Network Working Group
Request for Comments: 4302
Obsoletes: 2402
Category: Standards Track

S. Kent
BBN Technologies

December 2005

IP Authentication Header

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes an updated version of the IP Authentication Header (AH), which is designed to provide authentication services in IPv4 and IPv6. This document obsoletes RFC 2402 (November 1998).

Table of Contents

1. Introduction

This document assumes that the reader is familiar with the terms and concepts described in the "Security Architecture for the Internet Protocol" [Ken-Arch], hereafter referred to as the Security Architecture document. In particular, the reader should be familiar with the definitions of security services offered by the Encapsulating Security Payload (ESP) [Ken-ESP] and the IP Authentication Header (AH), the concept of Security Associations, the ways in which ESP can be used in conjunction with the Authentication Header (AH), and the different key management options available for ESP and AH.

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in RFC 2119 [Bra97].

The IP Authentication Header (AH) is used to provide connectionless integrity and data origin authentication for IP datagrams (hereafter referred to as just "integrity") and to provide protection against replays. This latter, optional service may be selected, by the receiver, when a Security Association (SA) is established. (The protocol default requires the sender to increment the sequence number used for anti-replay, but the service is effective only if the receiver checks the sequence number.) However, to make use of the Extended Sequence Number feature in an interoperable fashion, AH does impose a requirement on SA management protocols to be able to negotiate this new feature (see Section 2.5.1 below).

AH provides authentication for as much of the IP header as possible, as well as for next level protocol data. However, some IP header fields may change in transit and the value of these fields, when the packet arrives at the receiver, may not be predictable by the sender. The values of such fields cannot be protected by AH. Thus, the protection provided to the IP header by AH is piecemeal. (See Appendix A.)

AH may be applied alone, in combination with the IP Encapsulating Security Payload (ESP) [Ken-ESP], or in a nested fashion (see Security Architecture document [Ken-Arch]). Security services can be provided between a pair of communicating hosts, between a pair of communicating security gateways, or between a security gateway and a host. ESP may be used to provide the same anti-replay and similar integrity services, and it also provides a confidentiality (encryption) service. The primary difference between the integrity provided by ESP and AH is the extent of the coverage. Specifically, ESP does not protect any IP header fields unless those fields are encapsulated by ESP (e.g., via use of tunnel mode). For more details on how to use AH and ESP in various network environments, see the Security Architecture document [Ken-Arch].

Section 7 provides a brief review of the differences between this document and RFC 2402 [RFC2402].

2. Authentication Header Format

The protocol header (IPv4, IPv6, or IPv6 Extension) immediately preceding the AH header SHALL contain the value 51 in its Protocol (IPv4) or Next Header (IPv6, Extension) fields [DH98]. Figure 1 illustrates the format for AH.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Payload Len  |          RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Security Parameters Index (SPI)               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Sequence Number Field                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                Integrity Check Value-ICV (variable)           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1. AH Format

The following table refers to the fields that comprise AH, (illustrated in Figure 1), plus other fields included in the integrity computation, and illustrates which fields are covered by the ICV and what is transmitted.

	# of bytes	Requ’d [1]	What Integ Covers	What is Xmtd
IP Header	variable	M	[2]	plain
Next Header	1	M	Y	plain
Payload Len	1	M	Y	plain
RESERVED	2	M	Y	plain
SPI	4	M	Y	plain
Seq# (low-order 32 bits)	4	M	Y	plain
ICV	variable	M	Y[3]	plain
IP datagram [4]	variable	M	Y	plain
Seq# (high-order 32 bits)	4	if ESN	Y	not xmtd
ICV Padding	variable	if need	Y	not xmtd

[1] - M = mandatory
[2] - See Section 3.3.3, "Integrity Check Value Calculation", for details of which IP header fields are covered.
[3] - Zeroed before ICV calculation (resulting ICV placed here after calculation)
[4] - If tunnel mode -> IP datagram
If transport mode -> next header and data

The following subsections define the fields that comprise the AH format. All the fields described here are mandatory; i.e., they are always present in the AH format and are included in the Integrity Check Value (ICV) computation (see Sections 2.6 and 3.3.3).

Note: All of the cryptographic algorithms used in IPsec expect their input in canonical network byte order (see Appendix of RFC 791 [RFC791]) and generate their output in canonical network byte order. IP packets are also transmitted in network byte order.

AH does not contain a version number, therefore if there are concerns about backward compatibility, they MUST be addressed by using a signaling mechanism between the two IPsec peers to ensure compatible versions of AH, e.g., IKE [IKEv2] or an out-of-band configuration mechanism.

2.1. Next Header

The Next Header is an 8-bit field that identifies the type of the next payload after the Authentication Header. The value of this field is chosen from the set of IP Protocol Numbers defined on the web page of Internet Assigned Numbers Authority (IANA). For example, a value of 4 indicates IPv4, a value of 41 indicates IPv6, and a value of 6 indicates TCP.

2.2. Payload Length

This 8-bit field specifies the length of AH in 32-bit words (4-byte units), minus "2". Thus, for example, if an integrity algorithm yields a 96-bit authentication value, this length field will be "4" (3 32-bit word fixed fields plus 3 32-bit words for the ICV, minus 2). For IPv6, the total length of the header must be a multiple of 8-octet units. (Note that although IPv6 [DH98] characterizes AH as an extension header, its length is measured in 32-bit words, not the 64-bit words used by other IPv6 extension headers.) See Section 2.6, "Integrity Check Value (ICV)", for comments on padding of this field, and Section 3.3.3.2.1, "ICV Padding".

2.3. Reserved

This 16-bit field is reserved for future use. It MUST be set to "zero" by the sender, and it SHOULD be ignored by the recipient. (Note that the value is included in the ICV calculation, but is otherwise ignored by the recipient.)

2.4. Security Parameters Index (SPI)

The SPI is an arbitrary 32-bit value that is used by a receiver to identify the SA to which an incoming packet is bound. For a unicast SA, the SPI can be used by itself to specify an SA, or it may be used in conjunction with the IPsec protocol type (in this case AH). Because for unicast SAs the SPI value is generated by the receiver, whether the value is sufficient to identify an SA by itself or whether it must be used in conjunction with the IPsec protocol value is a local matter. The SPI field is mandatory, and this mechanism for mapping inbound traffic to unicast SAs described above MUST be supported by all AH implementations.

If an IPsec implementation supports multicast, then it MUST support multicast SAs using the algorithm below for mapping inbound IPsec datagrams to SAs. Implementations that support only unicast traffic need not implement this de-multiplexing algorithm.

