IPSECME Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

S. Klassert
A. Antony
secunet
C. Hopps
LabN Consulting, L.L.C.

2 March 2026

Enhanced Encapsulating Security Payload (EESP)
draft-ietf-ipsecme-eesp-03

Abstract

This document describes the Enhanced Encapsulating Security Payload (EESP) protocol, which builds on the existing IP Encapsulating Security Payload (ESP) protocol. It is designed to modernize and overcome limitations in the ESP protocol.

EESP adds Session IDs (e.g., to support CPU pinning and QoS support based on the inner traffic flow), changes some previously mandatory fields to optional, and moves the ESP trailer into the EESP header. Additionally, EESP adds header options adapted from IPv6 to allow for future extension. New header options are defined which add a crypt-offset to allow for exposing inner flow information for middlebox use.

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 3 September 2026.

Copyright Notice

Table of Contents

1. Introduction

Due to the absence of a version number in the packet header of the ESP protocol, ESP can’t be updated in a transparent way. Any updates to ESP must be negotiated by IKEv2 and are therefore indiscernible to intermediate devices such as routers and firewalls. In the recent past, several attempts were taken to introduce an inner-flow identifier for certain use cases. Examples are [I-D.ponchon-ipsecme-anti-replay-subspaces] and [I-D.he-ipsecme-vpn-shared-ipsecsa]. Such an identifier could also be used to perform ECMP based on the inner flows at intermediate

devices or endpoints. Additionally to that, there exists a specification of the [PSP] protocol that has the need of an inner-flow identifier, called Virtual Network Identifier (VNI) there. PSP also defines a Crypt Offset to expose parts of the headers of the inner packet. EESP provides a solution for all the aforementioned use cases. This document defines a Session ID that can be used as a flow identifier and a Crypt Offset Option. Future documents can define the meaning of additional Options for their particular use-case. With this, all existing and potential new use cases can be covered. For instance, it can be used for the case of [I-D.ponchon-ipsecme-anti-replay-subspaces] or [I-D.he-ipsecme-vpn-shared-ipsecsa] etc., or combinations thereof. EESP does not have a trailer as ESP had, instead the Next Header and Pad Length values are moved to the EESP header. Additionally, an optimized EESP header is defined which eliminates these 2 values when using simple IPv4 or IPv6 tunnel mode. EESP also does not define TFC padding, IP-TFS as of [RFC9347] SHOULD be used instead. A detailed discussion about the problems of the ESP protocol can be found in [I-D.mrossberg-ipsecme-multiple-sequence-counters].

EESP follows the Security Architecture for the Internet Protocol [RFC4301] and uses ESP as of [RFC4303] as reference. That means this document is seen as a modern version of ESP (with new protocol number), but it follows the design principles of ESP. Protocol parts that are not mentioned here MUST be handled exactly the way as specified in [RFC4303]. EESP neither updates nor obsoletes [RFC4303].

EESP focuses on modern algorithms, hence it defines the use of combined mode algorithms only. This means that the integrity option is always taken.

Though this document specifies IKEv2 as a negotiation protocol, EESP may use other protocols for negotiation and key derivation. The packet specification is portable to other keying protocol use cases, such as [PSP], and offers versioning at the packet level. The Crypt Offset and the Session ID can be used to cover the PSP use case.

1.1. Requirements Language

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].

1.2. Terminology

This document uses the following terms defined in IKEv2 [RFC7296]: Child SA, CREATE_CHILD_SA, IKE_AUTH exchange, USE_TRANSPORT_MODE

This document uses the following terms defined in [PSP]: PSP (a recursive acronym for PSP Security Protocol), Virtual Network Identifier (VNI), Crypt Offset.

This document uses the following terms defined in [RFC2992]: Equal-cost multi-path (ECMP)

This document uses the following terms defined in [RFC4303]: Encapsulating Security Payload (ESP).

This document uses the following terms defined in [I-D.mrossberg-ipsecme-multiple-sequence-counters]: Sub-Child SA.

This document uses the following terms defined in [RFC3948]: Non-ESP Marker.

2. Enhanced Encapsulating Security Payload Packet Format

This section defines the exact packet formats, the section is normative.

2.1. Top-Level EESP format

The (outer) protocol header (IPv4, IPv6, or Extension) that immediately precedes the EESP header SHALL contain the value TBD in its [Protocol] (IPv4) or Next Header (IPv6, Extension) field. Figure 1 illustrates the top-level format of an EESP packet. The EESP header is split into multiple parts.

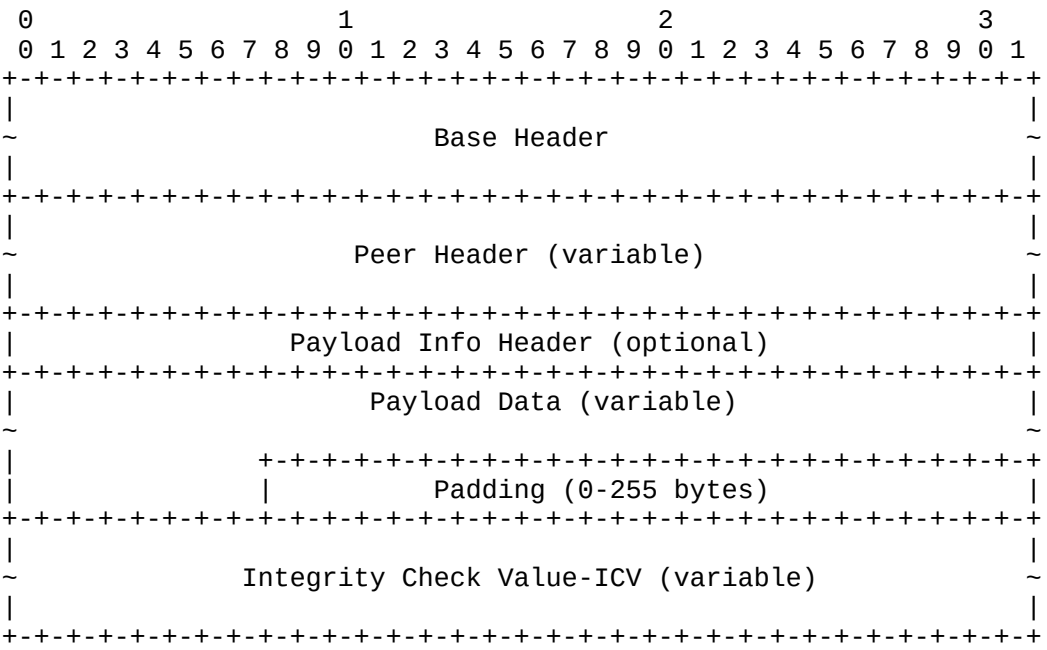


Figure 1: Top-Level Format of an EESP Packet

The packet starts with a 'Base Header' that can be used by protocol parsing engines of middleboxes such as routers or firewalls in addition to the IPsec peers that use it to route the packet to the correct cryptographic context.

The 'Peer Header' follows the 'Base Header'. The 'Peer Header' is used to support replay protection and to store cryptographic synchronization data, e.g., an Initialization Vector (IV) for the IPsec peer. The 'Peer Header' is only meaningful to the IPsec peers.

Unlike ESP, EESP does not have a trailer. Instead, these values have moved to a 'Payload Info Header' directly following the 'Peer Header'. The 'Payload Info Header' is encrypted by default, and therefore private to the IPsec peers. However, with a positive crypt offset (see Section 3.1.2), the 'Payload Info Header' might be left unencrypted. In this case, protocol parsing engines of middleboxes can act upon it (e.g., for telemetry).

The 'Payload Data' follows these 3 header parts, and has a structure that depends on the choice of encryption algorithm and mode.

'Padding' is an optional field following the 'Payload Data', primarily for alignment when using a block cipher.

Finally, the packet ends with an 'Integrity Check Value' (ICV) (see Section 4.4.2). The length of this ICV depends on the cryptographic suite.

2.2. Base Header

The 'Base Header' is comprised of a fixed base header followed by an optional 'Options' field. IPsec Peers and middleboxes MAY act upon the Base Header and any possible Options.

2.2.1. Fixed Base Header

The fixed portion of the base header is defined as follows.

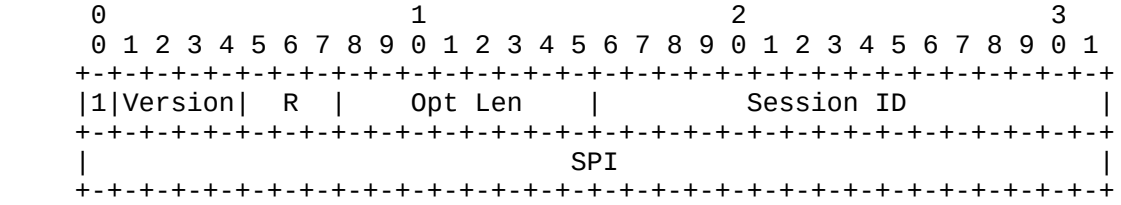


Figure 2: Fixed Base Header

ESP compatibility 1 bit : set to 1 for compatibility with ESP-in-UDP. ESP-in-UDP SAs MAY set this bit, the most significant bit of the SPI, to 0.

Version 4 bits: MUST be set to zero and checked by the receiver. If the version is different than an expected version number (e.g., negotiated via the control channel), then the packet MUST be dropped by the receiver. Future modifications to the EESP header require a new version number. In particular, the version of EESP defined in this document does not allow for any extensions. Intermediate nodes dealing with unknown versions are not necessarily able to parse the packet correctly. Intermediate treatment of such packets is policy-dependent (e.g., it may dictate dropping such packets).

Reserved (R) 3 bits: Reserved for future versions, MUST be set to zero and checked by the receiver. If the reserved bits are different to zero, the packet MUST be dropped by the receiver.

