WHITE PAPER

Intel® IPsec Acceleration

Author:

Stephen Doyle, Georgii Tkachuk

Contents

1 Introduction

IP Security (IPsec) is a network layer security protocol that is widely deployed and used in many network applications including Enterprise VPNs, wireless access networks and securing network virtualization tunnels. When IPsec is enabled on a network, the network nodes at each end of an IPsec tunnel must perform additional processing, for providing encryption and integrity protection, on IPsec packets. As network interface rates increase to 100Gbps and beyond, there is a desire and challenge to minimise the processing overheads associated with IPsec.

There are a variety of efficient IPsec implementation options available, ranging from a pure SW implementation using optimized CPU instructions to the use of hardware for offloading various aspects of IPsec packet processing.

This white paper summarizes some of these implementation options, provides some data on the expected performance achievable and provides guidance on when to use each option.

2 Outline

- Section 3 provides some brief background on IPsec and the need for acceleration of IPsec processing.
- Section 4 describes two of the IPsec acceleration options available.
- Section 5 compares different aspects of the IPsec implementation options.
- Section 6 presents some performance data for each implementation option.

3 Background

3.1 IPsec Overview

IPsec is a protocol suite that authenticates and encrypts IP packets. IPsec is described at a high level in RFC4301 "Security Architecture for the Internet Protocol" and in an additional set of RFCs which describe IPsec protocols and crypto algorithm usage for IPsec. The IPsec Encapsulating

Security Protocol (ESP) is the focus of this whitepaper and is described in detail in RFC4303.

3.1.1 Security Policy & Security Association

The decision to apply IPsec security to a specific network flow is specified by a security policy. This policy is set on a host for a particular network interface. The policy is stored on the host in a local Security Policy Database (SPD).

The SPD contains a set of rules with matching policies to apply when specific network flows match the rule. The rules are patterns matched against the headers of the network flow. The policies specify the cryptographic algorithms and the protocols to use to protect the matching network flow.

An example rule may specify that all network flows to/from a specific IP subnet should be encrypted and authenticated using the AES-GCM[1] algorithm and the ESP protocol.

When the policy requires IPsec for a particular network flow, the system creates a new entry, called a Security Association or SA, for that flow in a database called the Security Association Database (SADB). The SA contains the parameters for the required operations and protocols for that network flow as well as any required key material for the cryptographic algorithms. The SA for a specific flow is identified by an index to the SADB called the Security Parameter Index or SPI. The SPI is transmitted in the IPsec header with every packet to inform the receiver which security operations should be performed on that packet.

For transmit traffic, the SPD is consulted to determine if an outgoing packet needs to be protected and if so, which SA to use to protect the packet. If no SA is established, an Internet Key Exchange (IKE) is triggered to establish the SA. This scenario can happen in the case where the IPsec stack is configured to establish SAs on demand when the first packet is transmitted on the tunnel, rather than setting up SAs in advance. No packets are transmitted until the SA has been established.

For receive traffic, the SPD is consulted after a packet has been processed as required by the SA. The SPD lookup verifies that incoming packets were correctly protected according to the configured security policy.

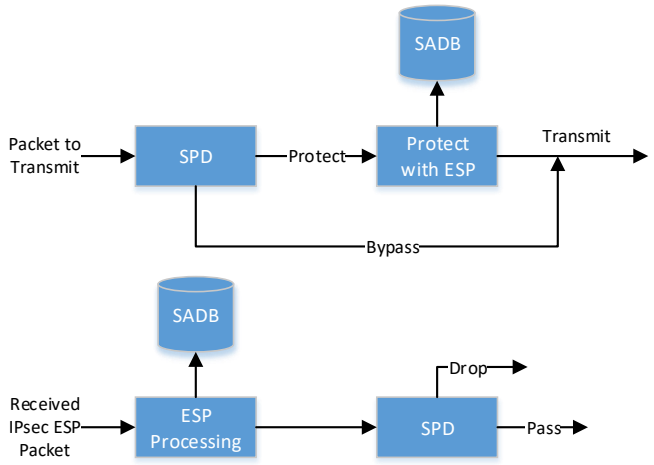


Figure 1 IPsec SADB & SPD

3.1.2 Key Exchange Protocols

The Internet Key Exchange (IKE) protocol offers a means to automatically negotiate security parameters and derive suitable keying material which is subsequently used to protect the data path traffic. IKE also manages the process of recreating or refreshing keys to ensure data confidentiality between peers. The basic operation of IKE can be broken down into two phases: IKE phase 1 and IKE phase 2.

