ippm
Internet-Draft
Intended status: Standards Track
Expires: 9 October 2024

F. Brockners
Cisco
S. Bhandari
Thoughtspot
T. Mizrahi
Huawei
J. Iurman
ULiege

7 April 2024

# Integrity of In-situ OAM Data Fields
# draft-ietf-ippm-ioam-data-integrity-08

## Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path in the network. IETF protocols require features to ensure their security. This document describes the integrity protection of IOAM-Data-Fields.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 9 October 2024.

## Copyright Notice

Table of Contents



## 1. Introduction

"In-situ" Operations, Administration, and Maintenance (IOAM) records OAM information within the packet while the packet traverses a particular network domain. The term "in-situ" refers to the fact that the OAM data is added to the data packets rather than being sent within packets specifically dedicated to OAM. IOAM is to complement mechanisms such as Ping or Traceroute. In terms of "active" or

"passive" OAM, "in-situ" OAM can be considered a hybrid OAM type. "In-situ" mechanisms do not require extra packets to be sent. IOAM adds information to the already available data packets and therefore cannot be considered passive. In terms of the classification given in [RFC7799], IOAM could be portrayed as Hybrid Type I. IOAM mechanisms can be leveraged where mechanisms using, e.g., ICMP do not apply or do not offer the desired results, such as proving that a certain traffic flow takes a pre-defined path, SLA verification for the data traffic, detailed statistics on traffic distribution paths in networks that distribute traffic across multiple paths, or scenarios in which probe traffic is potentially handled differently from regular data traffic by the network devices.

IOAM MUST be deployed in an IOAM-Domain. An IOAM-Domain is a set of nodes that use IOAM. An IOAM-Domain is bounded by its perimeter or edge. It is expected that all nodes in an IOAM-Domain are managed by the same administrative entity, that has means to select, monitor, and control the access to all the networking devices. As such, IOAM-Data-Fields are carried in clear within packets and there are no protections against any node or middlebox tampering with the data. IOAM-Data-Fields collected in an untrusted or semi-trusted environment require integrity protection to support critical operational decisions. Please refer to [RFC9197] for more details on IOAM-Domains.

The following considerations and requirements are to be taken into account in addition to addressing the problem of detectability of any integrity breach of the IOAM-Data-Fields collected:

1. IOAM data is processed by the data plane, hence viability of any method to prove integrity of the IOAM-Data-Fields must be feasible at data plane processing/forwarding rates (IOAM might be applied to all traffic a router forwards).

2. IOAM data is carried within packets. Additional space required to prove integrity of the IOAM-Data-Fields needs to be optimal, i.e. should not exceed the Maximum Transmission Unit (MTU) or have adverse effect on packet processing.

3. Protection against replay of old IOAM data should be possible. Without replay protection, a rogue node can present the old IOAM data, masking any ongoing network issues/activity and making the IOAM-Data-Fields collection useless.

This document defines a method to protect the integrity of IOAM-Data-Fields, using the IOAM Option-Types specified in [RFC9197] and [RFC9326] as an example. The method will similarly apply to future IOAM Option-Types.

## 2. Conventions

### 2.1. Requirements Language

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.

### 2.2. Abbreviations

Abbreviations used in this document:

OAM: Operations, Administration, and Maintenance

IOAM: In-situ OAM

POT: Proof of Transit

E2E: Edge to Edge

DEX: Direct Exporting

## 3. Threat Analysis

This section presents a threat analysis of integrity-related threats in the context of IOAM. The threats that are discussed are assumed to be independent of the lower layer protocols; it is assumed that threats at other layers are handled by security mechanisms that are deployed at these layers.

This document is focused on integrity protection for IOAM-Data-Fields. Thus the threat analysis includes threats that are related to or result from compromising the integrity of IOAM-Data-Fields. Other security aspects such as confidentiality are not within the scope of this document.

