Mboned
Internet-Draft
Intended status: Experimental
Expires: December 31, 2018

J. Holland
K. Rose
Akamai Technologies, Inc.

June 29, 2018

Asymmetric Manifest Based Integrity
draft-jholland-mboned-ambi-00

Abstract

This document introduces Asymmetric Manifest-Based Integrity (AMBI). AMBI allows each receiver 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 manifests for the data packets, over out-of-band communication channels.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 31, 2018.

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.

- 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) specifies 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 2.3) via an out-of-band communication channel that provides authentication and verifiable integrity.

Each manifest contains cryptographic hashes of packet payloads corresponding to specific packets in the authenticated data stream.

Three ways to authenticate a manifest are defined:

- Asymmetric signature of a message containing the manifest.
- Authenticated unicast stream providing a sequence of manifests.
- Using one of the prior two constructions to bootstrap a root manifest containing authentication information for further manifests. This we term "recursive authentication".

When using asymmetric signatures, recursive authentication allows the sender to amortize the computational overhead for a single asymmetric signature across enough data packets to sustain high data rates. When using a secure unicast stream, the recursive verification allows for scaling the authenticated data stream to more receivers than can otherwise be sustained with a limited set of trusted servers.

Upon successful verification of the contents 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 listed in the manifest.

An "anchor message", described in Section 2.2, provides the link between an authenticated data stream and the out-of-band channel of manifests that authenticates it.

Authenticating the integrity of the data packets depends on:

- authentication of the anchor message that provides the linkage between the manifest channel and the data stream; and
- the secrecy and cryptographic strength of private keys used for signing manifests, or the authentication of the secure unicast streams used for transmitting manifests; and

- o the difficulty of generating a collision for the packet hashes in the manifest.

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. (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 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.)

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Protocol Specification

2.1. Packet Identifiers

2.1.1. Overview

Packet identifiers are a sequence number contained within the authenticated payload of the packet. This sequence number is used inside a manifest to associate each packet hash with a specific packet. Each authenticated packet MUST contain a packet identifier. See Section 4.1 for a discussion of the security implications.

This document defines a new UDP option in Section 2.1.4 for use as a packet identifier.

Some multicast-capable transport protocols have a sequence number embedded in data packets in the protocol. The sequence numbers in

these protocols MAY be used instead of the new UDP option, to avoid introducing extra overhead in the authenticated data packets.

In Section 2.1.2, Section 2.1.3, and Section 2.1.4, this document defines some sample ways to specify packet identifiers based on such sequence numbers embedded in existing protocols.

Other appropriate sequence number systems may exist, such as the anti-replay Sequence Number field in Section 3.1 of [RFC6584], when NORM or FLUTE operates with an authentication profile that uses it (however, since that example already provides authentication, it is not added as an option in this document). The AMBI anchor message format can be extended in future documents to support those or other suitable schemes by adding values to the registry defined in Section 3.

In some deployments, in contrast to using the new UDP option, the approach of using an existing sequence number may carry a benefit because it requires no change to the stream of packets being authenticated, possibly enabling interoperability with legacy receivers.

2.1.2. RTP Sequence Number

Sequence number from Section 5.1 of [RFC3550].

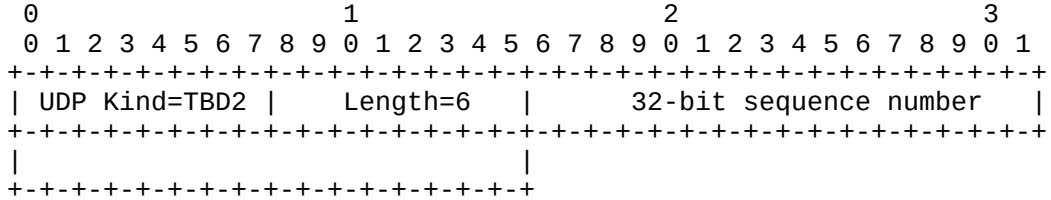
TBD: discussion of security consequences of using 16 bits- recommend a bigger hash in manifests for this case?

2.1.3. SRTP Sequence Number

Packet Index from Section 3.3.1 of [RFC3711].

2.1.4. UDP Option

Define a new UDP option [I-D.ietf-tsvwg-udp-options] (TBD2).



2.2. Anchor Message

2.2.1. Overview

An anchor message provides the information that makes it possible to associate the manifests with the data packets they authenticate. ID values that appear as text integers in the anchor message also appear in the manifest binary data, with the anchor message providing context on how to interpret the values.

An anchor message MAY be discovered and transmitted by any means which provides adequate source authentication and data integrity to meet the security needs of the receiver.

In order to support middle-box authentication, it is RECOMMENDED that senders arrange to distribute anchor messages according to the method outlined in Section 2.2.2.

