Stream: Internet Engineering Task Force (IETF)
RFC: 9145
Category: Standards Track
Published: December 2021
ISSN: 2070-1721
Authors: M. Boucadair, *Orange*; T. Reddy.K, *Akamai*; D. Wing, *Citrix*

RFC 9145
Integrity Protection for the Network Service Header (NSH) and Encryption of Sensitive Context Headers

Abstract

This specification presents an optional method to add integrity protection directly to the Network Service Header (NSH) used for Service Function Chaining (SFC). Also, this specification allows for the encryption of sensitive metadata (MD) that is carried in the NSH.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9145.

Table of Contents

1. Introduction

Many advanced Service Functions (SFs) are enabled for the delivery of value-added services. Typically, SFs are used to meet various service objectives such as IP address sharing, avoiding covert channels, detecting Denial-of-Service (DoS) attacks and protecting network infrastructures against them, network slicing, etc. Because of the proliferation of such advanced SFs together with complex service deployment constraints that demand more agile service delivery procedures, operators need to rationalize their service delivery logic and control its complexity while optimizing service activation time cycles. The overall problem space is described in [RFC7498].

[RFC7665] presents a data plane architecture addressing the problematic aspects of existing service deployments, including topological dependence and configuration complexity. It also describes an architecture for the specification, creation, and maintenance of Service Function Chains (SFCs) within a network, that is, how to define an ordered set of SFs and ordering constraints that must be applied to packets/flows selected as a result of traffic classification. [RFC8300] specifies the SFC encapsulation: Network Service Header (NSH).

The NSH data is unauthenticated and unencrypted, forcing a service topology that requires security and privacy to use a transport encapsulation that supports such features (Section 8.2 of [RFC8300]).

Note that some transport encapsulations (e.g., IPsec) only provide hop-by-hop security between two SFC data plane elements (e.g., two Service Function Forwarders (SFFs), SFF to SF) and do not provide SF-to-SF security of NSH metadata. For example, if IPsec is used, SFFs or SFs within a Service Function Path (SFP) that are not authorized to access the sensitive metadata (e.g., privacy-sensitive information) will have access to the metadata. As a reminder, the metadata referred to is information that is inserted by Classifiers or intermediate SFs and shared with downstream SFs; such information is not visible to the communication endpoints (Section 4.9 of [RFC7665]).

The lack of such capability was reported during the development of [RFC8300] and [RFC8459]. The reader may refer to Section 3.2.1 of [INTERNET-THREAT-MODEL] for a discussion on the need for more awareness about attacks from within closed domains.

This specification fills that gap for SFC (that is, it defines the "NSH Variable Header-Based Integrity" option mentioned in Section 8.2.1 of [RFC8300]). Concretely, this document adds integrity protection and optional encryption of sensitive metadata directly to the NSH (Section 4). The integrity protection covers the packet payload and provides replay protection (Section 7.4). Thus, the NSH does not have to rely upon an underlying transport encapsulation for security.

This specification introduces new Variable-Length Context Headers to carry fields necessary for integrity-protected NSH headers and encrypted Context Headers (Section 5). This specification is only applicable to NSH MD Type 0x02 (Section 2.5 of [RFC8300]). MTU considerations are discussed in Section 8. This specification is not applicable to NSH MD Type 0x01 (Section 2.4 of [RFC8300]) because that MD Type only allows a Fixed-Length Context Header whose size is 16 bytes; that is not sufficient to accommodate both the metadata and message integrity of the NSH data.

This specification limits access to NSH-supplied information along an SFP to entities that have a need to interpret it.

The mechanism specified in this document does not preclude the use of transport security. Such considerations are deployment specific.

It is out of the scope of this document to specify an NSH-aware control plane solution.

2. Terminology

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.

This document makes use of the terms defined in [RFC7665] and [RFC8300]. The term "transport encapsulation" used in this document refers to the outer encapsulation (e.g., Generic Routing Encapsulation (GRE), IPsec, and Generic Protocol Extension for Virtual eXtensible Local Area Network (VXLAN-GPE)) that is used to carry NSH-encapsulated packets as per Section 4 of [RFC8300].

The document defines the following terms:

SFC data plane element: Refers to NSH-aware SF, SFF, the SFC Proxy, or the Classifier as defined in the SFC data plane architecture [RFC7665] and further refined in [RFC8300].

SFC control element: Is a logical entity that instructs one or more SFC data plane elements on how to process NSH packets within an SFC-enabled domain.

Key Identifier: Is used to identify keys to authorized entities. See, for example, "kid" usage in [RFC7635].

NSH data: The NSH is composed of a Base Header, a Service Path Header, and optional Context Headers. NSH data refers to all the above headers and the packet or frame on which the NSH is imposed to realize an SFP.

NSH imposer: Refers to an SFC data plane element that is entitled to impose the NSH with the Context Headers defined in this document.

3. Assumptions and Basic Requirements

Section 2 of [RFC8300] specifies that the NSH data can be spread over three headers:

Base Header: Provides information about the service header and the payload protocol.

Service Path Header: Provides path identification and location within an SFP.

Context Header(s): Carries metadata (i.e., context data) along a service path.

The NSH allows sharing context information (a.k.a. metadata) with downstream NSH-aware data plane elements on a per-SFC/SFP basis. To that aim:

- The Classifier is instructed by an SFC control element about the set of context information to be supplied for a given service function chain.
- An NSH-aware SF is instructed by an SFC control element about any metadata it needs to attach to packets for a given service function chain. This instruction may occur any time during the validity lifetime of an SFC/SFP. For a given service function chain, the NSH-aware SF is also provided with an order for consuming a set of contexts supplied in a packet.
- An NSH-aware SF can also be instructed by an SFC control element about the behavior it should adopt after consuming context information that was supplied in the NSH. For example, the context information can be maintained, updated, or stripped.
- An SFC Proxy may be instructed by an SFC control element about the behavior it should adopt to process the context information that was supplied in the NSH on behalf of an NSH-unaware SF (e.g., the context information can be maintained or stripped). The SFC Proxy may also be instructed to add some new context information into the NSH on behalf of an NSH-unaware SF.

In reference to Table 1:

- Classifiers, NSH-aware SFs, and SFC proxies are entitled to update the Context Header(s).
- Only NSH-aware SFs and SFC proxies are entitled to update the Service Path Header.
- SFFs are entitled to modify the Base Header (TTL value, for example). Nevertheless, SFFs are not supposed to act on the Context Headers or look into the content of the Context Headers (Section 4.3 of [RFC7665]).

Thus, the following requirements:

- Only Classifiers, NSH-aware SFs, and SFC proxies must be able to encrypt and decrypt a given Context Header.

- Both encrypted and unencrypted Context Headers may be included in the same NSH.
- The solution must provide integrity protection for the Service Path Header.
- The solution must provide optional integrity protection for the Base Header. The implications of disabling such checks are discussed in Section 9.1.