Throughout the analysis there is a distinction between on-path and off-path attackers. As discussed in [RFC9055], on-path attackers are located in a position that allows interception and modification of in-flight protocol packets, whereas off-path attackers can only attack by generating protocol packets.

The analysis also includes the impact of each of the threats. Generally speaking, the impact of a successful attack on an OAM protocol [RFC7276] is a false illusion of nonexistent failures or preventing the detection of actual ones; in both cases, the attack

may result in denial of service (DoS). Furthermore, creating the false illusion of a nonexistent issue may trigger unnecessary processing in some of the IOAM nodes along the path, and may cause more IOAM-related data to be exported to the management plane than is conventionally necessary. Beyond these general impacts, threat-specific impacts are discussed in each of the subsections below.

### 3.1. Modification: IOAM-Data-Fields

Threat

An on-path attacker can modify the IOAM-Data-Fields of in-transit packets. The modification can either be applied to all packets or selectively applied to a subset of the en route packets. Maliciously modified IOAM-Data-Fields can for example mislead network diagnostics, result in incorrect network performance metrics, or could misguide network optimization efforts.

Impact

By systematically modifying the IOAM-Data-Fields of some or all of the in-transit packets, an attacker can create a false picture of the paths in the network, the existence of faulty nodes and their location, and the network performance.

### 3.2. Modification: IOAM Option-Type Headers

Threat

An on-path attacker can modify the header in IOAM Option-Types in order to change or disrupt the behavior of nodes processing IOAM-Data-Fields along the path.

Impact

Changing the header of IOAM Option-Types may have several implications. An attacker can maliciously increase the processing overhead in nodes that process IOAM-Data-Fields and increase the on-the-wire overhead of IOAM-Data-Fields, for example by modifying the IOAM-Trace-Type field in the IOAM Trace Option-Type header. An attacker can also prevent some of the nodes that process IOAM-Data-Fields from incorporating IOAM-Data-Fields, by modifying the RemainingLen field in the IOAM Trace Option-Type header. Another possibility for the attacker is to change the context of IOAM-Data-Fields by modifying the Namespace-ID field in IOAM Option-Type headers, which makes the integrity protection of IOAM-Data-Fields completely useless.

### 3.3. Injection: IOAM-Data-Fields

Threat

An attacker can inject packets with IOAM Option-Types and IOAM-Data-Fields. This threat is applicable to both on-path and off-path attackers.

Impact

This attack and its impacts are similar to Section 3.1.

### 3.4. Injection: IOAM Option-Type Headers

Threat

An attacker can inject packets with IOAM Option-Type headers, thus manipulating other nodes that process IOAM-Data-Fields in the network. This threat is applicable to both on-path and off-path attackers.

Impact

This attack and its impacts are similar to Section 3.2.

### 3.5. Replay

Threat

An attacker can replay packets with IOAM-Data-Fields. Specifically, an attacker may replay a previously transmitted IOAM Option-Type with a new data packet, therefore attaching old IOAM-Data-Fields to a fresh user packet. This threat is applicable to both on-path and off-path attackers.

Impact

By replaying old IOAM-Data-Fields, an attacker can create a false picture of the network status. The attacker could simulate a nonexistent failure, or incur non-required processing load on nodes that process these IOAM-Data-Fields.

### 3.6. Management and Exporting

Threat

Attacks that compromise the integrity of IOAM-Data-Fields can be applied at the management plane, e.g., by manipulating network management packets. Furthermore, the integrity of IOAM-Data-Fields that are exported to a receiving entity can also be compromised. Management plane attacks are not within the scope of this document; the network management protocol is expected to include inherent security capabilities. The integrity of exported data is also not within the scope of this document. It is expected that the specification of the export format will discuss the relevant security aspects.

Impact

Malicious manipulation of the management protocol can cause nodes that process IOAM-Data-Fields to malfunction, to be overloaded, or to incorporate unnecessary IOAM-Data-Fields into user packets. The impact of compromising the integrity of exported IOAM-Data-Fields is similar to the impacts of previous threats that were described in this section.

