ippm
Internet-Draft
Intended status: Standards Track
Expires: 26 February 2026

F. Brockners
Cisco
S. Bhandari
Databricks
T. Mizrahi
Huawei
J. Iurman
University of Liege
25 August 2025

Integrity Protection of In Situ Operations, Administration, and Maintenance (IOAM) Data Fields
draft-ietf-ippm-ioam-data-integrity-14

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 that can provide secure operation. 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 26 February 2026.

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 verifying 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 the 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.

Since arbitrary nodes and middleboxes can tamper with all packets data, including IOAM-Data-Fields, and the packets are (in general) processed by other intermediary nodes before they could arrive at a node that can verify the IOAM fields of the packet, there is little value in attempting to use cryptographic mechanisms to prevent such modifications to the IOAM fields in the packet. Instead, we limit ourselves to the "detectability problem", namely, to allow an endpoint to detect that such modification has occurred since the generation of the IOAM-Data-Fields. In addition to this detectability problem, the following considerations and requirements are to be taken into account in constructing an IOAM integrity mechanism:

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 SHOULD 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] 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

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 an 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 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 fake picture of the network status. Potential consequences include an impact on network performance, a change in the recorded forwarding path of packets, either based on fake node positions or fake data provided by the attacker to fool the system that ingests IOAM-Data-Fields.

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 existing IOAM Option-Types, just like it is assumed for future IOAM Option-Types, may have several implications. The following list of examples is not exhaustive, and is based on IOAM Option-Types defined in [RFC9197] and [RFC9326]. An attacker could maliciously increase the processing overhead in nodes that process IOAM-Data-Fields and increase the on-the-wire overhead of IOAM-Data-Fields, by modifying the IOAM-Trace-Type field in the IOAM Trace Option-Type header. An attacker could 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 definition or interpretation of IOAM-Data-Fields by modifying the Namespace-ID field, which is common to all IOAM Option-Type headers. For IOAM-Namespaces, please refer to [RFC9197], Section 4.2. Without the right context (i.e., Namespace-ID), IOAM-Data-Fields become meaningless, just like data without metadata. An attacker could also set the Loopback flag in the IOAM Trace Option-Type header so that packet copies would be sent back by each node to the encapsulating node. Note that the modification of the header can lead to similar impacts described in Section 3.1.

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

In addition to replaying old packets in general, 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

The impacts of this attack are similar to those described in Section 3.1.

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 an illusion of congestion. Delay attacks are well known in the context of deterministic networks [RFC9055] and time 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. Note that the mechanisms in this document do not attempt to provide any mitigation against this threat.

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. All existing IOAM Option-Types defined in [RFC9197] are extended as follows:

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

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

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

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

The Direct Export (DEX) Option-Type [RFC9326] is not covered by the Integrity Protection Method defined in this document (see Section 5). This document focuses on the integrity protection of IOAM-Data-Fields, while DEX does not have IOAM-Data-Fields by definition. Therefore, DEX is considered out of scope for this document. DEX, as well as any IOAM Option-Type without IOAM-Data-Fields, MUST NOT use the Integrity Protection Method defined in this document.

The IOAM Integrity Protection header follows the IOAM Option-Type header and precedes IOAM-Data-Fields, when the IOAM Option-Type is an Integrity Protected Option-Type. It is defined as follows:

