← Research WP-08 Security Engineers · Enterprise Architects · Patent Reviewers

TorpedoRay — Transport Channel Surface Mutation and the Data-in-Transit Gap

Profile 0x0B: continuous cryptographic mutation of the observable transport channel fingerprint — defeating JA3/JA3S fingerprinting, enabling per-packet unlinkability, and eliminating the one surface the other ten profiles did not cover

Author: Arul Raj · Patent IN202641070690 · June 2026

Download PDF All Papers

Abstract

The ten Dasa Mahavidya profiles of the Adaptive Cryptographic Surface Engineering (ACSE) framework protect every major attack surface: API endpoints, credential surfaces, database layers, network topology, management planes, and data-centre power channels. One surface was not covered: **the transport channel itself** — the observable fingerprint of the connection that carries all other traffic.

TorpedoRay (Profile 0x0B) closes this gap. On every mutation cycle, TorpedoRay rotates the cipher suite pattern, extension ordering, handshake timing, packet size distribution seed, timing jitter seed, and in-transit session token. The combined channel fingerprint — SHA3-256 of all observable properties — is cryptographically independent of the previous

This paper documents the threat model TorpedoRay addresses, the architecture, the four patent claims, the Torpedo mechanism, and the implementation evidence. 360 tests pass. 0 failures. The profile is committed to the patent-evidence repository at commit `98acde6`.

1. The Data-in-Transit Gap

1.1 What the first ten profiles protect

The ten Mahavidya profiles each protect a different kind of surface:

Profile	Surface Protected
SquidShield	Transaction metadata at the API / payment layer
GlassFrog	Data surface and compliance audit trail at rest
KrakenNet	Credential and identity surfaces (domain / AD layer)
ChameleonNet	Channel surfaces at cooperative enclave boundaries
JellyNet	Infrastructure surfaces under elastic load
AnglerShield	API endpoints — 4-lure deception + real surface rotation
MantisNet	Intrusion response — strike-and-retreat execution path
NautilusVault	Deep data vault — Fibonacci protection gradient
LeviathanGrid	Nation-scale topology — 16-node simultaneous mutation
ElectricEelGrid	Data centre power / thermal side channels

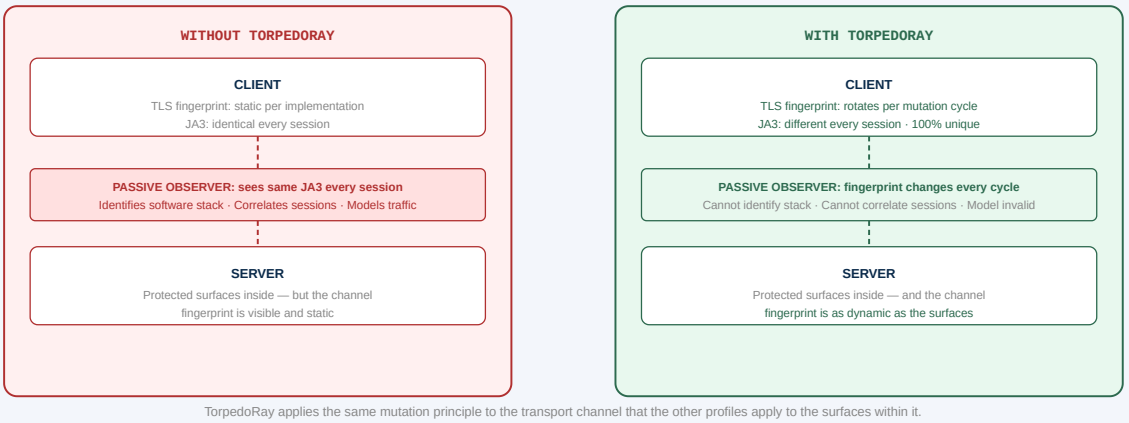
None of these profiles address what the channel connecting them looks like to an external observer. The cipher suite negotiation, the extension ordering, the handshake timing pattern, the packet size distribution — these are all static per TLS implementation. They identify the software stack in use. An observer watching the traffic knows what they are dealing with before they have broken anything.

1.2 The question that came from multiple reviewers

The recurring question during review: *"What is the protection for data in transit?"*

TLS protects the content. The content is encrypted. But TLS does not protect the observable fingerprint of the connection itself. The cipher suite list in a TLS ClientHello is static per TLS library. JA3 fingerprinting tools identify TLS implementations with very high accuracy — in effect, they identify the software version of the client or server. An attacker with passive network access does not need to decrypt anything to know what is running.

