Internet Engineering Task Force (IETF)
Request for Comments: 7402
Obsoletes: 5202
Category: Standards Track
ISSN: 2070-1721

P. Jokela
Ericsson Research NomadicLab
R. Moskowitz
HTT Consulting
J. Melen
Ericsson Research NomadicLab

April 2015

Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)

Abstract

This memo specifies an Encapsulating Security Payload (ESP) based mechanism for transmission of user data packets, to be used with the Host Identity Protocol (HIP). This document obsoletes RFC 5202.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7402.

Table of Contents

1. Introduction

In the Host Identity Protocol Architecture [HIP-ARCH], hosts are identified with public keys. The Host Identity Protocol (HIP) [RFC7401] base exchange allows any two HIP-supporting hosts to authenticate each other and to create a HIP association between themselves. During the base exchange, the hosts generate a piece of shared keying material using an authenticated Diffie-Hellman exchange.

The HIP base exchange specification [RFC7401] does not describe any transport formats or methods for user data to be used during the actual communication; it only defines that it is mandatory to implement the Encapsulating Security Payload (ESP) [RFC4303] based transport format and method. This document specifies how ESP is used with HIP to carry actual user data.

To be more specific, this document specifies a set of HIP protocol extensions and their handling. Using these extensions, a pair of ESP Security Associations (SAs) is created between the hosts during the base exchange. The resulting ESP Security Associations use keys drawn from the keying material (KEYMAT) generated during the base exchange. After the HIP association and required ESP SAs have been

established between the hosts, the user data communication is protected using ESP. In addition, this document specifies methods to update an existing ESP Security Association.

It should be noted that representations of Host Identity are not carried explicitly in the headers of user data packets. Instead, the ESP Security Parameter Index (SPI) is used to indicate the right host context. The SPIs are selected during the HIP ESP setup exchange. For user data packets, ESP SPIs (in possible combination with IP addresses) are used indirectly to identify the host context, thereby avoiding any additional explicit protocol headers.

HIP and ESP traffic have known issues with middlebox traversal (RFC 5207 [RFC5207]). Other specifications exist for operating HIP and ESP over UDP. (RFC 5770 [RFC5770] is an experimental specification, and others are being developed.) Middlebox traversal is out of scope for this document.

This document obsoletes RFC 5202.

2. Conventions Used in This Document

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 RFC 2119 [RFC2119].

3. Using ESP with HIP

The HIP base exchange is used to set up a HIP association between two hosts. The base exchange provides two-way host authentication and key material generation, but it does not provide any means for protecting data communication between the hosts. In this document, we specify the use of ESP for protecting user data traffic after the HIP base exchange. Note that this use of ESP is intended only for host-to-host traffic; security gateways are not supported.

To support ESP use, the HIP base exchange messages require some minor additions to the parameters transported. In the R1 packet, the Responder adds the possible ESP transforms in an ESP_TRANSFORM parameter before sending it to the Initiator. The Initiator gets the proposed transforms, selects one of those proposed transforms, and adds it to the I2 packet in an ESP_TRANSFORM parameter. In this I2 packet, the Initiator also sends the SPI value that it wants to be used for ESP traffic flowing from the Responder to the Initiator. This information is carried using the ESP_INFO parameter. When finalizing the ESP SA setup, the Responder sends its SPI value to the Initiator in the R2 packet, again using ESP_INFO.

3.1. ESP Packet Format

The ESP specification [RFC4303] defines the ESP packet format for IPsec. The HIP ESP packet looks exactly the same as the IPsec ESP transport format packet. The semantics, however, are a bit different and are described in more detail in the next subsection.

3.2. Conceptual ESP Packet Processing

ESP packet processing can be implemented in different ways in HIP. It is possible to implement it in a way that a standards compliant, unmodified IPsec implementation [RFC4303] can be used in conjunction with some additional transport checksum processing above it, and if IP addresses are used as indexes to the right host context.

When a standards compliant IPsec implementation that uses IP addresses in the Security Policy Database (SPD) and Security Association Database (SAD) is used, the packet processing may take the following steps. For outgoing packets, assuming that the upper-layer pseudo header has been built using IP addresses, the implementation recalculates upper-layer checksums using Host Identity Tags (HITs) and, after that, changes the packet source and destination addresses back to corresponding IP addresses. The packet is sent to the IPsec ESP for transport mode handling, and from there the encrypted packet is sent to the network. When an ESP packet is received, the packet is first put through the IPsec ESP transport mode handling, and after decryption, the source and destination IP addresses are replaced with HITs, and finally, upper-layer checksums are verified before passing the packet to the upper layer.

An alternative way to implement packet processing is the BEET (Bound End-to-End Tunnel) mode (see Appendix B). In BEET mode, the ESP packet is formatted as a transport mode packet, but the semantics of the connection are the same as for tunnel mode. The "outer" addresses of the packet are the IP addresses, and the "inner" addresses are the HITs. For outgoing traffic, after the packet has been encrypted, the packet’s IP header is changed to a new one that contains IP addresses instead of HITs, and the packet is sent to the network. When the ESP packet is received, the SPI value, together with the integrity protection, allow the packet to be securely associated with the right HIT pair. The packet header is replaced with a new header containing HITs, and the packet is decrypted. BEET mode is completely internal for a host and doesn’t require that the corresponding host implement it; instead, the corresponding host can have ESP transport mode and do HIT IP conversions outside ESP.

3.2.1. Semantics of the Security Parameter Index (SPI)

SPIs are used in ESP to find the right Security Association for received packets. The ESP SPIs have added significance when used with HIP; they are a compressed representation of a pair of HITs. Thus, SPIs MAY be used by intermediary systems in providing services like address mapping. Note that since the SPI has significance at the receiver, only the < DST, SPI >, where DST is a destination IP address, uniquely identifies the receiver HIT at any given point of time. The same SPI value may be used by several hosts. A single < DST, SPI > value may denote different hosts and contexts at different points of time, depending on the host that is currently reachable at the DST.

Each host selects for itself the SPI it wants to see in packets received from its peer. This allows it to select different SPIs for different peers. The SPI selection SHOULD be random; the rules of Section 2.1 of the ESP specification [RFC4303] must be followed. A different SPI SHOULD be used for each HIP exchange with a particular host; this is to avoid a replay attack. Additionally, when a host rekeys, the SPI MUST be changed. Furthermore, if a host changes over to use a different IP address, it MAY change the SPI.

One method for SPI creation that meets the above criteria would be to concatenate the HIT with a 32-bit random or sequential number, hash this (using SHA1), and then use the high-order 32 bits as the SPI.

The selected SPI is communicated to the peer in the third (I2) and fourth (R2) packets of the base HIP exchange. Changes in SPI are signaled with ESP_INFO parameters.

3.3. Security Association Establishment and Maintenance

3.3.1. ESP Security Associations

In HIP, ESP Security Associations are set up between the HIP nodes during the base exchange [RFC7401]. Existing ESP SAs can be updated later using UPDATE messages. The reason for updating the ESP SA later can be, for example, a need for rekeying the SA because of sequence number rollover.

Upon setting up a HIP association, each association is linked to two ESP SAs, one for incoming packets and one for outgoing packets. The Initiator’s incoming SA corresponds with the Responder’s outgoing one, and vice versa. The Initiator defines the SPI for its incoming association, as defined in Section 3.2.1. This SA is herein called

SA-RI, and the corresponding SPI is called SPI-RI. Respectively, the Responder's incoming SA corresponds with the Initiator's outgoing SA and is called SA-IR, with the SPI being called SPI-IR.

The Initiator creates SA-RI as a part of R1 processing, before sending out the I2, as explained in Section 6.4. The keys are derived from KEYMAT, as defined in Section 7. The Responder creates SA-RI as a part of I2 processing; see Section 6.5.

The Responder creates SA-IR as a part of I2 processing, before sending out R2; see Section 6.5. The Initiator creates SA-IR when processing R2; see Section 6.6.

The initial session keys are drawn from the generated keying material, KEYMAT, after the HIP keys have been drawn as specified in [RFC7401].

When the HIP association is removed, the related ESP SAs MUST also be removed.

3.3.2. Rekeying

After the initial HIP base exchange and SA establishment, both hosts are in the ESTABLISHED state. There are no longer Initiator and Responder roles, and the association is symmetric. In this subsection, the party that initiates the rekey procedure is denoted with I' and the peer with R'.