### 3.7. Delay

Threat

An on-path attacker may delay some or all of the in-transit packets that include IOAM-Data-Fields in order to create the false illusion of congestion. Delay attacks are well known in the context of deterministic networks [RFC9055] and synchronization [RFC7384], and may be somewhat mitigated in these environments by using redundant paths in a way that is resilient to an attack along one of the paths. This approach does not address the threat in the context of IOAM, as it does not meet the requirement to measure a specific path or to detect a problem along the path. It is noted that this threat is not within the scope of the threats that are mitigated in this document.

Impact

Since IOAM can be applied to a fraction of the traffic, an attacker can detect and delay only the packets that include IOAM-Data-Fields, thus preventing the authenticity of delay and load measurements.

### 3.8. Threat Summary

| Threat | In scope | Out of scope |
|---|---|---|
| Modification: IOAM-Data-Fields | + | |
| Modification: IOAM Option-Type Headers | + | |
| Injection: IOAM-Data-Fields | + | |
| Injection: IOAM Option-Type Headers | + | |
| Replay | + | |
| Management and Exporting | | + |
| Delay | | + |

Figure 1: Threat Analysis Summary

## 4. Integrity Protected Option-Types

This section defines new IOAM Option-Types. Their purpose is to carry IOAM-Data-Fields with integrity protection. Each of the IOAM Option-Types defined in [RFC9197] and [RFC9326] is extended as follows:

64 IOAM Pre-allocated Trace Integrity Protected Option-Type: corresponds to the IOAM Pre-allocated Trace Option-Type ([RFC9197]) with integrity protection.

65 IOAM Incremental Trace Integrity Protected Option-Type: corresponds to the IOAM Incremental Trace Option-Type ([RFC9197]) with integrity protection.

66 IOAM POT Integrity Protected Option-Type: corresponds to the IOAM POT Option-Type ([RFC9197]) with integrity protection.

67 IOAM E2E Integrity Protected Option-Type: corresponds to the IOAM E2E Option-Type ([RFC9197]) with integrity protection.

68 IOAM DEX Integrity Protected Option-Type: corresponds to the IOAM DEX Option-Type ([RFC9326]) with integrity protection.

The IOAM Integrity Protection Header follows the IOAM Option-Type header when the IOAM Option-Type is an Integrity Protected Option-Type. It is defined as follows:

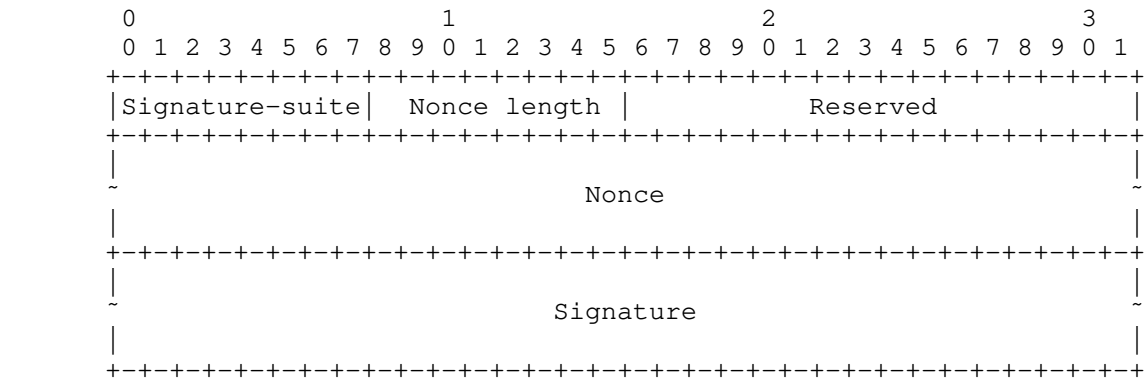



Figure 2: IOAM Integrity Protection Header

Signature-suite: 8-bit unsigned integer. This field defines the algorithm pair used to compute the digest and the signature over the IOAM-Data-Fields, as defined in Section 6.2.