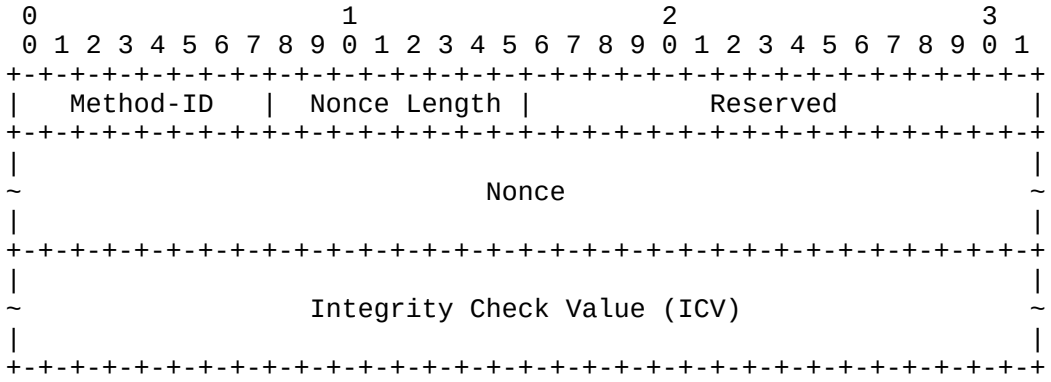


Figure 2: IOAM Integrity Protection header

Method-ID: 8-bit unsigned integer, see Section 6.2. It defines the Integrity Protection Method to compute the Integrity Check Value (ICV) field. If a node encounters an unknown value, it MUST NOT change the contents of the IOAM Integrity Protection header and MUST NOT change the contents of the IOAM-Data-Fields. In other words, the node MUST NOT process the IOAM Option-Type.

Nonce Length: 8-bit unsigned integer. It defines the length of the Nonce field, in octets.

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

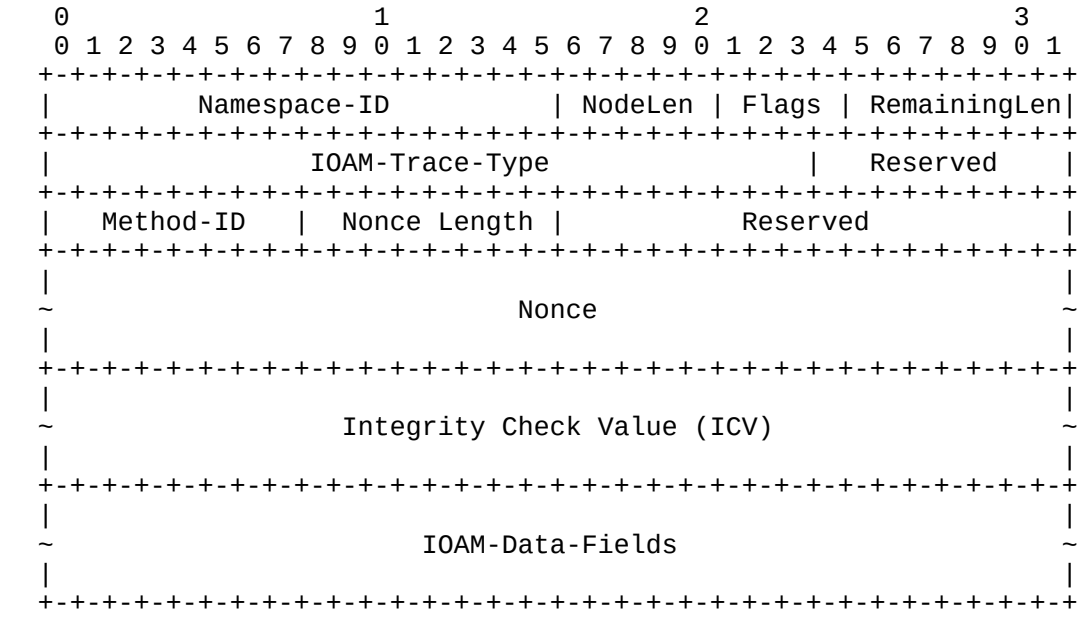
Nonce: Variable length field. Its size depends on the Nonce Length field.

Integrity Check Value (ICV): Variable length field. Its size depends on the Method-ID field.

In order to keep IOAM-Data-Fields aligned, the total length of the IOAM Integrity Protection header MUST be a multiple of 4 octets.