An existing HIP-created ESP SA may need updating during the lifetime of the HIP association. This document specifies the rekeying of an existing HIP-created ESP SA, using the UPDATE message. The ESP_INFO parameter introduced above is used for this purpose.

I' initiates the ESP SA updating process when needed (see Section 6.8). It creates an UPDATE packet with required information and sends it to the peer node. The old SAs are still in use, local policy permitting.

R', after receiving and processing the UPDATE (see Section 6.9), generates new SAs: SA-I'R' and SA-R'I'. It does not take the new outgoing SA into use, but still uses the old one, so there temporarily exist two SA pairs towards the same peer host. The SPI for the new outgoing SA, SPI-R'I', is specified in the received ESP_INFO parameter in the UPDATE packet. For the new incoming SA, R' generates the new SPI value, SPI-I'R', and includes it in the response UPDATE packet.

When I' receives a response UPDATE from R', it generates new SAs, as described in Section 6.9: SA-I'R' and SA-R'I'. It starts using the new outgoing SA immediately.

R' starts using the new outgoing SA when it receives traffic on the new incoming SA or when it receives the UPDATE ACK confirming completion of rekeying. After this, R' can remove the old SAs. Similarly, when the I' receives traffic from the new incoming SA, it can safely remove the old SAs.

3.3.3. Security Association Management

An SA pair is indexed by the 2 SPIs and 2 HITs (both local and remote HITs since a system can have more than one HIT). An inactivity timer is RECOMMENDED for all SAs. If the state dictates the deletion of an SA, a timer is set to allow for any late arriving packets.

3.3.4. Security Parameter Index (SPI)

The SPIs in ESP provide a simple compression of the HIP data from all packets after the HIP exchange. This does require a per HIT-pair Security Association (and SPI), and a decrease of policy granularity over other Key Management Protocols like Internet Key Exchange (IKE) [RFC7296].

When a host updates the ESP SA, it provides a new inbound SPI to and gets a new outbound SPI from its peer.

3.3.5. Supported Ciphers

All HIP implementations MUST support AES-128-CBC and AES-256-CBC [RFC3602]. If the Initiator does not support any of the transforms offered by the Responder, it should abandon the negotiation and inform the peer with a NOTIFY message about a non-supported transform.

In addition to AES-128-CBC, all implementations SHOULD implement the ESP NULL encryption algorithm. When the ESP NULL encryption is used, it MUST be used together with SHA-256 authentication as specified in Section 5.1.2.

When an authentication-only suite is used (NULL, AES-CMAC-96, and AES-GMAC are examples), the suite MUST NOT be accepted if offered by the peer unless the local policy configuration regarding the peer host is explicitly set to allow an authentication-only mode. This is to prevent sessions from being downgraded to an authentication-only mode when one side's policy requests privacy for the session.

3.3.6. Sequence Number

The Sequence Number field is MANDATORY when ESP is used with HIP. Anti-replay protection MUST be used in an ESP SA established with HIP. When ESP is used with HIP, a 64-bit sequence number MUST be used. This means that each host MUST rekey before its sequence number reaches 2^64.

When using a 64-bit sequence number, the higher 32 bits are NOT included in the ESP header, but are simply kept local to both peers. See [RFC4301].

3.3.7. Lifetimes and Timers

HIP does not negotiate any lifetimes. All ESP lifetimes are local policy. The only lifetimes a HIP implementation MUST support are sequence number rollover (for replay protection), and SHOULD support timing out inactive ESP SAs. An SA times out if no packets are received using that SA. Implementations SHOULD support a configurable SA timeout value. Implementations MAY support lifetimes for the various ESP transforms. Each implementation SHOULD implement per-HIT configuration of the inactivity timeout, allowing statically configured HIP associations to stay alive for days, even when inactive.

3.4. IPsec and HIP ESP Implementation Considerations

When HIP is run on a node where a standards compliant IPsec is used, some issues have to be considered.

The HIP implementation must be able to co-exist with other IPsec keying protocols. When the HIP implementation selects the SPI value, it may lead to a collision if not implemented properly. To avoid the possibility for a collision, the HIP implementation MUST ensure that the SPI values used for HIP SAs are not used for IPsec or other SAs, and vice versa.

Incoming packets using an SA that is not negotiated by HIP MUST NOT be processed as described in Section 3.2, paragraph 2. The SPI will identify the correct SA for packet decryption and MUST be used to identify that the packet has an upper-layer checksum that is calculated as specified in [RFC7401].

3.4.1. Data Packet Processing Considerations

For outbound traffic, the SPD (or coordinated SPDs, if there are two -- one for HIP and one for IPsec) MUST ensure that packets intended for HIP processing are given a HIP-enabled SA and that packets intended for IPsec processing are given an IPsec-enabled SA. The SP then MUST be bound to the matching SA, and non-HIP packets will not be processed by this SA. Data originating from a socket that is not using HIP MUST NOT have the checksum recalculated (as described in Section 3.2, paragraph 2), and data MUST NOT be passed to the SP or SA created by HIP.

It is possible that in the case of overlapping policies, the outgoing packet would be handled by both IPsec and HIP. In this case, it is possible that the HIP association is end to end, while the IPsec SA is for encryption between the HIP host and a security gateway. In the case of a security gateway ESP association, the ESP always uses tunnel mode.

In the case of IPsec tunnel mode, it is hard to see during the HIP SA processing if the IPsec ESP SA has the same final destination. Thus, traffic MUST be encrypted with both the HIP ESP SA and the IPsec SA when the IPsec ESP SA is used in tunnel mode.

In the case of IPsec transport mode, the connection endpoints are the same. However, for HIP data packets it is not possible to avoid HIP SA processing, while mapping the HIP data packet's IP addresses to the corresponding HITs requires SPI values from the ESP header. In the case of a transport mode IPsec SA, the IPsec encryption MAY be skipped to avoid double encryption, if the local policy allows.

3.4.2. HIP Signaling Packet Considerations

In general, HIP signaling packets should follow the same processing as HIP data packets.

In the case of IPsec tunnel mode, the HIP signaling packets are always encrypted using an IPsec ESP SA. Note that this hides the HIP signaling packets from the eventual HIP middleboxes on the path between the originating host and the security gateway.

In the case of IPsec transport mode, the HIP signaling packets MAY skip the IPsec ESP SA encryption if the local policy allows. This allows the eventual HIP middleboxes to handle the passing HIP signaling packets.

4. The Protocol

In this section, the protocol for setting up an ESP association to be used with a HIP association is described.

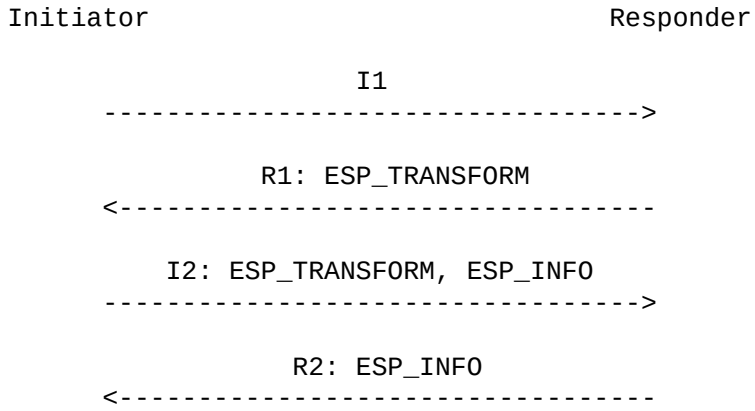
4.1. ESP in HIP

4.1.1. IPsec ESP Transport Format Type

The HIP handshake signals the TRANSPORT_FORMAT_LIST parameter in the R1 and I2 messages. This parameter contains a list of the supported HIP transport formats of the sending host, in the order of preference. The transport format type for IPsec ESP is the type number of the ESP_TRANSFORM parameter, i.e., 4095.

4.1.2. Setting Up an ESP Security Association

Setting up an ESP Security Association between hosts using HIP is performed by including parameters in the last three messages (R1, I2, and R2 messages) of the four-message HIP base exchange.



The R1 message contains the ESP_TRANSFORM parameter, in which the sending host defines the possible ESP transforms it is willing to use for the ESP SA.

Including the ESP_TRANSFORM parameter in the R1 message adds clarity to the TRANSPORT_FORMAT_LIST but may initiate negotiations for possibly unselected transforms. However, resource-constrained devices will most likely restrict support to a single transform for the sake of minimizing ROM overhead, and the additional parameter adds negligible overhead with unconstrained devices.

