An Introduction to the Helion IPsec ESP Engine

Overview

IPsec (short for IP security) is defined by a set of protocols which were developed by the Internet Engineering Task Force (IETF) to allow secure communication of IP datagrams over an untrusted network such as the Internet to create a virtual private network (VPN). The latest IPsec standards are defined in a series of Request For Comments (RFC 4301 to 4309) published in December 2005. The set of protocols described therein provide the access control, encryption, authentication, data integrity and key exchange mechanisms required to ensure data security between two communicating network devices at the IP layer.

The IPsec protocol most commonly employed for securing traffic traversing an IP network is Encapsulating Security Payload (ESP) which provides confidentiality, data origin authentication, connectionless integrity, an anti-replay service, and limited traffic flow confidentiality for IPv4 and IPv6 traffic. ESP may be used to provide security either between a pair of network hosts, between a pair of security gateways, or between a security gateway and a network host.

It is the ESP protocol of the IPsec standards we concentrate on in this document, and in particular how the Helion ESP Engine may be utilised to perform hardware acceleration of the key cryptographic algorithms and packet processing required to implement an efficient, high performance IPsec ESP compliant device in FPGA or ASIC. A full description of the ESP protocol is beyond the scope of this document and reference should be made to RFC 4303 for more detailed information. We start by introducing some basic ESP concepts.

An Introduction to ESP

The Helion ESP Engine is designed to provide hardware acceleration of the core ESP encryption and integrity algorithms used to provide ESP confidentiality and integrity services. In addition to greatly increasing the data throughput of the underlying encryption and integrity algorithms, offloading ESP packet processing into hardware allows the system CPU to concentrate on the higher level IPsec processing tasks which due to their complexity are more suited to software e.g. the Internet Key Exchange (IKE) which is used to establish and maintain secure connections, as well as perform authentication and key exchanges with other IPsec endpoints.

In essence, IPsec provides a secure boundary which acts as a limited functionality firewall between a protected environment such as a company LAN, and an unprotected environment such as the Internet. IPsec ensures that traffic crossing the boundary between the two environments is subject to access controls which determine whether a particular IP packet should be allowed through unimpeded, have a security service applied using ESP, or be discarded.

Three security service combinations are defined for use with ESP which provide either confidentiality only, integrity only, or both confidentiality and integrity protection for IP traffic. All ESP implementations must implement the latter two security services but all three are supported by the Helion ESP Engine. Although they can be offered independently, it is common for both confidentiality and integrity to be used together. It should be noted that although RFC 4303 states that the confidentiality only service may be supported, its use is currently under review in light of a fundamental weakness which has been highlighted by recent active attacks which would otherwise not be possible with the use of an integrity service. An anti-replay service may also be employed by a receiving IPsec endpoint where the integrity service is in use.

Security Association (SA)

The Security Association (SA) is a concept which is fundamental to IPsec and it is necessary for all implementations of ESP to support them. An SA may be considered a one-way connection between two communicating IPsec endpoints that has an associated ESP security service (including the encryption and integrity algorithms to be used) assigned to it in order to protect the IP traffic carried on it. Since an SA is unidirectional, in a typical situation where the traffic between two endpoints is bi-directional, a pair of SAs is required i.e. one to secure the traffic in each direction.

Internet Key Exchange (IKE)

The Internet Key Exchange (IKE) protocol is responsible for mutual authentication, the establishment of cryptographic keys, and the selection of cryptographic algorithms to be used for securing IP packets travelling between two IPsec endpoints. It is also responsible for negotiating the establishment of new SA connections and their ongoing management, updating and deletion on expiry.

IKE version 2 (RFC 4306) first establishes an IKE SA between the two communicating IPsec endpoints. This IKE SA defines a secure channel between the endpoints which may be used for subsequent secure message exchanges. It is used to efficiently create and maintain further SAs as required for use by ESP, and to propose a set of cryptographic algorithms which may be used to protect the IP traffic carried on them. The IKE SA is used by the endpoints to exchange IKE management messages using a simple request/response protocol.

Typically there are four messages required (two in either direction) to establish the IKE SA and the first ESP SA. The first exchange of an IKE session (referred to as IKE_SA_INIT) negotiates cryptographic algorithms for the IKE SA as well as exchanging cryptographic nonce and public key values. The second exchange (referred to as IKE_AUTH) is the first to be protected using the security parameters generated by the first exchange. This exchanges details of each endpoint's identity and certificate for authentication purposes as well as establishing the first SA for use by ESP. Further message exchanges using the IKE SA may then take place to either create further ESP SAs (referred to as CREATE_CHILD_SA), to report error conditions at either endpoint, to delete an SA which has expired, as well as any other general housekeeping (referred to as INFORMATIONAL).