In many secure multicast architectures, e.g., [RFC3740], a central Group Controller/Key Server unilaterally assigns the group security association's SPI. This SPI assignment is not negotiated or coordinated with the key management (e.g., IKE) subsystems that reside in the individual end systems that comprise the group. Consequently, it is possible that a group security association and a unicast security association can simultaneously use the same SPI. A multicast-capable IPsec implementation MUST correctly de-multiplex inbound traffic even in the context of SPI collisions.

Each entry in the Security Association Database (SAD) [Ken-Arch] must indicate whether the SA lookup makes use of the destination, or destination and source, IP addresses, in addition to the SPI. For multicast SAs, the protocol field is not employed for SA lookups. For each inbound, IPsec-protected packet, an implementation must conduct its search of the SAD such that it finds the entry that matches the "longest" SA identifier. In this context, if two or more SAD entries match based on the SPI value, then the entry that also matches based on destination, or destination and source, address comparison (as indicated in the SAD entry) is the "longest" match. This implies a logical ordering of the SAD search as follows:

1. Search the SAD for a match on {SPI, destination address, source address}. If an SAD entry matches, then process the inbound AH packet with that matching SAD entry. Otherwise, proceed to step 2.

2. Search the SAD for a match on {SPI, destination address}. If an SAD entry matches, then process the inbound AH packet with that matching SAD entry. Otherwise, proceed to step 3.

3. Search the SAD for a match on only {SPI} if the receiver has chosen to maintain a single SPI space for AH and ESP, or on {SPI, protocol} otherwise. If an SAD entry matches, then process the inbound AH packet with that matching SAD entry. Otherwise, discard the packet and log an auditable event.

In practice, an implementation MAY choose any method to accelerate this search, although its externally visible behavior MUST be functionally equivalent to having searched the SAD in the above order. For example, a software-based implementation could index into a hash table by the SPI. The SAD entries in each hash table bucket's linked list are kept sorted to have those SAD entries with the longest SA identifiers first in that linked list. Those SAD entries having the shortest SA identifiers are sorted so that they are the last entries in the linked list. A hardware-based implementation may be able to effect the longest match search intrinsically, using commonly available Ternary Content-Addressable Memory (TCAM) features.

The indication of whether source and destination address matching is required to map inbound IPsec traffic to SAs MUST be set either as a side effect of manual SA configuration or via negotiation using an SA management protocol, e.g., IKE or Group Domain of Interpretation (GDOI) [RFC3547]. Typically, Source-Specific Multicast (SSM) [HC03] groups use a 3-tuple SA identifier composed of an SPI, a destination multicast address, and source address. An Any-Source Multicast group SA requires only an SPI and a destination multicast address as an identifier.

The set of SPI values in the range 1 through 255 is reserved by the Internet Assigned Numbers Authority (IANA) for future use; a reserved SPI value will not normally be assigned by IANA unless the use of the assigned SPI value is specified in an RFC. The SPI value of zero (0) is reserved for local, implementation-specific use and MUST NOT be sent on the wire. (For example, a key management implementation might use the zero SPI value to mean "No Security Association Exists" during the period when the IPsec implementation has requested that its key management entity establish a new SA, but the SA has not yet been established.)

2.5. Sequence Number

This unsigned 32-bit field contains a counter value that increases by one for each packet sent, i.e., a per-SA packet sequence number. For a unicast SA or a single-sender multicast SA, the sender MUST increment this field for every transmitted packet. Sharing an SA among multiple senders is permitted, though generally not recommended. AH provides no means of synchronizing packet counters among multiple senders or meaningfully managing a receiver packet counter and window in the context of multiple senders. Thus, for a multi-sender SA, the anti-reply features of AH are not available (see Sections 3.3.2 and 3.4.3).

The field is mandatory and MUST always be present even if the receiver does not elect to enable the anti-replay service for a specific SA. Processing of the Sequence Number field is at the discretion of the receiver, but all AH implementations MUST be capable of performing the processing described in Section 3.3.2, "Sequence Number Generation", and Section 3.4.3, "Sequence Number Verification". Thus, the sender MUST always transmit this field, but the receiver need not act upon it.

The sender’s counter and the receiver’s counter are initialized to 0 when an SA is established. (The first packet sent using a given SA will have a sequence number of 1; see Section 3.3.2 for more details on how the sequence number is generated.) If anti-replay is enabled (the default), the transmitted sequence number must never be allowed to cycle. Thus, the sender’s counter and the receiver’s counter MUST be reset (by establishing a new SA and thus a new key) prior to the transmission of the 2^32nd packet on an SA.

2.5.1. Extended (64-bit) Sequence Number

To support high-speed IPsec implementations, a new option for sequence numbers SHOULD be offered, as an extension to the current, 32-bit sequence number field. Use of an Extended Sequence Number (ESN) MUST be negotiated by an SA management protocol. Note that in IKEv2, this negotiation is implicit; the default is ESN unless 32-bit sequence numbers are explicitly negotiated. (The ESN feature is applicable to multicast as well as unicast SAs.)

The ESN facility allows use of a 64-bit sequence number for an SA. (See Appendix B, "Extended (64-bit) Sequence Numbers", for details.) Only the low-order 32 bits of the sequence number are transmitted in the AH header of each packet, thus minimizing packet overhead. The high-order 32 bits are maintained as part of the sequence number counter by both transmitter and receiver and are included in the computation of the ICV, but are not transmitted.

2.6. Integrity Check Value (ICV)

This is a variable-length field that contains the Integrity Check Value (ICV) for this packet. The field must be an integral multiple of 32 bits (IPv4 or IPv6) in length. The details of ICV processing are described in Section 3.3.3, "Integrity Check Value Calculation", and Section 3.4.4, "Integrity Check Value Verification". This field may include explicit padding, if required to ensure that the length of the AH header is an integral multiple of 32 bits (IPv4) or 64 bits (IPv6). All implementations MUST support such padding and MUST insert only enough padding to satisfy the IPv4/IPv6 alignment requirements. Details of how to compute the required padding length are provided below in Section 3.3.3.2, "Padding". The integrity algorithm specification MUST specify the length of the ICV and the comparison rules and processing steps for validation.

3. Authentication Header Processing

3.1. Authentication Header Location

AH may be employed in two ways: transport mode or tunnel mode. (See the Security Architecture document for a description of when each should be used.)