Opt Len 8 bits: Length in bytes of the 'Options' field.

Session ID 16 bits: The Session ID covers additional information

that might be used to identify the SA. For instance, it can be used to encode a Sub SA ID. The meaning of that field is opaque and MAY be negotiated by IKEv2. This document defines the use of the Session ID as a Sub SA ID. Other use cases are not covered in this document.

Security Parameter Index (SPI) 32 bits: 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.

2.2.2. Base Header Options

The base header 'Options' field is optional, its size is given in the fixed header field 'Opt Len' and may be zero if no options are present.

When present, the 'Options' field carries a variable number of type-length-value (TLV) encoded options. The format of these options has been derived from the IPv6 extension header options as defined in Section 4.2 of [RFC8200], with the following exceptions. No special meaning is attached to the top 3 bits of the option type value, and the processing order of the options is not restricted.

Option type values are allocated from one of two ranges of values. One range is used for standardized option types and the second range is reserved for private options.

This document defines 3 initial standard option types, 'Pad1 Option', 'PadN Option', and 'Crypt Offset Option'. These options are defined in section Section 3.1.

Private options use 'Option Type' values from the private option reserved range and can be used for any purposes that are out of scope for standardization. For example, they can be used to encode hardware specific information, such as used encryption/authentication algorithms as done in [PSP].

2.2.2.1. Options Field End-Alignment

When options are present, padding options (i.e., 'Pad1' and 'PadN') MUST be used to align the fields following the 'Options' field. This alignment is dictated by the packet format, see Section 2.5.

2.3. Peer Header

The ’Peer Header’ follows the ’Base Header’ and ’Options’ field. The Peer Header is private to the IPsec peers, middleboxes MUST NOT act upon the Peer Header fields. Peer Header fields are optional and MUST be negotiated by IKEv2 or any other appropriate protocol; therefore it is not parsable by middleboxes. This document defines two Peer Header fields, a ’Sequence Number’ and an ’Initialization Vector’; the format is shown below. Future documents can define additional Peer Header fields based on their needs.

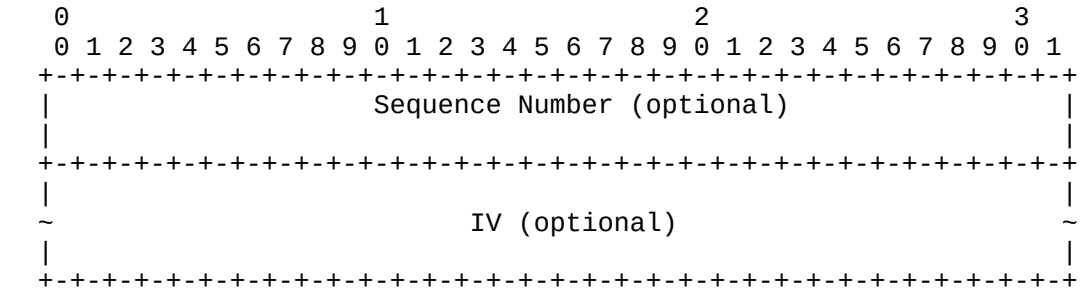


Figure 3: Peer Header

If present, the ’Sequence Number’ is a full 64bit sequence number. EESP only support 64bit sequence numbers, a.k.a ESN and transmits the entire sequence number on each packet. The actual size of the ’Initialization Vector’ depends on the choice of the cipher suite.

The ’Sequence Number’ and ’Initialization Vector’ fields are defined in the following sections.

2.3.1. Sequence Number

The sequence number field is used for replay protection. This unsigned 64-bit field contains a counter value that increases 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. The sequence number MUST increase strictly monotonically, sequence numbers MUST NOT repeat and MUST NOT cycle for any given SA. 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 2^64nd packet on an SA. If Sub SAs are used, the sender and receiver maintain a counter for each Sub SA. In this case, 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 2^64nd packet on a Sub SA. Implementations that do replay protection SHOULD increase the

sequence number by one for each sent packet. Even if recommended to increase the sequence number by one, implementations MAY employ other methods to increase the sequence number, as long as the aforementioned requirements are met. Sharing an SA among multiple senders is permitted, though generally not recommended. This document 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. However, EESP is capable to handle packet counters among multiple senders. This can be done by defining a new Base Header Option that covers a 'Sender ID'. Similar to the Session ID, this Sender ID can be used as an additional Sub SA ID (see Section 2.9). Defining such an Option is left for future documents.

Replay protection SHOULD be enabled. However, on multicast or in datacenter environments where the upper layer protocols ensure replay protection, it can be disabled. Disabling replay protection MUST be negotiated by IKEv2. In this case the sequence number field is omitted.

In contrast to ESP, where the receiver alone decides whether to disable replay protecton, it is negotiated in EESP so that sender and receiver can agree on it.

2.3.2. Initialization Vector

If the algorithm used to encrypt the payload requires cryptographic synchronization data, e.g., an Initialization Vector (IV), then this data is carried explicitly in the 'Peer Header' which is in front of the encrypted part of the packet. Any encryption algorithm that requires such explicit, per-packet synchronization data MUST indicate the length, any structure for such data, and the location of this data as part of an RFC specifying how the algorithm is used with EESP. (Typically, the IV immediately precedes the ciphertext. See Table 1) If such synchronization data is implicit, the algorithm for deriving the data MUST be part of the algorithm definition RFC. (If included, cryptographic synchronization data, e.g., an Initialization Vector (IV), usually is not encrypted per se (see Table 1), although it sometimes is referred to as being part of the ciphertext.)

Counter mode algorithms MAY use the 64-bit counter as the Initialization Vector (IV) in the Sequence number Field, as specified [RFC8750]. This option, Implicit Initialization Vector (IIV) saves the size of IV on each packet. Whether or not this option is selected is determined as part of Security Association (SA) establishment.

2.4. Payload Info Header

The Payload Info Header is needed if the contained payload is not a single IPv4 or IPv6 packet (e.g., when using Transport Mode, BEET Mode [RFC7402],

or IP-TFS [RFC9347]). It is not needed on tunnel mode because this information can be derived from the inner IPv4 or IPv6 header. This document specifies a full and an optimized packet format. The Payload Info Header is present in the full packet format, but not in the optimized packet format (see Section 2.8). The packet format depends on the used Mode. The full packet format MUST be used on all Modes that can't derive the 'Next Header' and 'Pad Length' from the following inner header. Modes that can derive these information MUST use the optimized packet format. IPsec peers and middleboxes (if Crypt Offset is positive, see Section 3.1.2) MAY act upon the Payload Info Header.

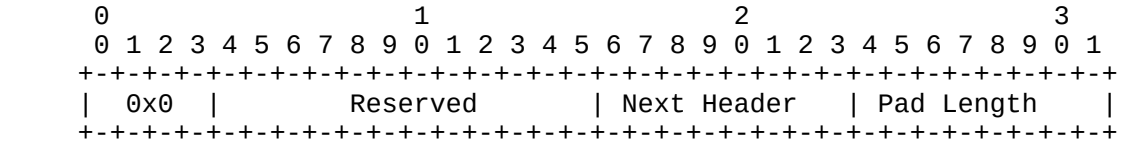


Figure 4: Payload Info Header

2.4.1. Next Header

The Next Header is an 8-bit field that identifies the type of data contained in the Payload Data field, e.g., a next layer header and data. The value of this field is chosen from the set of IP Protocol Numbers defined on the web page of the IANA (e.g., a value of 6 indicates TCP and a value of 17 indicates UDP).

2.4.2. Pad Length

The Pad Length field indicates the number of pad bytes immediately following the payload data and is used to align the ICV field. The range of valid values is 0 to 255, where a value of zero indicates that no Padding bytes are present.

2.5. Payload Data

Payload Data is a variable-length field containing data from the original IP packet. The Payload Data field is mandatory and is an integral number of bytes in length.

Note that the beginning of the next layer protocol header MUST be aligned relative to the beginning of the EESP header as follows. For IPv4, this alignment is a multiple of 4 bytes. For IPv6, the alignment is a multiple of 8 bytes.

2.6. Padding (for Encryption)

Two primary factors require or motivate use of the Padding field.

* If an encryption algorithm is employed that requires the plaintext to be a multiple of some number of bytes, e.g., the block size of a block cipher, the Padding field is used to fill the plaintext (consisting of the Payload Data, Padding, and Payload Info Header) to the size required by the algorithm.

* Padding also may be required, irrespective of encryption algorithm requirements, to ensure that the resulting ciphertext terminates on a 4-byte boundary to make sure the ICV is properly aligned.

The sender MAY add 0 to 255 bytes of padding. Inclusion of the Padding field in an EESP packet is optional, subject to the requirements noted above, but all implementations MUST support generation and consumption of padding.

If Padding bytes are needed but the algorithm does not specify the padding contents, then the Padding bytes MUST be set to 0.

If an algorithm imposes constraints on the values of the bytes used for padding, they MUST be specified by the RFC defining how the algorithm is employed with EESP. If the algorithm requires checking of the values of the bytes used for padding, this too MUST be specified in that RFC.

2.7. Integrity Check Value (ICV)

The Integrity Check Value is a variable-length field computed over the Encrypted Payload and Additional Authenticated Data, as defined in [ADD Construction]. The length of the field is specified by the algorithm selected and associated with the SA. The algorithm specification MUST specify the length of the ICV and the comparison rules and processing steps for validation.

2.8. Full and Optimized Packet Formats

The two possible packet formats are described in this section. The packet format containing the 'Payload Info Header' is called the "Full EESP packet format", while the packet format without the 'Payload Info Header' is the called the "Optimized EESP packet format". When IPv4 or IPv6 tunnel mode is used, the optimized packet format MUST be used, ommiting the 'Payload Info Header'. In this optimized mode the payload will always start with an IPv4 or IPv6 header. IPv4 or IPv6 packets always start with a Version field at the first nibble, so it is possible to identify IPv4 and IPv6 by reading the first nibble of the inner packet, and there is no need for a next header field. Additionally, IPv4 and IPv6 also have a field describing the overall size of the inner packet, so a pad length field is also not needed as it can be derived. When transport mode, BEET mode, or IP-TFS is used, it is not possible to derive this information, so the full packet format, including the 'Payload Info Header' MUST be used.

