
ECDHE-ECDSA-AES128-GCM-SHA256:
ECDHE-RSA-AES256-GCM-SHA384:
ECDHE-ECDSA-AES256-GCM-SHA384:
DHE-RSA-AES128-GCM-SHA256:
ECDHE-RSA-AES128-SHA256:
DHE-RSA-AES128-SHA256:
ECDHE-RSA-AES256-SHA384:
DHE-RSA-AES256-SHA384:
ECDHE-RSA-AES256-SHA256:
DHE-RSA-AES256-SHA256:
HIGH:
!aNULL:
!eNULL:
!EXPORT:
!DES:
!RC4:
!MD5:
!PSK:
!SRP:
!CAMELLIA
```

Reverse Proxy

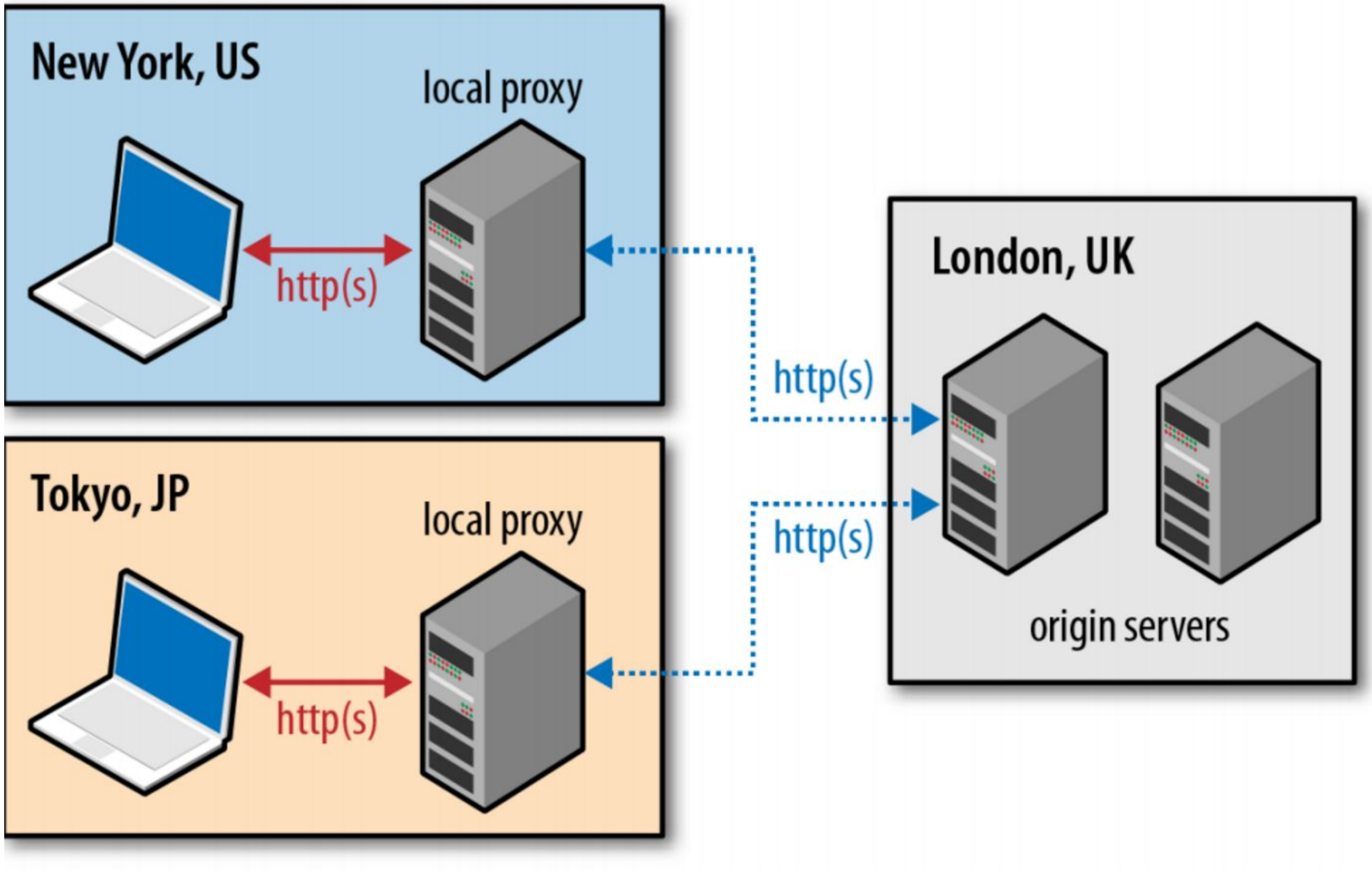