3.1.1. Transport Mode

In transport mode, AH is inserted after the IP header and before a next layer protocol (e.g., TCP, UDP, ICMP, etc.) or before any other IPsec headers that have already been inserted. In the context of IPv4, this calls for placing AH after the IP header (and any options that it contains), but before the next layer protocol. (Note that the term "transport" mode should not be misconstrued as restricting its use to TCP and UDP.) The following diagram illustrates AH transport mode positioning for a typical IPv4 packet, on a "before and after" basis.

```
                  BEFORE APPLYING AH
            ----------------------------
      IPv4  |orig IP hdr  |     |      |
            |(any options)| TCP | Data |
            ----------------------------

                  AFTER APPLYING AH
            -------------------------------------------------------
      IPv4  |original IP hdr (any options) | AH | TCP |    Data   |
            -------------------------------------------------------
            |<- mutable field processing ->|<- immutable fields ->|
            |<----- authenticated except for mutable fields ----->|
```

In the IPv6 context, AH is viewed as an end-to-end payload, and thus should appear after hop-by-hop, routing, and fragmentation extension headers. The destination options extension header(s) could appear before or after or both before and after the AH header depending on the semantics desired. The following diagram illustrates AH transport mode positioning for a typical IPv6 packet.

```
                       BEFORE APPLYING AH
            ---------------------------------------
      IPv6  |             | ext hdrs |     |      |
            | orig IP hdr |if present| TCP | Data |
            ---------------------------------------

                      AFTER APPLYING AH
           ------------------------------------------------------------
     IPv6  |             |hop-by-hop, dest*, |    | dest |     |      |
           |orig IP hdr  |routing, fragment. | AH | opt* | TCP | Data |
           ------------------------------------------------------------
           |<--- mutable field processing -->|<-- immutable fields -->|
           |<---- authenticated except for mutable fields ----------->|

                * = if present, could be before AH, after AH, or both
```

ESP and AH headers can be combined in a variety of modes. The IPsec Architecture document describes the combinations of security associations that must be supported.

Note that in transport mode, for "bump-in-the-stack" or "bump-in-the-wire" implementations, as defined in the Security Architecture document, inbound and outbound IP fragments may require an IPsec implementation to perform extra IP reassembly/fragmentation in order to both conform to this specification and provide transparent IPsec support. Special care is required to perform such operations within these implementations when multiple interfaces are in use.

3.1.2. Tunnel Mode

In tunnel mode, the "inner" IP header carries the ultimate (IP) source and destination addresses, while an "outer" IP header contains the addresses of the IPsec "peers," e.g., addresses of security gateways. Mixed inner and outer IP versions are allowed, i.e., IPv6 over IPv4 and IPv4 over IPv6. In tunnel mode, AH protects the entire inner IP packet, including the entire inner IP header. The position of AH in tunnel mode, relative to the outer IP header, is the same as for AH in transport mode. The following diagram illustrates AH tunnel mode positioning for typical IPv4 and IPv6 packets.

```
         ----------------------------------------------------------------
    IPv4 |                              |    | orig IP hdr*  |   |      |
         |new IP header * (any options) | AH | (any options) |TCP| Data |
         ----------------------------------------------------------------
         |<- mutable field processing ->|<------ immutable fields ----->|
         |<- authenticated except for mutable fields in the new IP hdr->|

         --------------------------------------------------------------
    IPv6 |           | ext hdrs*|    |            | ext hdrs*|   |    |
         |new IP hdr*|if present| AH |orig IP hdr*|if present|TCP|Data|
         --------------------------------------------------------------
         |<--- mutable field -->|<--------- immutable fields -------->|
         |       processing     |
         |<-- authenticated except for mutable fields in new IP hdr ->|

           * = if present, construction of outer IP hdr/extensions and
               modification of inner IP hdr/extensions is discussed in
               the Security Architecture document.
```

3.2. Integrity Algorithms

The integrity algorithm employed for the ICV computation is specified by the SA. For point-to-point communication, suitable integrity algorithms include keyed Message Authentication Codes (MACs) based on symmetric encryption algorithms (e.g., AES [AES]) or on one-way hash functions (e.g., MD5, SHA-1, SHA-256, etc.). For multicast communication, a variety of cryptographic strategies for providing integrity have been developed and research continues in this area.

3.3. Outbound Packet Processing

In transport mode, the sender inserts the AH header after the IP header and before a next layer protocol header, as described above. In tunnel mode, the outer and inner IP header/extensions can be interrelated in a variety of ways. The construction of the outer IP header/extensions during the encapsulation process is described in the Security Architecture document.

3.3.1. Security Association Lookup

AH is applied to an outbound packet only after an IPsec implementation determines that the packet is associated with an SA that calls for AH processing. The process of determining what, if any, IPsec processing is applied to outbound traffic is described in the Security Architecture document.

3.3.2. Sequence Number Generation

The sender’s counter is initialized to 0 when an SA is established. The sender increments the sequence number (or ESN) counter for this SA and inserts the low-order 32 bits of the value into the Sequence Number field. Thus, the first packet sent using a given SA will contain a sequence number of 1.

If anti-replay is enabled (the default), the sender checks to ensure that the counter has not cycled before inserting the new value in the Sequence Number field. In other words, the sender MUST NOT send a packet on an SA if doing so would cause the sequence number to cycle. An attempt to transmit a packet that would result in sequence number overflow is an auditable event. The audit log entry for this event SHOULD include the SPI value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.

The sender assumes anti-replay is enabled as a default, unless otherwise notified by the receiver (see Section 3.4.3) or if the SA was configured using manual key management. Thus, typical behavior of an AH implementation calls for the sender to establish a new SA when the Sequence Number (or ESN) cycles, or in anticipation of this value cycling.

If anti-replay is disabled (as noted above), the sender does not need to monitor or reset the counter, e.g., in the case of manual key management (see Section 5). However, the sender still increments the counter and when it reaches the maximum value, the counter rolls over back to zero. (This behavior is recommended for multi-sender, multicast SAs, unless anti-replay mechanisms outside the scope of this standard are negotiated between the sender and receiver.)

If ESN (see Appendix B) is selected, only the low-order 32 bits of the sequence number are transmitted in the Sequence Number field, although both sender and receiver maintain full 64-bit ESN counters. However, the high-order 32 bits are included in the ICV calculation.

Note: If a receiver chooses not to enable anti-replay for an SA, then the receiver SHOULD NOT negotiate ESN in an SA management protocol. Use of ESN creates a need for the receiver to manage the anti-replay window (in order to determine the correct value for the high-order bits of the ESN, which are employed in the ICV computation), which is generally contrary to the notion of disabling anti-replay for an SA.

