ESP Echo Protocol

draft-ietf-ipsecme-encrypted-esp-ping-05

Antony Antony

IETF 127, November 2026

Why ESP Echo? A re-cap

ESP Echo = **encrypted mode** + **unencrypted mode** (formerly Encrypted ESP Ping and ESP Ping)

Encrypted mode (post-IKE, inside an established SA):

- authenticated liveness check, per SA / per path
- Reply path SPI targets one specific path when multiple SAs exist
  - e.g. primary/backup path — wired vs 4G/5G failover
  - e.g. RFC 9611 multipath scenarios
- IKEv2 capability negotiation → silence reliably means path failure

Unencrypted mode (SPI 7 / 8, pre-IKE):

- tests raw ESP (protocol 50) reachability before IKE negotiation
- unauthenticated, no ICV — proves the protocol passes, nothing more
- no capability negotiation — silence is ambiguous

Draft changes: -03 to -05

- **Merged** ESP Ping and Encrypted ESP Ping into one draft, retitled **ESP Echo Protocol**
  - encrypted mode (SA SPI) and unencrypted mode (fixed SPI 7 / 8)
  - both use Next Header 144 (AGGFRAG_PAYLOAD)
- New co-authors (-04): Lorenzo Colitti, Jen Linkova, Nathan Harold
  - -05: Michael Richardson stepped down as co-author.
- **Updates RFC 4303** (fixed SPI 7 and 8), and RFC 9347
- IKEv2 notification renamed: `ENCRYPTED_PING_SUPPORTED` → `ESP_ECHO_SUPPORTED`

New since version -03: protocol details

- Fixed 12-octet header: sub-type, R flag, length, ID, seq, **Reply path SPI**
  - Reply path SPI always present (0 = reply on any SA)
- Reply SPI validated against responder's SAD: prevents reflection / amplification
- Echo is **standalone**: never aggregated or fragmented with IP-TFS data
- **Exempt from inbound SPD traffic selector checks** (no inner IP header)
- Unencrypted mode: NULL encryption, no ICV, seq 0; no reply does **not** prove ESP is blocked
- UDP/TCP encapsulation: SPI 7 / 8 accepted without an SA; over TCP it is liveness only, not an MTU probe
- New Implementation Status section (Linux XFRM)

Linux implementation

draft-ietf-ipsecme-encrypted-esp-ping-05

Modelled on the ICMP **ping socket** (`net/ipv4/ping.c`): `socket(AF_INET, SOCK_DGRAM, IPPROTO_ESP)`

Linux kernel implementation: Majority of the code is in `net/ipv4/esp_ping.c`, `net/ipv6/esp6_ping.c` `net/xfrm/xfrm_esp_ping.c`

UAPI most changes are in: `include/uapi/linux/in.h` `include/uapi/linux/xfrm.h` `include/uapi/linux/ip.h`

Questions?

github:antonyantony linux strongswan busybox iproute2