SFC Data Plane Element	Insert, remove, or replace the NSH			Update the NSH	
	Insert	Remove	Replace	Decrement Service Index	Update Context Header(s)
Classifier	+		+		+
Service Function Forwarder (SFF)		+			
Service Function (SF)				+	+
Service Function Chaining (SFC) Proxy	+	+		+	+

Table 1: Summary of NSH Actions

4. Design Overview

4.1. Supported Security Services

This specification provides the functions described in the following subsections.

4.1.1. Encrypt All or a Subset of Context Headers

The solution allows encrypting all or a subset of NSH Context Headers by Classifiers, NSH-aware SFs, and SFC proxies.

As depicted in Table 2, SFFs are not involved in data encryption.

Data Plane Element	Base and Service Path Headers Encryption	Context Header Encryption
Classifier	No	Yes
SFF	No	No
NSH-aware SF	No	Yes

Data Plane Element	Base and Service Path Headers Encryption	Context Header Encryption
SFC Proxy	No	Yes
NSH-unaware SF	No	No

Table 2: Encryption Function Supported by SFC Data Plane Elements

Classifier(s), NSH-aware SFs, and SFC proxies are instructed with the set of Context Headers (privacy-sensitive metadata, typically) that must be encrypted. Encryption keying material is only provided to these SFC data plane elements.

The control plane may indicate the set of SFC data plane elements that are entitled to supply a given Context Header (e.g., in reference to their identifiers as assigned within the SFC-enabled domain). It is out of the scope of this document to elaborate on how such instructions are provided to the appropriate SFC data plane elements nor to detail the structure used to store the instructions.

The Service Path Header (Section 2 of [RFC8300]) is not encrypted because SFFs use the Service Index (SI) in conjunction with the Service Path Identifier (SPI) for determining the next SF in the path.

4.1.2. Integrity Protection

The solution provides integrity protection for the NSH data. Two levels of assurance (LoAs) are supported.

The first level of assurance is where all NSH data except the Base Header are integrity protected (Figure 1). In this case, the NSH imposer may be a Classifier, an NSH-aware SF, or an SFC Proxy. SFFs are not provided with authentication material. Further details are discussed in Section 5.1.

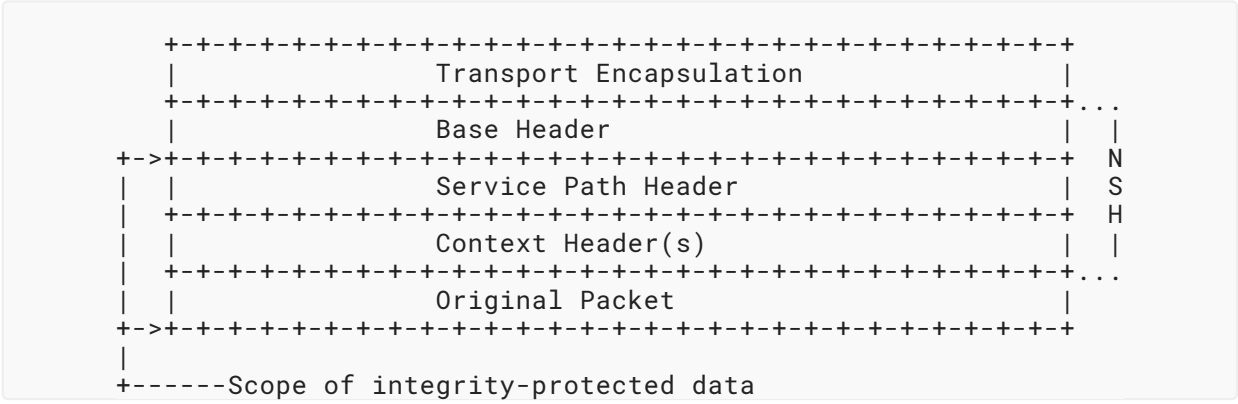

Figure 1: First Level of Assurance

The second level of assurance is where all NSH data, including the Base Header, are integrity protected (Figure 2). In this case, the NSH imposer may be a Classifier, an NSH-aware SF, an SFF, or an SFC Proxy. Further details are provided in Section 5.2.

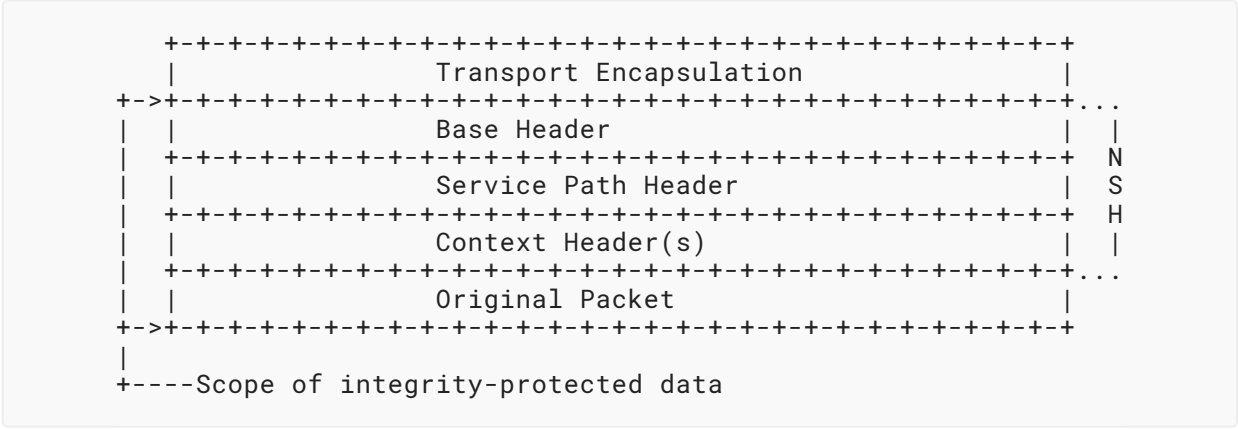

Figure 2: Second Level of Assurance

The integrity-protection scope is explicitly signaled to NSH-aware SFs, SFFs, and SFC proxies in the NSH by means of a dedicated MD Type (Section 5).

In both levels of assurance, the Context Headers and the packet on which the NSH is imposed are subject to integrity protection.

Table 3 classifies the data plane elements as being involved or not involved in providing integrity protection for the NSH.

Data Plane Element	Integrity Protection
Classifier	Yes
SFF	No (first LoA); Yes (second LoA)
NSH-aware SF	Yes
SFC Proxy	Yes
NSH-unaware SF	No

Table 3: Integrity Protection Supported by SFC Data Plane Elements

4.2. One Secret Key, Two Security Services

The Authenticated Encryption with Associated Data (AEAD) interface defined in Section 5 of [RFC5116] is used to encrypt the Context Headers that carry sensitive metadata and to provide integrity protection for the encrypted Context Headers.