3.3.3. Integrity Check Value Calculation

The AH ICV is computed over:

- o IP or extension header fields before the AH header that are either immutable in transit or that are predictable in value upon arrival at the endpoint for the AH SA
- o the AH header (Next Header, Payload Len, Reserved, SPI, Sequence Number (low-order 32 bits), and the ICV (which is set to zero for this computation), and explicit padding bytes (if any))
- o everything after AH is assumed to be immutable in transit
- o the high-order bits of the ESN (if employed), and any implicit padding required by the integrity algorithm

3.3.3.1. Handling Mutable Fields

If a field may be modified during transit, the value of the field is set to zero for purposes of the ICV computation. If a field is mutable, but its value at the (IPsec) receiver is predictable, then that value is inserted into the field for purposes of the ICV calculation. The Integrity Check Value field is also set to zero in preparation for this computation. Note that by replacing each field's value with zero, rather than omitting the field, alignment is preserved for the ICV calculation. Also, the zero-fill approach ensures that the length of the fields that are so handled cannot be changed during transit, even though their contents are not explicitly covered by the ICV.

As a new extension header or IPv4 option is created, it will be defined in its own RFC and SHOULD include (in the Security Considerations section) directions for how it should be handled when calculating the AH ICV. If the IP (v4 or v6) implementation encounters an extension header that it does not recognize, it will discard the packet and send an ICMP message. IPsec will never see the packet. If the IPsec implementation encounters an IPv4 option that it does not recognize, it should zero the whole option, using the second byte of the option as the length. IPv6 options (in Destination Extension Headers or the Hop-by-Hop Extension Header) contain a flag indicating mutability, which determines appropriate processing for such options.

3.3.3.1.1. ICV Computation for IPv4

3.3.3.1.1.1. Base Header Fields

The IPv4 base header fields are classified as follows:

Immutable
- Version
- Internet Header Length
- Total Length
- Identification
- Protocol (This should be the value for AH.)
- Source Address
- Destination Address (without loose or strict source routing)

Mutable but predictable
- Destination Address (with loose or strict source routing)

Mutable (zeroed prior to ICV calculation)
- Differentiated Services Code Point (DSCP) (6 bits, see RFC 2474 [NBBB98])
- Explicit Congestion Notification (ECN) (2 bits, see RFC 3168 [RFB01])
- Flags
- Fragment Offset
- Time to Live (TTL)
- Header Checksum

DSCP - Routers may rewrite the DS field as needed to provide a desired local or end-to-end service, thus its value upon reception cannot be predicted by the sender.

ECN - This will change if a router along the route experiences congestion, and thus its value upon reception cannot be predicted by the sender.

Flags - This field is excluded because an intermediate router might set the DF bit, even if the source did not select it.

Fragment Offset - Since AH is applied only to non-fragmented IP packets, the Offset Field must always be zero, and thus it is excluded (even though it is predictable).

TTL - This is changed en route as a normal course of processing by routers, and thus its value at the receiver is not predictable by the sender.

Header Checksum - This will change if any of these other fields change, and thus its value upon reception cannot be predicted by the sender.

3.3.3.1.1.2. Options

For IPv4 (unlike IPv6), there is no mechanism for tagging options as mutable in transit. Hence the IPv4 options are explicitly listed in Appendix A and classified as immutable, mutable but predictable, or mutable. For IPv4, the entire option is viewed as a unit; so even though the type and length fields within most options are immutable in transit, if an option is classified as mutable, the entire option is zeroed for ICV computation purposes.

3.3.3.1.2. ICV Computation for IPv6

3.3.3.1.2.1. Base Header Fields

The IPv6 base header fields are classified as follows:

Immutable
- Version
- Payload Length
- Next Header
- Source Address
- Destination Address (without Routing Extension Header)

Mutable but predictable
- Destination Address (with Routing Extension Header)

Mutable (zeroed prior to ICV calculation)
- DSCP (6 bits, see RFC2474 [NBBB98])
- ECN (2 bits, see RFC3168 [RFB01])
- Flow Label (*)
- Hop Limit

(*) The flow label described in AHv1 was mutable, and in RFC 2460 [DH98] was potentially mutable. To retain compatibility with existing AH implementations, the flow label is not included in the ICV in AHv2.

3.3.3.1.2.2. Extension Headers Containing Options

IPv6 options in the Hop-by-Hop and Destination Extension Headers contain a bit that indicates whether the option might change (unpredictably) during transit. For any option for which contents may change en-route, the entire "Option Data" field must be treated as zero-valued octets when computing or verifying the ICV. The

Option Type and Opt Data Len are included in the ICV calculation. All options for which the bit indicates immutability are included in the ICV calculation. See the IPv6 specification [DH98] for more information.

3.3.3.1.2.3. Extension Headers Not Containing Options

The IPv6 extension headers that do not contain options are explicitly listed in Appendix A and classified as immutable, mutable but predictable, or mutable.

3.3.3.2. Padding and Extended Sequence Numbers

3.3.3.2.1. ICV Padding

As mentioned in Section 2.6, the ICV field may include explicit padding if required to ensure that the AH header is a multiple of 32 bits (IPv4) or 64 bits (IPv6). If padding is required, its length is determined by two factors:

- the length of the ICV
- the IP protocol version (v4 or v6)

For example, if the output of the selected algorithm is 96 bits, no padding is required for IPv4 or IPv6. However, if a different length ICV is generated, due to use of a different algorithm, then padding may be required depending on the length and IP protocol version. The content of the padding field is arbitrarily selected by the sender. (The padding is arbitrary, but need not be random to achieve security.) These padding bytes are included in the ICV calculation, counted as part of the Payload Length, and transmitted at the end of the ICV field to enable the receiver to perform the ICV calculation. Inclusion of padding in excess of the minimum amount required to satisfy IPv4/IPv6 alignment requirements is prohibited.

3.3.3.2.2. Implicit Packet Padding and ESN

If the ESN option is elected for an SA, then the high-order 32 bits of the ESN must be included in the ICV computation. For purposes of ICV computation, these bits are appended (implicitly) immediately after the end of the payload, and before any implicit packet padding.

For some integrity algorithms, the byte string over which the ICV computation is performed must be a multiple of a blocksize specified by the algorithm. If the IP packet length (including AH and the 32 high-order bits of the ESN, if enabled) does not match the blocksize requirements for the algorithm, implicit padding MUST be appended to the end of the packet, prior to ICV computation. The padding octets

MUST have a value of zero. The blocksize (and hence the length of the padding) is specified by the algorithm specification. This padding is not transmitted with the packet. The document that defines an integrity algorithm MUST be consulted to determine if implicit padding is required as described above. If the document does not specify an answer to this, then the default is to assume that implicit padding is required (as needed to match the packet length to the algorithm’s blocksize.) If padding bytes are needed but the algorithm does not specify the padding contents, then the padding octets MUST have a value of zero.