Nonce length: 8-bit unsigned integer. This field specifies the length of the Nonce in octets.

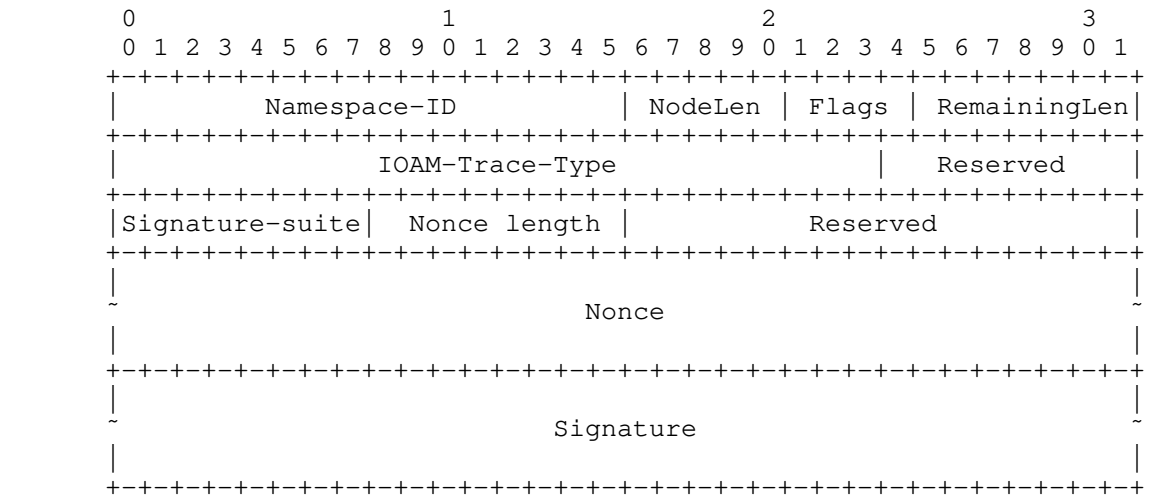
Reserved: 16-bit Reserved field. MUST be set to zero upon transmission and ignored upon receipt.

Nonce: Variable length field with length specified in Nonce length.

Signature: Digital signature value generated by the algorithm pair specified by Signature-suite. The Signature length depends on the Signature-suite value.

## 4.1. Integrity Protected Trace Option-Types

Both the IOAM Pre-allocated Trace Option-Type header and the IOAM Incremental Trace Option-Type header, as defined in [RFC9197], are followed by the Integrity Protection header when the IOAM Option-Type is respectively set to the IOAM Pre-allocated Trace Integrity Protected Option-Type or the IOAM Incremental Trace Integrity Protected Option-Type:




#### 4.1.1. Header fields for integrity protection

The following IOAM Pre-allocated or Incremental Option-Type header fields are involved in the integrity protection of IOAM-Data-Fields:

1. Namespace-ID

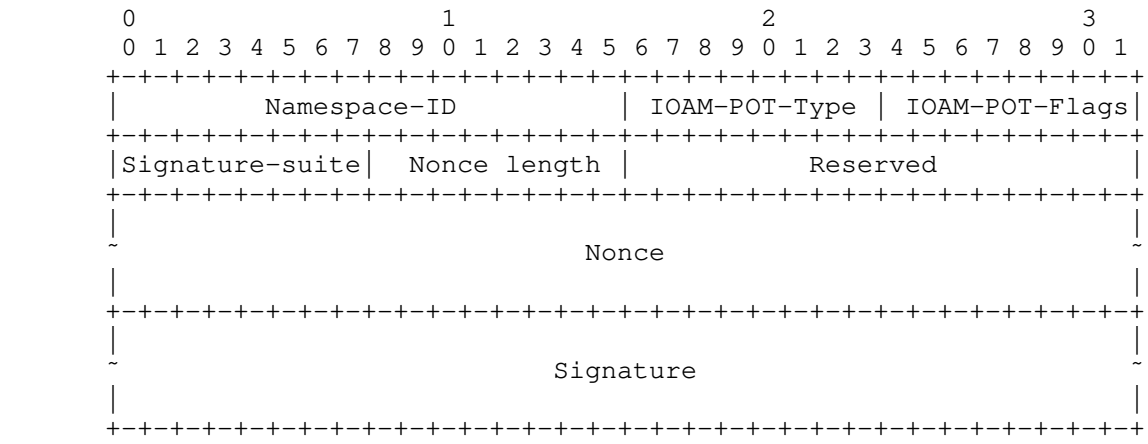
2. NodeLen

3. Flags: only bits 1 (Loopback) and 2 (Active)

4. IOAM-Trace-Type

### 4.2. Integrity Protected POT Option-Type

The IOAM POT Option-Type header, as defined in [RFC9197], is followed by the Integrity Protection header when the IOAM Option-Type is set to the IOAM POT Integrity Protected Option-Type:




#### 4.2.1. Header fields for integrity protection

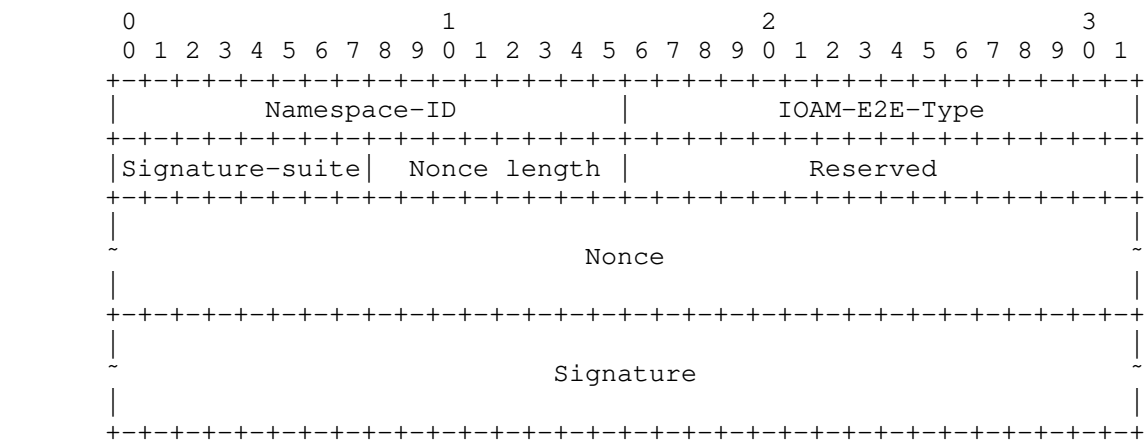
The following IOAM POT Option-Type header fields are involved in the integrity protection of IOAM-Data-Fields:

1. Namespace-ID

2. IOAM-POT-Type

### 4.3. Integrity Protected E2E Option-Type

The IOAM E2E Option-Type header, as defined in [RFC9197], is followed by the Integrity Protection header when the IOAM Option-Type is set to the IOAM E2E Integrity Protected Option-Type:



#### 4.3.1. Header fields for integrity protection

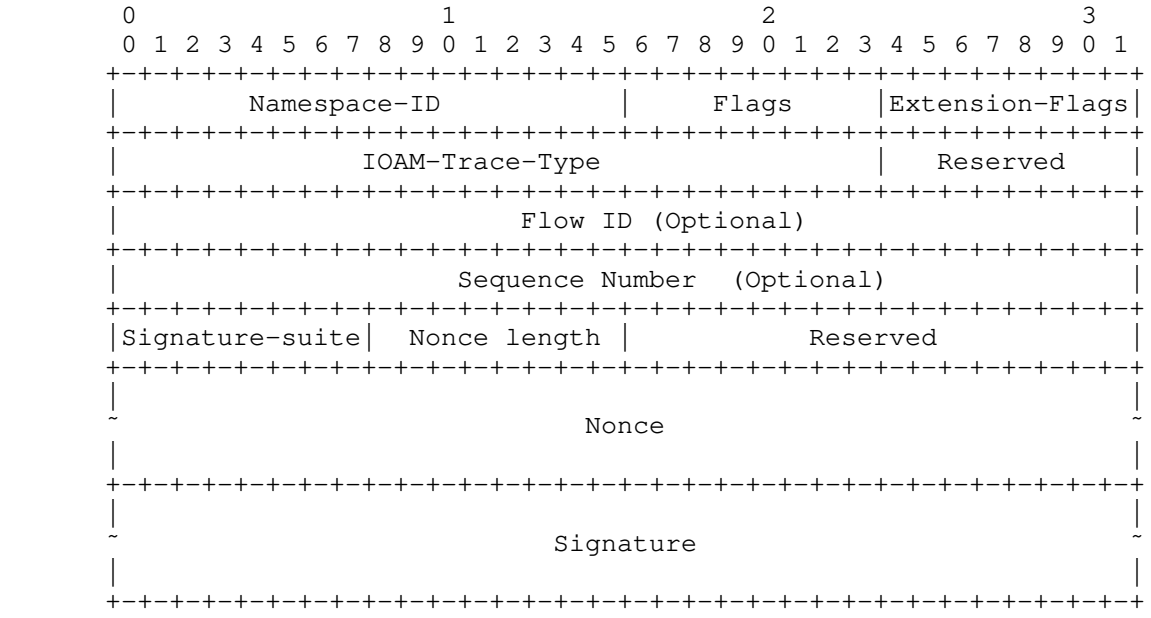
The following IOAM E2E Option-Type header fields are involved in the integrity protection of IOAM-Data-Fields:

1. Namespace-ID

2. IOAM-E2E-Type

### 4.4. Integrity Protected DEX Option-Type

The IOAM DEX Option-Type header, as defined in [RFC9326], is followed by the Integrity Protection header when the IOAM Option-Type is set to the IOAM DEX Integrity Protected Option-Type:




#### 4.4.1. Header fields for integrity protection

The following IOAM DEX Option-Type header fields are involved in the integrity protection of IOAM-Data-Fields:

1. Namespace-ID

2. Extension-Flags: only bits 0 (Flow ID) and 1 (Sequence Number)

3. IOAM-Trace-Type

The optional fields (i.e., Flow ID and Sequence Number) are treated as optional IOAM-Data-Fields, not header fields.

## 5. Integrity Protection Method

This section defines a method that uses a symmetric key based signature algorithm for integrity protection of IOAM-Data-Fields. In case of performance concerns, the method can be applied to a subset of the traffic by using sampling of data.

The symmetric key based signature algorithm MUST use SHA-256 ([SHS]) as the Digest Algorithm, and MUST use Advanced Encryption Standard ([AES]) with a key length of 256 bits and the Galois/Counter Mode ([NIST.800-38D]) as the Signature Algorithm (AES-256-GCM). The corresponding Signature Suite Identifier is 1, as defined in Section 6.2. As a consequence, the signature consumes 32 octets. The Integrity Protection Method is defined by the following steps:

1. The encapsulating node creates a nonce and stores it in the Nonce field of the IOAM Integrity Protection Header (the Nonce length field is set accordingly). The Signature-suite field is set to 1. The signature is generated over the hash of header fields (see Section 4.1.1, Section 4.2.1, Section 4.3.1, or Section 4.4.1, for the exact list of header fields to include in the signature, depending on the IOAM Integrity Protected Option-Type) and IOAM-Data-Fields it has inserted, i.e., sign(hash(Header-Fields || IOAM-Data-Fields)), with the Nonce field provided as the nonce. IOAM-Data-Fields supposed to be modified by other IOAM nodes on the path MUST be excluded from the signature (e.g., the POT Cumulative field). The signature is stored in the Signature field of the IOAM Integrity Protection Header.

