SFC M. Boucadair
Internet-Draft Orange
Intended status: Standards Track T. Reddy
Expires: July 26, 2021 McAfee
D. Wing
Citrix
January 22, 2021

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

draft-ietf-sfc-nsh-integrity-03

Abstract

This specification adds integrity protection directly to the Network Service Header (NSH) used for Service Function Chaining (SFC). Also, this specification allows to encrypt sensitive metadata that is carried in the NSH.

Status of This Memo

Copyright Notice

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 logics and master their complexity while optimising 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 [RFC8300], forcing a service topology that requires security and privacy to use a transport encapsulation that supports such features. Note that some transport encapsulation (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) not authorized to access the privacy-sensitive metadata will have access to the metadata. As a reminder, the metadata referred to is an 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 [I-D.arkko-farrell-arch-model-t] for a discussion on the need for more awareness about attacks from within closed domains.

This specification fills that gap. Concretely, this document adds integrity protection and optional encryption of sensitive metadata directly to the NSH (Section 4); integrity protects 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 and confidentiality.

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 limits thus access to an information along an SFP to entities that have a need to interpret it.

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 document defines the following terms:

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

- SFC control element: 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: A key identifier used to identify and deliver 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 to share context information (a.k.a., metadata) with downstream NSH-aware data elements on a per SFC/SFP basis. To that aim:

The control plane is used to instruct the Classifier about the set of context information to be supplied for a given service function chain.

The control plane is also used to instruct an NSH-aware SF 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. The control plane may indicate, for a given service function chain, an order for consuming a set of contexts supplied in a packet.

An NSH-aware SF can also be instructed about the behavior it should adopt after consuming a context information that was supplied in the NSH. For example, the context can be maintained, updated, or stripped.

An SFC Proxy may be instructed 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 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 Figure 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 Path 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.

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. That is, some Context Headers may be protected while others do not need to be protected.

- The solution MUST provide integrity protection for the Service Path Header.

- The solution MAY provide integrity protection for the Base Header. The implications of disabling such checks are discussed in Section 9.1.

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

Figure 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 to encrypt all or a subset of NSH Context Headers by Classifiers, NSH-aware SFs, and SFC proxies.

As depicted in Table 1, SFFs are not involved in data encryption. This document enforces this design approach by encrypting Context Headers with keys that are not supplied to SFFs, thus enforcing this limitation by protocol (rather than requirements language).

Data Plane Element	Base and Service Headers Encryption	Metadata Encryption
Classifier	No	Yes
SFF	No	No
NSH-aware SF	No	Yes
SFC Proxy	No	Yes
NSH-unaware SF	No	No

Table 1: Encryption Function Supported by SFC Data Plane Elements

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

The control plane may also 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 Service Index (SI) in conjunction with 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.

A first level of assurance where all NSH data except the Base Header are integrity protected (Figure 2). In this case, the NSH imposer may be a Classifier, an NSH-aware SF, or an SFC Proxy. SFFs are not

thus provided with authentication material. Further details are discussed in Section 5.1.

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                Transport Encapsulation                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...
   |                Base Header                             |  |
+->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  N
|  |                Service Path Header                     |  S
|  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  H
|  |                Context Header(s)                       |  |
|  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...
|  |                Original Packet                         |
+->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|
+------Scope of integrity protected data
```

Figure 2: First Level of Assurance

A second level of assurance where all NSH data, including the Base Header, are integrity protected (Figure 3). 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.

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                Transport Encapsulation                 |
+->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...
|  |                Base Header                             |  |
|  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  N
|  |                Service Path Header                     |  S
|  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  H
|  |                Context Header(s)                       |  |
|  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...
|  |                Original Packet                         |
+->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|
+----Scope of integrity protected data
```

Figure 3: Second Level of Assurance

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

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

Table 2 lists the roles of SFC data plane elements 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 2: Integrity Protection Supported by SFC Data Plane Elements

4.2. One Secret Key, Two Security Services

The authenticated encryption algorithm defined in [RFC7518] is used to provide NSH data integrity and to encrypt the Context Headers that carry privacy-sensitive metadata.

The authenticated encryption algorithm provides a unified encryption and authentication operation which turns plaintext into authenticated ciphertext and vice versa. The generation of secondary keys MAC_KEY and ENC_KEY from the secret key (K) is discussed in Section 5.2.2.1 of [RFC7518]:

- o The ENC_KEY is used for encrypting the Context Headers and the message integrity of the NSH data is calculated using the MAC_KEY.

- o If the Context Headers are not encrypted, the Hashed Message Authentication Mode (HMAC) algorithm discussed in [RFC4868] is used to integrity protect the NSH data.