3.3.4. Fragmentation

If required, IP fragmentation occurs after AH processing within an IPsec implementation. Thus, transport mode AH is applied only to whole IP datagrams (not to IP fragments). An IPv4 packet to which AH has been applied may itself be fragmented by routers en route, and such fragments must be reassembled prior to AH processing at a receiver. (This does not apply to IPv6, where there is no router-initiated fragmentation.) In tunnel mode, AH is applied to an IP packet, the payload of which may be a fragmented IP packet. For example, a security gateway or a "bump-in-the-stack" or "bump-in-the-wire" IPsec implementation (see the Security Architecture document for details) may apply tunnel mode AH to such fragments.

NOTE: For transport mode -- As mentioned at the end of Section 3.1.1, bump-in-the-stack and bump-in-the-wire implementations may have to first reassemble a packet fragmented by the local IP layer, then apply IPsec, and then fragment the resulting packet.

NOTE: For IPv6 -- For bump-in-the-stack and bump-in-the-wire implementations, it will be necessary to examine all the extension headers to determine if there is a fragmentation header and hence that the packet needs reassembling prior to IPsec processing.

Fragmentation, whether performed by an IPsec implementation or by routers along the path between IPsec peers, significantly reduces performance. Moreover, the requirement for an AH receiver to accept fragments for reassembly creates denial of service vulnerabilities. Thus, an AH implementation MAY choose to not support fragmentation and may mark transmitted packets with the DF bit, to facilitate Path MTU (PMTU) discovery. In any case, an AH implementation MUST support generation of ICMP PMTU messages (or equivalent internal signaling for native host implementations) to minimize the likelihood of fragmentation. Details of the support required for MTU management are contained in the Security Architecture document.

3.4. Inbound Packet Processing

If there is more than one IPsec header/extension present, the processing for each one ignores (does not zero, does not use) any IPsec headers applied subsequent to the header being processed.

3.4.1. Reassembly

If required, reassembly is performed prior to AH processing. If a packet offered to AH for processing appears to be an IP fragment, i.e., the OFFSET field is nonzero or the MORE FRAGMENTS flag is set, the receiver MUST discard the packet; this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address, and (in IPv6) the Flow ID.

NOTE: For packet reassembly, the current IPv4 spec does NOT require either the zeroing of the OFFSET field or the clearing of the MORE FRAGMENTS flag. In order for a reassembled packet to be processed by IPsec (as opposed to discarded as an apparent fragment), the IP code must do these two things after it reassembles a packet.

3.4.2. Security Association Lookup

Upon receipt of a packet containing an IP Authentication Header, the receiver determines the appropriate (unidirectional) SA via lookup in the SAD. For a unicast SA, this determination is based on the SPI or the SPI plus protocol field, as described in Section 2.4. If an implementation supports multicast traffic, the destination address is also employed in the lookup (in addition to the SPI), and the sender address also may be employed, as described in Section 2.4. (This process is described in more detail in the Security Architecture document.) The SAD entry for the SA also indicates whether the Sequence Number field will be checked and whether 32- or 64-bit sequence numbers are employed for the SA. The SAD entry for the SA also specifies the algorithm(s) employed for ICV computation, and indicates the key required to validate the ICV.

If no valid Security Association exists for this packet the receiver MUST discard the packet; this is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address, and (in IPv6) the Flow ID.

(Note that SA management traffic, such as IKE packets, does not need to be processed based on SPI, i.e., one can de-multiplex this traffic separately based on Next Protocol and Port fields, for example.)

3.4.3. Sequence Number Verification

All AH implementations MUST support the anti-replay service, though its use may be enabled or disabled by the receiver on a per-SA basis. Anti-replay is applicable to unicast as well as multicast SAs. However, this standard specifies no mechanisms for providing anti-replay for a multi-sender SA (unicast or multicast). In the absence of negotiation (or manual configuration) of an anti-replay mechanism for such an SA, it is recommended that sender and receiver checking of the Sequence Number for the SA be disabled (via negotiation or manual configuration), as noted below.

If the receiver does not enable anti-replay for an SA, no inbound checks are performed on the Sequence Number. However, from the perspective of the sender, the default is to assume that anti-replay is enabled at the receiver. To avoid having the sender do unnecessary sequence number monitoring and SA setup (see Section 3.3.2, "Sequence Number Generation"), if an SA establishment protocol such as IKE is employed, the receiver SHOULD notify the sender, during SA establishment, if the receiver will not provide anti-replay protection.

If the receiver has enabled the anti-replay service for this SA, the receive packet counter for the SA MUST be initialized to zero when the SA is established. For each received packet, the receiver MUST verify that the packet contains a Sequence Number that does not duplicate the Sequence Number of any other packets received during the life of this SA. This SHOULD be the first AH check applied to a packet after it has been matched to an SA, to speed rejection of duplicate packets.

Duplicates are rejected through the use of a sliding receive window. How the window is implemented is a local matter, but the following text describes the functionality that the implementation must exhibit.

The "right" edge of the window represents the highest, validated Sequence Number value received on this SA. Packets that contain sequence numbers lower than the "left" edge of the window are rejected. Packets falling within the window are checked against a list of received packets within the window.

If the ESN option is selected for an SA, only the low-order 32 bits of the sequence number are explicitly transmitted, but the receiver employs the full sequence number computed using the high-order 32 bits for the indicated SA (from his local counter) when checking the received Sequence Number against the receive window. In constructing the full sequence number, if the low-order 32 bits carried in the packet are lower in value than the low-order 32 bits of the receiver’s sequence number counter, the receiver assumes that the high-order 32 bits have been incremented, moving to a new sequence number subspace. (This algorithm accommodates gaps in reception for a single SA as large as 2**32-1 packets. If a larger gap occurs, additional, heuristic checks for re-synchronization of the receiver’s sequence number counter MAY be employed, as described in Appendix B.)

If the received packet falls within the window and is not a duplicate, or if the packet is to the right of the window, then the receiver proceeds to ICV verification. If the ICV validation fails, the receiver MUST discard the received IP datagram as invalid. This is an auditable event. The audit log entry for this event SHOULD include the SPI value, date/time, Source Address, Destination Address, the Sequence Number, and (in IPv6) the Flow ID. The receive window is updated only if the ICV verification succeeds.

A MINIMUM window size of 32 packets MUST be supported, but a window size of 64 is preferred and SHOULD be employed as the default. Another window size (larger than the MINIMUM) MAY be chosen by the receiver. (The receiver does NOT notify the sender of the window size.) The receive window size should be increased for higher-speed environments, irrespective of assurance issues. Values for minimum and recommended receive window sizes for very high-speed (e.g., multi-gigabit/second) devices are not specified by this standard.