The secondary keys "MAC_KEY" and "ENC_KEY" are generated from the input secret key (K) as follows; each of these two keys is an octet string:

MAC_KEY: Consists of the initial MAC_KEY_LEN octets of K, in order. The MAC_KEY is used for calculating the message integrity of the NSH data. In other words, the integrity protection provided by the cryptographic mechanism is extended to also provide protection for the unencrypted Context Headers and the packet on which the NSH is imposed.

ENC_KEY: Consists of the final ENC_KEY_LEN octets of K, in order. The ENC_KEY is used as the symmetric encryption key for encrypting the Context Headers.

The Hashed Message Authentication Code (HMAC) algorithm discussed in [RFC4868] is used to protect the integrity of the NSH data. The algorithm for implementing and validating HMACs is provided in [RFC2104].

The advantage of using both AEAD and HMAC algorithms (instead of just AEAD) is that NSH-aware SFs and SFC proxies only need to recompute the message integrity of the NSH data after decrementing the SI and do not have to recompute the ciphertext. The other advantage is that SFFs do not have access to the ENC_KEY and cannot act on the encrypted Context Headers, and (in the case of the second level of assurance) SFFs do have access to the MAC_KEY. Similarly, an NSH-aware SF or SFC Proxy not allowed to decrypt the Context Headers will not have access to the ENC_KEY.

The authenticated encryption algorithm or HMAC algorithm to be used by SFC data plane elements is typically controlled using the SFC control plane. Mandatory-to-implement authenticated encryption and HMAC algorithms are listed in Section 4.3.

The authenticated encryption process takes four inputs, each of which is an octet string: a secret key (ENC_KEY, referred to as "K" in [RFC5116]), a plaintext (P), associated data (A) (which contains the data to be authenticated but not encrypted), and a nonce (N). A ciphertext (C) is generated as an output as discussed in Section 2.1 of [RFC5116].

In order to decrypt and verify, the cipher takes ENC_KEY, N, A, and C as input. The output is either the plaintext or an error indicating that the decryption failed as described in Section 2.2 of [RFC5116].

4.3. Mandatory-to-Implement Authenticated Encryption and HMAC Algorithms

Classifiers, NSH-aware SFs, and SFC proxies **MUST** implement the AES_128_GCM [GCM][RFC5116] algorithm and **SHOULD** implement the AES_192_GCM and AES_256_GCM algorithms.

Classifiers, NSH-aware SFs, and SFC proxies **MUST** implement the HMAC- SHA-256-128 algorithm and **SHOULD** implement the HMAC-SHA-384-192 and HMAC-SHA-512-256 algorithms.

SFFs **MAY** implement the aforementioned cipher suites and HMAC algorithms.

> Note: The use of the AES_128_CBC_HMAC_SHA_256 algorithm would have avoided the need for a second 128-bit authentication tag, but due to the nature of how Cipher Block Chaining (CBC) mode operates, AES_128_CBC_HMAC_SHA_256 allows for the malleability of plaintexts. This malleability allows for attackers that know the Message Authentication Code (MAC) key but

not the encryption key to make changes in the ciphertext and, if parts of the plaintext are known, create arbitrary blocks of plaintext. This specification mandates the use of separate AEAD and MAC protections to prevent this type of attack.

4.4. Key Management

The procedure for the allocation/provisioning of secret keys (K) and the authenticated encryption algorithm or MAC_KEY and HMAC algorithm is outside the scope of this specification. As such, this specification does not mandate the support of any specific mechanism.

The document does not assume nor preclude the following:

- The same keying material is used for all the service functions used within an SFC-enabled domain.
- Distinct keying material is used per SFP by all involved SFC data path elements.
- Per-tenant keys are used.

In order to accommodate deployments relying upon keying material per SFC/SFP and also the need to update keys after encrypting NSH data for a certain amount of time, this document uses key identifiers to unambiguously identify the appropriate keying material and associated algorithms for MAC and encryption. This use of in-band identifiers addresses the problem of the synchronization of keying material.

Additional information on manual vs. automated key management and when one should be used over the other can be found in [RFC4107].

The risk involved in using a group-shared symmetric key increases with the size of the group to which it is shared. Additional security issues are discussed in Section 9.

4.5. New NSH Variable-Length Context Headers

New NSH Variable-Length Context Headers are defined in Section 5 for NSH data integrity protection and, optionally, encryption of Context Headers carrying sensitive metadata. Concretely, an NSH imposer includes (1) the key identifier to identify the keying material, (2) the timestamp to protect against replay attacks (Section 7.4), and (3) MAC for the target NSH data (depending on the integrity-protection scope) calculated using MAC_KEY and, optionally, Context Headers encrypted using ENC_KEY.

An SFC data plane element that needs to check the integrity of the NSH data uses the MAC_KEY and HMAC algorithm for the key identifier being carried in the NSH.

An NSH-aware SF or SFC Proxy that needs to decrypt some Context Headers uses ENC_KEY and the decryption algorithm for the key identifier being carried in the NSH.

Section 7 specifies the detailed procedure.

4.6. Encapsulation of NSH within NSH

As discussed in Section 3 of [RFC8459], an SFC-enabled domain (called an upper-level domain) may be decomposed into many sub-domains (called lower-level domains). In order to avoid maintaining state to restore upper-level NSH information at the boundaries of lower-level domains, two NSH levels are used: an Upper-NSH that is imposed at the boundaries of the upper-level domain and a Lower-NSH that is pushed by the Classifier of a lower-level domain in front of the original NSH (Figure 3). As such, the Upper-NSH information is carried along the lower-level chain without modification. The packet is forwarded in the top-level domain according to the Upper-NSH, while it is forwarded according to the Lower-NSH in a lower-level domain.

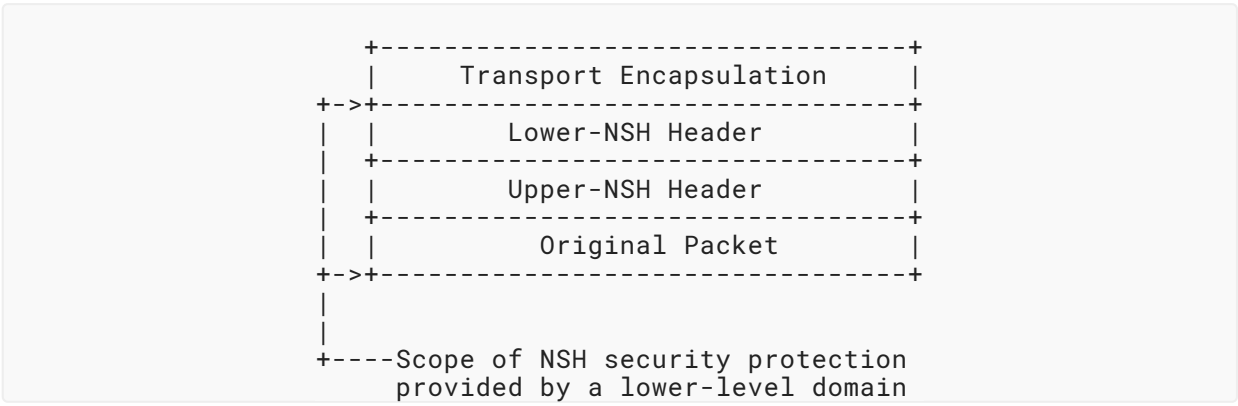

Figure 3: Encapsulation of NSH within NSH

SFC data plane elements of a lower-level domain include the Upper-NSH when computing the MAC.

