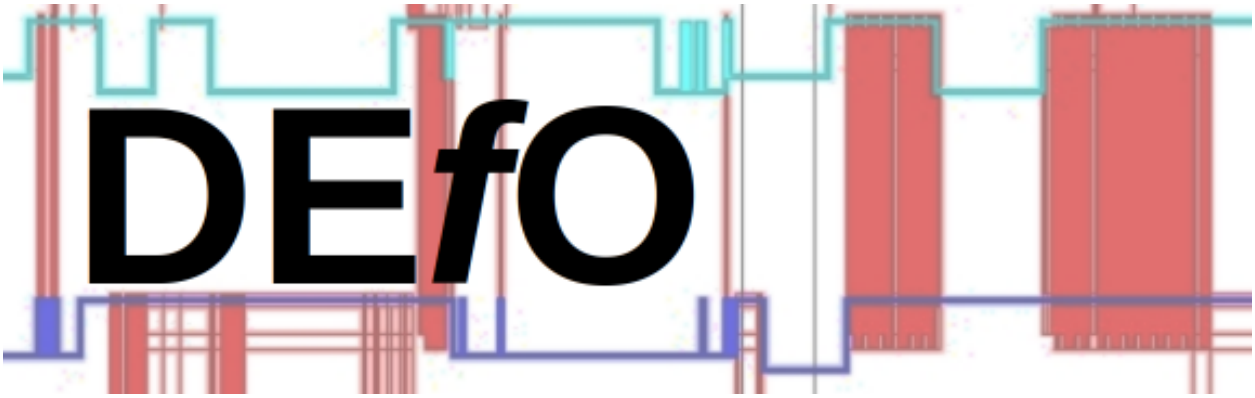

ECH Interoperability Report

Stephen Farrell

Trinity College Dublin/Tolerant Networks Ltd.

stephen.farrell@cs.tcd.ie

work-in-progress 2024/12/05:21:19 UTC

Abstract

The abstract.

Index Terms

Encrypted Client Hello (ECH), Interoperability

I. Introduction

Deployments of the Transport Layer Security (TLS [1]) protocol expose the name of the server (e.g. the web site DNS name) via the Server Name Indication (SNI) field in the first message sent (the ClientHello). The Encrypted Client Hello (ECH) [2] extension to TLS is a privacy-enhancing scheme that aims to address this leak.

This report describes the current state of ECH interoperability.

Comments, additions or corrections are welcome. Those can be sent via email to the author, or as issues or PRs (preferably for the latex files) in the github repository for this report which is https://github.com/defo-project/ech-interop-report.

II. SOFTWARE

This section describes libraries, clients and servers that support ECH.

A. Libraries

The libraries listed in Table I have some support for ECH. Of the libraries listed, all of the OpenSSL variants, python and libcurl were developed by the DEfO project.

TABLE I: Libraries with ECH

Name	Details
OpenSSL-sftcd	Source: https://github.com/sftcd/openssl/tree/ECH-draft-13c Version: 3.4.0-dev fork of master ECH support: client and server, TLS only (no DTLS, no QUIC) Comment: this is the DEfO project's main development branch
OpenSSL-defo	Source: https://github.com/defo-project/openssl/ Version: 3.5.0-dev fork of master, same ECH code as OpenSSL-sftcd ECH support: client and server, TLS only (no DTLS, no QUIC) Comment: this is used for DEfO project CI builds and tests
OpenSSL-feature-branch	Source: https://github.com/openssl/openssl/tree/feature/ech Version: 3.5.0-dev feature branch in upstream repo ECH support: stub APIs so far, next PR will have ECH client, then server after Comment: this is the DEfO-project's target for ECH PRs, and is where ECH code will end up prior to eventually being merged to master
boringssl	Source: https://boringssl.googlesource.com/boringssl Version: boringssl doesn't do versions, last local build 2024-11-29 ECH support: ECH client and server, ECH for TLS, QUIC and (possibly) DTLS Comment: in production use (chromium et al), a little more limited in HPKE suites than OpenSSL - only KEMs are x25519 and p256
NSS	Source: https://github.com/nss-dev/nss.git Version: NSS 3.108 ECH support: ECH client and server, ECH for TLS (unsure of DTLS/QUIC)

Continued on next page

TABLE I: Libraries with ECH (Continued)