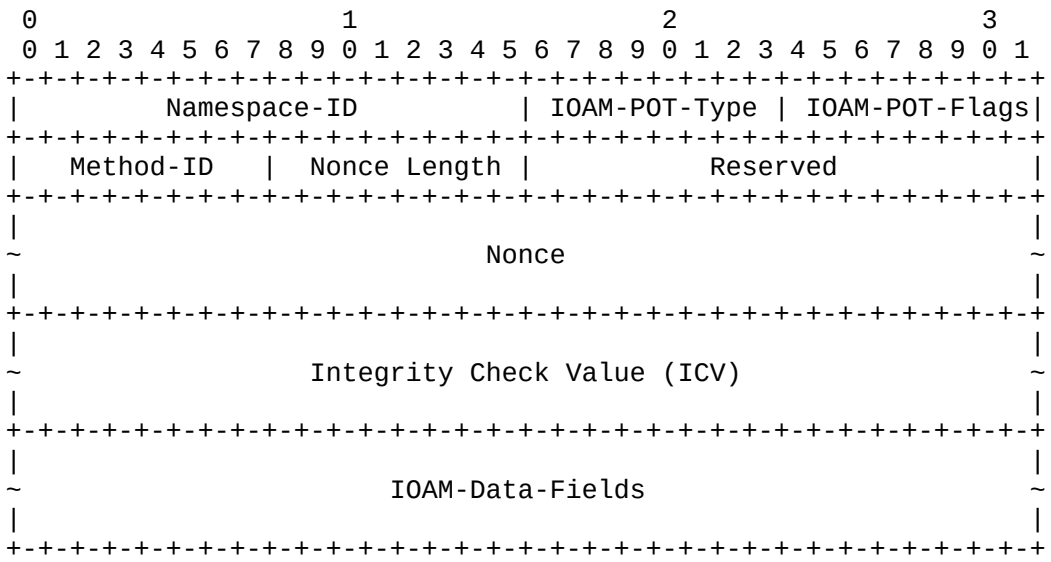
4.1. Integrity Protected Trace Option-Types

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 IOAM Integrity Protection header when the IOAM Option-Type is respectively set to the IOAM Integrity Protected Pre-allocated Trace Option-Type or the IOAM Integrity Protected Incremental Trace Option-Type:



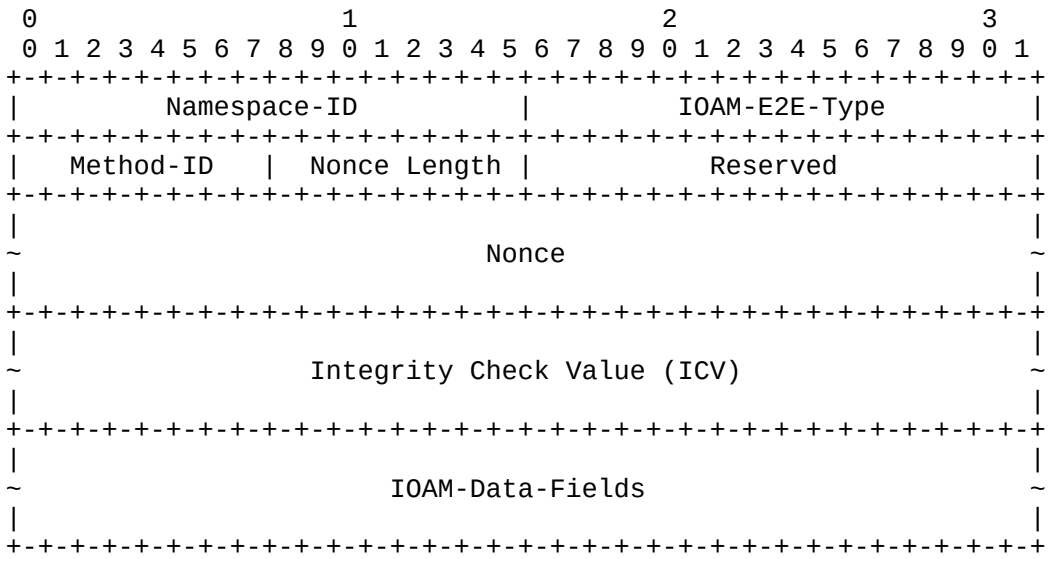
4.2. Integrity Protected POT Option-Type

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



4.3. Integrity Protected E2E Option-Type

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



5. Integrity Protection Method

This document defines a new method that uses a symmetric key based algorithm for the integrity protection of IOAM-Data-Fields. The method MUST use AES-GMAC ([AES] [NIST.800-38D]), a block cipher mode of operation providing data origin authentication, which is also a specialization of the Galois/Counter Mode (GCM). The GCM authenticated encryption operation has four inputs: a secret key, an Initialization Vector (IV), a plaintext, and Additional Authenticated Data (AAD). It has two outputs: a ciphertext whose length is identical to the plaintext and an Authentication Tag. GMAC is the special case of GCM in which the plaintext has a length of zero. Therefore, the empty ciphertext output is ignored, and the only output is the Authentication Tag. Although GMAC supports all AES key sizes (128, 192, 256 bits), it is always RECOMMENDED to use the longest key size when possible.