Diagram: High Performance Browser Networking

Mozilla SSL Configuration Generator is your friend: https://mozilla.github.io/server-side-tls/ssl-config-generator/

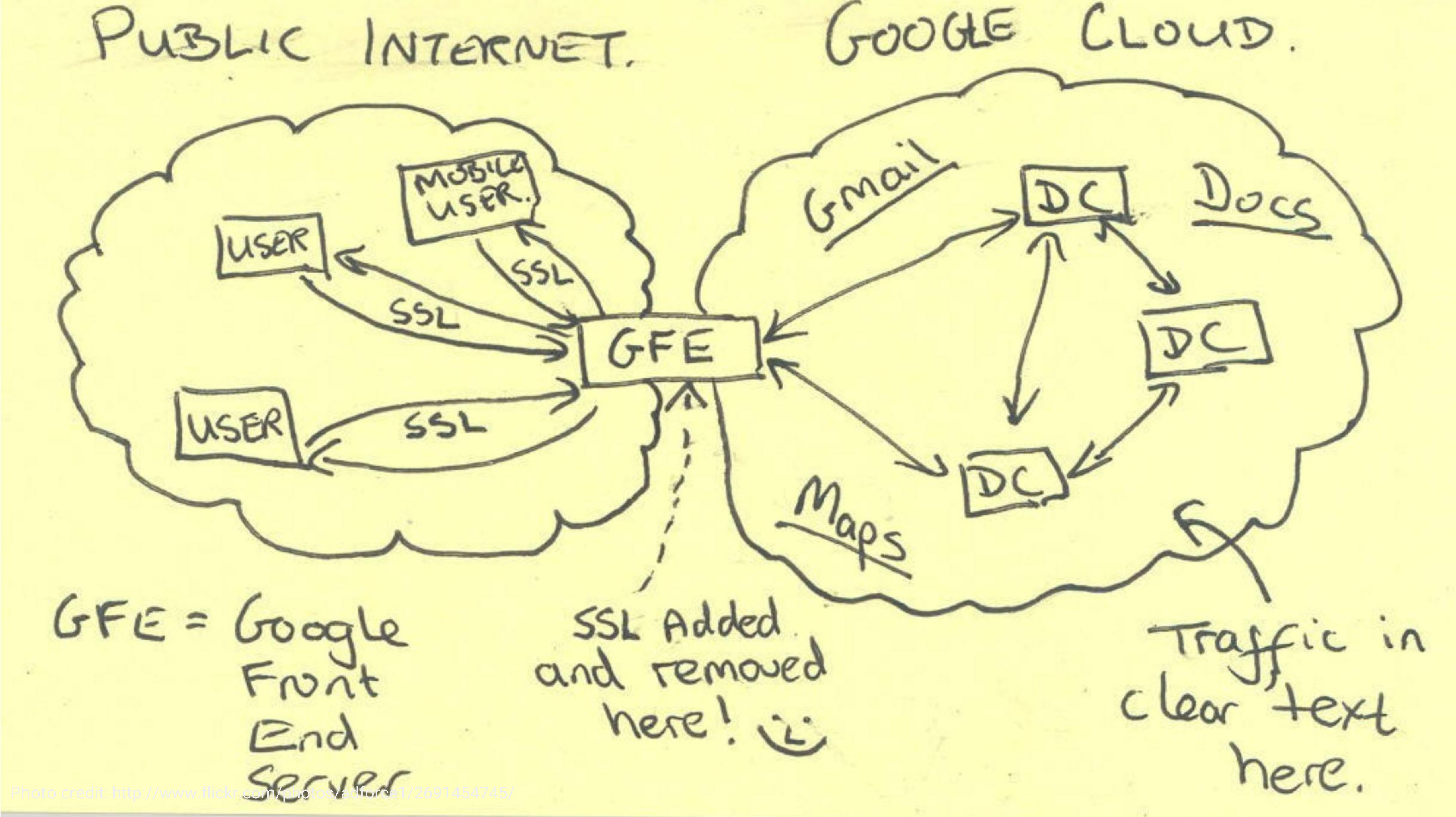

PUBLIC INTERNET.
GOOGLE CLOUD.
MOBILE USER.
USER
USER
SSL
SSL
SSL
GFE
Gmail
DC
Docs
DC
DC
Maps
GFE = Google Front End Server
SSL Added and removed here!
Traffic in clear text here.