3.4.4. Integrity Check Value Verification

The receiver computes the ICV over the appropriate fields of the packet, using the specified integrity algorithm, and verifies that it is the same as the ICV included in the ICV field of the packet. Details of the computation are provided below.

If the computed and received ICVs match, then the datagram is valid, and it is accepted. If the test fails, then the receiver MUST discard the received IP datagram as invalid. This is an auditable event. The audit log entry SHOULD include the SPI value, date/time received, Source Address, Destination Address, and (in IPv6) the Flow ID.

Implementation Note:

> Implementations can use any set of steps that results in the same result as the following set of steps. Begin by saving the ICV value and replacing it (but not any ICV field padding) with zero. Zero all other fields that may have been modified during transit. (See Section 3.3.3.1, "Handling Mutable Fields", for a discussion of which fields are zeroed before performing the ICV calculation.)

If the ESN option is elected for this SA, append the high-order 32 bits of the ESN after the end of the packet. Check the overall length of the packet (as described above), and if it requires implicit padding based on the requirements of the integrity algorithm, append zero-filled bytes to the end of the packet (after the ESN if present) as required. Perform the ICV computation and compare the result with the saved value, using the comparison rules defined by the algorithm specification. (For example, if a digital signature and one-way hash are used for the ICV computation, the matching process is more complex.)

4. Auditing

Not all systems that implement AH will implement auditing. However, if AH is incorporated into a system that supports auditing, then the AH implementation MUST also support auditing and MUST allow a system administrator to enable or disable auditing for AH. For the most part, the granularity of auditing is a local matter. However, several auditable events are identified in this specification, and for each of these events a minimum set of information that SHOULD be included in an audit log is defined. Additional information also MAY be included in the audit log for each of these events, and additional events, not explicitly called out in this specification, also MAY result in audit log entries. There is no requirement for the receiver to transmit any message to the purported sender in response to the detection of an auditable event, because of the potential to induce denial of service via such action.

5. Conformance Requirements

Implementations that claim conformance or compliance with this specification MUST fully implement the AH syntax and processing described here for unicast traffic, and MUST comply with all requirements of the Security Architecture document [Ken-Arch]. Additionally, if an implementation claims to support multicast traffic, it MUST comply with the additional requirements specified for support of such traffic. If the key used to compute an ICV is manually distributed, correct provision of the anti-replay service would require correct maintenance of the counter state at the sender, until the key is replaced, and there likely would be no automated recovery provision if counter overflow were imminent. Thus, a compliant implementation SHOULD NOT provide this service in conjunction with SAs that are manually keyed.

The mandatory-to-implement algorithms for use with AH are described in a separate RFC [Eas04], to facilitate updating the algorithm requirements independently from the protocol per se. Additional algorithms, beyond those mandated for AH, MAY be supported.

6. Security Considerations

Security is central to the design of this protocol, and these security considerations permeate the specification. Additional security-relevant aspects of using the IPsec protocol are discussed in the Security Architecture document.

7. Differences from RFC 2402

This document differs from RFC 2402 [RFC2402] in the following ways.

- o SPI -- modified to specify a uniform algorithm for SAD lookup for unicast and multicast SAs, covering a wider range of multicast technologies. For unicast, the SPI may be used alone to select an SA, or may be combined with the protocol, at the option of the receiver. For multicast SAs, the SPI is combined with the destination address, and optionally the source address, to select an SA.
- o Extended Sequence Number -- added a new option for a 64-bit sequence number for very high-speed communications. Clarified sender and receiver processing requirements for multicast SAs and multi-sender SAs.
- o Moved references to mandatory algorithms to a separate document [Eas04].

8. Acknowledgements

The author would like to acknowledge the contributions of Ran Atkinson, who played a critical role in initial IPsec activities, and who authored the first series of IPsec standards: RFCs 1825-1827. Karen Seo deserves special thanks for providing help in the editing of this and the previous version of this specification. The author also would like to thank the members of the IPsec and MSEC working groups who have contributed to the development of this protocol specification.

9. References

9.1. Normative References

[Bra97] Bradner, S., "Key words for use in RFCs to Indicate Requirement Level", BCP 14, RFC 2119, March 1997.

[DH98] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[Eas04] 3rd Eastlake, D., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4305, December 2005.

[Ken-Arch] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC1108] Kent, S., "U.S. Department of Defense Security Options for the Internet Protocol", RFC 1108, November 1991.

9.2. Informative References

[AES] Advanced Encryption Standard (AES), Federal Information Processing Standard 197, National Institutes of Standards and Technology, November 26, 2001.

[HC03] Holbrook, H. and B. Cain, "Source Specific Multicast for IP", Work in Progress, November 3, 2002.

[IKEv2] Kaufman, C., Ed., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[Ken-ESP] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[NBBB98] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFB01] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, July 1988.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1385] Wang, Z., "EIP: The Extended Internet Protocol", RFC 1385, November 1992.

[RFC1393] Malkin, G., "Traceroute Using an IP Option", RFC 1393, January 1993.

[RFC1770] Graff, C., "IPv4 Option for Sender Directed Multi-Destination Delivery", RFC 1770, March 1995.

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, February 1997.

[RFC2402] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[RFC3547] Baugher, M., Weis, B., Hardjono, T., and H. Harney, "The Group Domain of Interpretation", RFC 3547, July 2003.

[RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, March 2004.

Appendix A: Mutability of IP Options/Extension Headers

A1. IPv4 Options

This table shows how the IPv4 options are classified with regard to "mutability". Where two references are provided, the second one supercedes the first. This table is based in part on information provided in RFC 1700, "ASSIGNED NUMBERS", (October 1994).