IKE phase 1 is used to negotiate the parameters and key material required to establish an ISAKMP Security Association (ISAKMP SA). The ISAKMP SA is then used to protect future IKE exchanges and to setup a secure channel for negotiating IPsec SAs in IKE phase 2.

IKE phase 2 is used to negotiate the parameters and key material required to establish two unidirectional IPsec SAs for incoming and outgoing traffic. The IPsec SAs are then used to protect network data path traffic.

3.1.3 IPsec Security Protocols

Encapsulating Security Payload (ESP) is used to provide confidentiality, data origin authentication and integrity for IPv4/IPv6 and to provide protection against replays. It is described in RFC4303.

Authentication Header (AH) is used to provide integrity and data origin authentication for IPv4/v6 and to provide protection against replays. AH does not provide any confidentiality protection. It is described in RFC4302.

Both ESP and AH support two modes of use: transport mode and tunnel mode. In transport mode, protection is provided

[1] The use of AES-GCM for IPsec ESP is described in RFC4106.

for upper layer protocols. In tunnel mode, protection is provided for tunnelled IP packets.

In today's networks, IPsec ESP is the most common IPsec protocol deployed and is the focus of this whitepaper.

3.1.4 Tunnel Establishment

The establishment of an IPsec tunnel can be broken down into 5 main steps:

1. **Tunnel initiation**: IPsec tunnel initiation can be triggered manually or automatically when network traffic is flagged for protection according to the IPsec security policy configured in the IPsec peers. In both cases the IKE process starts for the tunnel.
2. **IKE Phase 1**: Establishes parameters and key material to protect IKE phase 2.
3. **IKE Phase 2**: Establishes parameters and key material to protect the data transfer phase. These parameters and key material form the IPsec SAs.
4. **Data transfer**: Incoming and outgoing network traffic is encapsulated according to the algorithms and parameters provided by the negotiated IPsec SA to provide confidentiality and integrity.
5. **Tunnel termination**: A tunnel is closed when its IPsec SAs terminate through deletion or by timing out. An IPsec SA can time out when a specified number of seconds have elapsed or when a specified number of bytes have passed through the tunnel.

3.1.5 IPsec Cryptographic Algorithm Trends

The IETF provides regular updates on cryptographic algorithm implementation requirements and usage guidance for IPsec. These updates are driven vulnerabilities and advancements in cryptographic algorithms, while also ensuring that deployed IPsec implementations are interoperable. RFC8221 (October 2017) contains the latest guidance on algorithm usage for IPsec. Table 1 shows the changes in IETF guidance on the use of the most commonly used cryptographic algorithms for IPsec ESP.

Table 1 IPsec Algorithm Guidance Trends

Algorithm[2]	RFC4835 *2007 -> 2014*	RFC7321 2014 -> 2017	RFC8221 *2017 ->*
Combined mode algorithms (AEAD)			
AES-GCM	MAY	SHOULD+	MUST
AES-CCM		MAY	SHOULD
ChaCha20-Poly1305			SHOULD
Encryption Algorithms			
AES-CBC	MUST	MUST	MUST
AES-CTR	SHOULD	MAY	MAY
3DES-CBC	MUST-	MAY	SHOULD NOT
Authentication Algorithms			
HMAC-SHA1	MUST	MUST	MUST-
HMAC-SHA256			MUST
HMAC-SHA512			SHOULD
AES-GMAC	MAY	SHOULD+	MAY
HMAC-MD5	MAY		MUST NOT

RFC8221 specifies that authentication must always be used with encryption. It also describes "the fastest and most modern method is to use ESP with a combined mode cipher, such as an AEAD cipher, that handles encryption/decryption and authentication in a single step". The RFC specifies AES-GCM with a MUST status while ChaCha20-Poly1305 and AES-CCM are given a SHOULD status. This status reflects the recommended support for these algorithms in an IPsec implementation.

