TLS Extension for EchoPulse Parameter Negotiation in the ClientHello / ServerHello

Abstract

This document specifies a new TLS 1.3 extension, `echopulse_parameters`, to facilitate the negotiation and communication of EchoPulse-specific parameters during the handshake. This extension allows clients and servers to agree on the specific variant of the EchoPulse KEM to be used, ensuring deterministic graph mutation and consistent key encapsulation/decapsulation across the communication peers. It defines the structure of the extension data and outlines the synchronization model for the EchoPulse graph state.

Status of This Memo

Copyright Notice

Table of Contents

1. Extension Overview

The EchoPulse Key Encapsulation Mechanism (KEM) [ECHO] relies on a publicly known symbolic graph that deterministically evolves over time through a mutation function $\mu(G,t)$. For successful key exchange in TLS 1.3, both client and server must agree on and synchronize their understanding of the current state of this graph.

This document defines the `echopulse_parameters` extension, identified by a new `ExtensionType` codepoint. This extension is sent in the ClientHello to propose EchoPulse configuration options and may be echoed in the ServerHello to indicate the server's chosen parameters.

The extension carries parameters essential for identifying the base graph, the mutation schedule, the symbolic path length, and the preferred hash function used internally by EchoPulse.

2. Use in TLS 1.3

The `echopulse_parameters` extension MAY be included in the ClientHello message if a `NamedGroup` corresponding to EchoPulse (e.g., `DRAFT_ECHOPULSE_KEM` or a hybrid variant as defined in [TLS-ECHO]) is offered in the `key_share` extension.

If the server selects an EchoPulse-related `NamedGroup` from the client's `key_share` extension, it MUST include an `echopulse_parameters` extension in its ServerHello. This ServerHello extension will echo the specific parameters chosen by the server from the client's proposals. If the client did not send an `echopulse_parameters` extension, but offered an EchoPulse KEM and the server chose it, the server MUST NOT send an `echopulse_parameters` extension. In such a case, default EchoPulse parameters (as defined by the `NamedGroup` specification) are used.

If a client offers an `echopulse_parameters` extension but the server does not select an EchoPulse-related `NamedGroup`, the server MUST NOT send an `echopulse_parameters` extension.

The extension type `echopulse_parameters` is requested from IANA (provisional value: `0xFE01`).

3. Extension Format

The `ExtensionData` for the `echopulse_parameters` extension contains a sequence of fields, defined using a length-prefixed structure. All multi-byte fields are encoded in network byte order (big-endian).

```
struct {
uint16 graph_id;
uint8 mutation_schedule_id;
uint8 symbol_path_length;
uint8 hash_mode;
opaque reserved_future_use<0..255>; // For future extensions,
// padded to a fixed length
} EchoPulseParameters;
```

* `graph_id`: A 16-bit identifier for the base symbolic graph
  (G_0) that both client and server are expected to use. This
  allows for multiple pre-agreed or pre-loaded base graphs on a
  device. A `graph_id` of `0x0000` MAY indicate a default or
  dynamically generated graph if specified by the `NamedGroup`.

* `mutation_schedule_id`: An 8-bit identifier for the specific
  sequence of mutation rules or algorithm parameters applied by
  $\mu(G,t)$. This allows different evolution patterns for the
  graph over time. A `mutation_schedule_id` of `0x00` MAY indicate
  a default or standard schedule.

* `symbol_path_length`: An 8-bit unsigned integer indicating the
  length, in bytes, of the symbolic path r (the encapsulated
  random payload) for the EchoPulse KEM. This corresponds to L_K
  in the KEM context. For example, `0x20` (32) for a 256-bit
  output key.

* `hash_mode`: An 8-bit identifier specifying the internal hash
  function used by EchoPulse for key derivation ($H(v_{enc} || r)$).
  This aligns with the `H_adapter` concept in [ECHO_HASH].
  Registry for `hash_mode` values (provisional):
  * `0x00`: Reserved
  * `0x01`: SHA3-256
  * `0x02`: BLAKE2s (256-bit output)
  * `0x03`: SHAKE128 (256-bit output)
  * `0x04`: Prehash + AES (AES-128-GCM-PRF-like)
  * `0xFF`: Reserved for GREASE / future use.