2.2.2. DNS-based Anchor URI Bootstrap

When a middle box tries to process a join for a specific source, if it is configured to perform authentication on SSM multicast channels it can forward, it SHOULD make a DNS request for ambi.(reverse-source-ip).ip6.arpa or ambi.(reverse-source-ip).in-addr.arpa, for IPv6 or IPv4 source addresses.

When AMBI is provided to authenticate traffic from this source IP, this domain name SHOULD be configured with a TXT field which contains a URI that can be used to securely fetch an anchor message that describes all the AMBI- authenticatable traffic from this source IP.

Other methods MAY be used to discover and transfer an anchor message. The description of alternate methods is out of scope for this document.

TBD: consider breaking up anchor message to avoid large, frequently changing anchors for sources with many groups.

TBD: consider graceful rollover for anchors, instead of synchronized update of anchor hash.

2.2.3. Anchor Message YANG model

```
<CODE BEGINS> file "ietf-ambi-anchor.yang"
module ietf-ambi-anchor {
    yang-version 1.1;

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

```
    prefix "ambi";

    import ietf-yang-types {
        prefix "yang";
        reference "RFC6991 Section 3";
    }

    import ietf-inet-types {
        prefix "inet";
        reference "RFC6991 Section 4";
    }

    import ietf-routing-types {
        prefix "rt-types";
        reference "RFC8294";
    }

    organization "IETF";

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

    description
        "This module contains the definition for the AMBI anchor
         message data type.

         Copyright (c) 2018 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
         (http://trustee.ietf.org/license-info).

         This version of this YANG module is part of
         draft-jholland-mboned-ambi,
         see the internet draft itself for full legal notices.";

    revision 2018-06-27 {
        description "Initial revision.";
        reference
          "";
```

```
    }

    /* TBD: copied some from https://tools.ietf.org/html/rfc8177,
       but the model doesn't seem to match what I want. is there another
       I can import instead of making these here? Or a registry
       to reference? */
    identity crypto-hash {
        description
            "Base identity of cryptographic hash options. ";
    }

    identity sha-256 {
        base crypto-hash;
        description
            "The SHA-256 algorithm.";
    }

    identity blake2b {
        base crypto-hash;
        description
            "The BLAKE2b algorithm.";
    }

    identity crypto-signature {
        description
            "Base identity of cryptographic asymmetric signature
             options.";
    }

    identity ed25519 {
        base crypto-signature;
        description
            "The Ed25519 algorithm.";
    }

    identity rsa {
        base crypto-signature;
        description
            "The RSA algorithm.";
    }

    identity sequence-type {
        description
            "Base identity for sequence number type options.";
    }

    identity rtp {
        base sequence-type;
```

```
        description
            "The sequence number from RTP.";
    }

    identity srtp {
        base sequence-type;
        description
            "The sequence number from SRTP.";
    }

    identity udp {
        base sequence-type;
        description
            "The sequence number from UDP.";
    }
    typedef key-identifier {
        type uint16 {
            range 1..65535;
        }
        description "Key identifier within a manifest";
    }

    typedef bitrate {
        type string {
          pattern '[1-9][0-9]*[GMK]?bps';
        }
        description "Bit-rate of a data stream";
    }

    typedef packetrate {
        type string {
          pattern '[1-9][0-9]*[GMK]?pps';
        }
        description "Packet rate of a data stream";
    }

    typedef manifest-transport {
        type union {
            type leafref {
                path "/anchor/data_stream/id";
            }
            type inet:uri;
        }
        description "Transport method for a manifest stream";
    }

    container anchor {
        container self {
```

```
            presence "An anchor message exists";
            description
                "Self-referential properties about the anchor message";
            leaf uri {
                type inet:uri;
                mandatory true;
                description
                    "The canonical URI for this anchor message.";
            }
            leaf version {
                type uint16;
                mandatory true;
                description
                    "The version number for this anchor message.";
            }
            leaf hash_algorithm {
                type identityref {
                    base crypto-hash;
                }
                mandatory true;
                description
                    "The algorithm for the anchor message hash provided
                     in a manifest.";
            }
            leaf hash_bits {
                type uint16;
                mandatory true;
                description
                    "The number of bits for the anchor's hash provided
                     in a manifest.";
            }
            leaf expires {
                type yang:date-and-time;
                mandatory true;
                description
                    "The expiration time for this anchor message.";
            }
        }
        description "Anchor message for AMBI";

        list public_key {
            key id;
            description "Public key for non-recursive manifest";
            leaf id {
                type key-identifier;
                mandatory true;
                description
                    "The key identifier referenced in a manifest.";
```