A more detailed description of the IKE protocol is beyond the scope of this introduction and reference should be made to RFC 4306 for further information.

ESP Modes

ESP supports two modes of operation, each of which provides security at different layers of the network stack. Transport mode as its name implies, may be used to add security at the transport layer e.g. TCP or UDP. Tunnel mode is used to provide security at the network layer by securing IP packets. For transport mode the ESP packet header is inserted after the original IP packet header but before the transport packet header, whilst for tunnel mode the ESP header is inserted before the original IP packet header. Figure 1. illustrates the modes using a TCP next layer protocol within IPv4 packets as an example.

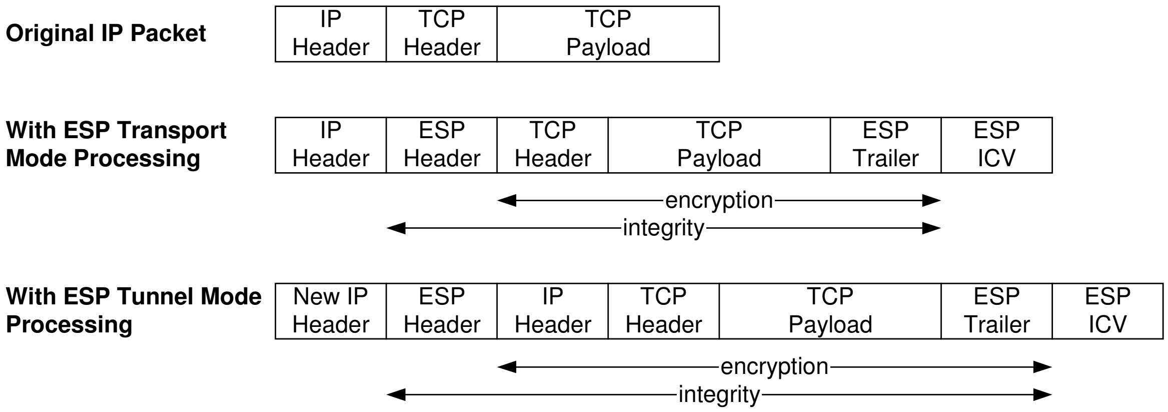

Figure 1. ESP Transport and Tunnel Mode Processing

In transport mode only the TCP packet is encapsulated into the ESP packet. The original IP header is re-used with the packet length field changed to reflect the increase caused by ESP processing, and the protocol field is changed to show that an ESP header is next rather than the original TCP header. RFC 4301 decrees that transport mode must be supported by all IPsec hosts and may optionally be supported by security gateways. Transport mode is normally used for providing end-to-end security for host-to-host communications over an IP network.

In tunnel mode the whole original IP packet is encapsulated and a new IP header is used to transmit the ESP packet onto the network. The new IP header will reflect the endpoints of the “tunnel” in the new source and destination address fields, while the ultimate source and destination are hidden when the confidentiality service is used. RFC 4301 decrees that tunnel mode must be supported by all IPsec hosts and security gateways. Tunnel mode is normally used to provide security for communications between two security gateways (e.g. network routers or firewalls) over an insecure IP network. The Helion ESP Engine has been designed to perform ESP packet processing for both transport and tunnel modes.

In transport mode the ESP Engine accepts as input a transport layer (TCP, UDP) packet and outputs an ESP packet in the encrypt direction. The higher level application is responsible for amending and prepending the outer IP header prior to transmission. In the decrypt direction the ESP Engine accepts as input an ESP packet and outputs the original transport layer packet. The higher level application is responsible for removing the outer IP header prior to input to the ESP Engine.

Similarly, for tunnel mode the ESP Engine accepts as input an IP packet and outputs an ESP packet in the encrypt direction. The higher level application is responsible for creating and prepending the outer IP header prior to transmission. In the decrypt direction the ESP Engine accepts as input an ESP packet and outputs the original IP packet. The higher level application is responsible for removing the outer IP packet header prior to input to the ESP Engine.

ESP Packet Format

Figure 2. shows the format of an ESP packet

1. before an IP header is attached to an outgoing packet from the Helion ESP Engine in the encrypt direction, or
2. after an IP header is removed from an incoming packet prior to input to the Helion ESP Engine in the decrypt direction.

It also illustrates which parts of the ESP packet are protected by encryption and integrity algorithms when the confidentiality and integrity services are enabled. What follows is a brief description of the fields which constitute the ESP packet.