* `reserved_future_use`: An opaque field used for future extensions

   or for padding to a fixed total extension length for GREASE
   compatibility (see Section 5). The length is determined by the
   `Extension.length` field of the TLS 1.3 `Extension` structure.
   When sent by the client, it MAY contain random bytes or be zero-padded.
   When echoed by the server, it MUST contain the same length as the
   client's `reserved_future_use` field, and its content MAY be
   zero-padded or mirror the client's content.

Total maximum length of `EchoPulseParameters` in ClientHello is
recommended to be 24-32 bytes for typical use cases, including
padding for GREASE.

4. Synchronization Semantics (Synchronization Model Architect)

Deterministic graph mutation requires both parties to derive an
identical graph state G_t for the KEM operation. The time index
t for $\mu(G, t)$ is derived from shared, verifiable parameters
within the TLS 1.3 handshake.

Shared State Derivation for Time Index (t):

The time index t is derived using the `graph_id`, the negotiated
TLS session's properties, and cryptographic binding to the handshake
transcript.

$t = \text{HKDF-Expand-Label}(\text{Handshake Secret}, \text{"echopulse time"}, \text{Transcript-Hash}, \text{time_index_length})$

Where:
* `Handshake Secret`: The Handshake Secret from the TLS 1.3
   KeySchedule [RFC8446, Section 7.1].
* `"echopulse time"`: A fixed label for HKDF expansion.
* `Transcript-Hash`: `Hash(ClientHello...ServerHelloDone)`
   (or equivalent up to point of KEM calculation) using the hash
   function agreed in the TLS 1.3 cipher suite (e.g., SHA256).
* `time_index_length`: A fixed length (e.g., 32 bits, 4 bytes)
   determining the entropy and range of t.

The derived t value ensures:
1. **Determinism:** Both client and server, having the same `Handshake Secret` and `Transcript-Hash`, will derive the same t.
2. **Liveness:** Each handshake produces a unique t due to the unique `Handshake Secret` and `Transcript-Hash`, preventing reuse of an identical G_t across sessions.
3. **Authentication:** The use of `Handshake Secret` binds t to the authenticated handshake transcript.

Protocol for Fallback or Invalid Parameter Handling:

* **Client Proposal:** The ClientHello MAY contain multiple `echopulse_parameters` extensions, each with a different set of parameters, possibly as part of a GREASE-like strategy (see Section 5). However, for negotiation, a client typically sends one preferred `echopulse_parameters` instance.
* **Server Selection:** The server processes the client's `echopulse_parameters` extension.
    * If the server supports the requested `graph_id`, `mutation_schedule_id`, `symbol_path_length`, and `hash_mode`, it MUST echo the exact same parameters in its ServerHello's `echopulse_parameters` extension.
    * If the server does not support any of the proposed parameters, it MUST NOT send an `echopulse_parameters` extension. If an EchoPulse `NamedGroup` was offered, the server MUST then select a different `NamedGroup` (e.g., a classical KEM or a different PQC KEM) from the client's `key_share` extension, or abort the handshake with an `unsupported_extension` or `illegal_parameter` alert.
* **Client Verification:** Upon receiving the ServerHello, the client MUST verify that the `echopulse_parameters` extension (if present) exactly matches its proposed parameters for the selected EchoPulse KEM. Any mismatch MUST result in a `handshake_failure` alert.

5. Compatibility and GREASE Handling (Wire Format & Compatibility Engineer)

5.1. Byte Layout (Length-Prefixed Fields):

The TLS `Extension` structure uses a `uint16` length for `ExtensionData`. The `EchoPulseParameters` structure within this `ExtensionData` is defined with fixed-size fields and a variable-length `reserved_future_use` field, ensuring predictable parsing.

```
struct {
ExtensionType extension_type; // echopulse_parameters (0xFE01)
uint16 length; // Total length of EchoPulseParameters struct
// EchoPulseParameters
uint16 graph_id;
uint8 mutation_schedule_id;
uint8 symbol_path_length;
uint8 hash_mode;
opaque reserved_future_use[length - 6]; // Calculated length, 6 bytes are fixed fields
} Extension;
```

The `reserved_future_use` field's length is `length - 6` bytes, ensuring the total `EchoPulseParameters` structure adheres to the `Extension.length`.