Name	Details
	Comment: in production use (firefox), a little more limited in HPKE suites than OpenSSL - only KEM is x25519
WolfSSL	Source: https://github.com/wolfSSL/wolfssl.git Version: 5.7.4 ECH support: ECH client and server, ECH for TLS (unsure of DTLS/QUIC) Comment: - ECH not built by default (needs "–enable-ech") - fails when HelloRetryRequest seen - https://github.com/wolfSSL/wolfssl/issues/6802
gnuTLS	Source: https://gitlab.com/gnutls/gnutls/blob/master/README.md Version: work-in-progress ECH support: interop untested (by DEfO-project) at this time Comment: an ECH merge request exists but has yet to be merged https://gitlab.com/gnutls/gnutls/-/merge_requests/1748
golang	Source: https://go.googlesource.com/go or https://github.com/golang/go/ Version: 1.23 or later required ECH support: client only, server coming in 1.24 (server code is merged) Comment: - golang tests were just (2024-12-02) added to our smokeping tests
rustls	Source: https://github.com/rustls/rustls/ Version: 0.23.19 ECH support: client only Comment: - rustls tests were just (2024-12-03) added to our smokeping tests
python	Source: https://github.com/defo-project/cpython Version: python 3.13/3.14 ECH support: TLS client only Comment:

Continued on next page

TABLE I: Libraries with ECH (Continued)

Name	Details
	- python tests were just (2024-12-05) added to our smokeping tests
libcurl	Source: part of https://github.com/curl/curl Version: 8.10.0-DEV ECH support: TLS client only Comment: not well tested other than via command line curl

B. Clients

Current versions of firefox (e.g. 133.0, used in our smokeping tests) and chromium (version 131.0.6778.85 is used in our tests) support ECH by default. Other chromium-derived browsers also support ECH, while we don't include them in our ongoing tests described below, we have manually verified that current versions of vivaldi, brave and opera also have ECH support. The Tor browser (a firefox derivative) doesn't seem to support ECH by default. To test if a browser supports ECH one can visit https://test.defo.ie/ or, for much more detail https://test.defo.ie/iframe_tests.html.

Browsers in general are still somewhat quirky in terms of whether or not ECH will succeed, even when ECH is properly setup. This seems at least partly due to the delay that can inherently occur between the browser receiving DNS answers for A/AAAA records, and the time at which the browser receives a DNS answer for an HTTPS record (containging ECH information). It seems to be the case that browsers may be "impatient" in that they start the TLS session without attempting ECH if that delay is too long, and that that happens sufficiently often to be noticeable. **TODO: Characterise browser oddities via some JS?**.

The only command line tool supporting ECH of which we're aware is curl. The DEfO project contributed ECH code as an experimental feature for curl that was merged to the master branch in April 2024. As an experimental feature, ECH is not built by default nor part of a release, so is only currently available to those who build from source. Curl supports using OpenSSL, boringssl or Wolfssl libraries for ECH. Build instrucftions are at https://github.com/curl/curl/blob/master/docs/ECH.md.

As a scripting tool, often used as part of a TLS client, Python also fits here. The DEfO project Cpython build adds ECH support to python via the OpenSSl library. That build is available

at https://github.com/defo-project/cpython with the new APIs required for ECH described at https://irl.github.io/cpython/library/ssl.html#encrypted-client-hello.

C. Servers

The servers listed in Table II support ECH. In each case, the ECH integration, using OpenSSL, was developed as part of the DEfO project.

TABLE II: Servers supporting ECH

Name	Details
lighttpd	Source: https://github.com/defo-project/lighttpd Version: 1.4.76.81 Comment: only shared-mode ECH
nginx	Source: https://github.com/defo-project/nginx Version: 1.27.3.4 Comment: both shared-mode and split-mode ECH
apache2	Source: https://github.com/defo-project/apache-httpd Version: 2.5.0.ech.351 Comment: only shared-mode ECH
haproxy	Source: https://github.com/defo-project/haproxy Version: 3.2.dev0.62 Comment: both shared-mode and split-mode ECH
OpenSSL s_server	Source: https://github.com/defo-project/openssl Version: 3.5.0-dev Comment: shared-mode only but can easily “force” HelloRetryRequest (HRR)

III. Services

A. Test Services

We can separate services into test and operational services. Table III lists known test services supporting ECH. All of those below .ie and my-own.net were setup by the DEfO project.