The selection of the packet format is determined by the encapsulation mode and by whether the values carried in the 'Payload Info Header' can be inferred from the decrypted payload. Table Table 1 summarizes which modes use which format. In this table, "Full" means that the 'Payload Info Header' is present, and "Optimized" means that the 'Payload Info Header' is absent (elided).

Mode	Packet format	Notes
IPv4/IPv6 tunnel mode	Optimized	Inner starts with IPv4/IPv6 header.
Transport mode	Full	Next Header/Pad Length cannot be inferred.
BEET mode	Full	Next Header/Pad Length cannot be inferred.
IP-TFS	Full	Payload starts with the IP-TFS header.

Table 1: EESP packet format selection by mode

The two packet formats are shown below. Figure 5 shows the full packet format used as the default for modes of operation. Figure 6 illustrates the resulting optimized packet format for use with IPv4 or IPv6 Tunnel Mode when the 'Payload Info Header' is elided.

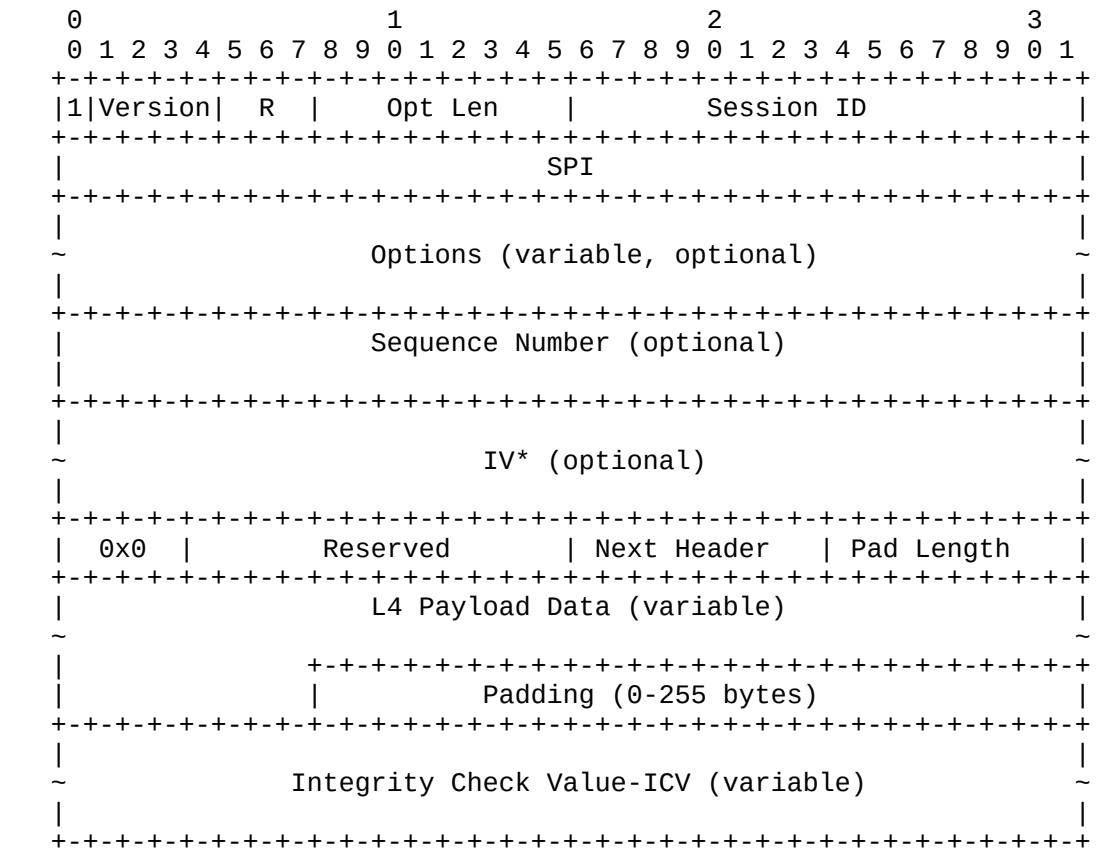


Figure 5: Full EESP packet format

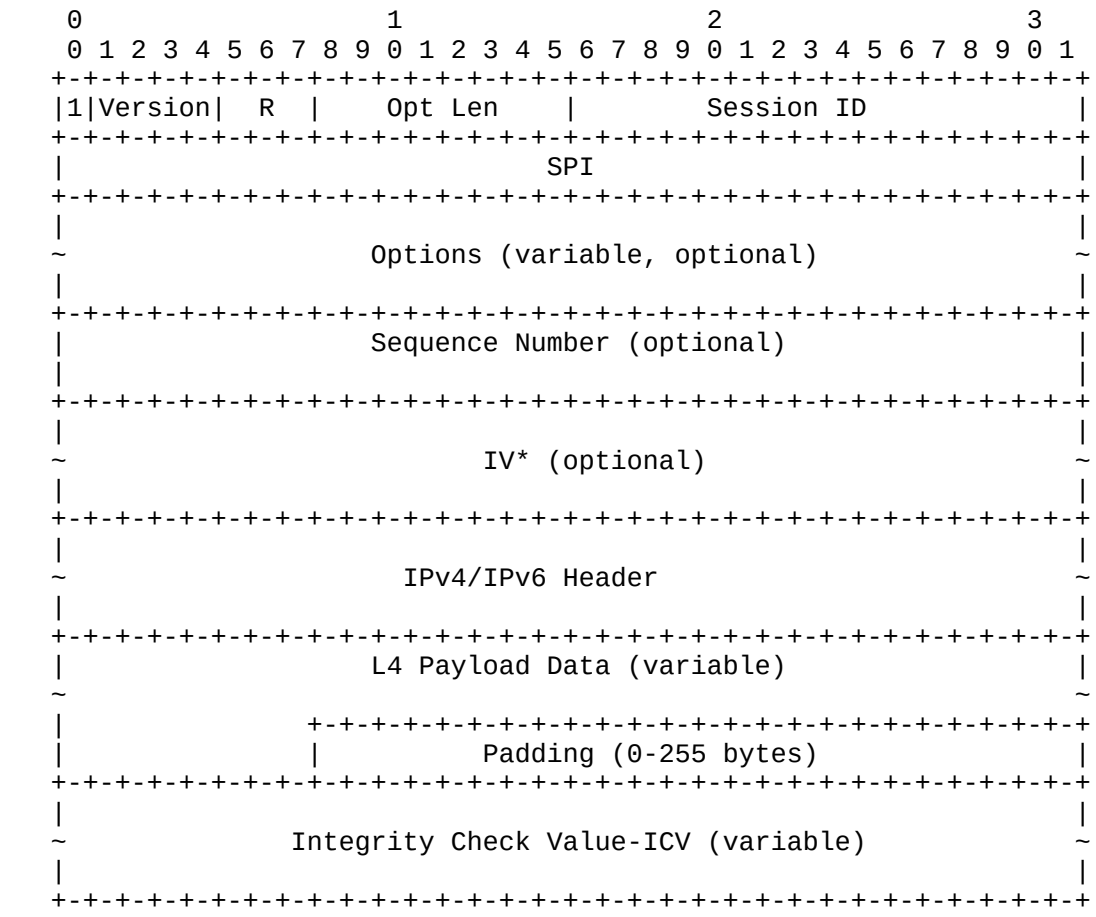


Figure 6: Optimized EESP packet format

[*] If included, cryptographic synchronization data, e.g., an 'Initialization Vector' (IV), usually is not encrypted per se, although it often is referred to as being part of the cipher-text. Unlike ESP, the IV is not considered to be a part of the payload data in EESP.

The explicit IV shown in Table 2 depends on the used algorithm and may be omitted. Because algorithms, modes and options are fixed when an SA is established, the detailed format of EESP packets for a given SA (including the 'Payload Data' substructure) is fixed for all traffic on the SA.

The table below refers to the fields in the preceding figures and illustrate how several categories of algorithmic options, each with a different processing model, affect the fields noted above. The processing details are described in later sections.

Field	# of bytes	Req'd [1]	Encrypt Covers	Integ Covers	Tx'd
Base Header	8	M		Y	plain
Options	variable	O		Y	plain
Sequence Number	8	O		Y	plain
IV	variable	O		Y	plain
Payload Info Hdr[4]	4	O	Y	Y	cipher [3]
Payload [2]	variable	M	Y	Y	cipher [3]
Padding	0-255	M	Y	Y	cipher [3]
ICV	variable	M			plain

Table 2: High level layout for fields of an EESP packet

* [1] M = mandatory; O = optional
* [2] If tunnel mode -> IP datagram. If BEET mode -> IP datagram. If transport mode -> next header and data. If IP-TFS, IP-TFS header and payload.
* [3] Ciphertext if encryption has been selected.
* [4] Not present in the Optimized Packet Format, otherwise mandatory.

In the table "optional" means that the field is omitted if the option is not selected, i.e., it is not present in the packet as transmitted or as formatted for computation of an ICV. Whether or not an option is selected depends on the used mode (Payload Info Header), or is determined as part of Security Association (SA) establishment. Thus, the format of EESP packets for a given SA is fixed for the duration of the SA. In contrast, "mandatory" fields are always present in the EESP packet format for all SAs.

