Transport Layer Security
Internet-Draft
Updates: draft-ietf-tls-esni-23 (if approved)
Intended status: Standards Track
Expires: 30 August 2025

N. Sullivan
Cryptography Consulting LLC

26 February 2025

Implicit ECH Configuration for TLS 1.3
draft-sullivan-tls-implicit-ech-00

Abstract

This document updates the TLS Encrypted ClientHello (ECH) specification [ECH-DRAFT] to support an implicit mode in ECH signaled by a new implicit_ech extension in ECHConfigContents. Clients that detect this extension override certain base ECH rules:

* They MAY choose any outer SNI instead of public_name.

* They MAY choose any value for the config_id without an application profile or being externally configured.

* They MAY use another value than ECHConfig.contents.public_name in the "server_name" extension (rather than they SHOULD use it)

Client-facing servers that include implicit_ech in the ECHConfig MUST accommodate flexible config_id usage as defined in Section 10.4. of [ECH-DRAFT]. This approach enables the removal of stable identifiers (fixed config ID and known public_name) that on-path adversaries can use to fingerprint a connection.

This improves upon the "Do Not Stick Out" design goal from Section 10.10.4 of [ECH-DRAFT] by allowing clients to choose unpredictable identifiers on the wire in the scenario where the set of ECH configurations the client encounters is small and therefore popular public_name or config_id values "stick out".

Note that this increases CPU usage in multi-key deployments because client-facing servers must perform uniform trial decryption to handle arbitrary config_id values.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Transport Layer Security mailing list (tls@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tls/.

Source for this draft and an issue tracker can be found at https://github.com/grittygrease/draft-sullivan-tls-implicit-ech.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Encrypted ClientHello (ECH) protocol [ECH-DRAFT] is designed to hide sensitive TLS handshake parameters, including the real SNI, from passive observers. In the base ECH model, the client sets its outer SNI to the public_name and config_id from the ECHConfig. Both of these can become stable fingerprints that on-path adversaries recognize.

In implicit mode, the client MAY:

1. Select any outer SNI (rather than the public_name).

2. Select any config_id instead of taking it from the ECH configuration (without an application profile or extenal configuration agreement).

Client-facing servers that publish or accept implicit ECH configurations must adjust key selection (e.g., single-key usage, uniform trial decryption), removing reliance on stable config IDs or well-known public_name values. This design helps conceal ECH usage from on-path adversaries, though deployments may see increased CPU usage.

This proposal also addresses a timing side-channel in GREASE vs. real ECH, by requiring client-facing servers supporting implicit ECH always to perform trial decryption as defined in Section 10.4. of [ECH-DRAFT] — ensuring consistent behavior regardless of ECH key validity.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Implicit ECH Extension

3.1. Extension Definition and Semantics

A new ECHConfig extension type is defined to indicate implicit mode. If this extension is present in ECHConfigContents.extensions, clients and client-facing servers follow the rules described here, overriding certain parts of the base ECH specification.

The extension has the following format:

```
enum { implicit_ech(TBD), (65535) } ECHConfigExtensionType;

struct { // No data; presence indicates "implicit" usage } ImplicitECHConfig;
```

The extension_data is zero-length. The presence of this extension in the ECHConfig signals to the client that the client-facing server is configured for implicit ECH and follows the requirements of this document.

3.2. Overridden Rules in the Base ECH Specification

When the implicit_ech extension is found in ECHConfigContents.extensions, the following rules in [ECH-DRAFT] are overridden:

* The requirements for choosing the config_id in the ClientHello (Section 6.1.1. of [ECH-DRAFT]). In implicit mode, the client MAY choose any value for the config_id.

* Outer SNI usage (Section 6.1 of [ECH-DRAFT] says the client SHOULD set the value of the "server_name" extension to ECHConfig.contents.public_name. In implicit mode, the client MAY choose any valid domain name for the outer SNI.

Note that the validation rules in Section 6.1.7 of [ECH-DRAFT] still apply and the client is still expected to validate that the certificate is valid for ECHConfig.contents.public_name (not the "server_name" chosen by the client) when the client-facing server rejects ECH.

4. Client Behavior

If the client sees the implicit_ech extension in an ECHConfig:

* It MAY select any valid DNS name for the "server_name" extension, ignoring public_name.