More traditional algorithms such as AES-CBC for encryption and HMAC-SHA2-256 are also given a MUST status within RFC8221. Implementations of these algorithms tend to be slower than AEAD algorithm implementations because of the need to perform encryption/decryption and authentication in separate steps. Although they are slower, they are given a MUST status to enable implementation interoperability with older deployed IPsec implementations that may not support AEAD algorithms.

Another emerging trend in cryptographic algorithm use for IPsec is the promotion of 256 bit cryptographic keys from a MAY status to a MUST status in RFC8221. While 128 bit keys are still widely used, and also have a MUST status, there is an expected migration towards 256 bit keys in the long term.

3.2 IPsec Offload

IPsec acceleration deals with improving the performance of IPsec processing relative to a baseline implementation where

[2] Key sizes are not explicitly shown. As of RFC8221, both 128 bit and 256 bit keys are a MUST.

all of the IPsec processing is performed by software running on a general purpose CPU.

The focus of the IPsec acceleration discussion in this whitepaper, is the acceleration of IPsec processing, for the ESP protocol, during the data transfer phase on an IPsec tunnel. For each packet, the main processing functions performed for the IPsec ESP protocol, during the data transfer phase are described in Table 2 IPsec Processing Tasks.

Table 2 IPsec Processing Tasks

IPsec ESP Transmit Packet Processing	IPsec ESP Receive Packet Processing
SA Lookup: Find the SA to process the packet	**SA Lookup**: Find the SA to process the packet
Encap: Add IPsec ESP headers and trailers to the packet	**Anti-Replay**: Check that the packet passes the anti-replay test.
Encrypt: Encrypt the ESP payload and generate the ICV field based on the integrity digest generated	**Decrypt**: Decrypt the ESP payload and validate the ICV field from the ESP trailer.
	Update anti-replay state: Anti-replay state can only be updated once both the anti-replay check nad the ICV checks pass.
	Decap: Remove the IPsec header and trailers.

IPsec acceleration can be used to reduce CPU cycles spent doing compute intensive operations. Within the IPsec processing tasks described above, the encryption and decryption tasks are the most compute intensive.

4 IPsec Offload Approaches

This section describes the various options for offloading IPsec processing to hardware.

Table 3 summarizes the IPsec offload approaches and identifies the processing tasks that are offloaded for each approach. Refer to Table 2 for a description of each processing task.

Table 3 Summary of IPsec Offload Approaches

ID	Name	SA Lookup[3]	Encap/ Decap	Encrypt/ Decrypt	Anti-Replay
A	Software	SW	SW	SW	SW
B	Lookaside acceleration of crypto operations	SW	SW	HW	SW

