Integrity Protection of IOAM Data Fields

draft-ietf-ippm-ioam-data-integrity-15

Justin Iurman, Frank Brockners, Shwetha Bhandari, Tal Mizrahi

IETF 124, IPPM WG
November 5, 2025

Document status (-15)

- Version -15 addresses Giuseppe Fioccola's review (Ack received on the mailing list)
- The authors believe that -15 also addresses Benjamin Kaduk's second secdir review

Issue 1/2: ☑ solved

Nonce reuse on transit nodes:

- mandatory for Trace Option-Types, but…
- needed much more text to ensure that no key/nonce reuse would *ever* occur

In scope: ***when*** keys are exchanged

Out of scope: ***how*** keys are exchanged

When the key-specific counter of an encapsulating node reaches the maximum value (i.e., 2^64), the encapsulating node MUST stop applying integrity protection with that key and start using a new one. Ideally, before that limit is reached, the key management system SHOULD rotate the key and notify any Validator that needs to validate messages protected by that key. The Key ID field provides an easy way to avoid conflicts during key rotations with other IOAM nodes that apply integrity protection. When an encapsulating node reaches its maximum use of 256 distinct keys (i.e., see the Key ID field) and is also about to reach the limit of its key-specific counter, the key management system MUST rotate the keys of all IOAM nodes in the IOAM-Domain and notify any Validator with the new keys. In addition, the internal integrity protection state of all IOAM nodes, which is useful to detect and prevent key-nonce reuse, MUST be reset. Overall, the key management system MUST NOT (ever) redistribute an old key to any of the IOAM nodes in the IOAM-Domain.

An encapsulating node that participates in the integrity protection of IOAM-Data-Fields MAY retain on durable storage the current key in use and the corresponding key-specific counter, such that a power interruption or system crash (or reboot in general) does not lead to nonce reuse. If it is not possible for some reason, the encapsulating node MUST NOT reuse the key and the key management system MUST rotate the key before the encapsulating node resumes integrity protection. In case of a power interruption or system crash (or reboot in general) on any transit node that participates in the integrity protection of IOAM-Data-Fields, the transit node MUST NOT reuse the key and the key management system MUST rotate the key before the transit node resumes integrity protection.

To illustrate with an example, it would take 7 years for a single key of an encapsulating node to reach the limit of 2^64 1500-byte packets on a 1 Pbps (Petabits per second) link at line rate, while it would take approximately 143 days with 64-byte packets. For 256 distinct keys of an encapsulating node, it would take 1792 and 100 years respectively. This is a worst-case scenario, as such link capacity is not common and IOAM may not be applied to all packets.

Issue 2/2: solved

Header fields protection vs. interoperability:

5.2. Integrity Protection of header fields

The main objective of the Integrity Protection Method defined in this document is to provide integrity protection of IOAM-Data-Fields. However, some Option-Type header fields are crucial for IOAM-Data-Fields (e.g., the IOAM Namespace-ID, which is common to all IOAM Option-Types). Without such fields, IOAM-Data-Fields cannot be reliably interpreted. As a consequence, the integrity of any immutable Option-Type header field MUST be protected, while the integrity of any mutable Option-Type header field MUST NOT be protected.

With GMAC, the bit length of the AAD MUST be a multiple of 8. In other words, the AAD is made up of whole octets. Masking specific bits of Option-Type header fields allows this constraint to be respected, even for fields that are single bits (e.g., flags) or not aligned on natural boundaries. The list of Option-Type header fields and their corresponding integrity protection masks to be applied (i.e., a logical AND operation) are defined in Section 6.2.

For interoperability, having a dynamic registry to specify which header fields participate in integrity protection may not seem optimal at first glance. However, from a vendor perspective, integrity protection is just a feature built on top of IOAM. Therefore, if two different vendors providing IOAM integrity protection are not interoperable, it probably means that their core IOAM implementations are not interoperable in the first place. Since IOAM has no versions, it does not make sense to introduce versions in the integrity protection feature either. This is the reason why it is acceptable to have a dynamic registry for this purpose and to update it accordingly over time.

ID	Description	Protected Header Fields	Reference
0x00	See Section 5	Pre-allocated Trace and Incremental Trace: - Namespace-ID (mask = 0xffff) - NodeLen + Flags + RemainingLen (mask = 0xfb00) - IOAM-Trace-Type (mask = 0xffffff) - Reserved (mask = 0x00) POT: - Namespace-ID (mask = 0xffff) - IOAM-POT-Type (mask = 0xff) IOAM-POT-Flags (mask = 0x00) E2E: - Namespace-ID (mask = 0xffff) - IOAM-E2E-Type (mask = 0xffff)	This document
0x01 - 0xFE	Unassigned		
0xFF	Reserved		This document

Issue 2/2: ☑ solved

Reminder from Bangkok (IETF 122):

Solution for header fields <u>and</u> interop?

Current solution: define header fields to be included, new ones are excluded (one-time fixed definition)

Opposite solution: versioning (perfect? maybe, but probably extreme)

- IOAM Option-Types (RFC 9197) don’t have versions
- Would be (even) less hardware friendly…

Proposed trade-off solution: define header fields to be included, new ones are added to the registry

- Integrity protection seen as a feature on top of IOAM Option-Types
- This *feature* (if present) is aligned and up-to-date with its IOAM implementation
- If Vendor_A_IOAM **!=** Vendor_B_IOAM → no interoperability
- If Vendor_A_IOAM **==** Vendor_B_IOAM → interoperability (both IOAM <u>and</u> IOAM Integrity)

IETF 122, IPPM, March 2025 3

Next step

- The document is ready for a second WGLC
- Feedback or comments welcome