2. A transit node generates a signature over the hash of IOAM-Data-Fields it has inserted, i.e., sign(hash(IOAM-Data-Fields)), with the Signature field provided as the nonce. IOAM-Data-Fields modified in-place by the transit node MUST be excluded from the signature (e.g., the POT Cumulative field). The signature is stored in the Signature field of the IOAM Integrity Protection Header.

    * If the transit node does not insert IOAM-Data-Fields (e.g., it only modifies IOAM-Data-Fields in-place, or does nothing), then the transit node MUST NOT generate a signature and MUST NOT update the Signature field.

3. In the context of the Integrity Protection Method, a node that performs the validation of the integrity protection is referred to as a "Validator". The role of a Validator is to recompute the signature by iteratively following the previous steps of the method, in the same order and up to itself, using the respective symmetric keys. The recomputed signature is then compared to the Signature field. As a result, the Validator can detect if the IOAM-Data-Fields integrity is intact or was altered. The validation is trivial in some cases (e.g., with POT Type-0, E2E or DEX Option-Types), where only the encapsulating node generates a signature, as specified above by this method. For other cases where transit nodes also generate a signature (e.g., with Trace Option-Types), node-ids MUST be included in IOAM-Data-Fields. Details on how the mapping between node-ids and keys is implemented on a Validator are outside the scope of this document.

    * Each node that takes actions triggered by fields in the IOAM Integrity Protected Option-Type header MUST act as a Validator. Otherwise, an attacker could modify the IOAM header along the path and change the actions a node performs. Examples:

        - For an Integrity Protected Trace Option-Type (Pre-allocated or Incremental), each transit node MUST act as a Validator, if either the IOAM Loopback or Active mode is used.

        - For an Integrity Protected DEX Option-Type, each transit node MUST act as a Validator.

    * The decapsulating node MUST act as a Validator. The decapsulating node MUST NOT generate a signature based on IOAM-Data-Fields it has inserted, if any, and therefore MUST NOT update the Signature field.

The method assumes that symmetric keys have been distributed to the respective nodes as well as the Validator(s). The details of the mechanisms used for key distribution are outside the scope of this document.

## 6. IANA Considerations

### 6.1. IOAM Option-Types

IANA is requested to define the following new code points in the "IOAM Option-Type" registry:

64 IOAM Pre-allocated Trace Integrity Protected Option-Type (see

Section 4)

65 IOAM Incremental Trace Integrity Protected Option-Type (see Section 4)

66 IOAM POT Integrity Protected Option-Type (see Section 4)

67 IOAM E2E Integrity Protected Option-Type (see Section 4)

68 IOAM DEX Integrity Protected Option-Type (see Section 4)

A document defining a new IOAM Integrity Protected Option-Type MUST define the IOAM Option-Type header fields involved in the integrity protection of IOAM-Data-Fields, as done in Section 4.1.1, Section 4.2.1, Section 4.3.1, and Section 4.4.1 of this document.

## 6.2. IOAM Integrity Protection Signature Suite

IANA is requested to define a new registry named "IOAM Integrity Protection Signature Suite", inside the "In Situ OAM (IOAM)" registry group.

The new registry defines 256 code points to identify the digest and signature algorithms used in the Signature-suite field, as explained in Section 4. The following code points are defined in this document:

| Signature Suite Identifier | Digest Algorithm | Signature Algorithm | Specification Pointer |
|---|---|---|---|
| 0x00 | Reserved | Reserved | This document |
| 0x01 | SHA-256 | AES-256-GCM | This document |
| 0x02 ... 0xFE | Unassigned | Unassigned | |
| 0xFF | Reserved | Reserved | This document |

Figure 3: IOAM Integrity Protection Signature Suite

Code points 2-254 are available for assignment via the "IETF Review" process, as per [RFC8126].