In this method, the GMAC Authentication Tag MUST NOT be truncated, meaning its size MUST always be 16 octets (i.e., a full Authentication Tag). Below, we refer to the GMAC Initialization Vector (IV) as the nonce, and to the GMAC Authentication Tag as the Integrity Check Value (ICV).

5.1. Encapsulating node

The encapsulating node generates a nonce based on mandatory requirements defined in Section 5.1.1 and stores it in the Nonce field of the IOAM Integrity Protection header (the Nonce Length field is set accordingly). The Method-ID field MUST be set to 1, as defined in Section 6.2.

The Integrity Check Value (ICV) is the result of a GMAC operation over a selection of header fields (see Section 5.1.2) and immutable IOAM-Data-Fields added by the encapsulating node. In the case the Integrity Protected Pre-allocated Trace Option-Type is used, the encapsulating node includes the IOAM-Data-Fields that correspond to itself, i.e., "node data list [n]" (see [RFC9197]), into the GMAC operation. With the nonce provided to GMAC, the encapsulating node performs the following operation:

* AAD = (header fields || node’s IOAM-Data-Fields)

* ICV = GMAC(AAD)

The encapsulating node stores the ICV in the corresponding field of the IOAM Integrity Protection header.

5.1.1. Nonce requirements

The size of the nonce MUST be 12 octets. The nonce MUST be based on the "Deterministic Construction" ([NIST.800-38D], Sec. 8), as follows:

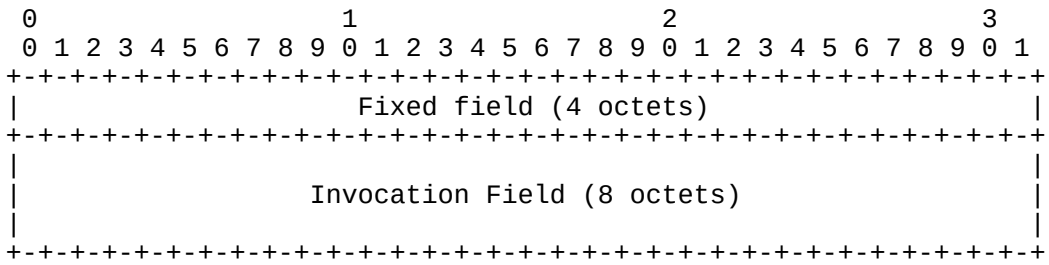


Figure 3: Structure of the Nonce field (12 octets)

* The nonce is the concatenation of two fields, called the Fixed field and the Invocation field. The leading (i.e., leftmost) 32 bits of the nonce MUST hold the Fixed field, while the trailing (i.e., rightmost) 64 bits MUST hold the Invocation field. It means a limit of 2^32 distinct devices, and a limit of 2^64 invocations for a given key.

* The Fixed field identifies an encapsulating node and MUST be unique (e.g., the unique IOAM identifier).

* The Invocation field identifies the sets of inputs (i.e., packets) to the authenticated encryption function in that encapsulating node and MUST be unique for each packet. It typically is either an integer counter or a linear feedback shift register that is driven by a primitive polynomial to ensure a maximal cycle length. In either case, the Invocation field increments upon each invocation of the authenticated encryption function.

5.1.2. Selection 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. Without them, IOAM-Data-Fields are meaningless. Therefore, the integrity of such header fields MUST be protected too.

For Trace Option-Types, here is the list of header fields that participate (in that order) in the integrity protection of IOAM-Data-Fields:

1. Namespace-ID

2. IOAM-Trace-Type

The NodeLen field is not included in the list because it can be deduced from the IOAM-Trace-Type field. Other header fields that are not included in the list are either mutable or only useful for processing Trace Option-Types (i.e., they don’t provide context or meaning to IOAM-Data-Fields).

For a POT Option-Type, here is the list of header fields that participate (in that order) in the integrity protection of IOAM-Data-Fields:

1. Namespace-ID

2. IOAM-POT-Type

Other header fields that are not included in the list are either mutable or only useful for processing a POT Option-Type (i.e., they don’t provide context or meaning to IOAM-Data-Fields).