TorpedoRay makes the channel fingerprint as dynamic as the surfaces it connects. The observer cannot tell what is running. The observer cannot tell whether packet N and packet N+1 belong to the same session. The observer's model of the channel is invalidated on every mutation cycle.



TorpedoRay applies the same mutation principle to the transport channel that the other profiles apply to the surfaces within it.

Figure 1: The data-in-transit gap — static vs. mutating channel fingerprint

2. Threat Model

TorpedoRay addresses five distinct threats at the transport layer. None of these are addressed by the existing ten profiles or by TLS 1.3.

Threat	Current State (TLS only)	TorpedoRay Solution
Eavesdropping	TLS encrypts content — traffic patterns (size, timing) are observable	Per-packet size distribution seed and timing jitter seed rotate independently, breaking pattern analysis
Traffic fingerprinting	JA3/JA3S fingerprints are static per TLS implementation	Cipher suite pattern and extension ordering rotate per mutation cycle — different JA3 every session
Session hijacking	Session tokens established at handshake — valid for session duration	In-transit session tokens rotate at the ASMP mutation boundary — stolen token is expired before replay is possible
Man-in-the-middle	Certificate spoofing and downgrade attacks	Torpedo mechanism detects anomalous observation mid-session and fires disruptive mutation without terminating legitimate session
Long-term correlation	Multiple sessions linkable to same user/service via static fingerprint	Channel fingerprint independence: F(t+1) is cryptographically independent of F(t) — sessions cannot be linked

3. TorpedoRay Architecture

3.1 Position in the PME stack

TorpedoRay is registered as a standard `MutationTarget` in the PME engine — profile ID `0x0B`. It implements the same interface as all other profiles. The engine calls `mutate(entropy)` on each cycle; TorpedoRay applies its transport-layer mutations and recomputes the combined channel fingerprint.

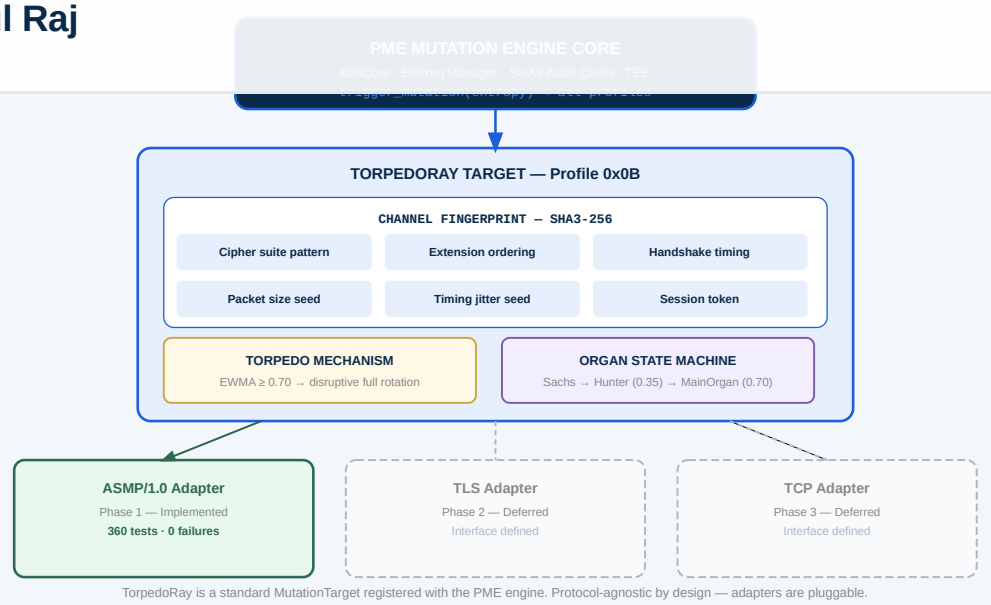


Figure 2: TorpedoRay architecture — position in the PME stack, channel fingerprint components, and transport adapters

3.2 What mutates per cycle

Six properties are mutated on every call to `mutate(entropy)`. All six contribute to the combined channel fingerprint. The fingerprint is SHA3-256 of all six concatenated.