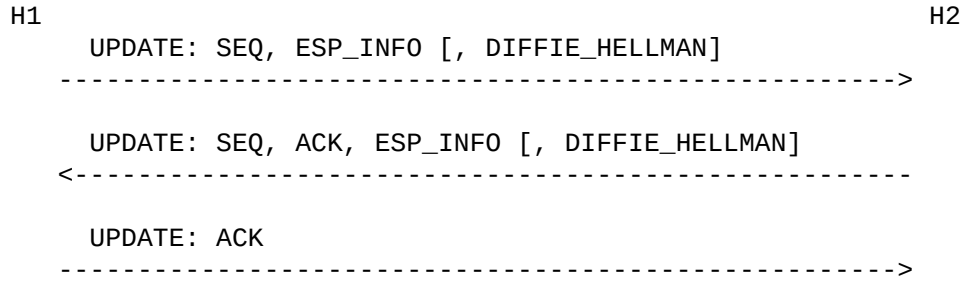
The I2 message contains the response to an ESP_TRANSFORM received in the R1 message. The sender must select one of the proposed ESP transforms from the ESP_TRANSFORM parameter in the R1 message and include the selected one in the ESP_TRANSFORM parameter in the I2 packet. In addition to the transform, the host includes the ESP_INFO parameter containing the SPI value to be used by the peer host.

In the R2 message, the ESP SA setup is finalized. The packet contains the SPI information required by the Initiator for the ESP SA.

4.1.3. Updating an Existing ESP SA

The update process is accomplished using three messages. The HIP UPDATE message is used to update the parameters of an existing ESP SA. The UPDATE mechanism and message are defined in [RFC7401], and the additional parameters for updating an existing ESP SA are described here.

The following picture shows a typical exchange when an existing ESP SA is updated. Messages include SEQ and ACK parameters required by the UPDATE mechanism.



The host willing to update the ESP SA creates and sends an UPDATE message. The message contains the ESP_INFO parameter containing the old SPI value that was used, the new SPI value to be used, and the index value for the keying material, giving the point from where the next keys will be drawn. If new keying material must be generated, the UPDATE message will also contain the DIFFIE_HELLMAN parameter defined in [RFC7401].

The host receiving the UPDATE message requesting update of an existing ESP SA MUST reply with an UPDATE message. In the reply message, the host sends the ESP_INFO parameter containing the corresponding values: old SPI, new SPI, and the keying material index. If the incoming UPDATE contained a DIFFIE_HELLMAN parameter, the reply packet MUST also contain a DIFFIE_HELLMAN parameter.

5. Parameter and Packet Formats

In this section, new and modified HIP parameters are presented, as well as modified HIP packets.

5.1. New Parameters

Two HIP parameters are defined for setting up ESP transport format associations in HIP communication and for rekeying existing ones. Also, the NOTIFICATION parameter, described in [RFC7401], has two error values defined for this specification.

Parameter	Type	Length	Data
ESP_INFO	65	12	Remote's old SPI, new SPI, and other info
ESP_TRANSFORM	4095	variable	ESP Encryption and Authentication Transform(s)

5.1.1. ESP_INFO

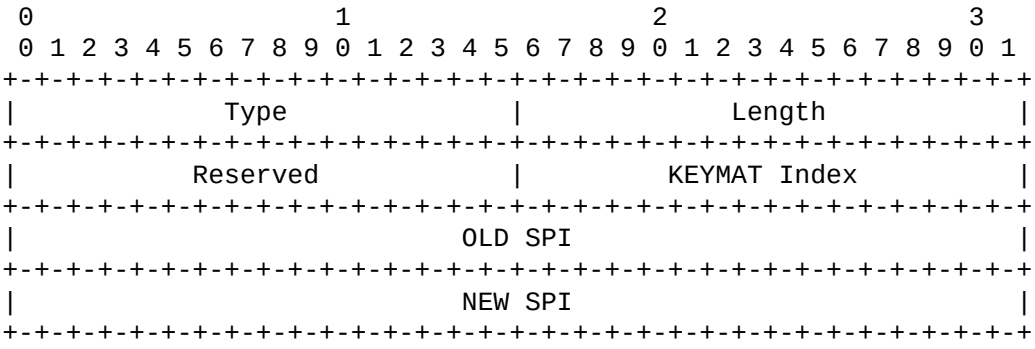
During the establishment and update of an ESP SA, the SPI value of both hosts must be transmitted between the hosts. In addition, hosts need the index value to the KEYMAT when they are drawing keys from the generated keying material. The ESP_INFO parameter is used to transmit the SPI values and the KEYMAT index information between the hosts.

During the initial ESP SA setup, the hosts send the SPI value that they want the peer to use when sending ESP data to them. The value is set in the NEW SPI field of the ESP_INFO parameter. In the initial setup, an old value for the SPI does not exist; thus, the OLD SPI field value is set to zero. The OLD SPI field value may also be zero when additional SAs are set up between HIP hosts, e.g., in the case of multihomed HIP hosts [RFC5206]. However, such use is beyond the scope of this specification.

The KEYMAT index value points to the place in the KEYMAT from where the keying material for the ESP SAs is drawn. The KEYMAT index value is zero only when the ESP_INFO is sent during a rekeying process and new keying material is generated.

During the life of an SA established by HIP, one of the hosts may need to reset the Sequence Number to one and rekey. The reason for rekeying might be an approaching sequence number wrap in ESP, or a local policy on the use of a key. Rekeying ends the current SAs and starts new ones on both peers.

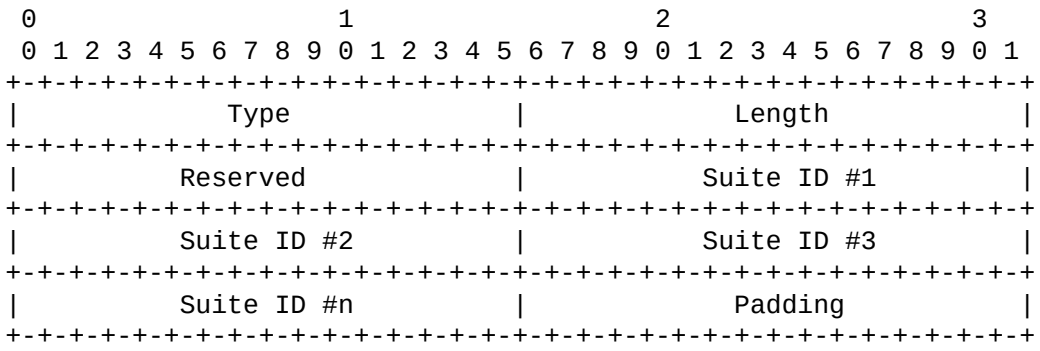
During the rekeying process, the ESP_INFO parameter is used to transmit the changed SPI values and the keying material index.



Type	65
Length	12
KEYMAT Index	index, in bytes, where to continue to draw ESP keys from KEYMAT. If the packet includes a new Diffie-Hellman key and the ESP_INFO is sent in an UPDATE packet, the field MUST be zero. If the ESP_INFO is included in base exchange messages, the KEYMAT Index must have the index value of the point from where the ESP SA keys are drawn. Note that the length of this field limits the amount of keying material that can be drawn from KEYMAT. If that amount is exceeded, the packet MUST contain a new Diffie-Hellman key.
OLD SPI	old SPI for data sent to address(es) associated with this SA. If this is an initial SA setup, the OLD SPI value is zero.
NEW SPI	new SPI for data sent to address(es) associated with this SA.

5.1.2. ESP_TRANSFORM

The ESP_TRANSFORM parameter is used during ESP SA establishment. The first party sends a selection of transform families in the ESP_TRANSFORM parameter, and the peer must select one of the proposed values and include it in the response ESP_TRANSFORM parameter.



Type	4095
Length	length in octets, excluding Type, Length, and padding.
Reserved	zero when sent, ignored when received.
Suite ID	defines the ESP Suite to be used.