Keying material used at the upper-level domain **SHOULD NOT** be the same as the one used by a lower-level domain.

5. New NSH Variable-Length Context Headers

This section specifies the format of new Variable-Length Context Headers that are used for NSH integrity protection and, optionally, Context Header encryption.

In particular, this section defines two "MAC and Encrypted Metadata" Context Headers, each having specific deployment constraints. Unlike Section 5.1, the level of assurance provided in Section 5.2 requires sharing MAC_KEY with SFFs. Both Context Headers have the same format as shown in Figure 4.

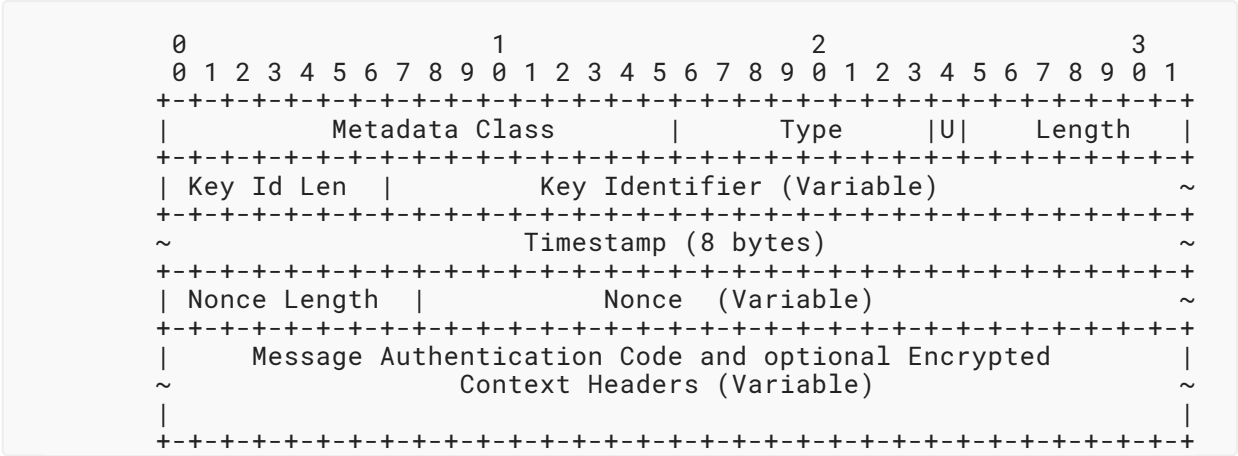

Figure 4: MAC and Encrypted Metadata Context Header

The "MAC and Encrypted Metadata" Context Headers are padded out to a multiple of 4 bytes as per Section 2.2 of [RFC8300]. The "MAC and Encrypted Metadata" Context Header, if included, **MUST** always be the last Context Header.

5.1. MAC#1 Context Header

The MAC#1 Context Header is a Variable-Length Context Header that carries MAC for the Service Path Header, Context Headers, and the inner packet on which NSH is imposed, calculated using MAC_KEY and, optionally, Context Headers encrypted using ENC_KEY. The scope of the integrity protection provided by this Context Header is depicted in Figure 5.

This MAC scheme does not require sharing MAC_KEY with SFFs. It does not require recomputing the MAC by each SFF because of TTL processing. Section 9.1 discusses the possible threat associated with this level of assurance.

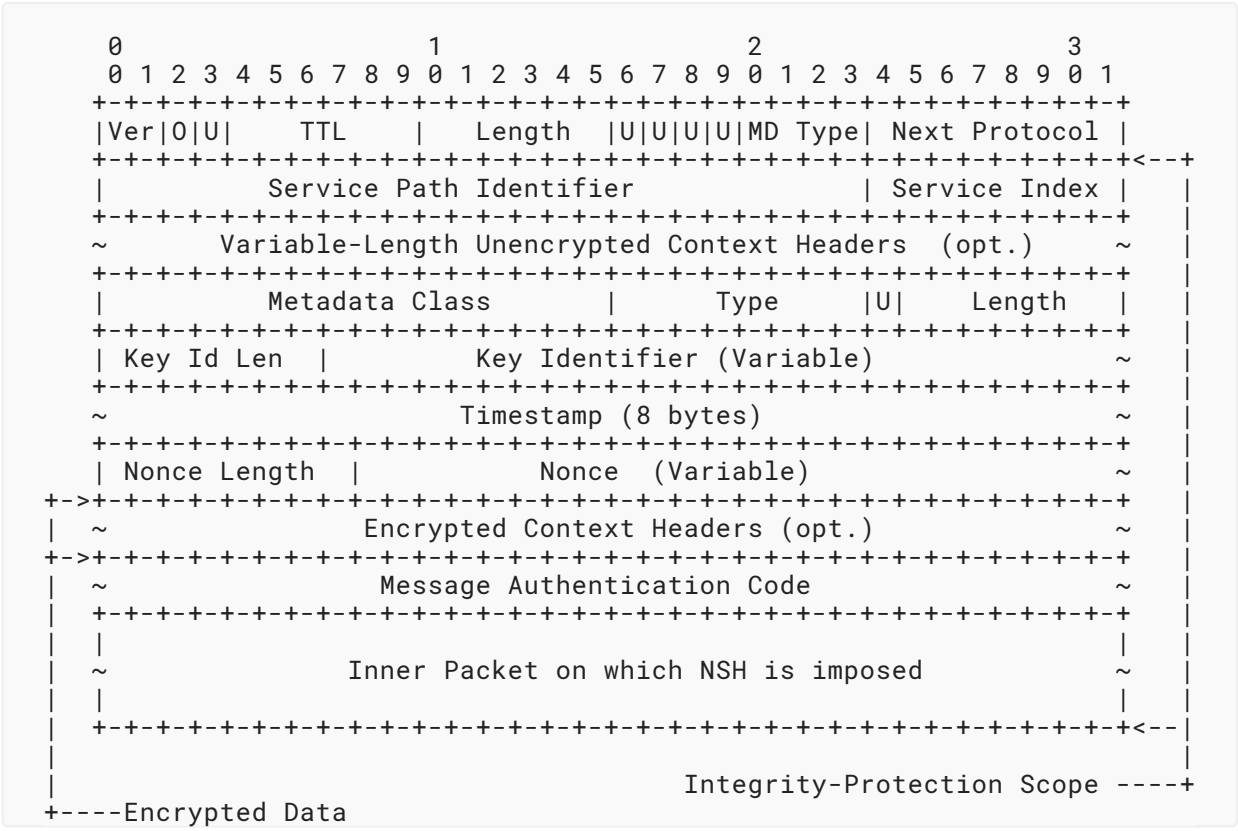

Figure 5: Scope of MAC#1

In reference to Figure 4, the description of the fields is as follows:

Metadata Class: **MUST** be set to 0x0 (Section 2.2 of [RFC8300]).