New registration requests MUST use the following template: the value of the requested code point, the associated digest algorithm name and signature algorithm name, and a reference to the document that defines the requested code point.

## 7. Security Considerations

Please refer to Section 3 for a threat analysis of integrity-related threats in the context of IOAM.

The Integrity Protection Method defined in this document (see Section 5) leverages symmetric keys. The symmetric keys need to be exchanged in a secure way between the nodes involved with integrity protection of IOAM-Data-Fields. The details of the key exchange are outside the scope of this document.

The Integrity Protection Method defined in this document requires additional per-packet processing by each node that uses it. Inappropriate use of the Integrity Protection Method might overload nodes and cause them to stop functioning properly. Operators deploying IOAM with the Integrity Protection Method MUST ensure that such overload situations are avoided. This could for example be achieved by applying IOAM only to a subset of the entire traffic.

The Nonce makes a signature chain unique but does not necessarily prevent replay attacks. To enable replay protection, the encapsulating node and the Validator(s) MUST agree on a common methodology to keep the Nonce valid only for a specific period of time, which is outside the scope of this document.

## 8. Acknowledgements

The authors would like to thank Santhosh N, Rakesh Kandula, Saiprasad Muchala, Al Morton, Greg Mirsky, Benjamin Kaduk, Mehmet Beyaz, and Martin Duke for their comments and advice.

## 9. References

### 9.1. Normative References

[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/info/rfc2119>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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/info/rfc8174>.

## 9.2. Informative References

[AES] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, 2001, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.

[NIST.800-38D] National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, 2001, <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>.

[RFC7276] Mizrahi, T., Sprecher, N., Bellagamba, E., and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools", RFC 7276, DOI 10.17487/RFC7276, June 2014, <https://www.rfc-editor.org/info/rfc7276>.

[RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <https://www.rfc-editor.org/info/rfc7384>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

[RFC9055] Grossman, E., Ed., Mizrahi, T., and A. Hacker, "Deterministic Networking (DetNet) Security Considerations", RFC 9055, DOI 10.17487/RFC9055, June 2021, <https://www.rfc-editor.org/info/rfc9055>.

[RFC9197] Brockners, F., Ed., Bhandari, S., Ed., and T. Mizrahi, Ed., "Data Fields for In Situ Operations, Administration, and Maintenance (IOAM)", RFC 9197, DOI 10.17487/RFC9197, May 2022, <https://www.rfc-editor.org/info/rfc9197>.

[RFC9326] Song, H., Gafni, B., Brockners, F., Bhandari, S., and T. Mizrahi, "In Situ Operations, Administration, and Maintenance (IOAM) Direct Exporting", RFC 9326, DOI 10.17487/RFC9326, November 2022, <https://www.rfc-editor.org/info/rfc9326>.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", NIST FIPS Publication 180-4, DOI 10.6028/NIST.FIPS.180-4, 2015, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

## Authors' Addresses

Frank Brockners
Cisco Systems, Inc.
Hansaallee 249, 3rd Floor
40549 DUESSELDORF
Germany
Email: fbrockne@cisco.com

Shwetha Bhandari
Thoughtspot
3rd Floor, Indiqube Orion, 24th Main Rd, Garden Layout, HSR Layout
Bangalore, KARNATAKA 560 102
India
Email: shwetha.bhandari@thoughtspot.com

Tal Mizrahi
Huawei
8-2 Matam
Haifa 3190501
Israel
Email: tal.mizrahi.phd@gmail.com

Justin Iurman
Universite de Liege
10, Allee de la decouverte (B28)
4000 Sart-Tilman
Belgium
Email: justin.iurman@uliege.be