The following Suite IDs can be used:

Suite ID	Value	
RESERVED	0	[RFC7402]
AES-128-CBC with HMAC-SHA1	1	[RFC3602], [RFC2404]
DEPRECATED	2	[RFC7402]
DEPRECATED	3	[RFC7402]
DEPRECATED	4	[RFC7402]
DEPRECATED	5	[RFC7402]
DEPRECATED	6	[RFC7402]
NULL with HMAC-SHA-256	7	[RFC2410], [RFC4868]
AES-128-CBC with HMAC-SHA-256	8	[RFC3602], [RFC4868]
AES-256-CBC with HMAC-SHA-256	9	[RFC3602], [RFC4868]
AES-CCM-8	10	[RFC4309]
AES-CCM-16	11	[RFC4309]
AES-GCM with an 8-octet ICV	12	[RFC4106]
AES-GCM with a 16-octet ICV	13	[RFC4106]
AES-CMAC-96	14	[RFC4493], [RFC4494]
AES-GMAC	15	[RFC4543]

The sender of an ESP transform parameter MUST make sure that there are no more than six (6) Suite IDs in one ESP transform parameter. Conversely, a recipient MUST be prepared to handle received transform parameters that contain more than six Suite IDs. The limited number of Suite IDs sets the maximum size of the ESP_TRANSFORM parameter. As the default configuration, the ESP_TRANSFORM parameter MUST contain at least one of the mandatory Suite IDs. There MAY be a configuration option that allows the administrator to override this default.

Mandatory implementations: AES-128-CBC with HMAC-SHA-256. NULL with HMAC-SHA-256 SHOULD also be supported (see also Section 3.3.5).

Under some conditions, it is possible to use Traffic Flow Confidentiality (TFC) [RFC4303] with ESP in BEET mode. However, the definition of such an operation is left for future work and must be done in a separate specification.

5.1.3. NOTIFICATION Parameter

The HIP base specification defines a set of NOTIFICATION error types. The following error types are required for describing errors in ESP Transform crypto suites during negotiation.

NOTIFICATION PARAMETER - ERROR TYPES	Value
NO_ESP_PROPOSAL_CHOSEN	18

None of the proposed ESP Transform crypto suites was acceptable.

INVALID_ESP_TRANSFORM_CHOSEN	19

The ESP Transform crypto suite does not correspond to one offered by the Responder.

5.2. HIP ESP Security Association Setup

The ESP Security Association is set up during the base exchange. The following subsections define the ESP SA setup procedure using both base exchange messages (R1, I2, R2) and UPDATE messages.

5.2.1. Setup during Base Exchange

5.2.1.1. Modifications in R1

The ESP_TRANSFORM contains the ESP modes supported by the sender, in the order of preference. All implementations MUST support AES-128-CBC [RFC3602] with HMAC-SHA-256 [RFC4868].

The following figure shows the resulting R1 packet layout.

The HIP parameters for the R1 packet:

```
IP ( HIP ( [ R1_COUNTER, ]
           PUZZLE,
           DIFFIE_HELLMAN,
           HIP_CIPHER,
           ESP_TRANSFORM,
           HOST_ID,
           [ ECHO_REQUEST, ]
           HIP_SIGNATURE_2 )
           [, ECHO_REQUEST ])
```

5.2.1.2. Modifications in I2

The ESP_INFO contains the sender's SPI for this association as well as the KEYMAT index from where the ESP SA keys will be drawn. The old SPI value is set to zero.

The ESP_TRANSFORM contains the ESP mode selected by the sender of R1. All implementations MUST support AES-128-CBC [RFC3602] with HMAC-SHA-256 [RFC4868].

The following figure shows the resulting I2 packet layout.

The HIP parameters for the I2 packet:

```
IP ( HIP ( ESP_INFO,
           [R1_COUNTER,]
           SOLUTION,
           DIFFIE_HELLMAN,
           HIP_CIPHER,
           ESP_TRANSFORM,
           ENCRYPTED { HOST_ID },
           [ ECHO_RESPONSE ,]
           HMAC,
           HIP_SIGNATURE
           [, ECHO_RESPONSE] ) )
```

5.2.1.3. Modifications in R2

The R2 contains an ESP_INFO parameter, which has the SPI value of the sender of the R2 for this association. The ESP_INFO also has the KEYMAT index value specifying where the ESP SA keys are drawn.

The following figure shows the resulting R2 packet layout.

The HIP parameters for the R2 packet:

```
IP ( HIP ( ESP_INFO, HMAC_2, HIP_SIGNATURE ) )
```

5.3. HIP ESP Rekeying

In this section, the procedure for rekeying an existing ESP SA is presented.

Conceptually, the process can be represented by the following message sequence using the host names I' and R' defined in Section 3.3.2. For simplicity, HMAC and HIP_SIGNATURE are not depicted, and DIFFIE_HELLMAN keys are optional. The UPDATE with ACK_I need not be piggybacked with the UPDATE with SEQ_R; it may be ACKed separately (in which case the sequence would include four packets).

```
        I'                                  R'

             UPDATE(ESP_INFO, SEQ_I, [DIFFIE_HELLMAN])
         ---------------------------------->
             UPDATE(ESP_INFO, SEQ_R, ACK_I, [DIFFIE_HELLMAN])
         <----------------------------------
             UPDATE(ACK_R)
         ---------------------------------->
```

Below, the first two packets in this figure are explained.

5.3.1. Initializing Rekeying

When HIP is used with ESP, the UPDATE packet is used to initiate rekeying. The UPDATE packet MUST carry an ESP_INFO and MAY carry a DIFFIE_HELLMAN parameter.

Intermediate systems that use the SPI will have to inspect HIP packets for those that carry rekeying information. The packet is signed for the benefit of the intermediate systems. Since intermediate systems may need the new SPI values, the contents cannot be encrypted.

The following figure shows the contents of a rekeying initialization UPDATE packet.

The HIP parameters for the UPDATE packet initiating rekeying:

```
IP ( HIP ( ESP_INFO,
           SEQ,
           [DIFFIE_HELLMAN, ]
           HMAC,
           HIP_SIGNATURE ) )
```

5.3.2. Responding to the Rekeying Initialization

The UPDATE ACK is used to acknowledge the received UPDATE rekeying initialization. The acknowledgment UPDATE packet MUST carry an ESP_INFO and MAY carry a DIFFIE_HELLMAN parameter.

Intermediate systems that use the SPI will have to inspect HIP packets for packets carrying rekeying information. The packet is signed for the benefit of the intermediate systems. Since intermediate systems may need the new SPI values, the contents cannot be encrypted.

The following figure shows the contents of a rekeying acknowledgment UPDATE packet.

The HIP parameters for the UPDATE packet:

```
IP ( HIP ( ESP_INFO,
           SEQ,
           ACK,
           [ DIFFIE_HELLMAN, ]
           HMAC,
           HIP_SIGNATURE ) )
```

5.4. ICMP Messages

ICMP message handling is mainly described in the HIP base specification [RFC7401]. In this section, we describe the actions related to ESP security associations.

5.4.1. Unknown SPI

If a HIP implementation receives an ESP packet that has an unrecognized SPI number, it MAY respond (subject to rate limiting the responses) with an ICMP packet with type "Parameter Problem", with the pointer pointing to the beginning of the SPI field in the ESP header.

6. Packet Processing

Packet processing is mainly defined in the HIP base specification [RFC7401]. This section describes the changes and new requirements for packet handling when the ESP transport format is used. Note that all HIP packets (currently protocol 139) MUST bypass ESP processing.

6.1. Processing Outgoing Application Data

Outgoing application data handling is specified in the HIP base specification [RFC7401]. When the ESP transport format is used, and there is an active HIP session for the given < source, destination > HIT pair, the outgoing datagram is protected using the ESP security association. The following additional steps define the conceptual processing rules for outgoing ESP protected datagrams.

1. Detect the proper ESP SA using the HITs in the packet header or other information associated with the packet.

2. Process the packet normally, as if the SA was a transport mode SA.

3. Ensure that the outgoing ESP protected packet has proper IP header format, depending on the used IP address family, and proper IP addresses in its IP header, e.g., by replacing HITs left by the ESP processing. Note that this placement of proper IP addresses MAY also be performed at some other point in the stack, e.g., before ESP processing.

6.2. Processing Incoming Application Data

Incoming HIP user data packets arrive as ESP protected packets. In the usual case, the receiving host has a corresponding ESP security association, identified by the SPI and destination IP address in the packet. However, if the host has crashed or otherwise lost its HIP state, it may not have such an SA.

The basic incoming data handling is specified in the HIP base specification. Additional steps are required when ESP is used for protecting the data traffic. The following steps define the conceptual processing rules for incoming ESP protected datagrams targeted to an ESP security association created with HIP.

1. Detect the proper ESP SA using the SPI. If the resulting SA is a non-HIP ESP SA, process the packet according to standard IPsec rules. If there are no SAs identified with the SPI, the host MAY send an ICMP packet as defined in Section 5.4. How to handle lost state is an implementation issue.