For a E2E Option-Type, here is the list of header fields that participate (in that order) in the integrity protection of IOAM-Data-Fields:

1. Namespace-ID

2. IOAM-E2E-Type

Those are all the header fields defined for a E2E Option-Type.

For new IOAM Integrity Protected Option-Types that intend to use the Integrity Protection Method defined in this document, the same logic can be applied to select header fields by following this simple rule:

* Only immutable IOAM Option-Type header fields that provide context or meaning to IOAM-Data-Fields are considered, others are excluded (e.g., mutable or reserved fields). For example, the Namespace-ID, which is common to all IOAM Option-Types, MUST always be included in the list. Once specified, the list MUST NOT change for interoperability reasons.

5.2. Transit node

For a transit node, the Integrity Check Value (ICV) is the result of a GMAC operation over the received ICV field and immutable IOAM-Data-Fields added by the transit node. In the case the Integrity Protected Pre-allocated Trace Option-Type is used, a transit node includes the IOAM-Data-Fields that correspond to itself (the ones it updated), i.e., "node data list [i]" (see [RFC9197]), into the GMAC operation. With the received Nonce field provided to GMAC, the transit node performs the following operation:

* AAD = (ICV field || node’s IOAM-Data-Fields)

* ICV = GMAC(AAD)

The transit node updates the ICV field in the IOAM Integrity Protection header.

If the transit node does not add any immutable IOAM-Data-Fields (e.g., it only modifies mutable IOAM-Data-Fields or does nothing), and if the transit node, in case the Integrity Protected Pre-allocated Trace Option-Type is used, does not update the "node data list" array, then the transit node MUST NOT update the ICV field in the IOAM Integrity Protection header.

A transit node MUST NOT add or remove the IOAM Integrity Protection header. Also, a transit node MUST NOT modify the Method-ID field, the Nonce-Length field, and the Nonce field in the IOAM Integrity Protection header.

5.3. Decapsulating node

The decapsulating node MAY perform the function of the Validator. If it does, please refer to Section 5.4.

If the decapsulating node does not perform the function of the Validator, which is an alternative to put the Validator out of the forwarding path in case of performance concerns, the decapsulating node MUST send the entire IOAM Integrity Protected Option-Type to the Validator. The method to send it to the Validator is out of scope for this document. Before that, the decapsulating node updates the ICV field in the IOAM Integrity Protection header. The Integrity Check Value (ICV) is the result of a GMAC operation over the received ICV field and immutable IOAM-Data-Fields added by the decapsulating node. In the case the Integrity Protected Pre-allocated Trace Option-Type is used, the decapsulating node includes the IOAM-Data-Fields that correspond to itself (the ones it updated), i.e., "node data list [i]" (see [RFC9197]), into the GMAC operation. With the received Nonce field provided to GMAC, the decapsulating node performs the following operation:

* AAD = (ICV field || node’s IOAM-Data-Fields)

* ICV = GMAC(AAD)

If the decapsulating node does not add any immutable IOAM-Data-Fields (e.g., it only modifies mutable IOAM-Data-Fields or does nothing), and if the decapsulating node, in case the Integrity Protected Pre-allocated Trace Option-Type is used, does not update the "node data list" array, then the decapsulating node MUST NOT update the ICV field in the IOAM Integrity Protection header.

The decapsulating node MUST NOT add the IOAM Integrity Protection header. Also, the decapsulating node MUST NOT modify the Method-ID field, the Nonce-Length field, and the Nonce field in the IOAM Integrity Protection header.

5.4. Validator

A node that performs the validation of the integrity protection is referred to as the Validator. This method assumes that symmetric keys have been distributed to the Validator only. The details of the mechanisms used for key distribution are outside the scope of this document. We refer to the method as "Zero Trust" in the sense that neither the encapsulating node, nor transit nodes, nor any other non-IOAM nodes need to be trusted. The Validator is the only point of trust, meaning the method is considered a full integrity protection of IOAM-Data-Fields.