2.9. Session ID as Sub SA ID

This section specifies the use of the Session ID as a Sub SA ID. The use of the Session ID as a Sub SA ID MUST be negotiated by IKEv2, or any other suitable protocol. In this case, Session ID is used as a 16 bits Sub SA ID. Sub SA IDs were initially defined in [I-D.ponchon-ipsecme-anti-replay-subspaces] and called 'Replay Subspaces' there.

Each number of the 16 bits Sub SA ID encodes a single 64 bit anti-replay sequence number space. This means that each core, path, or QoS class, or any combination of those, can then use their own unique anti-replay sequence number subspace. Each anti-replay sequence number subspace uses Sequence Numbers as specified in section Section 2.3.1.

To make sure that at most 2^64 sequence numbers are used for a given key, a KDF MUST be used used to derive a separate key for each anti-replay sequence number subspace (see Section 5). In this case, the full 64 bits of each anti-replay sequence number subspace can be used.

Sub SAs can be created "on the fly" within the IPsec data-plane. Sub SAs streamline traffic flow management, reduce overhead, and enable more efficient lifecycle operations.

A pair of EESP SAs combined with multiple unidirectional Sub SAs provides a more flexible approach to carrying asymmetric traffic patterns, particularly in high-speed environments. Sub SAs reduces overhead, improves resource utilization, and enhances scalability for large-scale deployments. In many use cases, several unidirectional SAs used, while others are unused which can result in unnecessary overhead for SA management, rekeying, and resource consumption. Furthermore, using multiple bidirectional Child SAs for granular traffic flows often leads to additional setup delays and complex lifetime management. This inefficiency is particularly acute in high-throughput or low-latency environments, where rapid setup and teardown of SAs is essential to maintain performance.

Each Sub SA is identified by a Sub SA ID, which MUST be carried in each EESP packet in the Session ID field—consistent with the negotiation of the EESP Child SA. This Sub SA ID is used to derive a unique key.

Particularly implementations with hardware offload, MAY derive Sub SA keys dynamically on a per-packet basis. This mitigates the risk of data-plane performance degradation caused by a large number of keys.

AEAD transforms such as AES-GCM [RFC4106], [RFC8750] requires that the IV never repeat within a single Sub SA. Because each Sub SA uses a distinct key, the IV MAY be reused across different Sub SAs, satisfying the requirement that each key be paired with a unique IV. Implementations MUST also maintain an independent sequence number space for each Sub SA when full 64-bit sequence numbers are in use. For a given Sub SA key, sequence numbers MUST remain unique and monotonically increasing to meet cryptographic requirements.

2.9.1. Sender Behavior

This section defines the IPsec sender’s behavior when transmitting packets using an IPsec Child SA that has been previously configured or negotiated with IKEv2 to use at most N different sequence number subspace IDs.

The sender MAY set the sequence number subspace ID to any value between 0 and N-1. How the different subspace IDs are used is up to the implementation, but as an example, the sender could use different subspace ID values per path or per processing core (or combination of both).

The sender MUST NOT use any subspace ID values greater or equal to N (since the IPsec Child SA has been configured to use at most N different values). This requirement was introduced to improve the implementation performance, as opposed to allowing the sender to use arbitrary subspace ID values.

The sender MUST maintain one sequence number counter per sequence number subspace that it makes use of. But the sender MAY use only some (and as few as a single one) of the available N subspace ID values between 0 and N-1.

When transmitting a packet, the sender MUST use the sequence number counter associated with the sequence number subspace in use for that packet.

2.9.2. Receiver Behavior

This section defines the IPsec receiver’s behavior when receiving packets using an IPsec SA that has been previously configured or negotiated to use at most N different sequence number subspace IDs.

The receiver MUST maintain one anti-replay window and counter for each sequence number subspace being used.

When receiving a packet, the receiver MUST use the anti-replay window and counter associated with the sequence number subspace identified with the subspace ID field.

The receiver MUST drop any packet received with a subspace ID value greater or equal to N. Receiving such packets is an auditable event. The audit log entry for this event SHOULD include the SPI value, subspace ID value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.

Note: Since the sender may decide to only use a subset of the available N subspace values, the receiver MAY reactively allocate an anti-replay window when receiving the first packet for a given subspace. When doing so, the receiver SHOULD first check the authenticity of the packet before allocating the new anti-replay window.

3. EESP Header Options

The EESP header 'Options' field carries a variable number of type-length-value (TLV) encoded "options" of the following format:

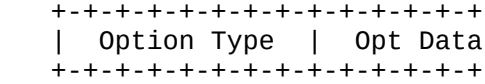


Figure 7: EESP Header Option Format

Option Type 8-bit identifier of the type of option.

Opt Data Len 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

Option Data Variable-length field. Option-Type-specific data.

The overall length of all Options is limited to 255 bytes by the OptLen field in the 'Base Header'.

3.1. EESP Option Types

This document defines two padding options 'Pad1' and 'PadN', and a 'Crypt Offset Option'. Future documents can define additional options. Appendix A of [RFC8200] contains applicable formatting guidelines for designing new options.

3.1.1. Padding Options

Individual options may have specific alignment requirements, to ensure that multi-octet values within Option Data fields fall on natural boundaries. The alignment requirement of an option is specified using the notation xn+y, meaning the 'Option Type' must appear at an integer multiple of x octets from the start of the 'Options' field, plus y octets. For example:

* 2n means any 2-octet offset from the start of the 'Options' field.

* 8n+2 means any 8-octet offset from the start of the 'Options' field, plus 2 octets.

Unless otherwise specified EESP options have no alignment requirements.

There are two padding options which are used when necessary to align subsequent options and to pad out the containing options field. These padding options must be recognized by all implementations:

3.1.1.1. Pad1 option

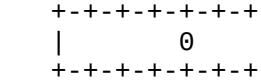


Figure 8: Pad1 Option

Note: the format of the Pad1 option is a special case -- it does not have length and value fields.

The 'Pad1' option is used to insert one octet of padding into the Options field. If more than one octet of padding is required, the 'PadN' option, described next, should be used, rather than multiple 'Pad1' options.

3.1.1.2. PadN option

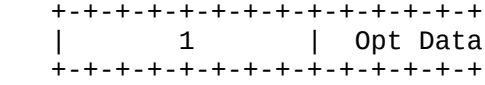


Figure 9: PadN Option

The 'PadN' option is used to insert two or more octets of padding into the 'Options' field. For N octets of padding, the Opt Data Len field contains the value N-2, and the 'Option Data' consists of N-2 zero-valued octets.

3.1.2. EESP Crypt Offset Option

This option is typically used within one Datacenter use case such as [PSP]. When enabled, full packet format with Payload Info Header MUST be used; for the intermediate router to have Next Header.

The Crypt Offset can vary on a per packet basis. The maximum allowed Crypt Offset MUST be negotiated by IKEv2 or any other appropriate protocol. Packets with a Crypt Offset greater than the negotiated maximum MUST be dropped by the receiver. The receiver SHOULD cryptographically process such packets anyway. The action in case of a correct ICV value depends on local policy. However, it is recommended to tear down the connection as it can't be considered as secure anymore.

Receiving such packets is an auditable event. The audit log entry for this event SHOULD include the SPI value, subspace ID value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.

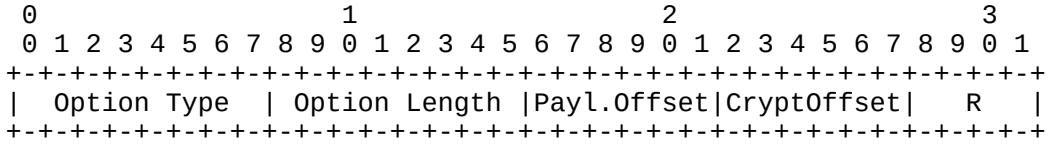


Figure 10: Crypt Offset Option

Option Type 8 bits: See Section 3

Option Length 8 bits: See Section 3

Payload Offset 6 bits: The offset from the start of the fixed header to the start of the payload header (or the payload for optimized packet format) measured in 4-octet units.

CryptOffset 6 bits: The offset from the start of the payload header (or the payload for optimized packet format) to the start of the encrypted portion of the packet, measured in 4-octet units. The resulting value MUST NOT be larger than the size of the inner packet.

R[eserved] 4-bits: Reserved MUST be sent 0 and ignored on receipt.

4. Enhanced Encapsulating Security Protocol Processing

4.1. EESP Header Location

EESP may be employed in multiple ways. To secure end-to-end network traffic, transport mode may be used. For the VPN use case, tunnel and BEET mode may be employed.

4.1.1. Layer 4 Encapsulation Modes

Layer 4 Encapsulation Modes are transport mode and BEET mode [RFC7402] Layer 4 Encapsulation Modes distinguish from tunnel mode on the position of the EESP header in the packet. On Layer 4 Encapsulation Modes the EESP header is inserted between the original IPv4/IPv6 header and the following Layer 4 header. Layer 4 Encapsulation Modes MUST use the Full Packet Format. In contrast to this, in tunnel mode the full ipv4/IPv6 datagram is encapsulated. This means the the EESP header is placed in front of the original IPv4/IPv6 datagram and a new ’outer IPv4/IPv6 header’ is added in front of the EESP header. The following sections illustrate the positioning of the EESP header.

Note that in Layer 4 Encapsulation Modes, 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.

4.1.1.1. Transport Mode Processing

In transport mode, EESP is inserted after the IP header and before a next layer protocol, e.g., TCP, UDP, ICMP, etc. In the context of IPv4, this translates to placing EESP after the IP header (and any options that it contains), but before the next layer protocol. (If AH is also applied to a packet, it is applied to the EESP header, Payload and ICV, if present.) (Note that the term "transport" mode should not be misconstrued as restricting its use to TCP and UDP.) The following diagram illustrates EESP transport mode positioning for a typical IPv4 packet, on a "before and after" basis. (This and subsequent diagrams in this section show the ICV field, the presence of which is a function of the security services and the algorithm/ mode selected.)