2. If the SPI matches with an active HIP-based ESP SA, the IP addresses in the datagram are replaced with the HITs associated with the SPI. Note that this IP-address-to-HIT conversion step MAY also be performed at some other point in the stack, e.g., after ESP processing. Note also that if the incoming packet has IPv4 addresses, the packet must be converted to IPv6 format before replacing the addresses with HITs (such that the transport checksum will pass if there are no errors).

3. The transformed packet is next processed normally by ESP, as if the packet were a transport mode packet. The packet may be dropped by ESP, as usual. In a typical implementation, the result of successful ESP decryption and verification is a datagram with the associated HITs as source and destination.

4. The datagram is delivered to the upper layer. Demultiplexing the datagram to the right upper-layer socket is performed as usual, except that the HITs are used in place of IP addresses during the demultiplexing.

6.3. HMAC and SIGNATURE Calculation and Verification

The new HIP parameters described in this document, ESP_INFO and ESP_TRANSFORM, must be protected using HMAC and signature calculations. In a typical implementation, they are included in R1, I2, R2, and UPDATE packet HMAC and SIGNATURE calculations as described in [RFC7401].

6.4. Processing Incoming ESP SA Initialization (R1)

The ESP SA setup is initialized in the R1 message. The receiving host (Initiator) selects one of the ESP transforms from the presented values. If no suitable value is found, the negotiation is terminated. The selected values are subsequently used when generating and using encryption keys, and when sending the reply packet. If the proposed alternatives are not acceptable to the system, it may abandon the ESP SA establishment negotiation, or it may resend the I1 message within the retry bounds.

After selecting the ESP transform and performing other R1 processing, the system prepares and creates an incoming ESP security association. It may also prepare a security association for outgoing traffic, but since it does not have the correct SPI value yet, it cannot activate it.

6.5. Processing Incoming Initialization Reply (I2)

The following steps are required to process the incoming ESP SA initialization replies in I2. The steps below assume that the I2 has been accepted for processing (e.g., has not been dropped due to HIT comparisons as described in [RFC7401]).

- The ESP_TRANSFORM parameter is verified, and it MUST contain a single value in the parameter; and it MUST match one of the values offered in the initialization packet.

- The ESP_INFO NEW SPI field is parsed to obtain the SPI that will be used for the Security Association outbound from the Responder and inbound to the Initiator. For this initial ESP SA establishment, the old SPI value MUST be zero. The KEYMAT Index field MUST contain the index value to the KEYMAT from where the ESP SA keys are drawn.

- The system prepares and creates both incoming and outgoing ESP security associations.

- Upon successful processing of the initialization reply message, the possible old Security Associations (as left over from an earlier incarnation of the HIP association) are dropped and the new ones are installed, and a finalizing packet, R2, is sent. Possible ongoing rekeying attempts are dropped.

6.6. Processing Incoming ESP SA Setup Finalization (R2)

Before the ESP SA can be finalized, the ESP_INFO NEW SPI field is parsed to obtain the SPI that will be used for the ESP Security Association inbound to the sender of the finalization message R2. The system uses this SPI to create or activate the outgoing ESP security association used for sending packets to the peer.

6.7. Dropping HIP Associations

When the system drops a HIP association, as described in the HIP base specification, the associated ESP SAs MUST also be dropped.

6.8. Initiating ESP SA Rekeying

During ESP SA rekeying, the hosts draw new keys from the existing keying material, or new keying material is generated from where the new keys are drawn.

A system may initiate the SA rekeying procedure at any time. It MUST initiate a rekey if its incoming ESP sequence counter is about to overflow. The system MUST NOT replace its keying material until the rekeying packet exchange successfully completes.

Optionally, a system may include a new Diffie-Hellman key for use in new KEYMAT generation. New KEYMAT generation occurs prior to drawing the new keys.

The rekeying procedure uses the UPDATE mechanism defined in [RFC7401]. Because each peer must update its half of the security association pair (including new SPI creation), the rekeying process requires that each side both send and receive an UPDATE. A system will then rekey the ESP SA when it has sent parameters to the peer and has received both an ACK of the relevant UPDATE message and corresponding peer’s parameters. It may be that the ACK and the required HIP parameters arrive in different UPDATE messages. This is always true if a system does not initiate an ESP SA update but responds to an update request from the peer, and may also occur if two systems initiate update nearly simultaneously. In such a case, if the system has an outstanding update request, it saves the one parameter and waits for the other before completing rekeying.

The following steps define the processing rules for initiating an ESP SA update:

1. The system decides whether to continue to use the existing KEYMAT or to generate a new KEYMAT. In the latter case, the system MUST generate a new Diffie-Hellman public key.

2. The system creates an UPDATE packet, which contains the ESP_INFO parameter. In addition, the host may include the optional DIFFIE_HELLMAN parameter. If the UPDATE contains the DIFFIE_HELLMAN parameter, the KEYMAT Index in the ESP_INFO parameter MUST be zero, and the Diffie-Hellman Group ID must be unchanged from that used in the initial handshake. If the UPDATE does not contain DIFFIE_HELLMAN, the ESP_INFO KEYMAT Index MUST be greater than or equal to the index of the next byte to be drawn from the current KEYMAT.

3. The system sends the UPDATE packet. For reliability, the underlying UPDATE retransmission mechanism MUST be used.

4. The system MUST NOT delete its existing SAs, but continue using them if its policy still allows. The rekeying procedure SHOULD be initiated early enough to make sure that the SA replay counters do not overflow.

5. In case a protocol error occurs and the peer system acknowledges the UPDATE but does not itself send an ESP_INFO, the system may not finalize the outstanding ESP SA update request. To guard against this, a system MAY re-initiate the ESP SA update procedure after some time waiting for the peer to respond, or it MAY decide to abort the ESP SA after waiting for an implementation-dependent time. The system MUST NOT keep an outstanding ESP SA update request for an indefinite time.

To simplify the state machine, a host MUST NOT generate new UPDATEs while it has an outstanding ESP SA update request, unless it is restarting the update process.

6.9. Processing Incoming UPDATE Packets

When a system receives an UPDATE packet, it must be processed if the following conditions hold (in addition to the generic conditions specified for UPDATE processing in Section 6.12 of [RFC7401]):

1. A corresponding HIP association must exist. This is usually ensured by the underlying UPDATE mechanism.

2. The state of the HIP association is ESTABLISHED or R2-SENT.

If the above conditions hold, the following steps define the conceptual processing rules for handling the received UPDATE packet:

1. If the received UPDATE contains a DIFFIE_HELLMAN parameter, the received KEYMAT Index MUST be zero and the Group ID must match the Group ID in use on the association. If this test fails, the packet SHOULD be dropped and the system SHOULD log an error message.

2. If there is no outstanding rekeying request, the packet processing continues as specified in Section 6.9.1.

3. If there is an outstanding rekeying request, the UPDATE MUST be acknowledged, the received ESP_INFO (and possibly DIFFIE_HELLMAN) parameters must be saved, and the packet processing continues as specified in Section 6.10.

6.9.1. Processing UPDATE Packet: No Outstanding Rekeying Request

The following steps define the conceptual processing rules for handling a received UPDATE packet with the ESP_INFO parameter:

1. The system consults its policy to see if it needs to generate a new Diffie-Hellman key, and generates a new key (with same Group ID) if needed. The system records any newly generated or received Diffie-Hellman keys for use in KEYMAT generation upon finalizing the ESP SA update.

2. If the system generated a new Diffie-Hellman key in the previous step, or if it received a DIFFIE_HELLMAN parameter, it sets the ESP_INFO KEYMAT Index to zero. Otherwise, the ESP_INFO KEYMAT Index MUST be greater than or equal to the index of the next byte to be drawn from the current KEYMAT. In this case, it is RECOMMENDED that the host use the KEYMAT Index requested by the peer in the received ESP_INFO.

3. The system creates an UPDATE packet, which contains an ESP_INFO parameter and the optional DIFFIE_HELLMAN parameter. This UPDATE would also typically acknowledge the peer's UPDATE with an ACK parameter, although a separate UPDATE ACK may be sent.

4. The system sends the UPDATE packet and stores any received ESP_INFO and DIFFIE_HELLMAN parameters. At this point, it only needs to receive an acknowledgment for the newly sent UPDATE to finish the ESP SA update. In the usual case, the acknowledgment is handled by the underlying UPDATE mechanism.

