Mboned
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2020

J. Holland
K. Rose
Akamai Technologies, Inc.
March 09, 2020

Asymmetric Manifest Based Integrity
draft-jholland-mboned-ambi-05

Abstract

This document defines Asymmetric Manifest-Based Integrity (AMBI). AMBI allows each receiver or forwarder of a stream of multicast packets to check the integrity of the contents of each packet in the data stream. AMBI operates by passing cryptographically verifiable hashes of the data packets inside manifest messages, and sending the manifests over authenticated out-of-band communication channels.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Multicast transport poses security problems that are not easily
addressed by the same security mechanisms used for unicast transport.

The "Introduction" sections of the documents describing TESLA
[RFC4082], and TESLA in SRTP [RFC4383], and TESLA with ALC and NORM
[RFC5776] present excellent overviews of the challenges unique to
multicast authentication, briefly summarized here:

- A MAC based on a symmetric shared secret cannot be used because each packet has multiple receivers that do not trust each other, and using a symmetric shared secret exposes the same secret to each receiver.

- Asymmetric per-packet signatures can handle only very low bit-rates because of the computational overhead.

- An asymmetric signature of a larger message comprising multiple packets requires reliable receipt of all such packets, something that cannot be guaranteed in a timely manner even for protocols that do provide reliable delivery, and the retransmission of which may anyway exceed the useful lifetime for data formats that can otherwise tolerate some degree of loss.

Aymmetric Manifest-Based Integrity (AMBI) defines a method for receivers or middle boxes to cryptographically authenticate and verify the integrity of a stream of packets, by communicating packet "manifests" (described in Section 2.4) via an out-of-band communication channel that provides authentication and verifiable integrity.

Each manifest contains a message digest (described in Section 2.3) for each packet in a sequence of packets from the data stream, hereafter called a "packet digest". The packet digest incorporates a cryptographic hash of the packet contents and some identifying data from the packet, according to a defined digest profile for the data stream.

Each manifest MUST be delivered in a way that provides cryptographic integrity guarantees of the authenticity of the manifest. For example, TLS could be used to deliver a stream of manifests over a unicast data stream from a set of trusted senders to each receiver, or a protocol that asymmetrically signs each message could be used to transport authenticated manifests over a multicast channel. Note that a UDP-based protocol might drop or reorder manifests while still providing authentication.

Upon successful verification of a manifest and receipt of any subset of the corresponding data packets, the receiver has proof of the integrity of the contents of the data packets that are listed in the manifest.

Authenticating the integrity of the data packets depends on:

- the authenticity of the manifests; and

- the authenticity of the digest profile used for construction of the packet digests; and

- the difficulty of generating a collision for the packet digests contained in the manifest.

This document defines a YANG [RFC7950] module that augments the DORMS [I-D.draft-jholland-mboned-dorms-01] YANG module to provide a way to communicate a digest profile, described in Section 2.3.1, for construction of the packet digests, described in Section 2.3. When obtaining the digest profile by using DORMS, the authenticity of the data stream relies on a trust relationship with the DORMS server, since that anchors the authenticity of the digest profile for constructing packet digests.

1.1. Comparison with TESLA

AMBI and TESLA [RFC4082] and [RFC5776] attempt to achieve a similar goal of authenticating the integrity of streams of multicast packets. AMBI imposes a higher overhead, as measured in the amount of extra data required, than TESLA imposes. In exchange, AMBI provides non-repudiation (which TESLA does not), and relaxes the requirement for establishing an upper bound on clock synchronization between sender and receiver.

This tradeoff enables new capabilities for AMBI, relative to TESLA. In particular, when receiving multicast traffic from an untrusted transit network, AMBI can be used by a middle box to authenticate packets from a trusted source before forwarding traffic through the network, and the receiver also can separately authenticate the packets it receives.

This use case is not possible with TESLA because the data packets can't be authenticated until a key is disclosed, so either the middlebox has to forward data packets without first authenticating them, so that the receiver has them prior to key disclosure, or the middlebox has to hold packets until the key is disclosed, at which point the receiver can no longer establish their authenticity.

The other new capability is that because AMBI provides authentication information out of band, authentication can be retrofitted into some pre-existing deployments without changing the protocol of the data packets, under some restrictions outlined in Section 7. By contrast, TESLA requires a MAC to be added to each authenticated message.