The advantage of using the authenticated encryption algorithm is that NSH-aware SFs and SFC proxies only need to re-compute the message integrity of the NSH data after decrementing the Service Index (SI) and do not have to re-compute 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, only in 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 as input four octet strings: a secret key (K), a plaintext (P), Additional Authenticated Data (A) (which contains the data to be authenticated, but not encrypted), and an Initialization Vector (IV). The ciphertext value (E) and the Authentication Tag value (T) are provided as outputs.

In order to decrypt and verify, the cipher takes as input K, IV, A, T, and E. The output is either the plaintext or an error indicating that the decryption failed as described in Section 5.2.2.2 of [RFC7518].

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

Classifiers, NSH-aware SFs, and SFC proxies MUST implement the AES_128_CBC_HMAC_SHA_256 algorithm and SHOULD implement the AES_192_CBC_HMAC_SHA_384 and AES_256_CBC_HMAC_SHA_512 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 AES-GCM + HMAC may have CPU and packet size implications (need for a second 128-bit authentication tag).

4.4. Key Management

The procedure for the allocation/provisioning of secret keys (K) and 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 documents does not assume nor preclude the following:

- o The same keying material is used for all the service functions used within an SFC-enabled domain.

- o Distinct keying material is used per SFP by all involved SFC data path elements.

- o 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 certain amount of time, this document uses key identifier (kid)

to unambiguously identify the appropriate keying material. Doing so allows to address the problem of 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].

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 privacy-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) the Message Authentication Code (MAC) for the target NSH data (depending on the integrity protection scope) calculated using the 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 MAC_KEY and the 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 [RFC8459], an SFC-enabled domain (called, upper-level domain) may be decomposed into many sub-domains (called, lower-level domains). In order to avoid maintaining state to restore back upper-lower NSH information at the boundaries of lower-level domains, two NSH levels are used: an Upper-NSH which 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 4). 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.

```
                 +----------------------------------+
                 |     Transport Encapsulation      |
              +->+----------------------------------+
              |  |         Lower-NSH Header         |
              |  +----------------------------------+
              |  |         Upper-NSH Header         |
              |  +----------------------------------+
              |  |           Original Packet        |
              +->+----------------------------------+
              |
              |
              +----Scope of NSH security protection
                   provided by a lower-level domain
```

Figure 4: Encapsulation of NSH within NSH

SFC data plane elements of a lower-level domain includes 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 Headers 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 Section 5.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Metadata Class       |      Type     |U|    Length   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Key Length  |         Key Identifier (Variable)               ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ~                      Timestamp (8 bytes)                      ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | IV Length     |     Initialization Vector  (Variable)         ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |     Message Authentication Code and optional Encrypted        |
    ~                     Context Headers                           ~
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: 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].

5.1. MAC#1 Context Header

MAC#1 Context Header is a variable-length Context Header that carries the Message Authentication Code (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 6.

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

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |Ver|O|U|    TTL    |   Length  |U|U|U|U|MD Type| Next Protocol |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<--+
       |          Service Path Identifier              | Service Index |   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
       |                                                               |   |
       ~       Variable-Length Unencrypted Context Headers  (opt.)     ~   |
       |                                                               |   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
       |          Metadata Class       |      Type     |U|    Length   |   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
       | Key Length  |              Key Identifier                     ~   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
       ~                      Timestamp (8 bytes)                      ~   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
       | IV Length   |           Initialization Vector                 ~   |
    +-->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
    |  ~                 Context Headers to encrypt (opt.)             ~   |
    +-->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
    |  |                                                               |   |
    |  ~               Inner Packet on which NSH is imposed            ~   |
    |  |                                                               |   |
    |  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<--|
    |                                                                      |
    |                                                                      |
    |                                                                      |
    |                                       Integrity Protection Scope ----+
    +----Encrypted Data
```

Figure 6: Scope of MAC#1

In reference to Figure 5, the description of the fields is as follows:

- Metadata Class: MUST be set to 0x0 (Section 2.5.1 of [RFC8300]).

- Type: TBD1 (See Section 10)

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

- Length: Variable. Padding considerations are discussed in Section 2.5.1 of [RFC8300].

- Key Length: Variable. Carries the length of the key identifier.

- o Key Identifier: Carries a variable length Key Identifier object used to identify and deliver keys to SFC data plane elements. This identifier is helpful to accommodate deployments relying upon keying material per SFC/SFP. The key identifier helps in resolving the problem of synchronization of keying material.

- o Timestamp: Carries an unsigned 64-bit integer value that is expressed in seconds relative to 1970-01-01T00:00Z in UTC time. See Section 6 for more details.

- o IV Length: Carries the length of the IV (Section 5.2 of [RFC7518]). If encryption is not used, IV length is set to zero (that is, no "Initialization Vector" is included).

- o Initialization Vector: Carries the IV for authenticated encryption algorithm as discussed in Section 5.2 of [RFC7518].

- o The Additional Authenticated Data (defined in [RFC7518]) MUST be the Service Path header, the unencrypted Context headers, and the inner packet on which the NSH is imposed .

- o Message Authentication Code covering the entire NSH data excluding the Base header.

5.2. MAC#2 Context Header

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 7.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<--+
   |Ver|O|U|    TTL    |   Length  |U|U|U|U|MD Type| Next Protocol |   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
   |          Service Path Identifier              | Service Index |   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
   |                                                               |   |
   ~       Variable-Length Unencrypted Context Headers  (opt.)     ~   |
   |                                                               |   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
   |          Metadata Class       |      Type     |U|    Length   |   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
   | Key Length  |              Key Identifier                     ~   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
   ~                      Timestamp (8 bytes)                      ~   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
   | IV Length   |           Initialization Vector                 |   |
+->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|  ~                 Context Headers to encrypt (opt.)             ~   |
+->+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   |
|  |                                                               |   |
|  ~               Inner Packet on which NSH is imposed            ~   |
|  |                                                               |   |
|  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+<--|
|                                                                      |
|                                                                      |
|                                                                      |
|                                        Integrity Protection Scope ----+
+----Encrypted Data
```