TABLE III: Test Services with ECH

Name	Details
defo.ie	https://defo.ie/ech-check.php is often used to check ECH ECH keys are rotated hourly, private usable for 3 hours, ECHConfigList in DNS contains only latest public
draft-13.esni.defo.ie	different server technology (as listed in Table II) instances on different ports as listed at https://defo.ie/, e.g., https://draft-13.esni.defo.ie:10413 is served by nginx ECH keys are rotated hourly, private usable for 3 hours, ECHConfigList in DNS contains only latest public
test.defo.ie	hosts a number of ECH server setups with good and variously bad configurations - see https://test.defo.ie/iframes_tests which describes those and allows a browser to attempt connections to each via Iframes ECH keys/configs are static for these setups
foo.ie	https://foo.ie/ech-check.php was setup used to check the defo.ie setup was easily replicated ECH keys are rotated hourly, private usable for 3 hours, ECHConfigList in DNS contains only latest public
my-own.net	this was to test the impact of having the same ECH keys on port 443 (https://my-own.net/ech-check.php) and another port (https://my-own.net:8443/ech-check.php) - at one point that made a difference to browsers ECH keys are rotated hourly, private usable for 3 hours, ECHConfigList in DNS contains only latest public
tls-ech.dev	https://tls-ech.dev/ was setup by the boringssl developers as a test server that uses boringssl ECH keys/configs seem to be static for this setup
Cloudflare	https://cloudflare-ech.com/cdn-cgi/trace is a test page setup by cloudflare that reports on ECH success/failure apparently, the server implementation and infrastucrure are part of Cloudflare's normal setup a similar test service used to be available at https://crypto.cloudflare.com/cdn-cgi/trace but that was turned off around the time that ECH was re-enabled for Cloudflare customers

Continued on next page

TABLE III: Test Services with ECH (Continued)

Name	Details
	ECH keys are rotated hourly, private usable for N hours, ECHConfigList in DNS contains only latest public **TODO: Check what N for private key usability**
rfc5746.mywaifu.best	this ECH-enabled web page (https://rfc5746.mywaifu.best/) seems to have been setup as an ECH test site by someone using DEfO artefacts (nginx and documentation) but without any contact having happened between the person who set that up and any of the DEfO-project participants the person who set this up documented some of that at https://ckcr4lyf.github.io/tech-notes/services/nginx/nginx-ech.html

B. Operational Services

The only operational ECH service we know of is Cloudflare's deployment. However, that is non-negligible. Cloudflare earlier enabled ECH but disabled it soon after in October 2023 [1] as it caused some back-end issues. They then re-enabled ECH in October 2024. Our understanding of Cloudflare's deployment is that ECH is enabled by default for their "free" tier customers, but that paying customers have to take action to enable ECH.

In contrast, non-paying customers are not provided with a control to disable only ECH - they have to downgrade to TLSv1.2 in order to disable ECH. This is something that has been seen in recent weeks after the Russian government started blocking use of ECH to Cloudflare. [2]

C. Domain Probe Data

As part of our DEfO test setup, we have a web page where one can enter a host name and port and a script on our server will check if ECH is enabled for a web server at the name and port. (That's at https://test.defo.ie/domainechprobe.php.) Data is only stored if there is an HTTPS resource record for the name and port, if there is not we store nothing. In cases where this is an HTTPS resource record, we only store the name and port, whether the HTTPS value includes an "ech=" field, if there is, whether or not ECH worked using our ECH-enabled curl as

[1] https://community.cloudflare.com/t/early-hints-and-encrypted-client-hello-ech-are-currently-disabled-globally/567730

[2] https://betanews.com/2024/11/20/encrypted-client-hello-didnt-solve-censorship-but-still-may-have-a-role-to-play/

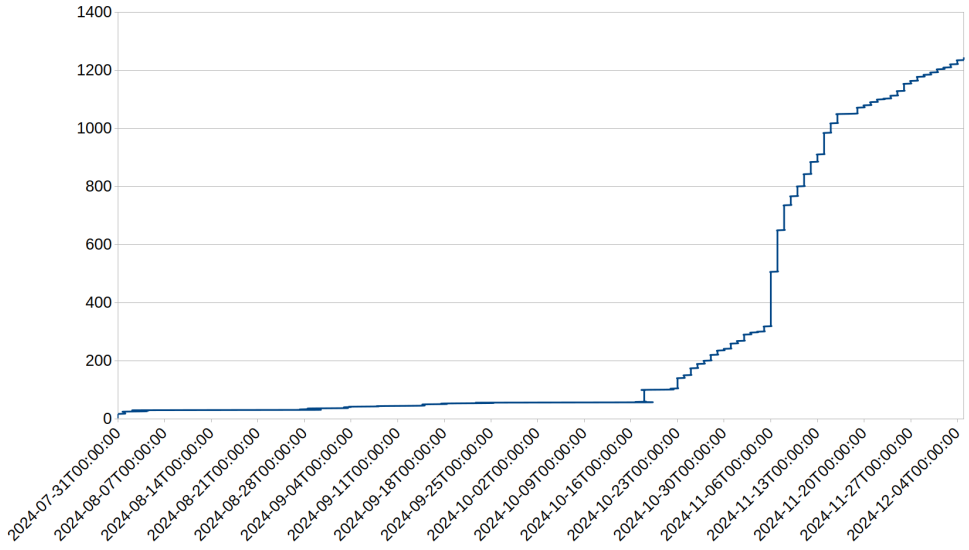

Fig. 1: Cumulative number of queries seen at https://test.defo.ie/domainechprobe.php versus time. The total number of queries is 1244.

the client. We also store the HTTPS resource record value, unless there are CNAMES or other kinds of re-direction involved.