1.2. Threat Model

TBD: Summarize the applicable threat model this protects against. A diagram plus a cleaned-up version of the on-list explanation here is probably appropriate: https://mailarchive.ietf.org/arch/msg/mboned/CG9FLjPwuno3MtvYvgNcD5p69I4/

1.3. 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 [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Protocol Operation

2.1. Overview

In order to authenticate a data packet, AMBI receivers need to hold these three pieces of information at the same time:

- o the data packet; and
- o an authenticated manifest containing the packet digest for the data packet; and
- o a digest profile defining the transformation from the data packet to its packet digest.

The manifests are delivered as a stream of manifests over an authenticated data channel. Manifest contents MUST be authenticated before they can be used to authenticate data packets.

The manifest stream is composed of an ordered sequence of manifests that each contain an ordered sequence of packet digests, corresponding to the original packets as sent from their origin, in the same order.

Note that a manifest contains potentially many packet digests, and its size can be tuned to fit within a convenient PDU (Protocol Data Unit) of the manifest transport stream, so that usually, many packet digests for the multicast data stream can be delivered per packet of the manifest transport. The intent is that even with unicast-based manifest transport, multicast-style efficiencies of scale can still be realized, with only a relatively small unicast overhead, when manifests use a unicast transport.

2.2. Buffering and Validation Windows

Using different communication channels for the manifest stream and the data stream introduces a possibility of desynchronization in the timing of the received data between the different channels, so receivers hold data packets and packet digests from the manifest stream in buffers for some duration while awaiting the arrival of their counterparts.

While holding a data packet, if the corresponding packet digest for that packet arrives in the manifest stream and can be authenticated, the data packet is authenticated.

While holding an authenticated packet digest, if the corresponding data packet arrives with a matching packet digest, the data packet is authenticated.

Once a data packet is authenticated, the corresponding packet digest can be discarded and the data packet can be further processed by the receiving application or forwarded through the receiving network. Authenticating a data packet consumes one packet digest and prevents re-learning, with a hold-down time equal to the hold time for packet digests. A different manifest might provide the same packet digest with the same packet sequence number, but the digest remains consumed if it has been used to authenticate a data packet.

If the receiver's hold duration for a data packet expires without authenticating the packet, the packet SHOULD be dropped as unauthenticated. If the hold duration of a manifest expires, packet digests last received in that manifest SHOULD be discarded. (Note that in some cases, packet digests can be sent redundantly in more than one manifest. In such cases, the latest received time for an authenticated packet digest should be used for the expiration time.)

Since packet digests are usually smaller than the data packets, it's RECOMMENDED that senders generate and send manifests with timing such that the packet digests in a manifest will typically be received by subscribed receivers before the data packets corresponding to those digests are received.

This strategy reduces the buffering requirements at receivers at, the cost of introducing some buffering of data packets at the sender, since data packets are generated before their packet digests can be added to manifests.

The RECOMMENDED default hold times at receivers are:

o 2 seconds for data packets

o 10 seconds for packet digests

The sender MAY recommend different values for specific data streams, in order to tune different data streams for different performance goals. The YANG model in Section 5 provides a mechanism for senders to communicate the sender's recommendation for buffering durations, when using DORMS.

Receivers SHOULD follow the recommendations for hold times provided by the sender, subject to their capabilities and any administratively configured limits on buffer sizes at the receiver.

However receivers MAY deviate from the values recommended by the sender for a variety of reasons. Decreasing the buffering durations recommended by the server increases the risk of losing packets, but can be an appropriate tradeoff for specific network conditions and hardware constraints on some devices.

TBD: should there be any reordering restrictions above and beyond the timing constraints?

2.2.1. Inter-packet Gap

It's RECOMMENDED that middle boxes forwarding buffered data packets preserve the inter-packet gap between packets, and that receiving libraries provide mechanisms to expose the network arrival times of packets to applications.

The purpose for this recommendation is to preserve the capability of receivers to use techniques for available bandwidth detection or network congestion based on observation of packet times. Examples of such techniques include paced chirping and pathrate.

Note that this recommendation SHOULD NOT prevent the transmission of an authenticated packet because the prior packet is unauthenticated. This recommendation only asks implementations to delay the transmission of an authenticated packet to correspond to the interpacket gap if an authenticated packet was previously transmitted and the authentication of the subsequent packet would otherwise burst the packets more quickly.

This does not prevent the transmission of packets out of order according to their order of authentication, only the timing of packets that are transmitted, after authentication, in the same order they were received.

For receiver applications, the time that the original packet was received from the network SHOULD be made available to the receiving application.