6.10. Finalizing Rekeying

A system finalizes rekeying when it has both received the corresponding UPDATE acknowledgment packet from the peer and successfully received the peer's UPDATE. The following steps are taken:

1. If the received UPDATE messages contain a new Diffie-Hellman key, the system has a new Diffie-Hellman key due to initiating an ESP SA update, or both, the system generates a new KEYMAT. If there is only one new Diffie-Hellman key, the old existing key is used as the other key.

2. If the system generated a new KEYMAT in the previous step, it sets the KEYMAT Index to zero, independent of whether the received UPDATE included a Diffie-Hellman key or not. If the system did not generate a new KEYMAT, it uses the greater KEYMAT Index of the two (sent and received) ESP_INFO parameters.

3. The system draws keys for new incoming and outgoing ESP SAs, starting from the KEYMAT Index, and prepares new incoming and outgoing ESP SAs. The SPI for the outgoing SA is the new SPI value received in an ESP_INFO parameter. The SPI for the incoming SA was generated when the ESP_INFO was sent to the peer. The order of the keys retrieved from the KEYMAT during the rekeying process is similar to that described in Section 7. Note that only IPsec ESP keys are retrieved during the rekeying process, not the HIP keys.

4. The system starts to send to the new outgoing SA and prepares to start receiving data on the new incoming SA. Once the system receives data on the new incoming SA, it may safely delete the old SAs.

6.11. Processing NOTIFY Packets

The processing of NOTIFY packets is described in the HIP base specification.

7. Keying Material

The keying material is generated as described in the HIP base specification. During the base exchange, the initial keys are drawn from the generated material. After the HIP association keys have been drawn, the ESP keys are drawn in the following order:

SA-gl ESP encryption key for HOST_g's outgoing traffic

SA-gl ESP authentication key for HOST_g's outgoing traffic

SA-lg ESP encryption key for HOST_l's outgoing traffic

SA-lg ESP authentication key for HOST_l's outgoing traffic

HOST_g denotes the host with the greater HIT value, and HOST_l denotes the host with the lower HIT value. When HIT values are compared, they are interpreted as positive (unsigned) 128-bit integers in network byte order.

The four HIP keys are only drawn from KEYMAT during a HIP I1->R2 exchange. Subsequent rekeys using UPDATE will only draw the four ESP keys from KEYMAT. Section 6.9 describes the rules for reusing or regenerating KEYMAT based on the rekeying.

The number of bits drawn for a given algorithm is the "natural" size of the keys, as specified in Section 6.5 of [RFC7401].

8. Security Considerations

In this document, the usage of ESP [RFC4303] between HIP hosts to protect data traffic is introduced. The security considerations for ESP are discussed in the ESP specification.

There are different ways to establish an ESP Security Association between two nodes. This can be done, e.g., using IKE [RFC7296]. This document specifies how the Host Identity Protocol is used to establish ESP Security Associations.

The following issues are new or have changed from the standard ESP usage:

- Initial keying material generation
- Updating the keying material

The initial keying material is generated using the Host Identity Protocol [RFC7401] using the Diffie-Hellman procedure. This document extends the usage of the UPDATE packet, defined in the base specification, to modify existing ESP SAs. The hosts may rekey, i.e., force the generation of new keying material using the Diffie-Hellman procedure. The initial setup of ESP SAs between the hosts is done during the base exchange, and the message exchange is protected using methods provided by the base exchange. Changes in connection parameters basically mean that the old ESP SA is removed and a new one is generated once the UPDATE message exchange has been completed. The message exchange is protected using the HIP association keys. Both HMAC and signing of packets are used.

9. IANA Considerations

The following changes to the "Host Identity Protocol (HIP) Parameters" registries have been made. In all cases, the changes updated the reference from [RFC5202] to this specification.

This document defines two Parameter Types and two NOTIFY Message Types for the Host Identity Protocol [RFC7401].

The parameters and their type numbers are defined in Sections 5.1.1 and 5.1.2, and they have been added to the "Parameter Types" namespace created by [RFC7401]. No new action regarding these values is required by this specification, other than updating the reference from [RFC5202] to this specification.

The new NOTIFICATION error types and their values are defined in Section 5.1.3, and they have been added to the "Notify Message Types" namespace created by [RFC7401]. No new action regarding these values is required by this specification, other than updating the reference from [RFC5202] to this specification.

Section 5.1.2 of this document defines values for "ESP Transform Suite IDs", which are registered in a new IANA registry, with an "IETF Review" registration procedure [RFC5226] for new values.

10. References

10.1. Normative References

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

[RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, November 1998, <http://www.rfc-editor.org/info/rfc2404>.

[RFC2410] Glenn, R. and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998, <http://www.rfc-editor.org/info/rfc2410>.

[RFC3602] Frankel, S., Glenn, R., and S. Kelly, "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, September 2003, <http://www.rfc-editor.org/info/rfc3602>.

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

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

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005, <http://www.rfc-editor.org/info/rfc4309>.

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, June 2006, <http://www.rfc-editor.org/info/rfc4493>.

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

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

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, May 2007, <http://www.rfc-editor.org/info/rfc4868>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, April 2015, <http://www.rfc-editor.org/info/rfc7401>.

10.2. Informative References

[HIP-ARCH] Moskowitz, R., Ed., and M. Komu, "Host Identity Protocol Architecture", Work in Progress, draft-ietf-hip-rfc4423-bis-09, October 2014.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981, <http://www.rfc-editor.org/info/rfc791>.

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

[RFC5202] Jokela, P., Moskowitz, R., and P. Nikander, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 5202, April 2008, <http://www.rfc-editor.org/info/rfc5202>.

[RFC5206] Nikander, P., Henderson, T., Vogt, C., and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, April 2008, <http://www.rfc-editor.org/info/rfc5206>.

[RFC5207] Stiemerling, M., Quittek, J., and L. Eggert, "NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication", RFC 5207, April 2008, <http://www.rfc-editor.org/info/rfc5207>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5770] Komu, M., Henderson, T., Tschofenig, H., Melen, J., and A. Keranen, "Basic Host Identity Protocol (HIP) Extensions for Traversal of Network Address Translators", RFC 5770, April 2010, <http://www.rfc-editor.org/info/rfc5770>.

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

Appendix A. A Note on Implementation Options

It is possible to implement this specification in multiple different ways. As noted above, one possible way of implementing this is to rewrite IP headers below IPsec. In such an implementation, IPsec is used as if it was processing IPv6 transport mode packets, with the IPv6 header containing HITs instead of IP addresses in the source and destination address fields. In outgoing packets, after IPsec processing, the HITs are replaced with actual IP addresses, based on the HITs and the SPI. In incoming packets, before IPsec processing, the IP addresses are replaced with HITs, based on the SPI in the incoming packet. In such an implementation, all IPsec policies are based on HITs and the upper layers only see packets with HITs in the place of IP addresses. Consequently, support of HIP does not conflict with other uses of IPsec as long as the SPI spaces are kept separate. Appendix B describes another way to implement this specification.

Appendix B. Bound End-to-End Tunnel Mode for ESP

This section introduces an alternative way of implementing the necessary functions for HIP ESP transport. Compared to the option of implementing the required address rewrites outside of IPsec, BEET has one implementation-level benefit. In a BEET-mode-based implementation, the address-rewriting information is kept in one place, at the SAD. On the other hand, when address rewriting is implemented separately, the implementation MUST make sure that the information in the SAD and the information in the separate address-rewriting database are kept in synchrony. As a result, the BEET-mode-based way of implementing this specification is RECOMMENDED over the separate implementation, as it binds the identities, encryption, and locators tightly together. It should be noted that implementing BEET mode doesn’t require that corresponding hosts implement it, as the behavior is only visible internally in a host.

BEET mode is a combination of IPsec tunnel and transport modes, and it provides some of the features from both. HIP uses HITs as the "inner" addresses and IP addresses as "outer" addresses, like IP addresses are used in tunnel mode. Instead of tunneling packets between hosts, a conversion between inner and outer addresses is made at end hosts, and the inner address is never sent on the wire after the initial HIP negotiation. BEET provides IPsec transport mode syntax (no inner headers) with limited tunnel mode semantics (fixed logical inner addresses -- the HITs -- and changeable outer IP addresses).

B.1. Protocol Definition

In this section, we define the exact protocol formats and operations.

B.1.1. Changes to Security Association Data Structures

A BEET mode Security Association contains the same data as a regular tunnel mode Security Association, with the exception that the inner selectors must be single addresses and cannot be subnets. The data includes the following:

- A pair of inner IP addresses.
- A pair of outer IP addresses.
- Cryptographic keys and other data as defined in Section 4.4.2 of RFC 4301 [RFC4301].