Type: 0x02 (see Section 10).

U: Unassigned bit (Section 2.5.1 of [RFC8300]).

Length: Indicates the length of the variable-length metadata in bytes. Padding considerations are discussed in Section 2.5.1 of [RFC8300].

Key Id Len: Variable. Carries the length of the key identifier in octets.

Key Identifier: Carries a variable-length Key Identifier object used to identify and deliver keys to SFC data plane elements. This identifier is helpful for accommodating deployments relying upon keying material per SFC/SFP. The key identifier helps to resolve the problem of synchronization of keying material. A single key identifier is used to look up both the ENC_KEY and the MAC_KEY associated with a key, and the corresponding encryption and MAC algorithms used with those keys.

Timestamp: Refer to Section 6 for more details about the structure of this field.

Nonce Length: Carries the length of the Nonce. If the Context Headers are only integrity protected, "Nonce Length" is set to zero (that is, no "Nonce" is included).

Nonce: Carries the Nonce for the authenticated encryption operation (Section 3.1 of [RFC5116]).

Encrypted Context Headers: Carries the optional encrypted Context Headers.

Message Authentication Code: Covers the entire NSH data, excluding the Base Header.

5.2. MAC#2 Context Header

The MAC#2 Context Header is a Variable-Length Context Header that carries the MAC for the entire NSH data calculated using MAC_KEY and, optionally, Context Headers encrypted using ENC_KEY. The scope of the integrity protection provided by this Context Header is depicted in Figure 6.

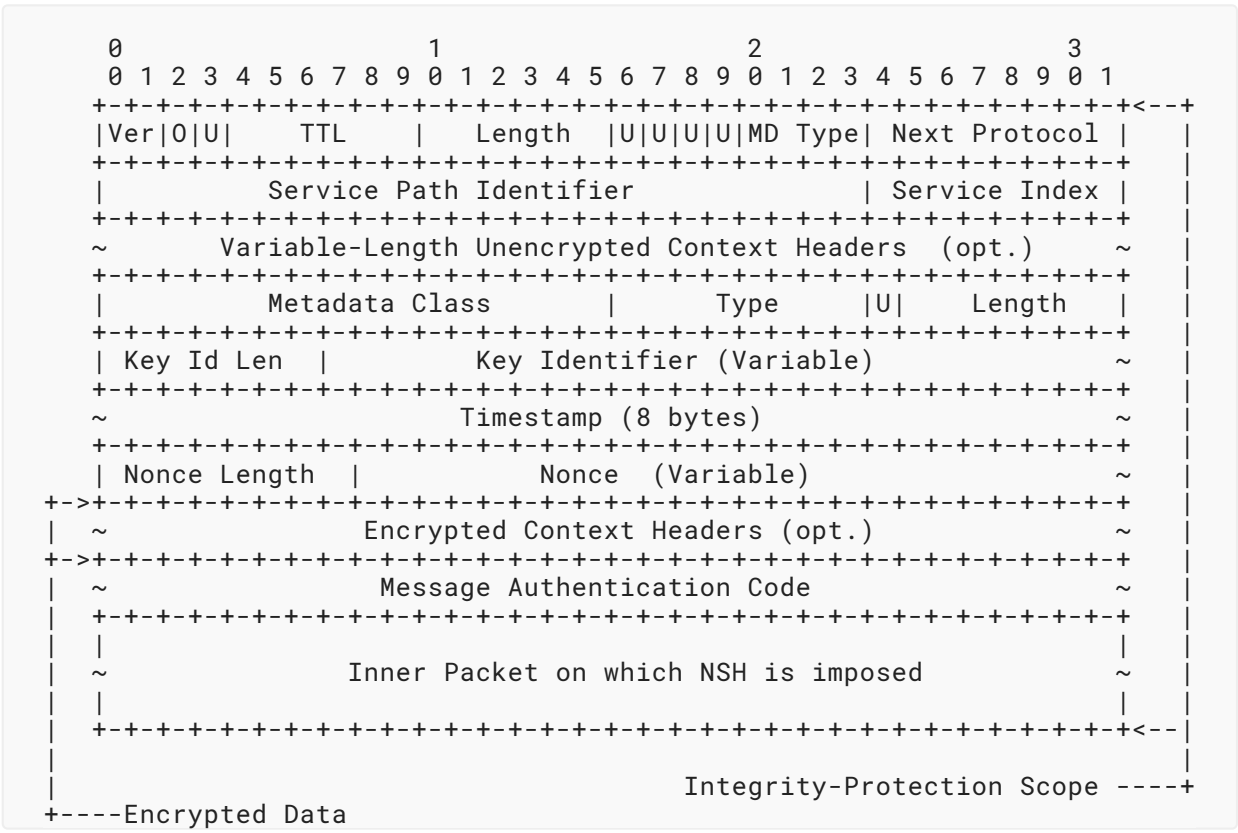

Figure 6: Scope of MAC#2

In reference to Figure 4, the description of the fields is as follows:

Metadata Class: **MUST** be set to 0x0 (Section 2.2 of [RFC8300]).

Type: 0x03 (see Section 10).

U: Unassigned bit (Section 2.5.1 of [RFC8300]).

Length: Indicates the length of the variable-length metadata in bytes. Padding considerations are discussed in Section 2.5.1 of [RFC8300].

Key Id Len: See Section 5.1.

Key Identifier: See Section 5.1.

Timestamp: See Section 6.

Nonce Length: See Section 5.1.

Nonce: See Section 5.1.

Encrypted Context Headers: Carries the optional encrypted Context Headers.

Message Authentication Code: Covers the entire NSH data.

6. Timestamp Format

This section follows the template provided in Section 3 of [RFC8877].

The format of the Timestamp field introduced in Section 5 is depicted in Figure 7.