2.3. Packet Digests

2.3.1. Digest Profile

A packet digest is a message digest for a data packet, built according to a digest profile defined by the sender.

The digest profile is defined by the sender, and specifies:

1. A cryptographically secure hash algorithm (REQUIRED)

2. A manifest stream identifier

3. Whether to hash the IP payload or the UDP payload. (see Section 2.3.1.1)

The hash algorithm is applied to a pseudoheader followed by the packet payload, as determined by the digest profile. The computed hash value is the packet digest.

TBD: there should also be a way to specify that only packets to a specific UDP port are applicable. I think this is not quite right today and probably should be done with a grouping in the yang model, so that the profile appears either inside a "protocol" container inside the (S,G) or inside the udp-stream inside the "protocol", but am not sure. Follow-up on this after the first reference implementation...

2.3.1.1. Payload Type

2.3.1.1.1. UDP vs. IP payload validation

When the digest profile indicates that UDP payloads are validated, the IP protocol for the packets MUST be UDP (0x11) and the payload used for calculating the packet digest includes only the UDP payload, with length as the number of UDP payload octets, as calculated by subtracting the size of the UDP header from the UDP payload length.

When the digest profile indicates that IP payloads are validated, the IP payload of the packet is used, using the outermost IP layer that contains the (S,G) corresponding to the (S,G) protected by the manifest. There is no restriction on the IP protocols that can be authenticated. The length field in the pseudoheader is calculated by

subtracting the IP Header Length from the IP length, and is equal to the number of octets in the payload for the digest calculation.

2.3.1.1.2. Motivation

Full IP payloads often aren't available to receivers without extra privileges on end user operating systems, so it's useful to provide a way to authenticate only the UDP payload, which is often the only portion of the packet available to many receiving applications.

However, for some use cases a full IP payload is appropriate. For example, when retrofitting some existing protocols, some packets may be predictable or frequently repeated. Use of an IPSec Authentication Header [RFC4302] is one way to disambiguate such packets. Even though the shared secret means the Authentication Header can't itself be used to authenticate the packet contents, the sequence number in the Authentication Header can ensure that specific packets are not repeated at the IP layer, and so it's useful for AMBI to have the capability to authenticate such packets.

Another example: some services might need to authenticate the UDP options [I-D.ietf-tsvwg-udp-options]. When using the UDP payload, the UDP options would not be part of the authenticated payload, but would be included when using the IP payload type.

Lastly, since (S,G) subscription operates at the IP layer, it's possible that some non-UDP protocols will need to be authenticated.

2.3.1.2. TBD: Packet contents?

TBD: Determine whether we need to support packet contents in the packet digest. If so, add to above list in Section 2.3.1:

o A set of bits from the packet contents (potentially empty)

The packet contents are a sequence of bits composed from a sequence of fixed bit (offset, length) pairs, as specified in xxxxxx. A useful choice for packet contents is to use sequence numbers in the application level protocol, such as with RTP [RFC3550], but any contents from the packet with a fixed bit offset and length can be used.

Providing variable packet contents in the packet digest increases the difficulty of attacking the hash by limiting the scope of legitimate data packets that can be matched when attempting to generate a hash collision.

The basic idea is to put an encoding here so that for example the RTP sequence number or the sequence number in an Authentication Header can be provided here in bulk (you give "value starts at bit 80 and is 16 bits long unsigned and increases by 1 per packet for the packets in the manifest with starting value 10", indicating that the 100 packets in the manifest have values 10-110 in their contents at the given location. Now those contents are prepended to the packet digest, and can be verified against the packets, as well as the hash of the contents).

For packet streams without a sequence number, we can instead incorporate a few high-entropy bits from the packet contents and NOT provide the value as a sequence number, but rather incorporate it in the digest values themselves. (Is this useful?)

Before defining this, I want to calculate how much overhead it buys us- how much can we truncate a good hash algorithm if we use this to add collision resistance? Might not be worthwhile, it's a significant increase in complexity. -jake 2019-08-31

If we need it, tentative addition to yang for the data profile looks like:

```
    list packet-contents {
      key offset;
      description "contents from the packet for the packet
          digest";
      leaf offset {
        type uint16;
        mandatory true;
        description "offset of the contents, in number of bits";
      }
      leaf length {
        type uint16;
        mandatory true;
        description "length of the contents, in number of bits";
      }
      leaf manifest-delivery {
        type enumeration {
          enum sequence;
          enum digest;
        }
        mandatory true;
        description "the way these content bits are delivered in
            the manifest";
      }
    }
```

The manifest-delivery would indicate whether the bits are a sequence number (in which case a section for a manifest with a start+step would be added ahead of the digests), or digest (indicating the bits appear inside each digest, ahead of the hash), and they would prepend in order to the packet digest, with sequence number bits inserted at the right bit location for the digest, based on earlier-appearing values, if any.