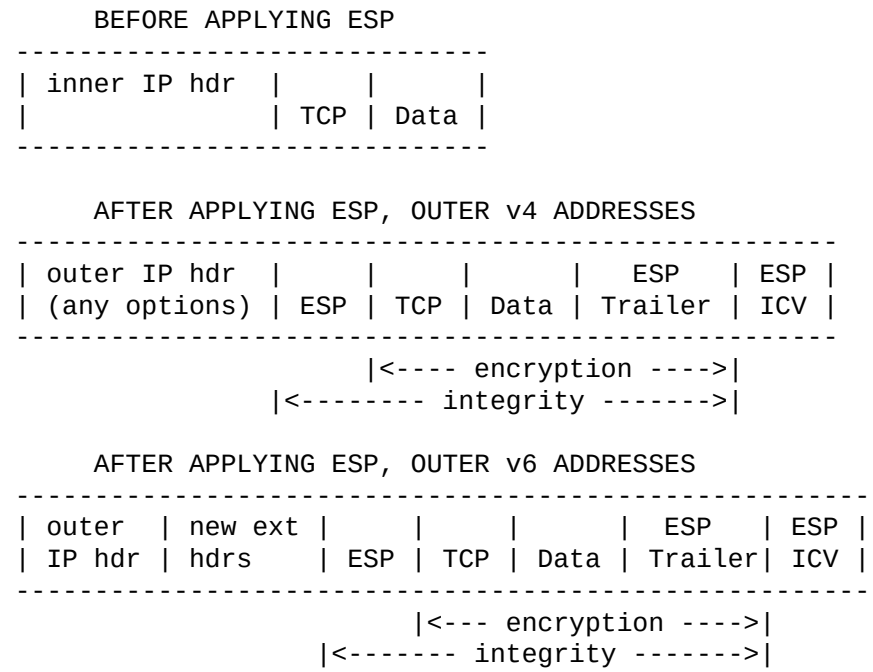
A conforming implementation MAY store the data in a way similar to a regular tunnel mode Security Association.

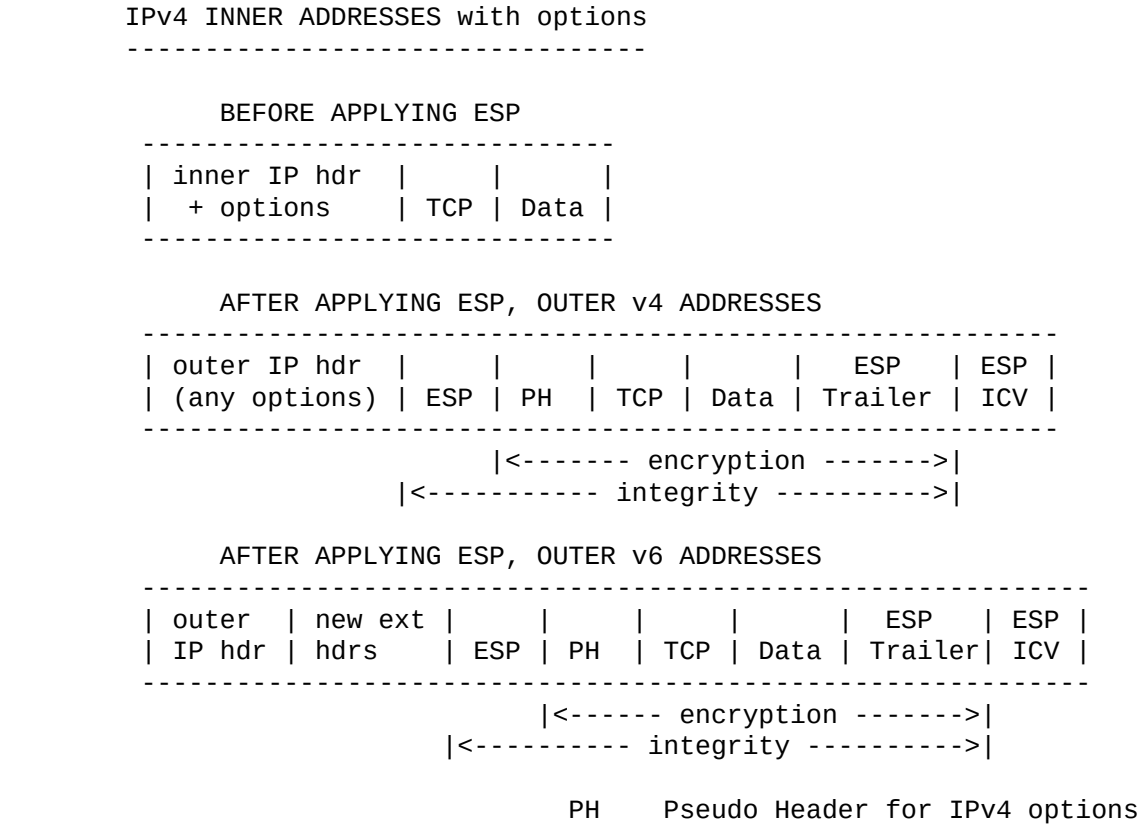
Note that in a conforming implementation the inner and outer addresses MAY belong to different address families. All implementations that support both IPv4 and IPv6 SHOULD support both IPv4-over-IPv6 and IPv6-over-IPv4 tunneling.

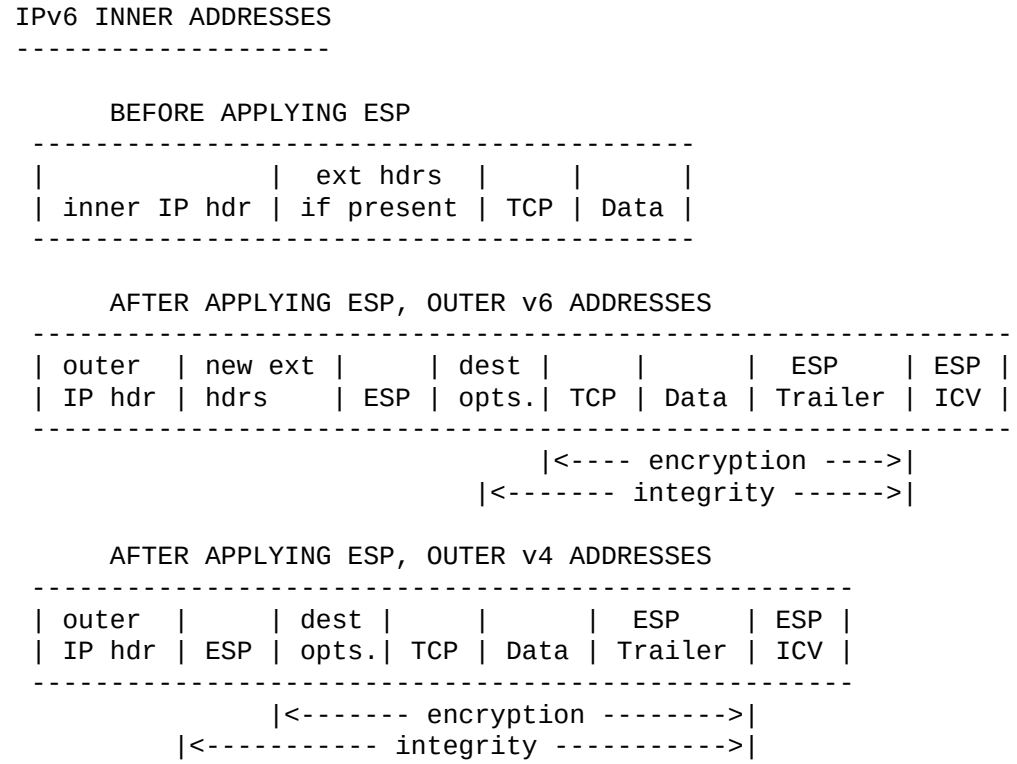
B.1.2. Packet Format

The wire packet format is identical to the ESP transport mode wire format as defined in Section 3.1.1 of [RFC4303]. However, the resulting packet contains outer IP addresses instead of the inner IP addresses received from the upper layer. The construction of the outer headers is defined in Section 5.1.2 of RFC 4301 [RFC4301]. The following diagram illustrates ESP BEET mode positioning for typical IPv4 and IPv6 packets.