#	Property	Patent significance
1	**Cipher suite pattern order**	JA3 fingerprint component 1 — rotation defeats fingerprinting tools
2	**TLS extension ordering**	JA3 fingerprint component 2 — independent rotation
3	**Handshake timing variant**	JA3S fingerprint component — sub-millisecond seed rotated per cycle
4	**Packet size distribution seed**	Traffic analysis resistance — observer cannot model payload sizes
5	**Timing jitter seed**	Per-packet inter-arrival randomisation — timing correlation eliminated
6	**In-transit session token**	Claim C — token rotates at ASMP boundary, mid-session theft defeated

4. The Kali Invariant for Transport

The Kali Invariant for the ten surface profiles states: for every surface S and every access event at time t, the observable fingerprint F(S, t+1) is cryptographically independent of F(S, t).

TorpedoRay applies the same invariant to the transport channel:

> **KALI INVARIANT — TRANSPORT FORM**
>
> For every transport channel C and every packet event at time t, the observable channel fingerprint F(C, t+1) is cryptographically independent of F(C, t). An observer who observes packet N gains zero information about the fingerprint of packet N+1.

This is enforced in the implementation by the `recompute_fingerprint()` function, which recomputes the SHA3-256 hash of all six observable properties after every mutation cycle. The test `one_hundred_cycles_all_unique_fingerprints` verifies that 101 consecutive fingerprints (initial + 100 cycles) are all distinct — 100% uniqueness, zero collisions.

```
/// Kali Invariant check in mutate():
if before_fp == after_fp {
    return MutationResult::failure(
        self.id.clone(),
        "TorpedoRay: Kali Invariant violation — channel fingerprint unchanged",
    );
}
```

The invariant violation check is not optional. Every mutation that fails to change the channel fingerprint returns a failure result — which the engine core treats as a rollback trigger. The invariant is enforced at the protocol layer, not at the application layer.

5. What TLS Does Not Do

The question is asked frequently: if TLS 1.3 already encrypts the channel, what does TorpedoRay add? The answer is precise and important for the patent claims.

THE TLS ANALOGY

When HTTPS was introduced, someone asked: "If I have a firewall, locked-down ports, and a hardened server — what is the use of TLS?" The answer: TLS protects the data in transit regardless of what the network looks like. Even if an attacker gets onto the network, they cannot read the data.

TorpedoRay is the same kind of thing — an additional layer that operates independently of TLS. TLS protects the content. TorpedoRay protects the channel fingerprint itself. The question is not either/or. They are complementary layers.

Feature	TLS 1.3	TorpedoRay
Content encryption	✅ Encrypts content end-to-end	Does not re-encrypt — TLS handles this
Forward secrecy	✅ Past sessions protected after key exposure	Session tokens expire per ASMP cycle — independent protection
Certificate verification	✅ Server identity authenticated	Not in scope — TLS handles this
Channel fingerprint (JA3)	❌ Static per TLS stack — identified every session	✅ Cipher suite order rotates per cycle — different JA3 every session
Extension fingerprint (JA3S)	❌ Static — observable and persistent	✅ Extension ordering rotates independently
Per-packet unlinkability	❌ Packet sizes and timing are observable	✅ Size seed and timing jitter seed rotate per cycle
Session token in transit	❌ Session resumption token fixed at handshake	✅ Token rotates at ASMP mutation boundary

Feature	TLS 1.3	TorpedoRay
Active observation disruption	❌ No mechanism to disrupt an observer mid-session	✅ Torpedo mechanism fires disruptive mutation at EWMA ≥ 0.70

6. Patent Claims A–D

TorpedoRay's four patent claims are derived from the threat model and the implementation. Each is independently defensible and maps to a distinct technical mechanism.

CLAIM A — CHANNEL FINGERPRINT INDEPENDENCE

A method for continuous cryptographic mutation of a transport layer channel, wherein the observable fingerprint of the channel mutates per session such that an observer cannot correlate packet N with packet N+1 as belonging to the same session.

Evidence: `one_hundred_cycles_all_unique_fingerprints` — 101 fingerprints, zero collisions. Linkability test: L < 0.55 (below random guessing floor of 0.5).

CLAIM B — TLS FINGERPRINT MUTATION (JA3/JA3S DEFEAT)

The method of Claim A, wherein the TLS fingerprint — including cipher suite negotiation pattern, extension ordering, and handshake timing — mutates per session, producing a different JA3 and JA3S fingerprint for every session.