2.3.2. Pseudoheader

When calculating the hash for the packet digest, the hash algorithm is applied to a pseudoheader followed by the payload from the packet. The complete sequence of octets used to calculate the hash is structured as follows:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Source Address (32 bits IPv4/128 bits IPv6)           |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Destination Address (32 bits IPv4/128 bits IPv6)        |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Zeroes     |   Protocol     |             Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port           |         Destination Port     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Manifest Identifier                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Payload Data                          |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.3.2.1. Source Address

The IPv4 or IPv6 source address of the packet.

2.3.2.2. Destination Address

The IPv4 or IPv6 destination address of the packet.

2.3.2.3. Zeroes

All bits set to 0.

2.3.2.4. Protocol

The IP Protocol field from IPv4, or the Next Header field for IPv6. When UDP payload is indicated, this value MUST be UDP (0x11).

2.3.2.5. Length

The length in octets of the Payload Data field, expressed as an unsigned 16-bit integer.

2.3.2.6. Source Port

The source port of the packet. Zeroes if using a protocol that does not use source ports.

2.3.2.7. Destination Port

The destination port of the packet. Zeroes if using a protocol that does not use destination ports.

TBD: there's something I hate about the source and destination ports. Maybe it should only be active in UDP-payload mode, instead of zeroes when not UDP? But I suspect there's a better approach than UDP-or-not, so it's this way for now, with hopes of finding something better in the next version.

2.3.2.8. Manifest Identifier

The 32-bit identifier for the manifest stream.

2.3.2.9. Payload Data

The payload data includes either the IP payload or the UDP payload, as indicated by the digest profile.

The payload type is configurable because when sending UDP, some legacy networks may strip the UDP option space, and it's necessary to provide a manifest stream capable of authentication that can interoperate with these networks. However, for non-UDP traffic or in order to authenticate the UDP options, some use cases may require support for authenticating the full IP payload.

2.4. Manifests

2.4.1. Manifest Layout

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Manifest Stream Identifier                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Manifest sequence number                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  First packet sequence number                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       Refresh Deadline        |      Packet Digest Count      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  ... Packet Content Expansions ...            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       ... Packet Digests ...                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.4.1.1. Manifest Stream Identifier

A 32-bit unsigned integer chosen by the sender.

2.4.1.2. Manifest Sequence Number

A monotonically increasing 32-bit unsigned integer. Each manifest sent by the sender increases this value by 1. On overflow it wraps to 0.

It's RECOMMENDED to expire the manifest stream and start a new stream for the data packets before a sequence number wrap is necessary.

2.4.1.3. First Packet Sequence Number

A monotonically increasing 32-bit unsigned integer. Each packet in the data stream increases this value by 1.

It's RECOMMENDED to expire the manifest stream and start a new stream for the data packets before a sequence number wrap is necessary.

Note: for redundancy, especially if using a manifest stream with unreliable transport, successive manifests MAY provide duplicates of the same packet digest with the same packet sequence number, using overapping sets of packet sequence numbers. When received, these reset the hold timer for the listed packet digests.

2.4.1.4. Refresh Deadline

A 16-bit unsigned integer number of seconds.

A zero value means the current digest profile for the current manifest stream is stable.

A nonzero value means that the authentication is transitioning to a new manifest stream, and the set of digest profiles SHOULD be refreshed by receivers that might stay joined longer than this duration, and a different manifest stream SHOULD be selected, before this many seconds have elapsed, in order to avoid a disruption. See Section 2.5.

2.4.1.5. Packet Digest Count

The count of packet digests in the manifest.

2.4.1.6. Packet Digests

Packet digests appended one after the other, aligned to 8-bit boundaries with zero padding (if the bit length of the digests are not multiples of 8 bits).

2.5. Transitioning to Other Manifest Streams

It's possible for multiple manifest streams authenticating the same data stream to be active at the same time. The different manifest streams can have different hash algorithms, manifest ids, and current packet sequence numbers for the same data stream. These result in different sets of packet digests for the same data packets, one digest per packet per digest profile.