Photo credit: http://www.flickr.com/photos/adforce1/2691454745/

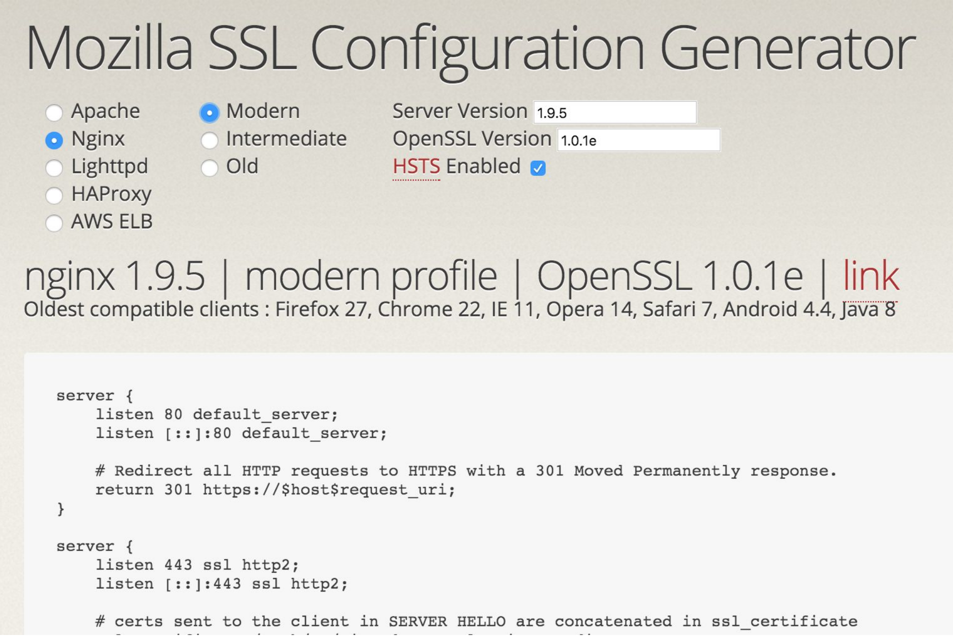
TL;DR

or

require('https')

Use Mozilla's SSL Configs

- https://mozilla.github.io/server-side-tls/ssl-config-generator/



Mozilla SSL Configuration Generator

○ Apache
● Nginx
○ Lighttpd
○ HAProxy
○ AWS ELB

● Modern
○ Intermediate
○ Old

Server Version 1.9.5
OpenSSL Version 1.0.1e
HSTS Enabled ☑

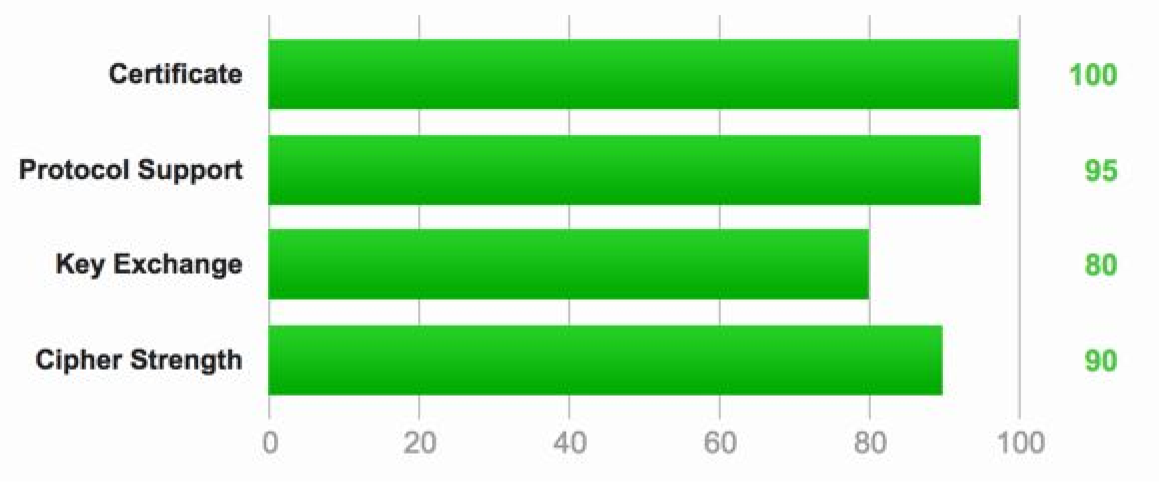
nginx 1.9.5 | modern profile | OpenSSL 1.0.1e | link

Oldest compatible clients : Firefox 27, Chrome 22, IE 11, Opera 14, Safari 7, Android 4.4, Java 8

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # Redirect all HTTP requests to HTTPS with a 301 Moved Permanently response.
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;

    # certs sent to the client in SERVER HELLO are concatenated in ssl_certificate
```

Verify your TLS configuration

- https://www.ssllabs.com/ssltest/



Thanks!

Deirdre Connolly
github.com/dconnolly
@durumcrustulum