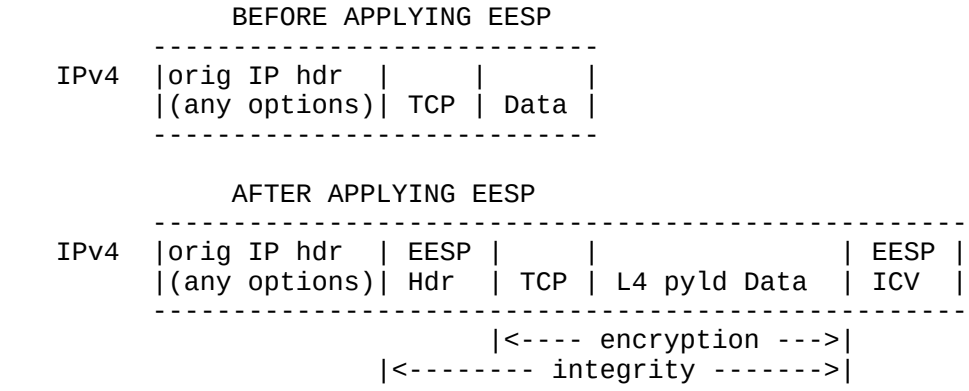
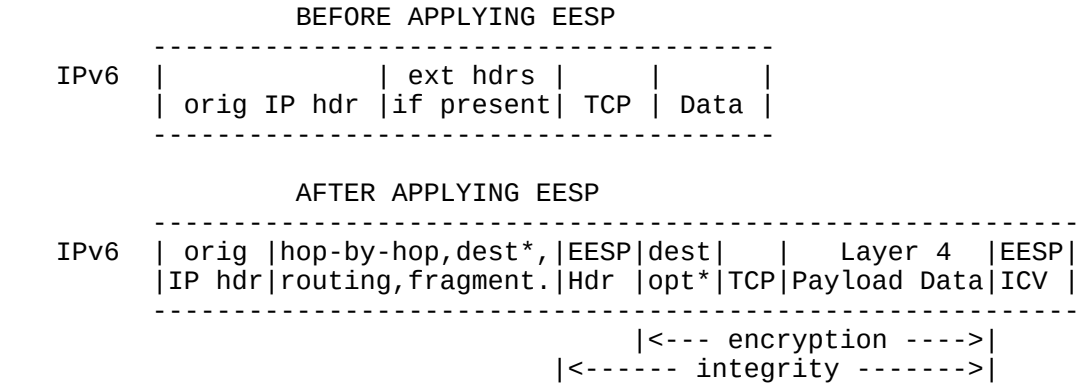


Figure 11: IPv4 Transport Mode

In the IPv6 context, EESP is viewed as an end-to-end payload, and thus should appear after hop-by-hop, routing, and fragmentation extension headers. Destination options extension header(s) could appear before, after, or both before and after the EESP header depending on the semantics desired. However, because EESP protects only fields after the EESP header, it generally will be desirable to place the destination options header(s) after the EESP header. The following diagram illustrates EESP transport mode positioning for a typical IPv6 packet.



* = if present, could be before EESP, after EESP, or both

Figure 12: IPv6 Transport Mode

4.1.1.2. BEET Mode Processing

In BEET mode, EESP is inserted exactly at the same position as it is done for transport mode. The original IP or IPv6 header is replaced by a new one. The new header SHOULD be negotiated by IKEv2 or any other suitable protocol.

In BEET mode, the EESP header is inserted between the (new) IP header and the next layer protocol header, as shown below, and the Full Packet Format MUST be used. The original IP header is replaced by a new one as specified in [RFC7402]. Any information required to reconstruct the original header is protected by EESP and processed according to [RFC7402].

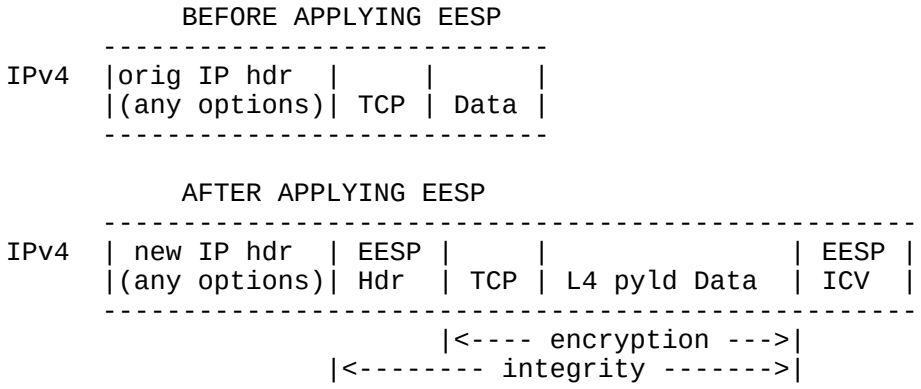
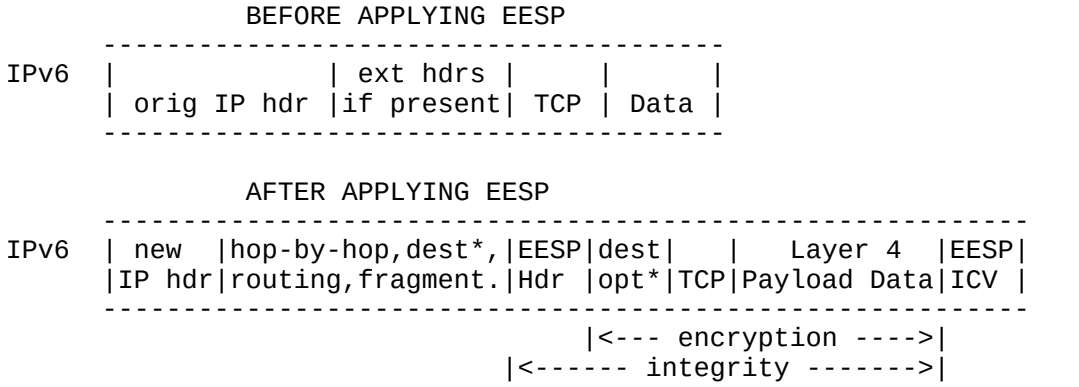


Figure 13: IPv4 BEET Mode



* = if present, could be before EESP, after EESP, or both

Figure 14: IPv6 BEET Mode

4.1.2. Tunnel Mode Processing

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, EESP protects the entire inner IP packet, including the entire inner IP header. The position of EESP in tunnel mode, relative to the outer IP header, is the same as for EESP in transport mode. In tunnel mode, the Optimized Packet Format MUST be used. The following diagram illustrates EESP tunnel mode positioning for typical IPv4 and IPv6 packets.

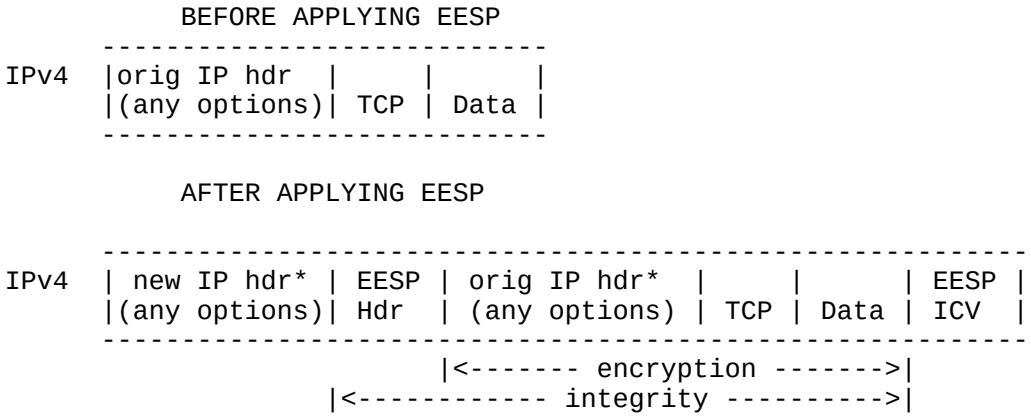


Figure 15: IPv4 Tunnel Mode

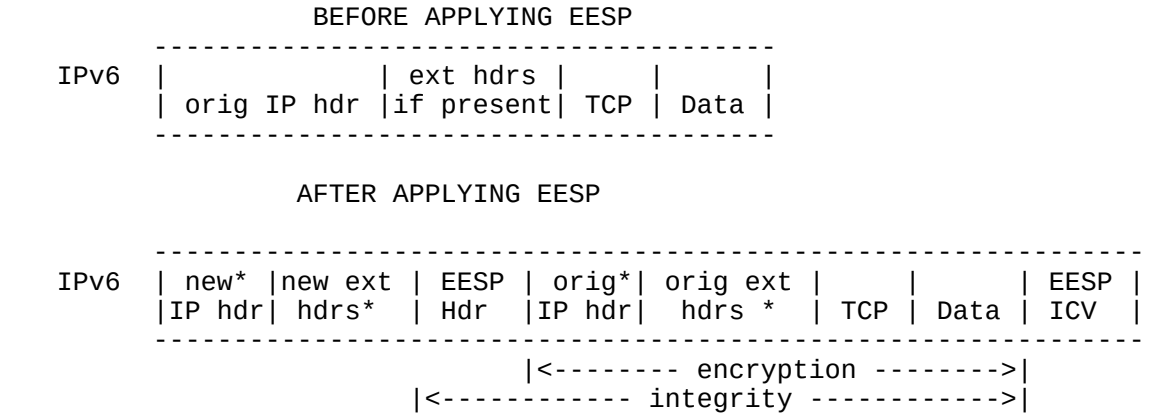


Figure 16: IPv6 Tunnel Mode

4.2. AAD Construction

Additional Authenticated Data (AAD) includes the Base Header, any Optional Headers and Peer Header.