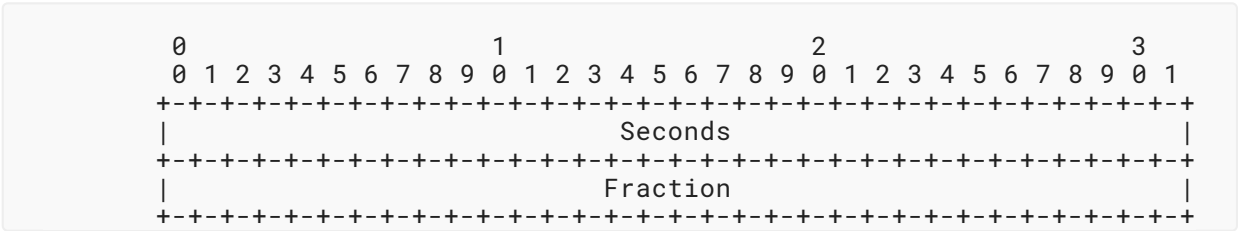

Figure 7: Timestamp Field Format

Timestamp field format:

Seconds: Specifies the integer portion of the number of seconds since the epoch.

+ Size: 32 bits

+ Units: Seconds

Fraction: Specifies the fractional portion of the number of seconds since the epoch.

+ Size: 32 bits

+ Units: The unit is $2^{(-32)}$ seconds, which is roughly equal to 233 picoseconds.

Epoch:

The epoch is 1970-01-01T00:00 in UTC time. Note that this epoch value is different from the one used in Section 6 of [RFC5905] (which will wrap around in 2036).

Leap seconds:
: This timestamp format is affected by leap seconds. The timestamp represents the number of seconds elapsed since the epoch minus the number of leap seconds.

Resolution:
: The resolution is $2^{(-32)}$ seconds.

Wraparound:
: This time format wraps around every 2^{32} seconds, which is roughly 136 years. The next wraparound will occur in the year 2106.

Synchronization aspects:
: It is assumed that SFC data plane elements are synchronized to UTC using a synchronization mechanism that is outside the scope of this document. In typical deployments, SFC data plane elements use NTP [RFC5905] for synchronization. Thus, the timestamp may be derived from the NTP-synchronized clock, allowing the timestamp to be measured with respect to the clock of an NTP server. Since this time format is specified in terms of UTC, it is affected by leap seconds (in a manner analogous to the NTP time format, which is similar). Therefore, the value of a timestamp during or slightly after a leap second may be temporarily inaccurate.

7. Processing Rules

The following subsections describe the processing rules for integrity-protected NSH and, optionally, encrypted Context Headers.

7.1. Generic Behavior

This document adheres to the recommendations in Section 8.1 of [RFC8300] for handling the Context Headers at both ingress and egress SFC boundary nodes (i.e., to strip the entire NSH, including Context Headers).

Failures of a Classifier to inject the Context Headers defined in this document **SHOULD** be logged locally while a notification alarm **MAY** be sent to an SFC control element. Failures of an NSH-aware node to validate the integrity of the NSH data **MUST** cause that packet to be discarded while a notification alarm **MAY** be sent to an SFC control element. The details of sending notification alarms (i.e., the parameters that affect the transmission of the notification alarms depending on the information in the Context Header such as frequency, thresholds, and content in the alarm) **SHOULD** be configurable.

NSH-aware SFs and SFC proxies **MAY** be instructed to strip some encrypted Context Headers from the packet or to pass the data to the next SF in the service function chain after processing the content of the Context Headers. If no instruction is provided, the default behavior for intermediary NSH-aware nodes is to maintain such Context Headers so that the information can be passed to the next NSH-aware hops. NSH-aware SFs and SFC proxies **MUST** reapply the integrity protection if any modification is made to the Context Headers (e.g., strip a Context Header, update the content of an existing Context Header, insert a new Context Header).

An NSH-aware SF or SFC Proxy that is not allowed to decrypt any Context Headers **MUST NOT** be given access to the ENC_KEY.

Otherwise, an NSH-aware SF or SFC Proxy that receives encrypted Context Headers, for which it is not allowed to consume a specific Context Header it decrypts (but consumes others), **MUST** keep that Context Header unaltered when forwarding the packet upstream.

Only one instance of a "MAC and Encrypted Metadata" Context Header (Section 5) is allowed in an NSH level. If multiple instances of a "MAC and Encrypted Metadata" Context Header are included in an NSH level, the SFC data plane element **MUST** process the first instance and ignore subsequent instances and **MAY** log or increase a counter for this event as per Section 2.5.1 of [RFC8300]. If NSH within NSH is used (Section 4.6), distinct LoAs may be used for each NSH level.

MTU and fragmentation considerations are discussed in Section 8.

7.2. MAC NSH Data Generation

After performing any Context Header encryption, the HMAC algorithm discussed in [RFC4868] is used to integrity protect the target NSH data. An NSH imposer inserts a "MAC and Encrypted Metadata" Context Header for integrity protection (Section 5).

The NSH imposer sets the MAC field to zero and then computes the message integrity for the target NSH data (depending on the integrity-protection scope discussed in Section 5) using the MAC_KEY and HMAC algorithm. It inserts the computed digest in the MAC field of the "MAC and Encrypted Metadata" Context Header. The length of the MAC is decided by the HMAC algorithm adopted for the particular key identifier.

The Message Authentication Code (T) computation process for the target NSH data with HMAC-SHA-256-128() can be illustrated as follows:

```
T = HMAC-SHA-256-128(MAC_KEY, target NSH data)
```

An entity in the SFP that updates the NSH **MUST** follow the above behavior to maintain message integrity of the NSH for subsequent validations.

7.3. Encrypted NSH Metadata Generation

An NSH imposer can encrypt Context Headers carrying sensitive metadata, i.e., encrypted and unencrypted metadata may be carried simultaneously in the same NSH packet (Sections 5 and 6).

In order to prevent pervasive monitoring [RFC7258], it is **RECOMMENDED** to encrypt all Context Headers. All Context Headers carrying privacy-sensitive metadata **MUST** be encrypted; by doing so, privacy-sensitive metadata is not revealed to attackers. Privacy-specific threats are discussed in Section 5.2 of [RFC6973].

Using the secret key (ENC_KEY) and authenticated encryption algorithm, the NSH imposer encrypts the Context Headers (as set, for example, in Section 3) and inserts the resulting payload in the "MAC and Encrypted Metadata" Context Header (Section 5). The additional authenticated

data input to the AEAD function is a zero-length byte string. The entire Context Header carrying sensitive metadata is encrypted (that is, including the MD Class, Type, Length, and associated metadata of each Context Header).

More details about the exact encryption procedure are provided in Section 2.1 of [RFC5116]. In this case, the associated data (A) input is zero length for AES Galois/Counter Mode (AES-GCM).

An authorized entity in the SFP that updates the content of an encrypted Context Header or needs to add a new encrypted Context Header **MUST** also follow the aforementioned behavior.

7.4. Timestamp for Replay Attack Prevention

The Timestamp imposed by an initial Classifier is left untouched along an SFP. However, it can be updated when reclassification occurs (Section 4.8 of [RFC7665]). The same considerations for setting the Timestamp are followed in both initial classification and reclassification (Section 6).