Figure 7: Scope of MAC#2

In reference to Figure 5, the description of the fields is as follows:

o Metadata Class: MUST be set to 0x0 (Section 2.5.1 of [RFC8300]).

o Type: TBD2 (See Section 10)

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

o Length: Variable. Padding considerations are discussed in Section 2.5.1 of [RFC8300].

o Key Length: See Section 5.1.

o Key Identifier: See Section 5.1.

- Timestamp: See Section 6.

- IV Length: See Section 5.1.

- Initialization Vector: See Section 5.1.

- The Additional Authenticated Data (defined in [RFC7518]) MUST be the entire NSH data (i.e., including the Base Header) excluding the Context Headers to be encrypted.

- Message Authentication Code covering the entire NSH data and optional encrypted Context Headers.

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 8.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Seconds                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Fraction                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: 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 1970-01-01T00:00Z in UTC time.

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 the NTP time format is affected by leap seconds, the current timestamp format is similarly affected. 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 [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 affecting the transmission of the notification alarms

depend on the information in the context header such as frequency, thresholds, and content in the alarm) SHOULD be configurable by the SFC control plane.

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 next NSH-aware hops. NSH-aware SFs and SFC proxies MUST re-apply the integrity protection if any modification is made to the Context Headers (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 "MAC and Encrypted Metadata" Context Header (Section 5) is allowed. If multiple instances of "MAC and Encrypted Metadata" Context Header are included in an NSH packet, the SFC data 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].

MTU and fragmentation considerations are discussed in Section 8.

7.2. MAC NSH Data Generation

If the Context Headers are not encrypted, 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 computes the message integrity for the target NSH data (depending on the integrity protection scope discussed in Section 5) using MAC_KEY and HMAC algorithm. It inserts the MAC in 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 can be illustrated as follows:

```
T = HMAC-SHA-256-128(MAC_KEY, A)
```

An entity in the SFP that intends to update 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 privacy-sensitive metadata, i.e., encrypted and unencrypted metadata may be carried simultaneously in the same NSH packet (Figure 9).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Ver|O|U|    TTL    |   Length  |U|U|U|U|MD Type| Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Service Path Identifier              | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~       Variable-Length Unencrypted Context Headers  (opt.)     ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                      Key Identifier                           ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                      Timestamp                                ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
~                   MAC and Encrypted Context Headers           ~
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: NSH with Encrypted and Unencrypted Metadata

In an SFC-enabled domain where pervasive monitoring [RFC7258] is possible, all Context Headers carrying privacy-sensitive metadata MUST be encrypted; doing so, privacy-sensitive metadata is not revealed to attackers. Privacy specific threats are discussed in Section 5.2 of [RFC6973].

Using K and authenticated encryption algorithm, the NSH imposer encrypts the Context Headers (as set by the control plane Section 3), computes the message integrity for the target NSH data, and inserts the resulting payload in the "MAC and Encrypted Metadata" Context Header (Section 5). The entire Context Header carrying a privacy-sensitive metadata is encrypted (that is, including the MD Class, Type, Length, and associated metadata of each Context Header).

The message Authentication Tag (T) and ciphertext (E) computation process can be illustrated as follows:

```
MAC_KEY = initial MAC_KEY_LEN octets of K,
ENC_KEY = final ENC_KEY_LEN octets of K,
E = CBC-PKCS7-ENC(ENC_KEY, P),
M = MAC(MAC_KEY, A || IV || E || AL),
T = initial T_LEN octets of M.
MAC and Encrypted Metadata = E || T
```

As specified in [RFC7518], the octet string (AL) is equal to the number of bits in the Additional Authenticated Data (A) expressed as a 64-bit unsigned big-endian integer.