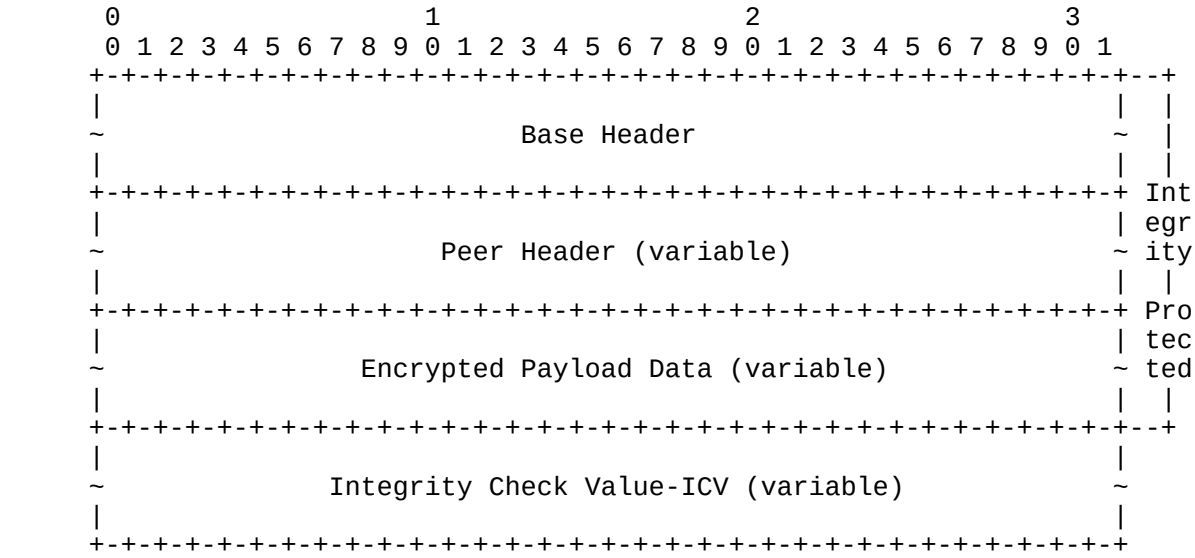


Figure 17: EESP AAD

Additionally, if a Crypt Offset is used, the AAD includes the associated data exposed due to the offset. Payload Data covered by the Crypt Offset is transmitted in the clear, but is still included in the AAD.

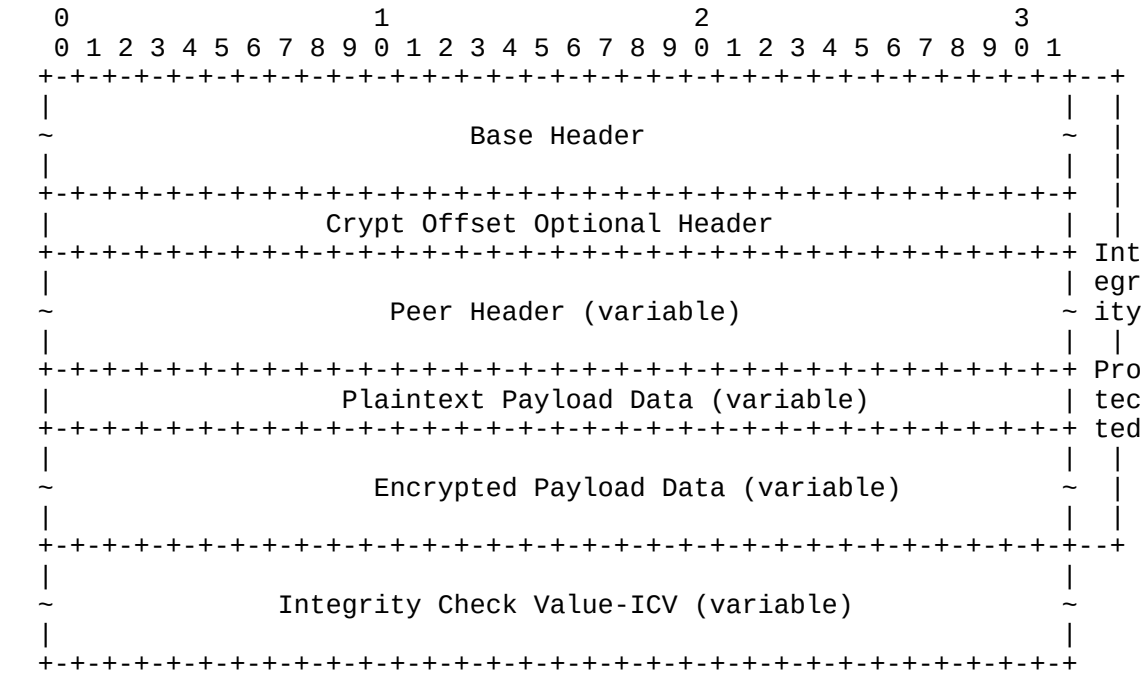


Figure 18: EESP Tunnel Mode AAD with Crypt Offset

As an example consider a Tunnel mode SA, with replay protection enabled and 8 bytes explicit IV carrying an IPv4 UDP packet with Crypt Offset 8 (8x4 = 32 bytes). Figure 19

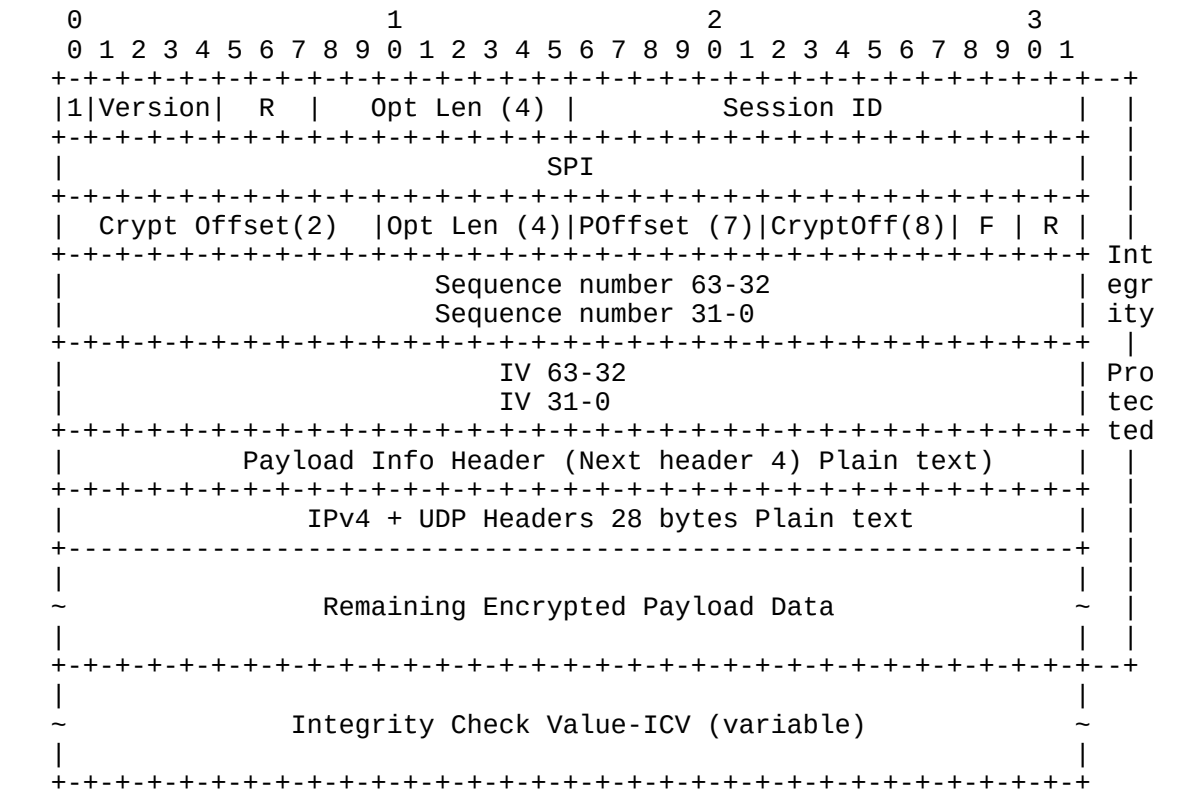


Figure 19: EESP Tunnel Mode AAD with Crypt Offset example

The AAD specifications apply to all EESP cipher suites used with EESP. This document updates [RFC4106] to define EESP-specific handling of Additional Authenticated Data (AAD) when using AES-GCM. For AES-GMAC [RFC4543], the AAD includes all headers, i.e. the entire EESP payload except the Integrity Check Value (ICV). This document also updates AAD processing for the ENCR_CHACHA20_POLY1305 cipher suite, as specified in [RFC7634].

4.3. Algorithms

EESP version 0 specifies combined mode algorithms only. Separate confidentiality and integrity algorithms MUST NOT be used with version 0 of EESP. This means that both, confidentiality and integrity services are provided always. Although not specified here, EESP can support separate confidentiality and integrity algorithms. In case using separate confidentiality and integrity algorithms becomes necessary, a new version of EESP MUST be defined.

The mandatory-to-implement algorithms for use with EESP described in a separate RFC, for e.g. RFC8221bis or another I.D., to facilitate updating the algorithm requirements independently from the protocol per se. Additional algorithms, beyond those mandated for EESP, MAY be supported.

4.3.1. Combined Mode Algorithms

The combined mode algorithm employed to protect an EESP packet is specified by the SA via which the packet is transmitted/received. Because IP packets may arrive out of order, and not all packets may arrive (packet loss), each packet must carry any data required to allow the receiver to establish cryptographic synchronization for decryption. This data may be carried explicitly, e.g., as an IV (as described above), or the data may be derived from the plaintext portions of the (outer IP or EESP) packet header. (Note that if plaintext header information is used to derive an IV, that information may become security critical and thus the protection boundary associated with the encryption process may grow.