5.2. Total Maximum Length:

The maximum recommended `length` for the `EchoPulseParameters` extension in ClientHello is 256 bytes. This allows for sufficient `reserved_future_use` padding without excessively increasing ClientHello size. For typical operations, the fixed fields (6 bytes) are sufficient.

5.3. GREASE-Compatibility Considerations:

To facilitate the deployment of new extensions and prevent ossification of the TLS ecosystem, the `echopulse_parameters` extension SHOULD be GREASE-compatible.

* **Client GREASE:** Clients MAY send `echopulse_parameters`

   extensions with randomly chosen, non-standard `graph_id`,
   `mutation_schedule_id`, `symbol_path_length`, `hash_mode`, and/or
   `reserved_future_use` values. Servers MUST ignore such
   extensions if the selected `NamedGroup` is not an EchoPulse
   KEM. If an EchoPulse KEM is selected, servers should still
   follow the negotiation rules (Section 4).
* **Server GREASE:** Servers MAY include a `reserved_future_use` field
   of arbitrary (but supported) length in their `echopulse_parameters`
   response, provided it does not alter the functional parameters.
   The contents of this field can be randomized.

5.4. Reserved Future-Use Fields:

The `reserved_future_use` field provides a mechanism for future
expansion of EchoPulse parameters without requiring a new extension
codepoint. New fields can be defined by extending the length of this
field and specifying their structure. Implementations MUST ignore
unknown content in this field, processing only the fields explicitly
defined in this version.

6. Security and Privacy Considerations

* **Parameter Exposure:** The parameters carried in this extension
   are publicly exchanged. They MUST NOT contain any sensitive
   or secret information that could compromise the KEM or user
   privacy.
* **Consistency Assurance:** The deterministic nature of EchoPulse
   requires strict consistency of parameters. Any deviation in
   `graph_id`, `mutation_schedule_id`, `symbol_path_length`, or
   `hash_mode` between client and server, or an incorrect derivation
   of the time index t, will lead to key derivation failure.
* **Side-Channel Implications:** The values exchanged in this
   extension could influence the execution path or memory access
   patterns of the EchoPulse KEM implementation. Implementations
   MUST ensure that processing these parameters and deriving the
   graph state G_t remains constant-time, preventing leakage
   of information about the chosen parameters or other secrets.

  This is particularly relevant for `symbol_path_length` influencing
  loop iterations.
* **Denial-of-Service:** A malicious client could send an
  `echopulse_parameters` extension requesting computationally
  expensive or resource-intensive EchoPulse parameters. Servers
  SHOULD apply limits on the complexity of accepted parameters
  to mitigate DoS risks.

7. IANA Considerations

IANA is requested to add the following entry to the "TLS ExtensionType" registry:

Value: TBD (Provisional: `0xFE01`)
TLS Name: `echopulse_parameters`
Recommended: Yes
Reference: This document
DTLS-OK: Y
TLS-1.3-OK: Y

IANA is also requested to create a new sub-registry named "TLS EchoPulse Hash Modes" within the "Transport Layer Security (TLS) Parameters" registry. This registry will use an 8-bit length and be governed by RFC Required specification.

Value	Description	Reference
0x00	Reserved	This document
0x01	SHA3-256	This document
0x02	BLAKE2s (256-bit output)	This document
0x03	SHAKE128 (256-bit output)	This document
0x04	Prehash + AES (AES-128-GCM-PRF-like)	This document
0xFF	Reserved for GREASE	This document

8. References

8.1. Normative References

[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/info/rfc8446>.

8.2. Informative References

[ECHO]   EchoPulse Project Documentation: "Security Properties of
the Graph Mutation Function $\mu(G, t)$ in the EchoPulse Protocol".
(Editor's Note: Placeholder for a more formal EchoPulse KEM spec.)

[ECHO_HASH] EchoPulse Project Documentation: "Constant-Time BLAKE2s
Implementation for EchoPulse: Embedded-Compatible
Cryptographic Hash Kernel".

[TLS-ECHO] EchoPulse Project Documentation:
"TLS_ECHOPULSE_WITH_AES_128_GCM_SHA256:
A Post-Quantum Key Encapsulation CipherSuite Based on EchoPulse".