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

An SFF or NSH-aware SF or SFC Proxy that only has access to the MAC_KEY, but not the ENC_KEY, computes the message Authentication Tag (T) after decrementing the TTL (by the SFF) or SI (by an SF or SFC Proxy) and replaces the Authentication Tag in the NSH with the computed Authentication Tag. Similarly, an NSH-aware SF (or SFC Proxy) that does not modify the encrypted Context headers also follows the aforementioned behavior.

The message Authentication Tag (T) computation process can be illustrated as follows:

```
M = MAC(MAC_KEY, A || IV || E || AL),
T = initial T_LEN octets of M.
```

7.4. Timestamp for Replay Attack

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 packet 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, for example, from the NSH data and Original packet (e.g., using SHA2). Such NSH-aware node will detect and reject duplicates. If for legitimate service reasons, some flows have to be duplicated but still share 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.

> Note: Within the timestamp delta window, defining a sequence number to protect against replay attacks may be considered. In such 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 receives an NSH packet, it MUST first ensure that a "MAC and Encrypted Metadata" Context Header is included. It MUST silently discard the message if the timestamp is invalid (Section 7.4). It MUST log an error at least once per the SPI for which the "MAC and Encrypted Metadata" Context Header is missing.

If the timestamp check is successfully passed, the SFC data plane element proceeds then with NSH data integrity validation. The SFC data plane element 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 for the key identifier. If the value of the newly generated digest is identical to the one enclosed in the NSH, the SFC data plane element is certain that the NSH data has not been tampered and validation is therefore successful. Otherwise, the NSH packet MUST be discarded.

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 decryption algorithm for the key identifier in the NSH.

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 5.2.2.2 of [RFC7518]).

8. MTU Considerations

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

- o Identify SFPs: this is typically the NSH Base Header and Service Path Header.
- o Carry metadata such those defined in Section 5.
- o Steer the traffic along the SFPs: 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 for network operators to 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 feature 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 transport encapsulations such as those discussed in [I-D.ietf-intarea-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 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 interaction between the SFC data plane elements and a key management system MUST NOT be transmitted in clear 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.

NSH data are exposed to several threats:

- A man-in-the-middle attacker modifying the NSH data.
- Attacker spoofing the NSH data.
- Attacker capturing and replaying the NSH data.
- Data carried in Context Headers revealing privacy-sensitive information to attackers.
- Attacker replacing the packet on which the NSH is imposed with a 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 [I-D.nguyen-sfc-security-architecture] are handled owing to the solution specified in this document, except for attacks dropping packets. Such attacks can be detected relying upon statistical analysis; such analysis is out of scope of this document. Also, if SFFs are not involved in the integrity checks, a misbehaving SFF which 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). In the meantime, an active attacker can also drop NSH packets. As such, this attack is not considered an attack against the security mechanism specified in the document.

No device other than the NSH-aware SFs in the SFC-enabled domain should be able to update the integrity protected NSH data. Similarly, 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 privacy-sensitive metadata. In other words, if 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 privacy-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).

9.3. Time Synchronization

Section 5.6 of [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

This document requests IANA to assign the following types from the "NSH IETF-Assigned Optional Variable-Length Metadata Types" (0x0000 IETF Base NSH MD Class) registry available at: https://www.iana.org/assignments/nsh/nsh.xhtml#optional-variable-length-metadata-types.

Value	Description	Reference
TBD1	MAC and Encrypted Metadata #1	[ThisDocument]
TBD2	MAC and Encrypted Metadata #2	[ThisDocument]

11. Acknowledgements

This document was edited 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, and Greg Mirsky for the comments.

Many thanks to Steve Hanna for the valuable secdir review.

12. References

12.1. 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>.

[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>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[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>.

[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>.

12.2. Informative References

[I-D.arkko-farrell-arch-model-t]
Arkko, J. and S. Farrell, "Challenges and Changes in the Internet Threat Model", draft-arkko-farrell-arch-model-t-04 (work in progress), July 2020.

[I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.

[I-D.nguyen-sfc-security-architecture]
Nguyen, T. and M. Park, "A Security Architecture Against Service Function Chaining Threats", draft-nguyen-sfc-security-architecture-00 (work in progress), November 2019.

[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>.

[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>.

Authors' Addresses

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Tirumaleswar Reddy
McAfee, Inc.
Embassy Golf Link Business Park
Bangalore, Karnataka 560071
India

Email: TirumaleswarReddy_Konda@McAfee.com

Dan Wing
Citrix Systems, Inc.
USA

Email: dwing-ietf@fuggles.com