- *Security Parameters Index (SPI)*

The SPI is an arbitrary 32-bit value which is used by the receiving endpoint to determine the SA to be used for the incoming packet. The endpoint may also use the IP header destination and source address as part of the SA lookup used to retrieve the security parameters required to process the ESP packet.

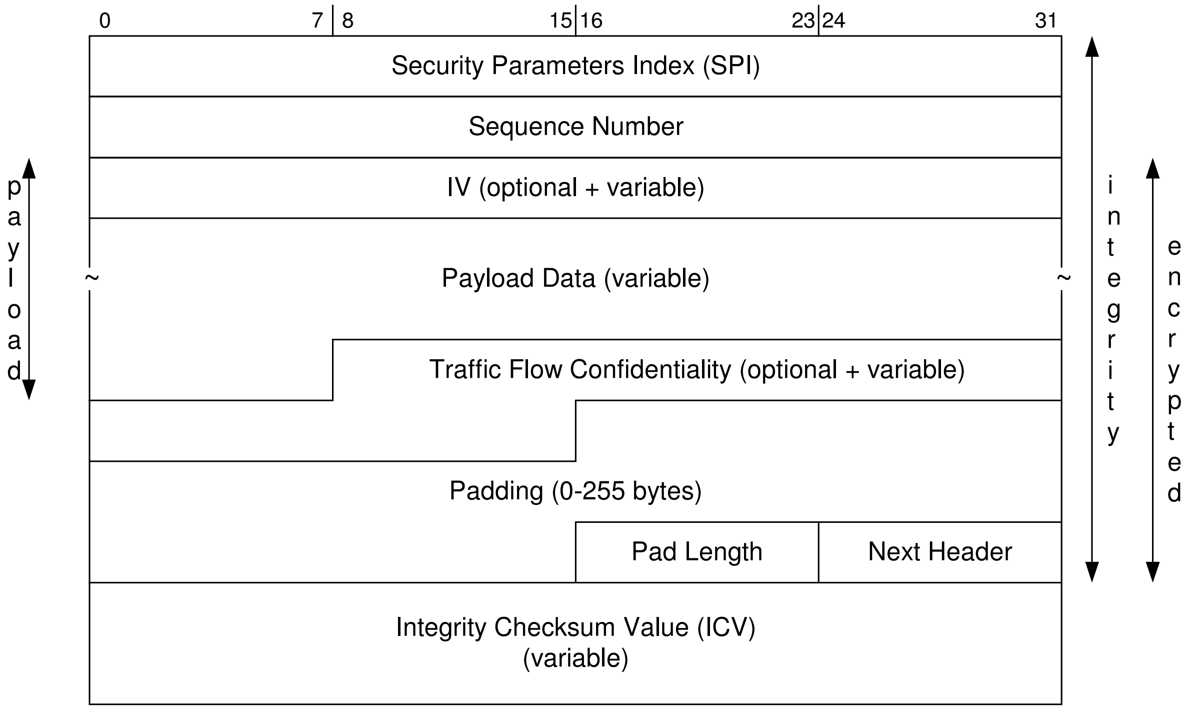

Figure 2. ESP Packet Format to RFC 4303

- *Sequence Number*

This is a 32-bit counter value which is increment by the sending endpoint for each packet sent on an SA. Where an integrity service is used this may optionally be used by the receiver to enforce an anti-replay service. On creation of an SA this value is reset to 0 by both endpoints. However, to ensure security is maintained the sender must never allow this counter value to wrap and a new SA should be established prior to the 2^{32}nd packet being transmitted to ensure continued secure communication between the endpoints.

- *Extended (64-bit) Sequence Number (ESN)*

The Sequence Number may be extended by a further 32 bits to provide a 64-bit Sequence Number for any SA which protects high-throughput IP network traffic. Only the least significant 32 bits of the ESN are transmitted in the ESP packet as described above. The most significant 32 bits are maintained as part of the sequence number counter associated with the SA by both endpoints. Although the most significant bits are not transmitted on the network they are included as part of the calculation of the Integrity Checksum Value (ICV).

NOTE: For IKEv1 the use of the ESN service is negotiated as part of the messaging protocol used during creation of an SA for ESP. For IKEv2, support for ESN is assumed and use of 32-bit Sequence Numbers must be explicitly negotiated.

The Helion ESP Engine fully supports handling of Extended Sequence Numbers.

- *IV and Payload Data*

This variable length field contains the original packet data of the type indicated by the IANA number contained in the Next Header field. Where a confidentiality service is in use, and the payload data is encrypted using an algorithm which requires synchronisation (e.g. AES-CBC) to ensure correct decryption at the receiver, the first bytes of the payload may contain an Initialisation Vector (IV). It should be noted that even though the IV is included as part of the Payload Data field it is not normally encrypted as such. When confidentiality is not used the IV is not present.