It's necessary sometimes to transition gracefully from one manifest stream to another. The Refresh Deadline field from the manifest is used to signal to receivers the need to transition.

When a receiver gets a nonzero refresh deadline in a manifest the sender SHOULD have an alternate manifest stream ready and available, and the receiver SHOULD learn the alternate manifest stream, join the new one, and leave the old one before the number of seconds given in the refresh deadline. After the refresh deadline has expired, a manifest stream MAY end.

The receivers SHOULD use a random value between now and one half the number of seconds in the deadline field, to spread the spike of load on the DORMS server during a large multicast event.

3. Transport Considerations

3.1. Overview

AMBI manifests MUST be authenticated, but any transport protocol providing authentication can be used. This section discusses several viable options for the use of an authenticating transport, and some associated design considerations.

TBD: extend the 'manifest-transport' in the YANG model to make an extensible mechanism to advertise different transport options for receiving manifest streams.

TBD: add ALTA to the list when and if it gets further along [I-D.draft-krose-mboned-alta-01]. Sending an authenticatable multicast stream (instead of the below unicast-based proposals) is a worthwhile goal, else a 1% unicast authentication overhead becomes a new unicast limit to the scalability.

3.2. HTTPS

This document defines a new media type 'application/ambi' for use with HTTPS.

An HTTPS stream carrying the 'application/ambi' media type is composed of a sequence of binary AMBI manifests. It is RECOMMENDED to use Chunked encoding.

Complete packet Digests from partially received manifests MAY be used by the receiver for authentication, even if the full manifest is not yet delivered.

3.3. DTLS

TBD: DTLS [RFC6347] can provide authentication for datagrams, so if manifests can be constructed to fit within datagrams, it is an appropriate choice. (IPSec is similar-worth adding as an option?).

This option provides no native redundancy or retransmission, but packet digests can be repeated in different manifests to provide some resilience to loss. Lost manifests that result in the loss of blocks of packet digests can be expensive, since they would make received data packets unauthenticatable. TBD: should we therefore not support this case?

3.4. DTLS + FECFRAME

DTLS for manifests that do not fit into single-packet payloads can still be delivered by using FECFRAME [RFC6363], particularly Reed-Solomon [RFC6865] or possibly Raptor [RFC6681]. This has some advantages compared to HTTPS because of the absence of HOL-blocking, while providing for tunable redundancy. This has some advantages relative to DTLS because of overhead reduction and non-integer redundancy tunability (e.g. 1.5 becomes a viable redundancy factor).

TBD: define this method, possibly in another RFC.

4. Examples

TBD: walk through some examples as soon as we have a build running. Likely to need some touching up.

5. YANG Module

5.1. Tree Diagram

```
module: ietf-ambi
  augment /dorms:metadata/dorms:sender/dorms:group/dorms:udp-stream:
    +--rw manifest-stream* [id]
       +--rw id                      uint32
       +--rw manifest-transport*     inet:uri
       +--rw hash-algorithm          ct:asymmetric-key-algorithm-t
       +--rw payload-type            enumeration
       +--rw data-hold-time-ms?      uint32
       +--rw digest-hold-time-ms?    uint32
```