For example, if one were to use the EESP Sequence Number to derive an IV, the Sequence Number generation logic (hardware or software) would have to be evaluated as part of the encryption algorithm implementation. In the case of FIPS 140-2 [NIST01], this could significantly extend the scope of a cryptographic module evaluation.)

Because EESP makes provision for padding of the plaintext, combined mode algorithms employed with EESP may exhibit either block or stream mode characteristics. The means by which a combined mode algorithm provides integrity for the payload, and for the header fields, may vary for different algorithm choices. In order to provide a uniform, algorithm-independent approach to invocation of combined mode algorithms, no payload substructure is defined.

To allow an EESP implementation to determine the MTU impact of a combined mode algorithm, the RFC for each algorithm used with EESP must specify a (simple) formula that yields encrypted payload size, as a function of the plaintext payload and EESP header sizes.

4.4. Outbound Packet Processing

In Layer 4 Encapsulation Modes, the sender encapsulates the next layer protocol information behind the EESP header fields, and retains the specified IP header (and any IP extension headers in the IPv6 context). 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.

4.4.1. Security Association Lookup

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

4.4.2. Packet Encryption and Integrity Check Value (ICV) Calculation

In this section, we speak in terms of encryption always being applied because of the formatting implications. This is done with the understanding that "no confidentiality" is offered, for instance, by using the AES-CMAC algorithm ([RFC4494]).

4.4.3. Combined Confidentiality and Integrity Algorithms

The Sender proceeds for combined confidentiality/integrity algorithm as follows:

1. Encapsulate into the EESP Payload Data field:

    * for transport and BEET mode -- just the original next layer protocol information.

    * for tunnel mode -- the entire original IP datagram.

2. Add any necessary (encryption) Padding.

3. Encrypt and integrity protect the result using the key and combined mode algorithm specified for the SA and using any required cryptographic synchronization data.

    * If explicit cryptographic synchronization data, e.g., an IV, is indicated, it is input to the combined mode algorithm per the algorithm specification and placed in the IV field of the peer header.

    * If implicit cryptographic synchronization data is employed, it is constructed and input to the encryption algorithm as per the algorithm specification.

    * The EESP header fields are inputs to the algorithm, as they must be included in the integrity check computation. The means by which these values are included in this computation are a function of the combined mode algorithm employed.

* The (explicit) ICV field MAY be a part of the EESP packet format. If one is not used, an analogous field usually will be a part of the ciphertext payload. The location of any integrity fields, and the means by which the EESP header fields are included in the integrity computation, are defined in Section 4.2.

4.4.4. Sequence Number Generation

Replay protection is negotiated by the IPsec peers. If a SA chooses to do replay protection, the sequence numbers are generated in the following way.

The sender’s counter SHOULD be initialized to 0 when an SA is established. The sender increments the sequence number counter for this SA and inserts this value into the Sequence Number field of the Peer Header. Note that 0 is not a valid sequence number. Thus, the minimal sequence number to use for the first packet sent using given SA 1. This means that the first packet sent using given SA will contain a sequence number of 1, or bigger. The most natural method to increase the sequence number is to increase only by one for each sent packet. This method SHOULD be implemented when possible. However, peers MAY choose different replay protection algorithms, i.e. not by using sequence numbers that are incremented by one for each packet. In case the peers choose such an algorithm, the sender MUST ensure that the sequence number is strictly monotonic increasing.

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 such 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, Session ID value, current date/time, Source Address, Destination Address, and (in IPv6) the cleartext Flow ID.

Typical behavior of an EESP implementation calls for the sender to establish a new SA when the Sequence Number of the SA cycles, or if sequence number subspaces are used any one of the subspaces cycles, or in anticipation of this values cycling.

If the key used to compute an ICV is manually distributed, a compliant implementation SHOULD NOT provide anti-replay service. If a user chooses to employ anti-replay in conjunction with SAs that are manually keyed, the sequence number counter at the sender MUST be correctly maintained across local reboots, etc., until the key is replaced.

4.4.5. Fragmentation

If necessary, fragmentation is performed after EESP processing within an IPsec implementation. Thus, transport and BEET mode, EESP is applied only to whole IP datagrams (not to IP fragments). An IP packet to which EESP has been applied may itself be fragmented by routers en route, and such fragments must be reassembled prior to EESP processing at a receiver. In tunnel mode, EESP is applied to an IP packet, which may be a fragment of an IP datagram. For example, a security gateway or a "bump-in-the-stack" or "bump-in-the-wire" IPsec implementation (as defined in the Security Architecture document) may apply tunnel mode EESP to such fragments.

NOTE: For Layer 4 Encapsulation Modes -- As mentioned at the end of Section 4.1.1, 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 EESP receiver to accept fragments for reassembly creates denial of service vulnerabilities. Thus, an EESP 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 EESP 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.

4.5. Inbound Packet Processing

4.5.1. Reassembly

If required, reassembly is performed prior to EESP processing. If a packet offered to EESP for processing appears to be an IP fragment, i.e., the OFFSET field is non-zero 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, Session ID value, date/time received, Source Address, Destination Address, Sequence Number, 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.

4.5.2. Security Association Lookup

Upon receipt of a packet containing an EESP 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 2.1. 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 2.1. (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 is present, and whether the (explicit) ICV field should be present (and if so, its size). Also, the SAD entry will specify the algorithms and keys to be employed for decryption and ICV computation (if applicable).

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, Session ID value, date/time received, Source Address, Destination Address, Sequence Number, and (in IPv6) the cleartext Flow ID.

4.5.3. Sequence Number Verification

All EESP implementations MUST support the anti-replay service, though its use may be enabled or disabled by negotiation 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 anti-replay service is enabled for this SA, the receive packet counter for each used Sub SA MUST be initialized to zero when the SA is established. For each received packet on a Sub SA, the receiver MUST verify that the packet contains a Sequence Number that does not duplicate the Sequence Number of any other packets received on this Sub SA during the life of the SA. This SHOULD be the first ESP check applied to a packet after it has been matched to an SA, to speed rejection of duplicate packets.

EESP permits two-stage verification of packet sequence numbers. This capability is important whenever an ESP implementation (typically the cryptographic module portion thereof) is not capable of performing decryption and/or integrity checking at the same rate as the interface(s) to unprotected networks. If the implementation is capable of such "line rate" operation, then it is not necessary to perform the preliminary verification stage described below.

The preliminary Sequence Number check is effected utilizing the Sequence Number value in the EESP Header and is performed prior to integrity checking and decryption. If this preliminary check fails,

the packet is discarded, thus avoiding the need for any cryptographic operations by the receiver. If the preliminary check is successful, the receiver cannot yet modify its local counter, because the integrity of the Sequence Number has not been verified at this point.

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 Sub 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 received packet falls within the window and is not a duplicate, or if the packet is to the right of the window, receiver proceeds with cryptographic processing, i.e. integrity check along with decryption. If the integrity check 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, Session ID value, date/time received, Source Address, Destination Address, the Sequence Number, and (in IPv6) the Flow ID. The receive window is updated only if the integrity verification succeeds. (If a combined mode algorithm is being used, then the integrity protected Sequence Number must also match the Sequence Number used for anti-replay protection.)

A minimum window size of 64 packets MUST be supported. 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-terabit/second) devices are not specified by this standard.

4.5.4. Packet Decryption and Integrity Check Value Verification

4.5.4.1. Combined Confidentiality and Integrity Algorithms

The receiver proceeds for combined mode algorithms as follows:

1. Decrypts and integrity checks the EESP Payload Info Header (if present), Payload Data, Padding, using the key, algorithm, algorithm mode, and cryptographic synchronization data (if any), indicated by the SA. The Base Header and the Peer Header are inputs to this algorithm, as they are required for the integrity check.

    * If explicit cryptographic synchronization data, e.g., an IV, is indicated, it is taken from the IV field and input to the decryption algorithm as per the algorithm specification.

    * If implicit cryptographic synchronization data, e.g., an IV, is indicated, a local version of the IV is constructed and input to the decryption algorithm as per the algorithm specification.

2. If the integrity check performed by the combined mode algorithm fails, the receiver MUST discard the received IP datagram as invalid; this is an auditable event. The log data SHOULD include the SPI value, Session ID value, date/time received, Source Address, Destination Address, the Sequence Number, and (in IPv6) the cleartext Flow ID.

3. Process any Padding as specified in the encryption algorithm specification, if the algorithm has not already done so.

4. The receiver checks the Next Header field. If the value is "59" (no next header), the (dummy) packet is discarded without further processing.

5. Extract the original IP datagram (tunnel mode) or transport-layer frame (layer 4 payload encapsulation modes) from the EESP Payload Data field.

The exact steps for reconstructing the original datagram depend on the mode. Transport and Tunnel mode are described in the Security Architecture document [RFC4301]. BEET Mode is described in [RFC7402] and IP-TFS in [RFC9347]. At a minimum, in an IPv6 context, the receiver SHOULD ensure that the decrypted data is 8-byte aligned, to facilitate processing by the protocol identified in the Next Header field.

5. Key Derivation for Sub SAs

When an EESP SA is using Sub SAs, each Sub SA (including the one with Session ID 0) uses separate keys. This allows each Sub SA to use its own independent Sequence Number and IV space.

In order to derive these keys, a Sub SA Key Derivation Function (SSKDF) MUST be configured as a property of the EESP SA if Sub SAs are to be used. If no SSKDF is configured, Sub SAs can’t be used.

If an SSKDF is set, the key material required for the EESP SA is determined by the key size of the negotiated SSKDF. This single key is called ’root’ key and is the basis for the keys derived for all Sub SAs.