- *Traffic Flow Confidentiality Padding (TFC)*

This optional variable length field may be used to disguise the characteristics of IPsec traffic passing between two endpoints by inserting arbitrary padding bytes between the Payload Data and Padding fields. TFC padding can only be used where the next layer protocol header contains an explicit packet length which allows the receiver to discard it. To ensure backward compatibility with older ESP implementations which may not support TFC, the SA management protocol must negotiate the use of the TFC service.

The Helion ESP Engine fully supports generation of arbitrary length TFC padding to section 2.7 of RFC 4303.

- *Padding*

Padding of the preceding payload data (with or without the addition of TFC padding) is required for two purposes. When encryption is used based on a block cipher mode, additional padding may be required to ensure that the ESP packet length is a multiple of the cipher block size prior to encryption. Even when no confidentiality service is associated with the SA and so no padding is required for encryption, it is still necessary to ensure that the ICV is aligned to a 4-byte boundary. For this reason all ESP implementations must support the generation and consumption of padding.

The Helion ESP Engine generates and consumes the default incrementing byte count padding scheme specified in RFC 4303. It also performs checking of the padding for received ESP packets in accordance with section 2.4 of RFC 4303.

- *Pad Length*

This 8-bit field indicates the number of ESP padding bytes in the preceding Padding field. A value of 0 indicates no Padding field is present.

- *Next Header*

This 8-bit field identifies the next header type contained in the ESP packet payload data as defined by the set of protocol numbers allocated by IANA.

♦ *Integrity Check Value (ICV)*

This variable length field is only present when an integrity service is selected. The exact length of this field depends on the integrity algorithm associated with the SA. The ICV is calculated and appended to the ESP packet in the encrypt direction. In the decrypt direction the ICV is compared with the value calculated from processing the received ESP packet for authentication and data integrity.

The Helion ESP Engine handles all ICV generation and checking.

An Example Helion ESP Engine Application

The block diagram in Figure 3. shows an example of a simple IPsec security gateway where a Helion ESP Engine performs hardware acceleration of the ESP processing required for packets crossing the IPsec boundary in either direction. Higher throughput IPsec implementations may utilise one or more ESP Engines to provide ESP hardware acceleration in each direction.

The CPU is at the heart of the IPsec boundary, providing the red/black separation between the protected local area IP network and an unprotected public IP network such as the Internet. The CPU is responsible for implementing the IPsec Security Policy Database (SPD) which must be present in all IPsec endpoints. The SPD determines how all inbound and outbound IP traffic passing across the IPsec boundary is to be handled. RFC4301 defines the three possible processing choices which must be enforced by the SPD as *Discard*, *Bypass*, and *Protect*. *Discard* is employed for IP packets which are not allowed to traverse the IPsec boundary; *Bypass* is employed for IP packets which are allowed to cross the boundary without any IPsec protection; and *Protect* is employed for IP packets that must have ESP protection applied, in which case the SPD must specify a related SA containing details of the associated ESP security service such as encryption and integrity algorithms, cryptographic keys and IVs, anti-replay counters etc.

By enforcing the IPsec Security Policy the CPU acts as a limited functionality IP layer firewall between the local area network and the Internet. The CPU only dispatches packets requiring ESP processing to the ESP Engine - all other packets are either discarded at the boundary, or bypass IPsec and so do not require processing. Inbound IPsec packets from the unprotected network should only be dispatched to the ESP Engine once it has been established that a valid SA exists and that any additional receiver checks e.g. anti-replay have been passed.