IPv4 INNER ADDRESSES




IPv4 INNER ADDRESSES with options
BEFORE APPLYING ESP
inner IP hdr + options
TCP
Data
AFTER APPLYING ESP, OUTER v4 ADDRESSES
outer IP hdr (any options)
ESP
PH
TCP
Data
ESP Trailer
ESP ICV
encryption
integrity
AFTER APPLYING ESP, OUTER v6 ADDRESSES
outer IP hdr
new ext hdrs
ESP
PH
TCP
Data
ESP Trailer
ESP ICV
encryption
integrity
PH Pseudo Header for IPv4 options



B.1.3. Cryptographic Processing

The outgoing packets MUST be protected exactly as in ESP transport mode [RFC4303]. That is, the upper-layer protocol packet is wrapped into an ESP header, encrypted, and authenticated exactly as if regular transport mode was used. The resulting ESP packet is subject to IP header processing as defined in Appendices B.1.4 and B.1.5. The incoming ESP protected messages are verified and decrypted exactly as if regular transport mode was used. The resulting cleartext packet is subject to IP header processing as defined in Appendices B.1.4 and B.1.6.

B.1.4. IP Header Processing

The biggest difference between BEET mode and the other two modes is in IP header processing. In the regular transport mode, the IP header is kept intact. In the regular tunnel mode, an outer IP header is created on output and discarded on input. In BEET mode, the IP header is replaced with another one on both input and output.

On the BEET mode output side, the IP header processing MUST first ensure that the IP addresses in the original IP header contain the inner addresses as specified in the SA. This MAY be ensured by proper policy processing, and it is possible that no checks are needed at the time of SA processing. Once the IP header has been verified to contain the right IP inner addresses, it is discarded. A new IP header is created, using the fields of the discarded inner header (except the IP addresses) to populate the fields of the new outer header. The IP addresses in the new header MUST be the outer tunnel addresses.

On the input side, the received IP header is simply discarded. Since the packet has been decrypted and verified, no further checks are necessary. A new IP header corresponding to a BEET mode inner header is created, using the fields of the discarded outer header (except the IP addresses) to populate the fields of the new inner header. The IP addresses in the new header MUST be the inner addresses.

As the outer header fields are used as a hint for creating the inner header, it must be noted that the inner header differs as compared to a tunnel mode inner header. In BEET mode, the inner header will have the Time to Live (TTL), Don't Fragment (DF) bit, and other option values from the outer header. The TTL, DF bit, and other option values of the inner header MUST be processed by the stack.

B.1.5. Handling of Outgoing Packets

The outgoing BEET mode packets are processed as follows:

1. The system MUST verify that the IP header contains the inner source and destination addresses, exactly as defined in the SA. This verification MAY be explicit, or it MAY be implicit, for example, as a result of prior policy processing. Note that in some implementations there may be no real IP header at this time but the source and destination addresses may be carried out of band. If the source address is still unassigned, it SHOULD be ensured that the designated inner source address would be selected at a later stage.

2. The IP payload (the contents of the packet beyond the IP header) is wrapped into an ESP header as defined in Section 3.3 of [RFC4303].

3. A new IP header is constructed, replacing the original one. The new IP header MUST contain the outer source and destination addresses, as defined in the SA. Note that in some implementations there may be no real IP header at this time but the source and destination addresses may be carried out of band.

In the case where the source address must be left unassigned, it SHOULD be ensured that the right source address is selected at a later stage. Other than the addresses, it is RECOMMENDED that the new IP header copies the fields from the original IP header.

4. If there are any IPv4 options in the original packet, it is RECOMMENDED that they are discarded. If the inner header contains one or more options that need to be transported between the tunnel endpoints, the sender MUST encapsulate the options as defined in Appendix B.1.7.

Instead of literally discarding the IP header and constructing a new one, a conforming implementation MAY simply replace the addresses in an existing header. However, if the RECOMMENDED feature of allowing the inner and outer addresses from different address families is used, this simple strategy does not work.

B.1.6. Handling of Incoming Packets

The incoming BEET mode packets are processed as follows:

1. The system MUST verify and decrypt the incoming packet successfully, as defined in Section 3.4 of [RFC4303]. If the verification or decryption fails, the packet MUST be discarded.

2. The original IP header is simply discarded, without any checks. Since the ESP verification succeeded, the packet can be safely assumed to have arrived from the right sender.

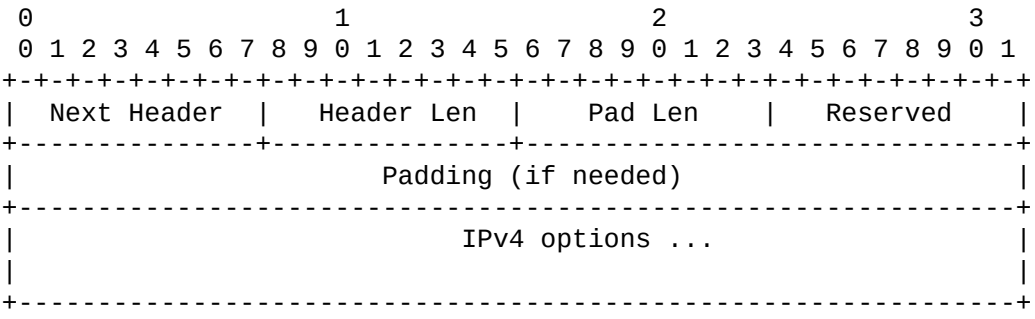
3. A new IP header is constructed, replacing the original one. The new IP header MUST contain the inner source and destination addresses, as defined in the SA. If the sender has set the ESP Next Header field to 94 and included the pseudo header as described in Appendix B.1.7, the receiver MUST include the options after the constructed IP header. Note that in some implementations the real IP header may have already been discarded and the source and destination addresses are carried out of band. In such a case, the out-of-band addresses MUST be the inner addresses. Other than the addresses, it is RECOMMENDED that the new IP header copies the fields from the original IP header.

Instead of literally discarding the IP header and constructing a new one, a conforming implementation MAY simply replace the addresses in an existing header. However, if the RECOMMENDED feature of allowing the inner and outer addresses from different address families is used, this simple strategy does not work.

B.1.7. Handling of IPv4 Options

In BEET mode, if IPv4 options are transported inside the tunnel, the sender MUST include a pseudo header after the ESP header. The pseudo header indicates that IPv4 options from the original packet are to be applied to the packet on the input side.

The sender MUST set the Next Header field in the ESP header to 94. The resulting pseudo header, including the IPv4 options, MUST be padded to an 8-octet boundary. The padding length is expressed in octets; valid padding lengths are 0 or 4 octets, as the original IPv4 options are already padded to a 4-octet boundary. The padding MUST be filled with No Operation (NOP) options as defined in Section 3.1 ("Internet Header Format") of [RFC0791] ("Internet Protocol"). The padding is added in front of the original options to ensure that the receiver is able to reconstruct the original IPv4 datagram. The Header Length field contains the length of the IPv4 options, and padding in 8-octet units.



Next Header	identifies the data following this header.
Length in octets	8-bit unsigned integer. Length of the pseudo header in 8-octet units, not including the first 8 octets.

The receiver MUST remove this pseudo header and padding as a part of BEET processing, in order to reconstruct the original IPv4 datagram. The IPv4 options included in the pseudo header MUST be added after the reconstructed IPv4 (inner) header on the receiving side.

Acknowledgments

This document was separated from the base Host Identity Protocol specification in the beginning of 2005. Since then, a number of people have contributed to the text by providing comments and modification proposals. The list of people includes Tom Henderson, Jeff Ahrenholz, Jan Melen, Jukka Ylitalo, and Miika Komu. Especially, the authors want to thank Pekka Nikander for his invaluable contributions to the document since the first draft version. The authors also want to thank Charlie Kaufman for reviewing the document with his eye on the usage of crypto algorithms.

Due to the history of this document, most of the ideas are inherited from the base Host Identity Protocol specification. Thus, the list of people in the Acknowledgments section of that specification is also valid for this document. Many people have given valuable feedback, and our apologies to anyone whose name is missing.

Authors' Addresses

Petri Jokela
Ericsson Research NomadicLab
JORVAS FIN-02420
Finland

Phone: +358 9 299 1
EMail: petri.jokela@nomadiclab.com

Robert Moskowitz
HTT Consulting
Oak Park, MI
United States

EMail: rgm@labs.htt-consult.com

Jan Melen
Ericsson Research NomadicLab
JORVAS FIN-02420
Finland

Phone: +358 9 299 1
EMail: jan.melen@nomadiclab.com