We can use this data to get some further insight into services for which ECH is enabled. For now, we are not publishing the raw data - while data for the most recent 50 queries is shown on the web page, if we wanted to publish the raw data, we'd have to enable some form of consent for users, and it's not clear that'd be a) easy or b) worthwhile. (We currently take no steps to try record who made which requests to this page.)

Figure 1 shows the (small) numbers of queries made since August 2024. We can see a noticeable uptick in accesses just-before and after Russia blocked Cloudflare's ECH service. We also see some usage (at least 15 cases) that appear to indicate some web site owners may be using this service to check whether or not they have successfully disabled ECH. (The pattern is two quick accesses to the same name/port, the first of which indicates ECH worked, and the second that the HTTPS RR no longer contains an "ech=" value.)

The 1244 queries involved 517 unique names (117 of which are under the ".ru" ccTLD). The most commonly queried name is "youtube.com" (131 times), for which ECH is not currently enabled. Some 354 names were only queried once, and 86 were queried twice.

Of the 1244 queries, 660 referred to a name that had an "ech=" in the relevant HTTPS record,

referring to 298 unique names. Within the 660 HTTPS values that contain an "ech=" value, we see 276 unique ECHConfigList values (4 of which are corrupt values published for our DEfO test services). Of the 298 unique names, 271 seem to be served by Cloudflare (based on the relevant name server names), 15 relate to test services described in Table III. At the time of writing, 10 of the names result in an NXDOMAIN or SERVFAIL response, 1 seems to correspond to a possible hobbyist site and 2 seem to (now) be parked domains or part of some advertising campaign. None of of those last 3 currently publish an HTTPS record.

So at least for the names entered to our domain probe page, we can seemingly conclude that Cloudflare have the only operational service at present.

IV. Interoperability

Our primary interoperability testing is based on our "smokeping"-like tests. While smokeping measures latency for names/addreses, our tests aim to establish whether and which ECH configurations interoperate or fail to do so.

We mainly do this via a set of hourly cronjobs running on the test.defo.ie VM that attempt to access 67 URLs (some deliberately with broken configurations) from six different clients (described in Table IV. We report results for the most recent 6 runs at https://test.defo.ie/smokeping-summary.php.

TABLE IV: Smokeping clients

Name	Details
chromium	headless browser tests via selenium Version: 131.0.6778.85 Python script: https://github.com/defo-project/ech-dev-utils/blob/main/scripts/selenium_test.py ECH implementation based on boringssl
firefox	headless browser tests via selenium Version: 133.0 Python script: https://github.com/defo-project/ech-dev-utils/blob/main/scripts/selenium_test.py ECH implementation based on NSS

Continued on next page

TABLE IV: Smokeping clients (Continued)

Name	Details
curl	bash script using curl Version: 8.11.1-DEV Bash script: https://github.com/defo-project/ech-dev-utils/blob/main/scripts/smoke_ech_curl.sh curl currently only pays attention the the first HTTPS resource record seen in DNS answers ECH implementation based on OpenSSL
golang	custom golang programme and bash script Version: golang 1.23 Golang programme: https://github.com/defo-project/ech-dev-utils/blob/main/scripts/ech_url.go Bash script: https://github.com/defo-project/ech-dev-utils/blob/main/scripts/smoke_ech_go.sh ECH implementation developed for golang
rustls	custom rustls programme and bash script Version: rustls 0.23.19 Rustls programme: https://github.com/defo-project/ech-dev-utils/blob/main/scripts/ech_url.rs Bash script: https://github.com/defo-project/ech-dev-utils/blob/main/scripts/smoke_ech_rs.sh ECH implementation developed for rustls
python	custom Cpython build and python script Version: python 3.13 Python script: https://github.com/defo-project/ech-dev-utils/blob/main/scripts/ech_url.py Bash script: https://github.com/defo-project/ech-dev-utils/blob/main/scripts/smoke_ech_py.sh ECH implementation based on OpenSSL

We can characterise the different kinds of test URLs used according to the server technology

used (basically one of the servers listed in Table II).

V. Conclusions

Conclude

Acknowledgements

Thanks to DEfO folks, OTF...

References

[1] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[2] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "TLS Encrypted Client Hello," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-esni-22, Sep. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-esni/22/