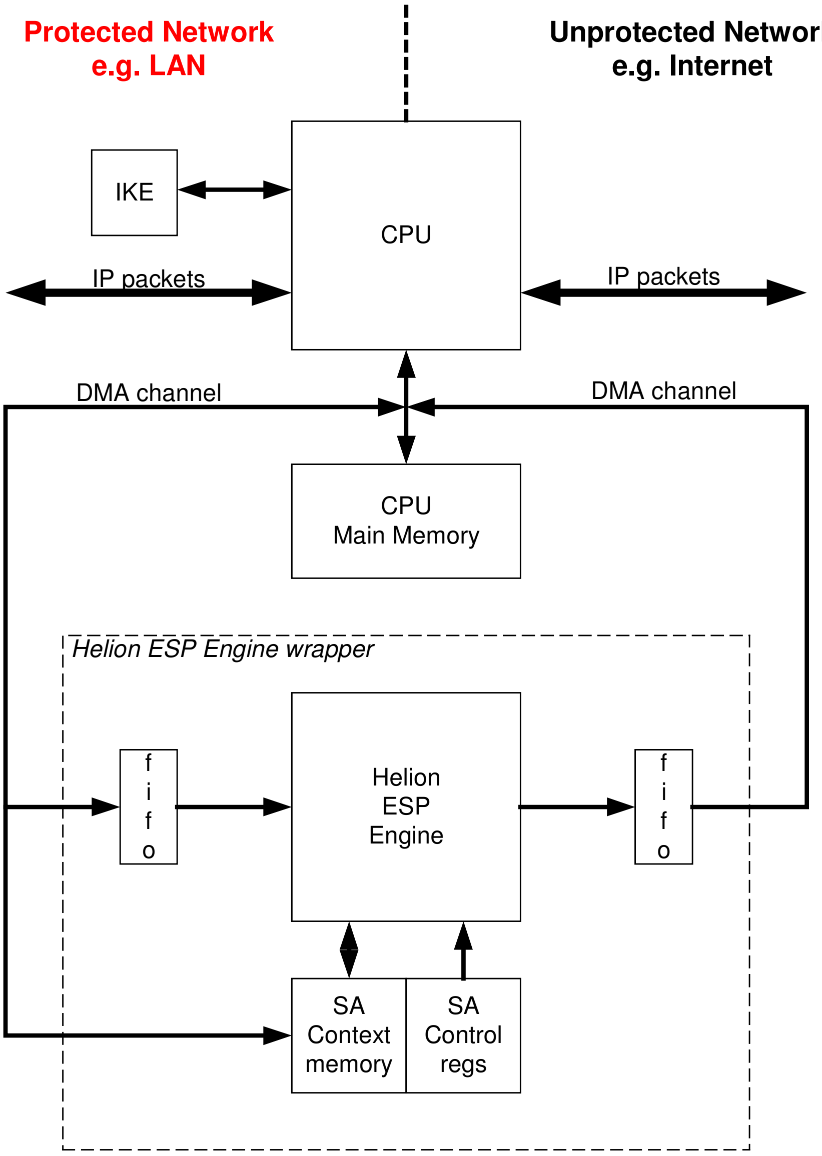

Figure 3. Example Helion ESP Engine Application

Figure 3. shows the Helion ESP Engine as a component within an application specific wrapper which provides packet buffering and interfaces to an external data bus used to transfer SA context and packet data; in this example a CPU data bus using DMA. The wrapper provides FIFO storage to buffer packet data at both the input and output of the ESP Engine which act as a sink and source for CPU DMA data respectively. It also provides memory and/or register storage for the SA context and control required to process the packet. The SA context is transferred by the CPU as a result of the SA lookup operation performed for the incoming packet e.g. SPI, Sequence Number, encryption and integrity keys/IVs, packet direction, security service type etc.

IKE may be implemented entirely by the CPU for IPsec applications where only a few endpoints need be maintained and as a result the IKE protocol presents a manageable overhead to the CPU software. However, for high throughput applications required to handle communications with many IPsec endpoints concurrently this overhead quickly becomes burdensome, and IKE processing will benefit greatly from the use of hardware acceleration. In particular, any highly computationally intensive operations such as those required by the Diffie-Hellman public key cryptography algorithm will greatly increase performance, as may accelerating the encryption and integrity algorithms required to afford protection to the IKE Encrypted Payloads transferred on IKE_SA. Helion provides a range of solutions, including 1024 and 2048-bit Diffie-Hellman IP cores for acceleration of key exchange protocols, as well a number of encryption and integrity cores which are suitable for acceleration of IKE payload security.

Summary

The Helion ESP Engine provides an ideal solution for the acceleration of ESP packet processing for implementing high throughput IPsec endpoints in FPGA or ASIC. Its modular architecture provides full support for all ESP encryption and integrity algorithms specified by RFC 4305, as well as the proposed combined mode algorithms such as AES-CCM and AES-GCM which may become future requirements for IPsec implementations. The Helion ESP Engine can be configured to support as few, or as many, encryption and integrity algorithms as required for a particular IPsec application. Typically only a subset of these algorithms will be implemented, but as a minimum at least one encryption algorithm and one integrity algorithm must be supported to allow the full range of security service types to be offered and to ensure IPsec compliance.

Version : V1.0.0

http://www.heliontech.com

Disclaimer. This publication is provided “as is” without warranty of any kind either expressed or implied. Helion Technology Limited believes that the information published in this document is true and accurate, but assumes no responsibility for errors or omissions. Helion Technology Limited shall not be liable in the event of incidental or consequential damages arising from the use of information supplied in this publication.