The EESP SA root key and selected SSKDF are then used as follows to derive key material for each Sub SA:

KEYMAT_sub = SSKDF(KEY_root, Session ID, L)

Where L is the total length of the key material KEYMAT_sub and the salt value is the full Session ID field of the Sub SA. The length of KEYMAT_sub and how it is used depends on the negotiated encryption algorithm.

Keys for Sub SAs may be derived immediately or on demand when the first packet is processed. Memory constrained implementations may even decide to derive the Sub SA keys on the fly for each received packet as only the EESP key has to be stored to derive the keys of all Sub SAs.

Because individual Sub SAs can’t be rekeyed, the complete EESP SA MUST be rekeyed when either a cryptographic limit or a time-based limit is reached for any individual Sub SA.

6. UDP Encapsulation

UDP encapsulation for EESP is largely the same as UDP encapsulation for ESP specified in [RFC3948]. The primary difference is that the UDP source port used by EESP Sub SAs may be different from the IKE SA source port. This allows more flexible handling of EESP traffic, particularly ECMP support along the path and in the NIC.

A receiver intending to support both ESP and EESP encapsulated in UDP must be able to distinguish inbound ESP and EESP traffic on the same UDP port. To be able to handle this, the SPIs for the incoming ESP SAs MUST be chosen in such a way, that they can be distinguished from the EESP base header. Since the most significant bit of the EESP base header is fixed to be one, this can be achieved if ESP SPIs are selected in such a way, that the most significant bit of the ESP SPIs is always set to zero.

6.1. UDP Encapsulation of Sub SAs

An EESP SA primarily uses UDP encapsulation to facilitate NAT traversal. However, an additional use case for UDP encapsulation is to introduce source port entropy, which supports ECMP or/and RSS (Receive Side Scaling) mechanisms. In such scenarios, the initiator MAY also use a distinct, ephemeral source port for Sub SA IDs greater than zero.

It is important to note that IKE messages MUST NOT utilize these ephemeral source ports. Instead, IKE traffic should be confined to the source and destination ports to ensure proper protocol operation and maintain compatibility with existing implementations.

When using ephemeral source ports, the receiver can only set the source port upon arrival of an EESP packet with that Sub SA ID. If the receiver is pre-populating a Sub SA, it may have to install it with a source port set to zero and, upon arrival of a packet, update the source port using a mapping change.

Additionally, when multiple Sub SAs exist, the receiver can maintain a mapping table to track the source port associated with each Sub SA independently. This ensures the packets of the same Sub SA therefore the same Layer 4 flows are steered to the same NIC queue or CPU to prevents state locking in handling packets associated with different Sub SAs.

7. Auditing

Not all systems that implement EESP will implement auditing. However, if EESP is incorporated into a system that supports auditing, then the EESP implementation MUST also support auditing and MUST allow a system administrator to enable or disable auditing for EESP. 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.

* No valid Security Association exists for a session. The audit log entry for this event SHOULD include the SPI value, Session ID value, date/time received, Source Address, Destination Address, Sequence Number, and (for IPv6) the cleartext Flow ID.

* A packet offered to EESP for processing appears to be an IP fragment, i.e., the OFFSET field is non-zero or the MORE FRAGMENTS flag is set. The audit log entry for this event SHOULD include the SPI value, Session ID value, date/time received, Source Address, Destination Address, Sequence Number, and (in IPv6) the Flow ID.

* Attempt to transmit a packet that would result in Sequence Number overflow. The audit log entry for this event SHOULD include the SPI value, Session ID value, current date/time, Source Address, Destination Address, Sequence Number, and (for IPv6) the cleartext Flow ID.

* The received packet fails the anti-replay checks. The audit log entry for this event SHOULD include the SPI value, Session ID value, date/time received, Source Address, Destination Address, the Sequence Number, and (in IPv6) the Flow ID.

* The integrity check fails. The audit log entry for this event SHOULD include the SPI value, Session ID value, date/time received, Source Address, Destination Address, the Sequence Number, and (for IPv6) the Flow ID.

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.

8. Conformance Requirements

Implementations that claim conformance or compliance with this specification MUST implement the EESP syntax and processing described here for unicast traffic, and MUST comply with all additional packet processing requirements levied by the Security Architecture document [RFC4301]. 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 requires correct maintenance of the counter state at the sender (across local reboots, etc.), 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 anti-replay service in conjunction with SAs that are manually keyed.

The mandatory-to-implement algorithms for use with EESP described in a separate RFC, for e.g. RFC8221bis or another I.D., to facilitate updating the algorithm requirements independently from the protocol per se. Additional algorithms, beyond those mandated for EESP, MAY be supported.

9. Security Considerations

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

10. IANA Considerations

10.1. EESP IP Protocol Number

This document requests IANA allocate an IP protocol number from "Protocol Numbers - Assigned Internet Protocol Numbers" registry

* Decimal: TBD

* Keyword: EESP

* Protocol: Enhanced Encapsulating Security Payload

* Reference: This document

10.2. EESP Versions Registry

This document requests IANA to create a registry called "EESP_VERSIONS Type Registry" under a new category named "EESP_VERSIONS Parameters".

* Name: EESP Versions Registry

* Description: EESP Base Header Version

* Reference: This document

The initial content for this registry is as follows:

Value	EESP Version	Reference
0	V0	[this document]
1-13	Unassigned	[this document]
14-15	Private Use	[this document]

Figure 20: EESP Version Initial Registry Values

10.3. EESP Options Registry

This document requests IANA to create a registry called "EESP_OPTIONS Type Registry" under a new category named "EESP_OPTIONS Parameters".

* Name: EESP Options Registry

* Description: EESP Base Header Options

* Reference: This document

The initial content for this registry is as follows:

Value	EESP Header Options Types	Reference
0	Pad1	[this document]
1	PadN	[this document]
2	Crypt Offset	[this document]
3-223	Unassigned	[this document]
224-255	Private	[this document]

Figure 21: Initial Registry Values

11. Implementation Status

[Note to RFC Editor: Please remove this section and the reference to [RFC7942] before publication.]

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Authors are requested to add a note to the RFC Editor at the top of this section, advising the Editor to remove the entire section before publication, as well as the reference to [RFC7942].

12. Acknowledgments

TBD

13. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC4494] Song, JH., Poovendran, R., and J. Lee, "The AES-CMAC-96 Algorithm and Its Use with IPsec", RFC 4494, DOI 10.17487/RFC4494, June 2006, <https://www.rfc-editor.org/info/rfc4494>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC7402] Jokela, P., Moskowitz, R., and J. Melen, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 7402, DOI 10.17487/RFC7402, April 2015, <https://www.rfc-editor.org/info/rfc7402>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC9347] Hopps, C., "Aggregation and Fragmentation Mode for Encapsulating Security Payload (ESP) and Its Use for IP Traffic Flow Security (IP-TFS)", RFC 9347, DOI 10.17487/RFC9347, January 2023, <https://www.rfc-editor.org/info/rfc9347>.

14. Informative References

[I-D.he-ipsecme-vpn-shared-ipsecsa]
He, Q., Pan, W., Chen, X., and B. Ding, "Shared Use of IPsec Tunnel in a Multi-VPN Environment", Work in Progress, Internet-Draft, draft-he-ipsecme-vpn-shared-ipsecsa-01, 8 July 2024, <https://datatracker.ietf.org/doc/html/draft-he-ipsecme-vpn-shared-ipsecsa-01>.

[I-D.mrossberg-ipsecme-multiple-sequence-counters]
Rossberg, M., Klassert, S., and M. Pfeiffer, "Broadening the Scope of Encapsulating Security Payload (ESP) Protocol", Work in Progress, Internet-Draft, draft-mrossberg-ipsecme-multiple-sequence-counters-02, 15 February 2024, <https://datatracker.ietf.org/doc/html/draft-mrossberg-ipsecme-multiple-sequence-counters-02>.

[I-D.ponchon-ipsecme-anti-replay-subspaces]
Ponchon, P., Shaikh, M., Dernaika, H., Pfister, P., and G. Solignac, "IPsec and IKE anti-replay sequence number subspaces for traffic-engineered paths and multi-core processing", Work in Progress, Internet-Draft, draft-ponchon-ipsecme-anti-replay-subspaces-03, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ponchon-ipsecme-anti-replay-subspaces-03>.

[NIST01] NIST, "Federal Information Processing Standards Publication 140-2 (FIPS PUB 140-2), "Security Requirements for Cryptographic Modules", Information Technology Laboratory, National Institute of Standards and Technology, May 25, 2001.".

[Protocol] IANA, "Assigned Internet Protocol Numbers", <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>.

[PSP] Google, "PSP Architecture Specification", <https://github.com/google/psp/blob/main/doc/PSP_Arch_Spec.pdf>.

[RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992, DOI 10.17487/RFC2992, November 2000, <https://www.rfc-editor.org/info/rfc2992>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <https://www.rfc-editor.org/info/rfc3948>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <https://www.rfc-editor.org/info/rfc4106>.

[RFC4543] McGrew, D. and J. Viega, "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, DOI 10.17487/RFC4543, May 2006, <https://www.rfc-editor.org/info/rfc4543>.

[RFC7634] Nir, Y., "ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec", RFC 7634, DOI 10.17487/RFC7634, August 2015, <https://www.rfc-editor.org/info/rfc7634>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8750] Migault, D., Guggemos, T., and Y. Nir, "Implicit Initialization Vector (IV) for Counter-Based Ciphers in Encapsulating Security Payload (ESP)", RFC 8750, DOI 10.17487/RFC8750, March 2020, <https://www.rfc-editor.org/info/rfc8750>.

Appendix A. Additional Stuff

TBD

Authors' Addresses

Steffen Klassert
secunet Security Networks AG
Email: steffen.klassert@secunet.com

Antony Antony
secunet Security Networks AG
Email: antony.antony@secunet.com

Christian Hopps
LabN Consulting, L.L.C.
Email: chopps@chopps.org