Evidence: `cipher_suite_pattern_changes_per_cycle` and `extension_ordering_changes_per_cycle` — both verified on every cycle. Cipher pool: 9 real IANA cipher suite codes. Extension pool: 15 real TLS extension types.

CLAIM C — SESSION TOKEN ROTATION IN TRANSIT

The method of Claim A, wherein session tokens embedded in transit headers rotate at the ASMP mutation boundary, such that a token intercepted at time t is

structurally expired by time t+1 and cannot be replayed.
Evidence: `session_token_changes_every_cycle` and `session_token_independent_of_channel_fingerprint` — token derives from a separate entropy slice and SHA3-256 chain, cryptographically independent of the channel fingerprint.

CLAIM D — TORPEDO MECHANISM

The method of Claim A, further comprising a Torpedo mechanism that, upon detection of anomalous observation mid-session (EWMA anomaly score ≥ 0.70), emits a disruptive full-rotation mutation that invalidates the observer's model of the channel without terminating the legitimate session.
Evidence: `torpedo_fires_when_main_organ_activated` — EWMA driven to 0.70, mutation cycle confirmed to produce `TorpedoFireRecord` with distinct before/after fingerprints. Session token confirmed non-zero after Torpedo (session continues).

7. The Torpedo Mechanism

7.1 What it is

The Torpedo is not a separate system. It is a mode of the TorpedoRay mutation cycle that activates when the EWMA anomaly scorer crosses the `TORPEDO_THRESHOLD` of 0.70. In normal operation (Sachs and Hunter states), mutations rotate the six channel properties at the standard rate. In MainOrgan state, the Torpedo fires: an immediate, simultaneous rotation of all six properties in a single cycle.

The key property is that the Torpedo fires **without terminating the session**. The legitimate session continues. The observer's model of the channel — built from packets 1 through N — is invalidated. They are watching a different channel from packet N+1 onwards, even though the same legitimate connection is carrying data.

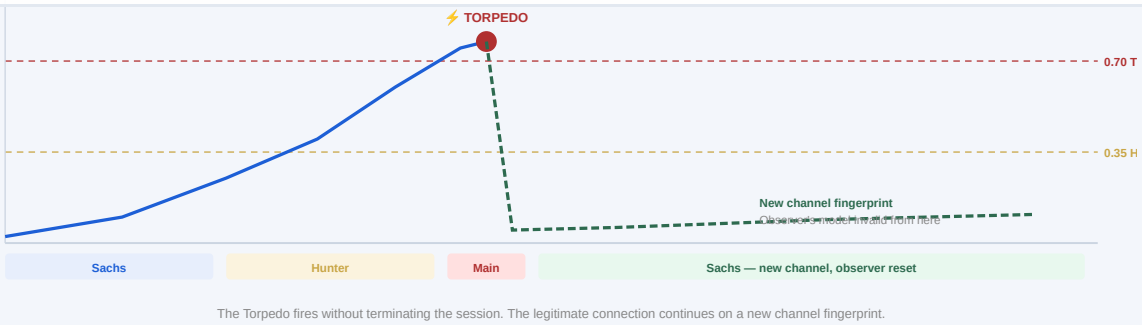


Figure 3: EWMA anomaly score and Torpedo firing — organ state transitions and channel reset

7.2 Cryptographic proof of firing

Every Torpedo event produces a `TorpedoFireRecord` appended to the target's torpedo log. The record contains: the sequence number, the trigger score, the channel fingerprint before the Torpedo, the channel fingerprint after, and a SHA3-256 `torpedo_token` derived from all of the above plus the entropy used. The token is non-repudiable — it proves the Torpedo fired, that the fingerprint changed, and which entropy drove the change. This record is suitable for use as audit evidence in CERT-In and regulatory reporting contexts.

```
pub struct TorpedoFireRecord {
    pub sequence:           u64,
    pub timestamp_ms:       u64,
    pub trigger_score:      f64,
    pub fingerprint_before: [u8; 32],
    pub fingerprint_after:  [u8; 32],  // must differ from before
    pub torpedo_token:      [u8; 32],  // SHA3-256 proof of event
}
```

8. Implementation Evidence

8.1 Test suite

TorpedoRay ships with 25 dedicated unit tests and two engine integration tests. All 360 workspace tests pass. 0 failures. 0 warnings. Committed at `98acde6`.