The received NSH is accepted by an NSH-aware node if the Timestamp (TS) in the NSH is recent enough to the reception time of the NSH (TSrt). The following formula is used for this check:

```
-Delta < (TSrt - TS) < +Delta
```

The Delta interval is a configurable parameter. The default value for the allowed Delta is 2 seconds. Special care should be taken when setting very low Delta values as this may lead to dropping legitimate traffic. If the timestamp is not within the boundaries, then the SFC data plane element receiving such packets **MUST** discard the NSH message.

Replay attacks within the Delta window may be detected by an NSH-aware node by recording a unique value derived from the packet, such as a unique value from the MAC field value. Such an NSH-aware node will detect and reject duplicates. If for legitimate service reasons some flows have to be duplicated but still share a portion of an SFP with the original flow, legitimate duplicate packets will be tagged by NSH-aware nodes involved in that segment as replay packets unless sufficient entropy is added to the duplicate packet. How such an entropy is added is implementation specific.

> Note: Within the timestamp Delta window, defining a sequence number to protect against replay attacks may be considered. In such a mode, NSH-aware nodes must discard packets with duplicate sequence numbers within the timestamp Delta window. However, in deployments with several instances of the same SF (e.g., cluster or load-balanced SFs), a mechanism to coordinate among those instances to discard duplicate sequence numbers is required. Because the coordination mechanism to comply with this requirement is service specific, this document does not include this protection.

All SFC data plane elements must be synchronized among themselves. These elements may be synchronized to a global reference time.

7.5. NSH Data Validation

When an SFC data plane element that conforms to this specification needs to check the validity of the NSH data, it **MUST** ensure that a "MAC and Encrypted Metadata" Context Header is included in a received NSH packet. The imposer **MUST** silently discard the packet and **MUST** log an error at least once per the SPI if at least one of the following is observed:

- the "MAC and Encrypted Metadata" Context Header is missing,
- the enclosed key identifier is unknown or invalid (e.g., the corresponding key expired), or
- the timestamp is invalid (Section 7.4).

If the timestamp check is successfully passed, the SFC data plane element proceeds with NSH data integrity validation. After storing the value of the MAC field in the "MAC and Encrypted Metadata" Context Header, the SFC data plane element fills the MAC field with zeros. Then, the SFC data plane element generates the message integrity for the target NSH data (depending on the integrity-protection scope discussed in Section 5) using the MAC_KEY and HMAC algorithm. If the value of the newly generated digest is identical to the stored one, the SFC data plane element is certain that the NSH data has not been tampered with and validation is therefore successful. Otherwise, the NSH packet **MUST** be discarded. The comparison of the computed HMAC value to the stored value **MUST** be done in a constant-time manner to thwart timing attacks.

7.6. Decryption of NSH Metadata

If entitled to consume a supplied encrypted Context Header, an NSH-aware SF or SFC Proxy decrypts metadata using (K) and a decryption algorithm for the key identifier in the NSH.

The authenticated encryption algorithm has only a single output, either a plaintext or a special symbol (FAIL) that indicates that the inputs are not authentic (Section 2.2 of [RFC5116]).

8. MTU Considerations

The SFC architecture prescribes that additional information be added to packets to:

- Identify SFPs: this is typically the NSH Base Header and Service Path Header.
- Carry metadata such as that defined in Section 5.
- Steer the traffic along the SFPs: This is realized by means of transport encapsulation.

This added information increases the size of the packet to be carried along an SFP.

Aligned with Section 5 of [RFC8300], it is **RECOMMENDED** that network operators increase the underlying MTU so that NSH traffic is forwarded within an SFC-enabled domain without fragmentation. The available underlying MTU should be taken into account by network operators when providing SFs with the required Context Headers to be injected per SFP and the size of the data to be carried in these Context Headers.

If the underlying MTU cannot be increased to accommodate the NSH overhead, network operators may rely upon a transport encapsulation protocol with the required fragmentation handling. The impact of activating such features on SFFs should be carefully assessed by network operators (Section 5.6 of [RFC7665]).

When dealing with MTU issues, network operators should consider the limitations of various tunnel mechanisms such as those discussed in [INTERNET-IP-TUNNELS].

9. Security Considerations

Data plane SFC-related security considerations, including privacy, are discussed in Section 6 of [RFC7665] and Section 8 of [RFC8300]. In particular, Section 8.2.2 of [RFC8300] states that attached metadata (i.e., Context Headers) should be limited to that which is necessary for correct operation of the SFP. Also, that section indicates that [RFC8165] discusses metadata considerations that operators can take into account when using NSH.

The guidelines for cryptographic key management are discussed in [RFC4107]. The group key management protocol-related security considerations discussed in Section 8 of [RFC4046] need to be taken into consideration.

The interaction between the SFC data plane elements and a key management system **MUST NOT** be transmitted unencrypted since this would completely destroy the security benefits of the integrity-protection solution defined in this document.

The secret key (K) must have an expiration time assigned as the latest point in time before which the key may be used for integrity protection of NSH data and encryption of Context Headers. Prior to the expiration of the secret key, all participating NSH-aware nodes **SHOULD** have the control plane distribute a new key identifier and associated keying material so that when the secret key is expired, those nodes are prepared with the new secret key. This allows the NSH imposer to switch to the new key identifier as soon as necessary. It is **RECOMMENDED** that the next key identifier and associated keying material be distributed by the control plane well prior to the secret key expiration time. Additional guidance for users of AEAD functions about rekeying can be found in [AEAD-LIMITS].

The security and integrity of the key-distribution mechanism is vital to the security of the SFC system as a whole.

NSH data is exposed to several threats:

- An on-path attacker modifying the NSH data.
- An attacker spoofing the NSH data.
- An attacker capturing and replaying the NSH data.
- Data carried in Context Headers revealing privacy-sensitive information to attackers.
- An attacker replacing the packet on which the NSH is imposed with a modified or bogus packet.

In an SFC-enabled domain where the above attacks are possible, (1) NSH data **MUST** be integrity protected and replay protected, and (2) privacy-sensitive NSH metadata **MUST** be encrypted for confidentiality preservation purposes. The Base and Service Path Headers are not encrypted.

MACs with two levels of assurance are defined in Section 5. Considerations specific to each level of assurance are discussed in Sections 9.1 and 9.2.

The attacks discussed in [ARCH-SFC-THREATS] are handled based on the solution specified in this document, with the exception of attacks dropping packets. Such attacks can be detected by relying upon statistical analysis; such analysis is out of the scope of this document. Also, if SFFs are not involved in the integrity checks, a misbehaving SFF that decrements SI while this should be done by an SF (SF bypass attack) will be detected by an upstream SF because the integrity check will fail.