```
            }
            leaf algorithm {
                type identityref {
                    base crypto-signature;
                }
                mandatory true;
                description
                    "The signature algorithm for use with this key.";
            }
            leaf signature_bits {
                type uint16;
                mandatory true;
                description
                    "The length of the signature provided in manifests
                     signed with this key.";
            }
            leaf value {
                type string;
                mandatory true;
                description
                    "The base64-encoded value of the public key.";
            }
        }

        list data_stream {
            key id;
            unique "source destination port";
            description "Stream of data packets to be authenticated";
            leaf id {
                type uint16;
                mandatory true;
                description
                    "The datastream_id referenced by a
                     manifest_stream.";
            }
            leaf source {
                type inet:ip-address;
                mandatory true;
                description
                    "The source IP address of the authenticated data
                     stream.";
            }
            leaf destination {
                type rt-types:ip-multicast-group-address;
                mandatory true;
                description
                    "The destination group IP address of the
                     authenticated data stream.";
```

```
            }
            leaf port {
                type uint16;
                mandatory true;
                description
                    "The destination UDP port of the authenticated data
                     stream.";
            }
            leaf max_bitrate {
                type bitrate;
                mandatory true;
                description
                    "The maximum bitrate expected for this data
                     stream.";
            }
            leaf max_packetrate {
                type packetrate;
                mandatory true;
                description
                    "The maximum packetrate expected for this data
                     stream.";
            }
            list authenticator {
                key manifest_id;
                description
                    "A manifest stream that authenticates this data";
                leaf manifest_id {
                    type leafref {
                        path "/anchor/manifest_stream/id";
                    }
                    mandatory true;
                    description
                        "The ID of a manifest stream that provides
                         authentication for this data stream.";
                }
            }
        }

        list manifest_stream {
            key id;
            description "Stream of manifests";
            leaf id {
                type uint16;
                mandatory true;
                description "The Manifest ID referenced in a manifest.";
            }
            leaf transport {
                type manifest-transport;
```

```
                mandatory true;
                description
                    "The ID of the data stream that carries this
                     manifest stream or a uri that provides a websocket
                     with the stream of manifests.";
            }
            leaf hash_algorithm {
                type identityref {
                    base crypto-hash;
                }
                mandatory true;
                description
                    "The hash algorithm for the packet hashes within
                     manifests in this stream.";
            }
            leaf hash_bits {
                type uint16;
                mandatory true;
                description
                    "The number of bits of hash provided for packet
                     hashes.";
            }
            leaf sequence_type {
                type identityref {
                    base sequence-type;
                }
                mandatory true;
                description
                    "The linkage to the data packet sequence numbers in
                     the manifest.";
            }
        }
    }
}
<CODE ENDS>
```

Figure 1: Anchor Message YANG model

2.2.4. Example Anchor Message

```
   {
     "ietf-ambi-anchor:anchor": {
       "self": {
         "uri": "https://example.com/ambi/anchor/example_1.json",
         "version": 1,
         "hash_algorithm": "blake2b",
         "hash_bits": 256,
         "expires": "2018-03-05T23:59:59Z"
```

```
      },
      "public_key": [
        {
          "id": 1,
          "algorithm": "ed25519",
          "signature_bits": 256,
          "value": "VGhpcyBpcyBub3QgYSBnb29kIGtleSB0byB1c2UuLi4NCg=="
        }
      ],
      "data_stream": [
        {
          "id": 10,
          "source": "192.0.2.10",
          "destination": "232.10.10.1",
          "port": 18001,
          "max_bitrate": "10Mbps",
          "max_packetrate": "1Kpps",
          "authenticator": [
            {
              "manifest_id": 1
            }
          ]
        },
        {
          "id": 20,
          "source": "192.0.2.10",
          "destination": "232.10.10.1",
          "port": 18002,
          "max_bitrate": "400Kbps",
          "max_packetrate": "40pps",
          "authenticator": [
            {
              "manifest_id": 2
            }
          ]
        }
      ],
      "manifest_stream": [
        {
          "id": 1,
          "transport": 20,
          "hash_algorithm": "blake2b",
          "hash_bits": 80,
          "sequence_type": "udp"
        },
        {
          "id": 2,
          "transport": "https://example.com/ambi/manifest_stream/3",
```