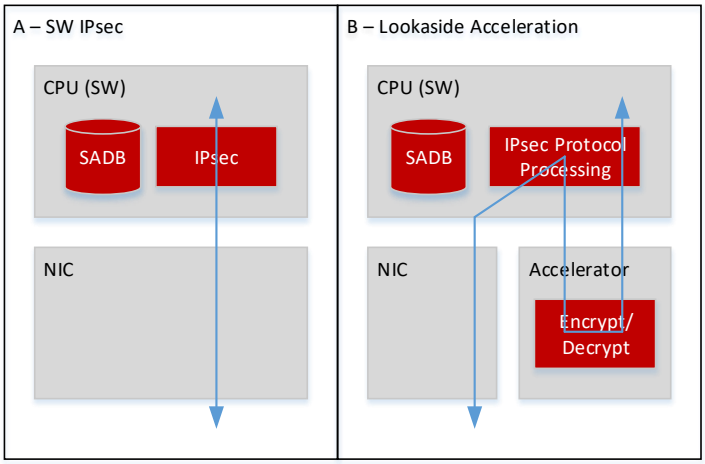


Figure 2 IPsec Offload Approaches

4.1 Software IPsec

IPsec processing in software is a baseline implementation model that other approaches can be compared against. It does not depend upon any specific hardware offload capability and so it is the most versatile approach. Excellent cryptographic performance can be achieved by specifying a cryptographic algorithm, for an IPsec SA, which takes advantage of available CPU capabilities. For example, on x86 CPUs the AES-NI and SHA-NI instructions[4] can be used to efficiently implement AES based and SHA based encryption and authentication algorithms. The IPsec trend towards the use of modern AEAD algorithms, specifically AES-GCM, fits well with x86 AES processing capabilities.

The primary characteristic of a pure software IPsec implementation, including those that are based on AESNI or SHANI, is that the processing cost per packet, measured in CPU cycles, is not fixed and instead varies based on the packet size. This is because each byte of the packet payload must be processed by the CPU as part of the encryption/decryption and authentication tasks. Because of

[3] Refer to Table 2 IPsec Processing Tasks" for a description of the processing performed in these tasks.

[4] Although the AES-NI and SHA-NI instructions are used to access hardware within the CPU, they could be considered as hardware acceleration. However, because they are offered via CPU instructions in the Intel standard instruction set, they are considered part of the software solution for the purposes of this whitepaper.

this, a pure software implementation tends to be most optimal for smaller packet sizes.

4.2 Lookaside Acceleration of IPsec

A cryptographic accelerator device can be used to offload the IPsec encrypt/decrypt operations. For example the Intel® QuickAssist Technology family of accelerators provide symmetric cryptographic acceleration services that can be used to offload both the encryption/decryption and authentication processing tasks for a packet in a single offload operation. These accelerators are widely available, either as part of the CPU chipset, in a System on Chip package, or in PCIe plugin cards.

For the purposes of discussion, the term lookaside acceleration is used to describe the use of such a standalone cryptographic accelerator.

When lookaside acceleration is used to offload encryption/decryption and authentication processing for an IPsec packet, the processing cost per packet, measured in CPU cycles, is constant, regardless of the packet size and regardless of the cryptographic algorithms used. There is however an overhead involved in sending offload requests to a lookaside accelerator and this overhead must be included as part of the per packet processing cost. Because of this "offload overhead", the use of lookaside acceleration for IPsec crypto processing is best suited to the processing of larger packet sizes.

The use of a lookaside IPsec offload has implications for the IPsec stack implementation. Optimal accelerator performance is achieved when there are multiple concurrent offload requests in flight. This allows the accelerator to pipeline and parallelize operations to achieve better performance. It does however mean that the software implementation of the IPsec stack must support the use of asynchronous cryptographic operations. One example of an efficient IPsec stack that supports the use of asynchronous cryptographic operations is the FD.io Vector Packet Processing (VPP) IPsec implementation using DPDK.

5 Comparison of IPsec Implementation Options

The charts shown in this section illustrate a relative comparison between each of the IPsec implementation options for a number of different vectors.

- **Single core throughput**: The maximum throughput obtained when a single CPU core is used for IPsec processing.
- **CPU cycle cost per IPsec packet**: The average CPU cost of processing an IPsec ESP packet, measured in CPU cycles.
- **CPU processing cost variability**: The variability in CPU packet processing cost based on packet size.
- **Packet latency**: Average latency incurred for IPsec processing on a packet.
- **PCIe Bandwidth**: Platform cost in terms of PCIe bandwidth required for support the IPsec offload option. As an example, the use of a lookaside accelerator that is not embedded with the CPU in a System on Chip typically requires PCIe lanes for connecting the lookaside accelerator to the CPU, with the number of lanes required increasing as the target throughput increases.
- **HW Support Availability**: Availability of hardware supporting the offload/acceleration features. Preferential rating is given to the solutions that are currently widely available.
- **Software support**: Availability of software supporting the offload/acceleration features. Preferential rating is given to the solutions that currently have wide support in either open source or commercial IPsec stacks.