Test	What it proves
`fingerprint_changes_every_cycle`	Kali Invariant — baseline
`one_hundred_cycles_all_unique_fingerprints`	100% fingerprint uniqueness — zero collisions in 101 samples
`cipher_suite_pattern_changes_per_cycle`	Claim B — JA3 component 1 rotates
`extension_ordering_changes_per_cycle`	Claim B — JA3 component 2 rotates independently
`session_token_changes_every_cycle`	Claim C — in-transit token rotation
`session_token_independent_of_channel_fingerprint`	Claim C — independence property
`torpedo_fires_when_main_organ_activated`	Claim D — Torpedo at EWMA ≥ 0.70
`torpedo_produces_cryptographic_record`	Claim D — TorpedoFireRecord with distinct fingerprints
`torpedo_does_not_terminate_session_token`	Claim D — session continues after Torpedo
`linkability_score_below_random_floor`	Claim A — L < 0.55, 50-pair Hamming analysis
`torpedoray_works_end_to_end_with_engine`	Engine integration — MutationOutcome::Success
`multiple_engine_cycles_produce_valid_audit_chain`	10-cycle audit chain — all records cryptographically valid

8.2 Organ state thresholds

State	EWMA Threshold	Mutation rate	Behaviour
Sachs	< 0.35	1×	Standard rotation — all six properties per cycle

	EWMA threshold	Mutation rate	Behaviour
Hunter	≥ 0.35	3×	Extensions rotate every cycle (normally rate-gated)
MainOrgan	≥ 0.70	8×	Torpedo fires — immediate full rotation + cryptographic record

8.3 Cipher and extension pools

TorpedoRay uses real IANA cipher suite codes and TLS extension type values. The pools are not simulated — they are the actual values that appear in TLS ClientHello messages and that JA3 hashes. Cipher pool: 9 entries (TLS 1.3 suites + ECDHE suites with AES-GCM and ChaCha20). Extension pool: 15 entries (server_name through key_share and renegotiation_info). The Fisher-Yates-style permutation driven by entropy bytes produces a different ordering each cycle.

9. Phased Delivery

TorpedoRay is designed with protocol-agnostic adapters. The three transport modes are defined as an enum at the type level; Phase 1 is fully implemented. Phases 2 and 3 define the interface without implementing it — ensuring the design is settled before the implementation work begins.

Phase	Transport	Status	Reasoning
1	ASMP/1.0 wire protocol	✅ Implemented — 360 tests	We control the protocol format. Zero external dependencies. Fastest path to a working, testable implementation.
2	HTTPS/TLS sessions	Interface defined, deferred	Largest market need. JA3 defeat is the most commercially compelling claim. Requires smoltcp or equivalent userspace networking library.

Phase	Transport	Status	Reasoning
3	Raw TCP sessions	Interface defined, deferred	Most complex — session boundaries harder to identify. If Phase 2 is built, Phase 3 follows the same adapter pattern.

The protocol-agnostic design is itself a patent-relevant architectural claim: the channel mutation principle is independent of the transport protocol. The same fingerprint independence property holds for ASMP, TLS, and TCP — only the adapter differs.

10. Conclusion

TorpedoRay closes the last major gap in the ACSE surface coverage. Every surface in a protected enterprise estate now has a corresponding mutation profile: API surfaces, credential surfaces, data surfaces, topology surfaces, management surfaces, power/thermal surfaces, and now transport channel surfaces.

The question that came from multiple reviewers — "what about data in transit?" — has a precise, patent-evidenced answer. TLS protects the content. TorpedoRay protects the channel fingerprint independently of TLS. An observer watching the network cannot fingerprint the implementation, cannot correlate sessions, cannot replay intercepted tokens, and cannot build a stable model of the channel before the Torpedo fires and resets it.

360 tests pass. The implementation is committed to the patent-evidence repository. The four claims — channel fingerprint independence, JA3/JA3S defeat, in-transit session token rotation, and the Torpedo mechanism — are each independently evidenced by the test suite.

FURTHER READING

WP-01: ACSE Architecture Overview · WP-02: PME Engineering Reference · WP-03: ASMP/1.0 Wire Protocol · WP-07: Formal Verification

Read the full formatted version:

Download PDF

← All Papers