* It MAY produce a random or arbitrary config_id, rather than using ECHConfigContents.key_config.config_id

Other aspects of the base ECH spec remain unchanged. In particular, the client still picks a cipher suite from key_config.cipher_suites, produces a valid HPKE ephemeral key, and encrypts ClientHelloInner into the payload field.

If the client-facing server issues an ECH retry hint (for example, in EncryptedExtensions), the client MUST still confirm that the server certificate is valid for the public_name from the ECHConfig used to establish the connection. Note that this may be a different name than the one sent in the outer SNI.

As described in Section 6.1.1 of [ECH-DRAFT], in the event of HRR, the config_id MUST be left unchanged for the second ClientHelloOuter.

5. Client-Facing Server Behavior

A client-facing server that supports Implicit ECH on an IP address shared with non-ECH services or GREASE ECH clients MUST attempt to decrypt the encrypted ClientHello for every incoming connection that presents an ECH extension. This requirement applies even if the config_id is not recognized, so GREASE and valid ECH connections appear indistinguishable from a timing perspective.

If the decryption attempt succeeds, the server proceeds with the handshake using the inner ClientHello and the appropriate certificate chain for the actual (inner) SNI. If the decryption attempt fails, there are two possibilities:

1. The client was connecting to a domain that does not support ECH

2. The client used a different ECHConfig than those currently supported on the client-facing server.

After trial decryption, if the server recognizes the outer SNI, has a certificate that covers it, and supports non-ECH connections for this domain, then the server proceeds with a standard TLS handshake based on the indicated SNI. Otherwise, the server MAY send an ECH retry hint in the ServerHello, accompanied by:

1. A newly issued or updated ECHConfig, possibly including the implicit flag again.

2. A server certificate that is valid for the public_name in one of the supported ECH configurations, ensuring the client can verify it.

If multiple ECH keys are in rotation, perform uniform trial decryption to avoid timing signals that reveal actual vs. unknown config_id usage. The server SHOULD attempt ECH decryption first to avoid revealing whether the config_id was recognized.

In this model, trial decryption on every connection ensures that GREASE and real ECH connections are handled uniformly, preventing timing side-channels.

If the client's ClientHello does not contain an ECH extension, the server proceeds with a standard TLS handshake based on the indicated SNI. This fallback behavior should remain unchanged from existing TLS handling. The presence or absence of ECH extension data in the ClientHello is the primary trigger for the server's ECH logic.

6. Deployment Considerations

Implicit config_id usage may require additional CPU overhead from trial decryption for GREASE ECH handshakes. A single-key environment simplifies ignoring the config_id and yields more uniform performance.

Supporting implicit ECH configurations limits the number of different ECH keys supported by a server on the same IP address since the outer SNI and config_id can no longer be used to choose the appropriate ECH configuration.

7. Security Considerations

7.1. Timing Side-Channels from Unknown IDs

In standard ECH, the server might quickly reject unknown hashed IDs. Implicit ECH requires the server to attempt uniform decryption for IDs, reducing the ECH vs. ECH GREASE timing gap.

7.2. Hiding the Known public_name

By not placing public_name in the actual outer SNI, on-path adversaries cannot block a known name. The client uses public_name only to authenticate ECH retry hints, so an active attacker cannot degrade ECH without a valid certificate.

7.3. CPU Overhead

Randomized config_id and outer SNI usage can lead to increased CPU usage from trial decryption. This cost grows if more than one ECH keys are in use on the same server. Operators should consider minimizing the number of active keys to mitigate this cost.

8. IANA Considerations

This document requests that IANA add the following entry to the "ECHConfig Extension" registry:

* Value: TBD (suggested code point for implicit_ech)

* Extension Name: implicit_ech

* Recommended: Yes

* Reference: This document

* Notes: If present, the ECHConfig is "implicit," enabling ephemeral config_id usage and flexible outer SNI.

9. Normative References

[ECH-DRAFT] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-23, 19 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-23>.

[I-D.draft-ietf-tls-esni-23] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-23, 19 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-23>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

Acknowledgments

Marwan Fayed and Chris Patton provided ideas and contributions to this draft.

Author’s Address

Nick Sullivan
Cryptography Consulting LLC
Email: nicholas.sullivan+ietf@gmail.com