Copy	Class	Opt. #	Name	Reference
IMMUTABLE -- included in ICV calculation				
0	0	0	End of Options List	[RFC791]
0	0	1	No Operation	[RFC791]
1	0	2	Security	[RFC1108] (historic but in use)
1	0	5	Extended Security	[RFC1108] (historic but in use)
1	0	6	Commercial Security	
1	0	20	Router Alert	[RFC2113]
1	0	21	Sender Directed Multi-Destination Delivery	[RFC1770]
MUTABLE -- zeroed				
1	0	3	Loose Source Route	[RFC791]
0	2	4	Time Stamp	[RFC791]
0	0	7	Record Route	[RFC791]
1	0	9	Strict Source Route	[RFC791]
0	2	18	Traceroute	[RFC1393]
EXPERIMENTAL, SUPERCEDED -- zeroed				
1	0	8	Stream ID	[RFC791, RFC1122 (Host Req)]
0	0	11	MTU Probe	[RFC1063, RFC1191 (PMTU)]
0	0	12	MTU Reply	[RFC1063, RFC1191 (PMTU)]
1	0	17	Extended Internet Protocol	[RFC1385, DH98 (IPv6)]
0	0	10	Experimental Measurement	
1	2	13	Experimental Flow Control	
1	0	14	Experimental Access Ctl	
0	0	15	???	
1	0	16	IMI Traffic Descriptor	
1	0	19	Address Extension	

NOTE: Use of the Router Alert option is potentially incompatible with use of IPsec. Although the option is immutable, its use implies that each router along a packet's path will "process" the packet and consequently might change the packet. This would happen on a hop-by-hop basis as the packet goes from router to router. Prior to being processed by the application to which the option contents are directed (e.g., Resource Reservation Protocol (RSVP)/Internet Group Management Protocol (IGMP)), the packet should encounter AH processing. However, AH processing would require that each router along the path is a member of a multicast-SA defined by the SPI. This might pose problems for packets that are not strictly source routed, and it requires multicast support techniques not currently available.

NOTE: Addition or removal of security labels (e.g., Basic Security Option (BSO), Extended Security Option (ESO), or Commercial Internet Protocol Security Option (CIPSO)) by systems along a packet’s path conflicts with the classification of these IP options as immutable and is incompatible with the use of IPsec.

NOTE: End of Options List options SHOULD be repeated as necessary to ensure that the IP header ends on a 4-byte boundary in order to ensure that there are no unspecified bytes that could be used for a covert channel.

A2. IPv6 Extension Headers

This table shows how the IPv6 extension headers are classified with regard to "mutability".

Option/Extension Name	Reference
MUTABLE BUT PREDICTABLE -- included in ICV calculation	
Routing (Type 0)	[DH98]
BIT INDICATES IF OPTION IS MUTABLE (CHANGES UNPREDICTABLY DURING TRANSIT)	
Hop-by-Hop options	[DH98]
Destination options	[DH98]
NOT APPLICABLE	
Fragmentation	[DH98]

Options -- IPv6 options in the Hop-by-Hop and Destination Extension Headers contain a bit that indicates whether the option might change (unpredictably) during transit. For any option for which contents may change en route, the entire "Option Data" field must be treated as zero-valued octets when computing or verifying the ICV. The Option Type and Opt Data Len are included in the ICV calculation. All options for which the bit indicates immutability are included in the ICV calculation. See the IPv6 specification [DH98] for more information.

Routing (Type 0) -- The IPv6 Routing Header "Type 0" will rearrange the address fields within the packet during transit from source to destination. However, the contents of the packet as it will appear at the receiver are known to the sender and to all intermediate hops. Hence, the IPv6 Routing Header "Type 0" is included in the Integrity Check Value calculation as mutable but predictable. The sender must order the field so that it appears as it will at the receiver, prior to performing the ICV computation.

Fragmentation -- Fragmentation occurs after outbound IPsec processing (Section 3.3) and reassembly occurs before inbound IPsec processing (Section 3.4). So the Fragmentation Extension Header, if it exists, is not seen by IPsec.

Note that on the receive side, the IP implementation could leave a Fragmentation Extension Header in place when it does re-assembly. If this happens, then when AH receives the packet, before doing ICV processing, AH MUST "remove" (or skip over) this header and change the previous header’s "Next Header" field to be the "Next Header" field in the Fragmentation Extension Header.

Note that on the send side, the IP implementation could give the IPsec code a packet with a Fragmentation Extension Header with Offset of 0 (first fragment) and a More Fragments Flag of 0 (last fragment). If this happens, then before doing ICV processing, AH MUST first "remove" (or skip over) this header and change the previous header’s "Next Header" field to be the "Next Header" field in the Fragmentation Extension Header.

Appendix B: Extended (64-bit) Sequence Numbers

B1. Overview

This appendix describes an Extended Sequence Number (ESN) scheme for use with IPsec (ESP and AH) that employs a 64-bit sequence number, but in which only the low-order 32 bits are transmitted as part of each packet. It covers both the window scheme used to detect replayed packets and the determination of the high-order bits of the sequence number that are used both for replay rejection and for computation of the ICV. It also discusses a mechanism for handling loss of synchronization relative to the (not transmitted) high-order bits.

B2. Anti-Replay Window

The receiver will maintain an anti-replay window of size W. This window will limit how far out of order a packet can be, relative to the packet with the highest sequence number that has been authenticated so far. (No requirement is established for minimum or recommended sizes for this window, beyond the 32- and 64-packet values already established for 32-bit sequence number windows. However, it is suggested that an implementer scale these values consistent with the interface speed supported by an implementation that makes use of the ESN option. Also, the algorithm described below assumes that the window is no greater than 2^31 packets in width.) All 2^32 sequence numbers associated with any fixed value for the high-order 32 bits (Seqh) will hereafter be called a sequence number subspace. The following table lists pertinent variables and their definitions.

Var. Name	Size (bits)	Meaning
W	32	Size of window
T	64	Highest sequence number authenticated so far, upper bound of window
Tl	32	Lower 32 bits of T
Th	32	Upper 32 bits of T
B	64	Lower bound of window
Bl	32	Lower 32 bits of B
Bh	32	Upper 32 bits of B
Seq	64	Sequence Number of received packet
Seql	32	Lower 32 bits of Seq
Seqh	32	Upper 32 bits of Seq

When performing the anti-replay check, or when determining which high-order bits to use to authenticate an incoming packet, there are two cases:

+ Case A: Tl >= (W - 1). In this case, the window is within one sequence number subspace. (See Figure 1)
+ Case B: Tl < (W - 1). In this case, the window spans two sequence number subspaces. (See Figure 2)

In the figures below, the bottom line ("----") shows two consecutive sequence number subspaces, with zeros indicating the beginning of each subspace. The two shorter lines above it show the higher-order bits that apply. The "====" represents the window. The "****" represents future sequence numbers, i.e., those beyond the current highest sequence number authenticated (ThTl).

```
        Th+1                         *********

        Th                =======*****

              --0--------+-----+-----0--------+-----------0--
                         Bl    Tl            Bl
                                        (Bl+2^32) mod 2^32
```

Figure 1 -- Case A

```
        Th                           ====**************

        Th-1                      ===

              --0-----------------+--0--+--------------+--0--
                                  Bl    Tl            Bl
                                                 (Bl+2^32) mod 2^32
```

Figure 2 -- Case B

B2.1. Managing and Using the Anti-Replay Window

The anti-replay window can be thought of as a string of bits where ‘W’ defines the length of the string. W = T - B + 1 and cannot exceed 2^32 - 1 in value. The bottom-most bit corresponds to B and the top-most bit corresponds to T, and each sequence number from Bl through Tl is represented by a corresponding bit. The value of the bit indicates whether or not a packet with that sequence number has been received and authenticated, so that replays can be detected and rejected.