Some events are logged locally with notification alerts sent by NSH-aware nodes to a Control Element. These events **SHOULD** be rate limited.

The solution specified in this document does not provide data origin authentication.

In order to detect compromised nodes, it is assumed that appropriate mechanisms to monitor and audit an SFC-enabled domain to detect misbehavior and to deter misuse are in place. Compromised nodes can thus be withdrawn from active service function chains using appropriate control plane mechanisms.

9.1. MAC#1

An active attacker can potentially modify the Base Header (e.g., decrement the TTL so the next SFF in the SFP discards the NSH packet). An active attacker can typically also drop NSH packets. As such, this attack is not considered an attack against the security mechanism specified in the document.

It is expected that specific devices in the SFC-enabled domain will be configured such that no device other than the Classifiers (when reclassification is enabled), NSH-aware SFs, and SFC proxies will be able to update the integrity-protected NSH data (Section 7.1), and no device other than the NSH-aware SFs and SFC proxies will be able to decrypt and update the Context Headers carrying sensitive metadata (Section 7.1). In other words, it is expected that the NSH-aware SFs and SFC proxies in the SFC-enabled domain are considered fully trusted to act on the NSH data. Only these elements can have access to sensitive NSH metadata and the keying material used to integrity protect NSH data and encrypt Context Headers.

9.2. MAC#2

SFFs can detect whether an illegitimate node has altered the content of the Base Header. Such messages must be discarded with appropriate logs and alarms generated (see Section 7.1).

Similar to Section 9.1, no device other than the NSH-aware SFs and SFC proxies in the SFC-enabled domain should be able to decrypt and update the Context Headers carrying sensitive metadata.

9.3. Time Synchronization

[RFC8633] describes best current practices to be considered in deployments where SFC data plane elements use NTP for time-synchronization purposes.

Also, a mechanism to provide cryptographic security for NTP is specified in [RFC8915].

10. IANA Considerations

IANA has added the following types to the "NSH IETF-Assigned Optional Variable-Length Metadata Types" registry (0x0000 IETF Base NSH MD Class) at <https://www.iana.org/assignments/nsh>.

Value	Description	Reference
0x02	MAC and Encrypted Metadata #1	RFC 9145
0x03	MAC and Encrypted Metadata #2	RFC 9145

Table 4: Additions to the NSH IETF-Assigned Optional Variable-Length Metadata Types Registry

11. References

11.1. Normative References

[GCM] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, DOI 10.6028/NIST.SP.800-38D, November 2007, <https://doi.org/10.6028/NIST.SP.800-38D>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <https://www.rfc-editor.org/info/rfc4107>.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <https://www.rfc-editor.org/info/rfc4868>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/info/rfc5116>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[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>.

[RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <https://www.rfc-editor.org/info/rfc8300>.

11.2. Informative References

[AEAD-LIMITS] Günther, F., Thomson, M., and C. A. Wood, "Usage Limits on AEAD Algorithms", Work in Progress, Internet-Draft, draft-irtf-cfrg-aead-limits-03, 12 July 2021, <https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-aead-limits-03>.

[ARCH-SFC-THREATS] THANG, N. C. and M. Park, "A Security Architecture Against Service Function Chaining Threats", Work in Progress, Internet-Draft, draft-nguyen-sfc-security-architecture-00, 24 November 2019, <https://datatracker.ietf.org/doc/html/draft-nguyen-sfc-security-architecture-00>.

[INTERNET-IP-TUNNELS] Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", Work in Progress, Internet-Draft, draft-ietf-intarea-tunnels-10, 12 September 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-intarea-tunnels-10>.

[INTERNET-THREAT-MODEL] Arkko, J. and S. Farrell, "Challenges and Changes in the Internet Threat Model", Work in Progress, Internet-Draft, draft-arkko-farrell-arch-model-t-04, 13 July 2020, <https://datatracker.ietf.org/doc/html/draft-arkko-farrell-arch-model-t-04>.

[RFC4046] Baugher, M., Canetti, R., Dondeti, L., and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046, DOI 10.17487/RFC4046, April 2005, <https://www.rfc-editor.org/info/rfc4046>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC7498] Quinn, P., Ed. and T. Nadeau, Ed., "Problem Statement for Service Function Chaining", RFC 7498, DOI 10.17487/RFC7498, April 2015, <https://www.rfc-editor.org/info/rfc7498>.

[RFC7635] Reddy, T., Patil, P., Ravindranath, R., and J. Uberti, "Session Traversal Utilities for NAT (STUN) Extension for Third-Party Authorization", RFC 7635, DOI 10.17487/RFC7635, August 2015, <https://www.rfc-editor.org/info/rfc7635>.

[RFC8165] Hardie, T., "Design Considerations for Metadata Insertion", RFC 8165, DOI 10.17487/RFC8165, May 2017, <https://www.rfc-editor.org/info/rfc8165>.

[RFC8459] Dolson, D., Homma, S., Lopez, D., and M. Boucadair, "Hierarchical Service Function Chaining (hSFC)", RFC 8459, DOI 10.17487/RFC8459, September 2018, <https://www.rfc-editor.org/info/rfc8459>.

[RFC8633] Reilly, D., Stenn, H., and D. Sibold, "Network Time Protocol Best Current Practices", BCP 223, RFC 8633, DOI 10.17487/RFC8633, July 2019, <https://www.rfc-editor.org/info/rfc8633>.

[RFC8877] Mizrahi, T., Fabini, J., and A. Morton, "Guidelines for Defining Packet Timestamps", RFC 8877, DOI 10.17487/RFC8877, September 2020, <https://www.rfc-editor.org/info/rfc8877>.

[RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <https://www.rfc-editor.org/info/rfc8915>.

Acknowledgements

This document was created as a follow-up to the discussion in IETF 104: <https://datatracker.ietf.org/meeting/104/materials/slides-104-sfc-sfc-chair-slides-01> (slide 7).

Thanks to Joel Halpern, Christian Jacquenet, Dirk von Hugo, Tal Mizrahi, Daniel Migault, Diego Lopez, Greg Mirsky, and Dhruv Dhody for the comments.

Many thanks to Steve Hanna for the valuable secdir review and Juergen Schoenwaelder for the opsdir review.

Thanks to Greg Mirsky for the Shepherd review.

Thanks to Alvaro Retana, Lars Eggert, Martin Duke, Erik Kline, Zaheduzzaman Sarker, Éric Vyncke, Roman Danyliw, Murray Kucherawy, John Scudder, and Benjamin Kaduk for the IESG review.

Authors' Addresses

Mohamed Boucadair
Orange
35000 Rennes
France
Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy.K
Akamai
Embassy Golf Link Business Park
Bangalore 560071
Karnataka
India
Email: kondtir@gmail.com

Dan Wing
Citrix Systems, Inc.
United States of America
Email: dwing-ietf@fuggles.com