5.2. Module

```
<CODE BEGINS> file ietf-ambi@2020-03-09.yang
module ietf-ambi {
    yang-version 1.1;

    namespace "urn:ietf:params:xml:ns:yang:ietf-ambi";
    prefix "ambi";

    import ietf-dorms {
        prefix "dorms";
        reference "I-D.jholland-mboned-dorms";
    }

    import ietf-inet-types {
        prefix "inet";
```

```
        reference "RFC6991 Section 4";
    }

    import ietf-crypto-types {
        prefix "ct";
        reference "draft-ietf-netconf-crypto-types";
    }

    organization "IETF";

    contact
        "Author:   Jake Holland
                   <mailto:jholland@akamai.com>
        ";

    description
    "Copyright (c) 2019 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX
     (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
     for full legal notices.

     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 (RFC 2119) (RFC 8174) when, and only when,
     they appear in all capitals, as shown here.

     This module contains the definition for the AMBI data types.
     It provides metadata for authenticating SSM channels as an
     augmentation to DORMS.";

    revision 2019-08-25 {
        description "Initial revision as an extension.";
        reference
          "";
    }

    augment
      "/dorms:metadata/dorms:sender/dorms:group/dorms:udp-stream" {
```

```
        description "Definition of the manifest stream providing
            integrity info for the data stream";

        list manifest-stream {
            key id;
            description "Definition of a manifest stream.";
            leaf id {
                type uint32;
                mandatory true;
                description "The Manifest ID referenced in a manifest.";
            }
            leaf-list manifest-transport {
                type inet:uri;
                description "A URI that provides a location for the
                    manifest stream";
            }
            leaf hash-algorithm {
                type ct:asymmetric-key-algorithm-t;
                mandatory true;
                description
                    "The hash algorithm for the packet hashes within
                     manifests in this stream.";
            }
            leaf payload-type {
                type enumeration {
                    enum udp {
                      description "The hash includes only the UDP
                          payload.";
                    }
                    enum ip {
                      description "The hash includes the full IP
                          payload.";
                    }
                }
                mandatory true;
                description "The contents of the payload for the
                    digest profile";
            }
            leaf data-hold-time-ms {
                type uint32;
                default 2000;
                description "The number of milliseconds to hold data
                    packets waiting for a corresponding digest before
                    discarding";
            }
            leaf digest-hold-time-ms {
                type uint32;
                default 10000;
```

```
                description "The number of milliseconds to hold packet
                    digests waiting for a corresponding data packet
                    before discarding";
            }
        }
    }
}

<CODE ENDS>
```

6. IANA Considerations

6.1. The YANG Module Names Registry

This document adds one YANG module to the "YANG Module Names" registry maintained at <https://www.iana.org/assignments/yang-parameters>. The following registrations are made, per the format in Section 14 of [RFC6020]:

```
      name:      ietf-ambi
      namespace: urn:ietf:params:xml:ns:yang:ietf-ambi
      prefix:    ambi
      reference: I-D.draft-jholland-mboned-ambi
```

6.2. Media Type

TBD: Register 'application/ambi' according to advice from: https://www.iana.org/form/media-types

TBD: check guidelines in https://tools.ietf.org/html/rfc5226

7. Security Considerations

7.1. Predictable Packets

Protocols that have predictable packets run the risk of offline attacks for hash collisions against those packets. When authenticating a protocol that might have predictable packets, it's RECOMMENDED to use a hash function secure against such attacks or to add content to the packets to make them unpredictable, such as an Authentication Header ([RFC4302]), or the addition of an ignored field with random content to the packet payload.

TBD: explain attack from generating malicious packets and then looking for collisions, as opposed to having to generate a collision on packet contents that include a sequence number and then hitting a match (especially expand on this if we do add Section 2.3.1.2).

TBD: follow the rest of the guidelines: https://tools.ietf.org/html/rfc3552

8. Acknowledgements

Many thanks to Daniel Franke, Eric Rescorla, Christian Worm Mortensen, Max Franke, and Albert Manfredi for their very helpful comments and suggestions.

9. References

9.1. Normative References

[I-D.draft-jholland-mboned-dorms-01]
Holland, J., "Discovery Of Restconf Metadata for Source-specific multicast", draft-jholland-mboned-dorms-01 (work in progress), September 2019.

[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>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, DOI 10.17487/RFC6363, October 2011, <https://www.rfc-editor.org/info/rfc6363>.

[RFC6681] Watson, M., Stockhammer, T., and M. Luby, "Raptor Forward Error Correction (FEC) Schemes for FECFRAME", RFC 6681, DOI 10.17487/RFC6681, August 2012, <https://www.rfc-editor.org/info/rfc6681>.

[RFC6865] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono, "Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6865, DOI 10.17487/RFC6865, February 2013, <https://www.rfc-editor.org/info/rfc6865>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[I-D.draft-krose-mboned-alta-01] Rose, K. and J. Holland, "Asymmetric Loss-Tolerant Authentication", draft-krose-mboned-alta-01 (work in progress), July 2019.

[I-D.ietf-tsvwg-udp-options] Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-08 (work in progress), September 2019.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/info/rfc3550>.

[RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, DOI 10.17487/RFC4082, June 2005, <https://www.rfc-editor.org/info/rfc4082>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, DOI 10.17487/RFC4383, February 2006, <https://www.rfc-editor.org/info/rfc4383>.

[RFC5776] Roca, V., Francillon, A., and S. Faurite, "Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 5776, DOI 10.17487/RFC5776, April 2010, <https://www.rfc-editor.org/info/rfc5776>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

Authors' Addresses

Jake Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: jakeholland.net@gmail.com

Kyle Rose
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: krose@krose.org