```
          "hash_algorithm": "blake2b",
          "hash_bits": 80,
          "sequence_type": "udp"
        }
      ]
    }
  }
```

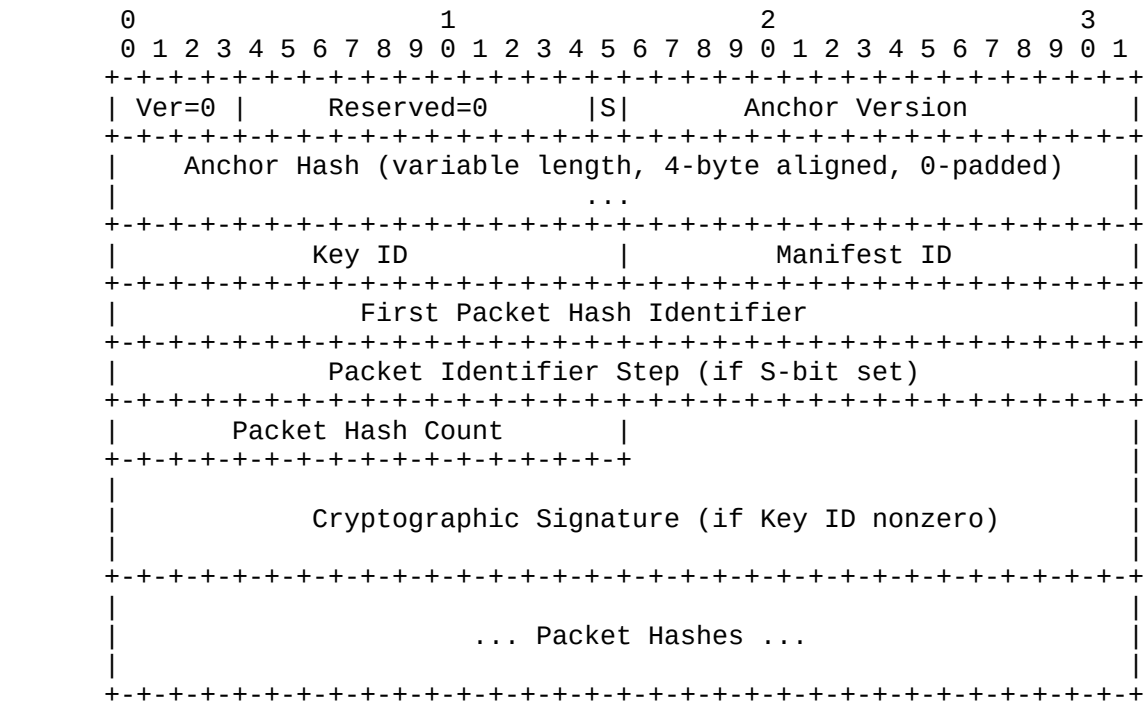
Figure 2: Example Anchor Message

2.3. Manifests

2.3.1. Overview

A manifest cannot be interpreted except in context of a known anchor message. In order for a manifest to be considered as potentially authenticating a set of packets, the anchor version MUST match the value in a known unexpired anchor message, and the hash MUST match the hash of the contents of that anchor message, according to the /anchor/self/hash_algorithm and /anchor/self/hash_bits fields, in order for a manifest to be accepted for use as evidence of authenticity and integrity.

2.3.2. Manifest Layout



3. IANA Considerations

TBD1: Request a "Specification Required" registry for Packet identifier methods, from https://tools.ietf.org/html/rfc5226#section-4.1 . Reserve anything beginning with "experimental:"?

TBD2: Add a new entry to the "UDP Option Kind" numbers registry: https://tools.ietf.org/html/draft-ietf-tsvwg-udp-options-02#section-14

TBD: check guidelines in https://tools.ietf.org/html/rfc5226 and remove this paragraph

Example from: https://tools.ietf.org/html/rfc5226#section-5.1

[TO BE REMOVED: Please add the yang model in Section 2.2.3 to: https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml

Name:ietf-ambi-anchor Maintained by IANA: N Namespace: urn:ietf:params:xml:ns:yang:ietf-ambi-anchor Prefix: anchor Reference: I-D.draft-jholland-mboned-ambi Notes:]

4. Security Considerations

4.1. Packet Identifiers

TBD: explain attack from generating malicious packets and then looking for collisions, as opposed to having to generate a collision including a sequence number and then hitting a match

TBD: DNSSEC vis-a-vis anchor url discovery. (we need a diagram about for middle-box handling of a revers-path propagated join?) Explain why malicious DNS could deny service, but cannot cause accepting attack packets.

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

5. References

5.1. Normative References

[I-D.ietf-tsvwg-udp-options]
Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-02 (work in progress), January 2018.

[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>.

[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>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

5.2. Informative References

[DIGSIGN] Rohatgi, P., "How to Sign Digital Streams", 1997, <https://link.springer.com/content/pdf/10.1007/BFb0052235.pdf>.

Published in CRYPTO ’97 "Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology", Pages 180-197

[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>.

[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>.

[RFC6584] Roca, V., "Simple Authentication Schemes for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 6584, DOI 10.17487/RFC6584, April 2012, <https://www.rfc-editor.org/info/rfc6584>.

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