When a packet with a 64-bit sequence number (Seq) greater than T is received and validated,

+ B is increased by (Seq - T)
+ (Seq - T) bits are dropped from the low end of the window
+ (Seq - T) bits are added to the high end of the window
+ The top bit is set to indicate that a packet with that sequence number has been received and authenticated
+ The new bits between T and the top bit are set to indicate that no packets with those sequence numbers have been received yet.
+ T is set to the new sequence number

In checking for replayed packets,

+ Under Case A: If Seql >= Bl (where Bl = Tl - W + 1) AND Seql <= Tl, then check the corresponding bit in the window to see if this Seql has already been seen. If yes, reject the packet. If no, perform integrity check (see Appendix B2.2 below for determination of SeqH).

+ Under Case B: If Seql >= Bl (where Bl = Tl - W + 1) OR Seql <= Tl, then check the corresponding bit in the window to see if this Seql has already been seen. If yes, reject the packet. If no, perform integrity check (see Appendix B2.2 below for determination of Seqh).

B2.2. Determining the Higher-Order Bits (Seqh) of the Sequence Number

Because only ‘Seql’ will be transmitted with the packet, the receiver must deduce and track the sequence number subspace into which each packet falls, i.e., determine the value of Seqh. The following equations define how to select Seqh under "normal" conditions; see Appendix B3 for a discussion of how to recover from extreme packet loss.

+ Under Case A (Figure 1):
  If Seql >= Bl (where Bl = Tl - W + 1), then Seqh = Th
  If Seql < Bl (where Bl = Tl - W + 1), then Seqh = Th + 1

+ Under Case B (Figure 2):
  If Seql >= Bl (where Bl = Tl - W + 1), then Seqh = Th - 1
  If Seql < Bl (where Bl = Tl - W + 1), then Seqh = Th

B2.3. Pseudo-Code Example

The following pseudo-code illustrates the above algorithms for anti-replay and integrity checks. The values for 'Seql', 'Tl', 'Th', and 'W' are 32-bit unsigned integers. Arithmetic is mod 2^32.

```
If (Tl >= W - 1)                                  Case A
    If (Seql >= Tl - W + 1)
        Seqh = Th
        If (Seql <= Tl)
            If (pass replay check)
                If (pass integrity check)
                    Set bit corresponding to Seql
                    Pass the packet on
                Else reject packet
            Else reject packet
        Else
            If (pass integrity check)
                Tl = Seql (shift bits)
                Set bit corresponding to Seql
                Pass the packet on
            Else reject packet
    Else
        Seqh = Th + 1
        If (pass integrity check)
            Tl = Seql (shift bits)
            Th = Th + 1
            Set bit corresponding to Seql
            Pass the packet on
        Else reject packet
Else                                          Case B
    If (Seql >= Tl - W + 1)
        Seqh = Th - 1
        If (pass replay check)
            If (pass integrity check)
                Set the bit corresponding to Seql
                Pass packet on
            Else reject packet
        Else reject packet
    Else
        Seqh = Th
        If (Seql <= Tl)
            If (pass replay check)
                If (pass integrity check)
                    Set the bit corresponding to Seql
                    Pass packet on
                Else reject packet
            Else reject packet
```

```
        Else
            If (pass integrity check)
                Tl = Seql (shift bits)
                Set the bit corresponding to Seql
                Pass packet on
            Else reject packet
```

B3. Handling Loss of Synchronization due to Significant Packet Loss

If there is an undetected packet loss of 2^32 or more consecutive packets on a single SA, then the transmitter and receiver will lose synchronization of the high-order bits, i.e., the equations in Appendix B2.2. will fail to yield the correct value. Unless this problem is detected and addressed, subsequent packets on this SA will fail authentication checks and be discarded. The following procedure SHOULD be implemented by any IPsec (ESP or AH) implementation that supports the ESN option.

Note that this sort of extended traffic loss seems unlikely to occur if any significant fraction of the traffic on the SA in question is TCP, because the source would fail to receive ACKs and would stop sending long before 2^32 packets had been lost. Also, for any bi-directional application, even ones operating above UDP, such an extended outage would likely result in triggering some form of timeout. However, a unidirectional application, operating over UDP, might lack feedback that would cause automatic detection of a loss of this magnitude, hence the motivation to develop a recovery method for this case.

The solution we've chosen was selected to:

+ minimize the impact on normal traffic processing.

+ avoid creating an opportunity for a new denial of service attack such as might occur by allowing an attacker to force diversion of resources to a re-synchronization process.
+ limit the recovery mechanism to the receiver because anti-replay is a service only for the receiver, and the transmitter generally is not aware of whether the receiver is using sequence numbers in support of this optional service. It is preferable for recovery mechanisms to be local to the receiver. This also allows for backward compatibility.

B3.1. Triggering Re-synchronization

For each SA, the receiver records the number of consecutive packets that fail authentication. This count is used to trigger the re-synchronization process, which should be performed in the background or using a separate processor. Receipt of a valid packet on the SA resets the counter to zero. The value used to trigger the re-synchronization process is a local parameter. There is no requirement to support distinct trigger values for different SAs, although an implementer may choose to do so.

B3.2. Re-synchronization Process

When the above trigger point is reached, a "bad" packet is selected for which authentication is retried using successively larger values for the upper half of the sequence number (Seqh). These values are generated by incrementing by one for each retry. The number of retries should be limited, in case this is a packet from the "past" or a bogus packet. The limit value is a local parameter. (Because the Seqh value is implicitly placed after the AH (or ESP) payload, it may be possible to optimize this procedure by executing the integrity algorithm over the packet up to the endpoint of the payload, then compute different candidate ICVs by varying the value of Seqh.) Successful authentication of a packet via this procedure resets the consecutive failure count and sets the value of T to that of the received packet.

This solution requires support only on the part of the receiver, thereby allowing for backward compatibility. Also, because re-synchronization efforts would either occur in the background or utilize an additional processor, this solution does not impact traffic processing and a denial of service attack cannot divert resources away from traffic processing.

Author's Address

Stephen Kent
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
USA

Phone: +1 (617) 873-3988
EMail: kent@bbn.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.