The Validator MUST recompute the ICV by iteratively following the previous steps of the method in the same order, using the respective symmetric keys received previously. The recomputed ICV is then compared to the received ICV field. As a result, the Validator can

detect if the integrity of IOAM-Data-Fields is intact or altered. The validation is one-step in some cases (e.g., with POT Type-0 or E2E), where only the encapsulating node updates the ICV, according to the definition of this method. For other cases where transit nodes also update the ICV (e.g., with Trace Option-Types), the Validator MUST identify these transit nodes in order to look up their respective keys. For that, a unique identifier of the node, such as the "node_id" for Trace Option-Types, MUST be included in IOAM-Data-Fields. Whatever the Option-Type, the nonce allows the encapsulating node to be identified (see Section 5.1.1). Details on how the mapping between those identifiers and keys is implemented on the Validator are outside the scope of this document.

The Validator MUST NOT update the ICV field in the IOAM Integrity Protection header. Since its role is to validate the integrity of IOAM-Data-Fields, it MUST trust itself.

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 (suggested) IOAM Integrity Protected Pre-allocated Trace Option-Type (see Section 4)

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

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

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

New IOAM Integrity Protected Option-Types that intend to use the Integrity Protection Method defined in this document MUST also specify a list of corresponding Option-Type header fields that participate in the integrity protection of IOAM-Data-Fields. See Section 5.1.2 as an example.

6.2. IOAM Integrity Protection Method Suite

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

This new registry defines 256 code points to identify IOAM Integrity Protection methods. The following code points are defined in this document:

ID	Description	Reference
0x00	Reserved	This document
0x01	Zero Trust (AES-GMAC, 16-octet ICV)	This document
0x02 ... 0xFE	Unassigned	
0xFF	Reserved	This document

Figure 4: IOAM Integrity Protection Method 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, a description of the method, and a reference to the document defining the 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.

There is an additional per-packet processing for each node that uses the Integrity Protection Method defined in this document. Inappropriate use of this Integrity Protection Method might overload nodes and cause them to stop functioning properly. Operators deploying IOAM with this 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, keeping in mind that only that IOAM subset would be integrity protected. When enabled, the integrity protection MUST be applied to the entire IOAM set, not a subset.

A compromised transit node could remove the Integrity Protection header and replace the IOAM Option-Type value with an equivalent one that has no integrity protection, in order to be able to modify IOAM-Data-Fields and bypass the Validator. Also, a compromised IOAM transit node could reinitialize both the Nonce and ICV fields in the Integrity Protection header, in order to pretend to be an encapsulating node and fool the Validator. To avoid such situations, the Validator MUST know all IOAM Namespace-IDs for which the integrity protection is enabled. For each of those Namespace-IDs, the Validator MUST know the corresponding IOAM encapsulating nodes. Implementation details are outside the scope of this document.

To ensure the integrity protection of IOAM-Data-Fields, the Integrity Protection Method defined in this document uses AES-GMAC ([AES] [NIST.800-38D]). In that context, a generated key MUST be fresh. Another important requirement is that the same combination of a nonce (i.e., the GMAC IV) and a key MUST NOT be used more than once. Otherwise, security guarantees are destroyed. This document specifies some requirements for the nonce (see Section 5.1.1) to ensure that those security properties are respected, and also to optimize key usage and avoid frequent key rotation. As an example, based on these nonce requirements, it would take 7 years for a key to reach the limit of 2^64 with 1500-byte packets on a 1 Pbps (Petabits per second) link at line rate, while it would take 170 days with 100-byte packets.

The nonce makes an ICV (i.e., the GMAC Authentication Tag) unique but does not necessarily prevent replay attacks. To enable replay protection, the encapsulating node and the Validator 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. However, a suggestion would be to put a 64-bit timestamp in the Invocation field of the nonce (see Section 5.1.1).

The Integrity Protection Method defined in this document is intended for Intra-IOAM-Domain use cases (i.e., no confidentiality, integrity protection only). For Inter-IOAM-Domain use cases, operators can use IPSec to securely transfer IOAM-Data-Fields between IOAM-Domains.

8. Acknowledgements

The authors would like to thank Santhosh N, Rakesh Kandula, Saiprasad Muchala, Al Morton, Greg Mirsky, Benjamin Kaduk, Mehmet Beyaz, Martin Duke, and Tianran Zhou 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>.

Authors' Addresses

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

Shwetha Bhandari
Databricks
Angkor West Building, Bagmane Capital Tech Park, Ferns City
Doddanekkundi, Mahadevpura, Bengaluru, Karnataka 560048
India
Email: shwetha.bhandari@databricks.com

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

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