The data in the chart reflects a relative comparison with the inner ring representing a baseline good level and the outer ring representing "best". For example, the latency with a software based implementation tends to be better than the latency with a lookaside implementation while the processing cost variability with a lookaside implementation is better than the variability in a SW implementation. The chart should not be used to infer linear performance comparisons between implementations.

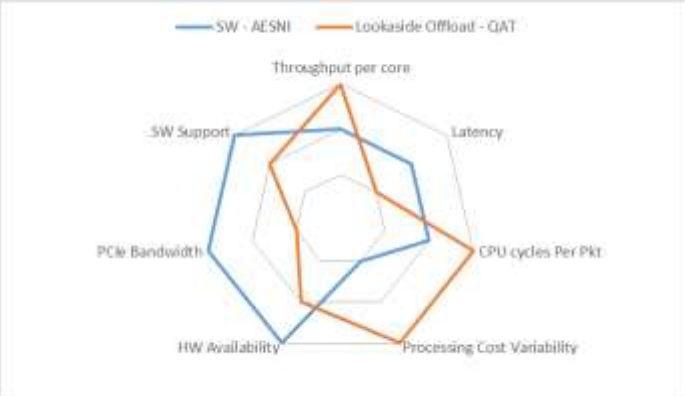


Figure 3 Comparison of IPsec Implementations

6 IPsec Performance Data

The performance data in this section is measured using VPP IPsec, v18.07, configured as an IPsec security gateway as illustrated in Figure 4. Clear text sustained throughput, measured with a 0.01% loss, is reported as the IPsec forwarding performance.

Throughput was measured separately with both software crypto using Intel® AESNI and with lookaside crypto acceleration using the 3 Intel® QuickAssist Technology devices integrated in the Intel® C621 chipset. In both cases the reported performance represents the maximum IPsec forwarding performance achieved using a single core of an Intel® Xeon® Platinum 8160 CPU. The test platform (labeled DUT in Figure 4) used Intel Ethernet Controller XXV710 cards (4x 25Gbps/card) and the benchmarking used 2 of these ports were for the IPsec interface (labeled GE2 on the DUT in Figure 4). An IPsec ESP SA per direction was configured for each port. The IPsec SAs were configured to use AES128-CBC+HMAC-SHA1.

Single core performance was chosen to better illustrate a throughput comparison between software based IPsec and IPsec with lookaside acceleration. By adding additional cores, over 100Gbps of IPsec can be achieved on this platform using Intel® AESNI® with 12 CPU cores while 85Gbps of IPsec throughput can be achieved using the 3 cores and the 3 lookaside Intel® QuickAssist Technology acceleration devices (1 core per device) available with the chipset. Using a 2nd core per acceleration device, 100Gbps of IPsec throughput can be achieved.

Additional performance data for a wider range of algorithms and CPUs are available from Intel®.

Figure 4 IPsec Forwarding Benchmark Configuration

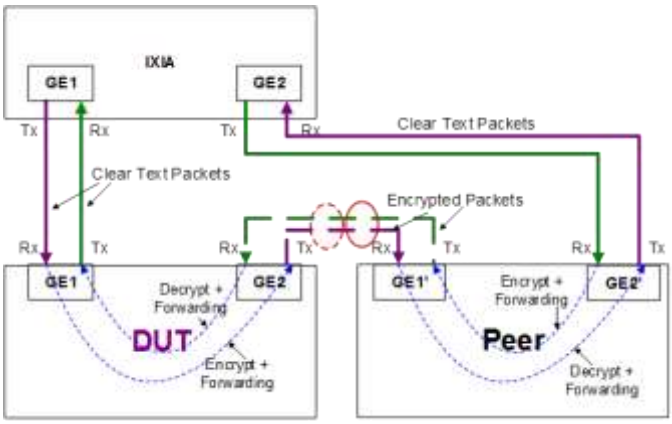
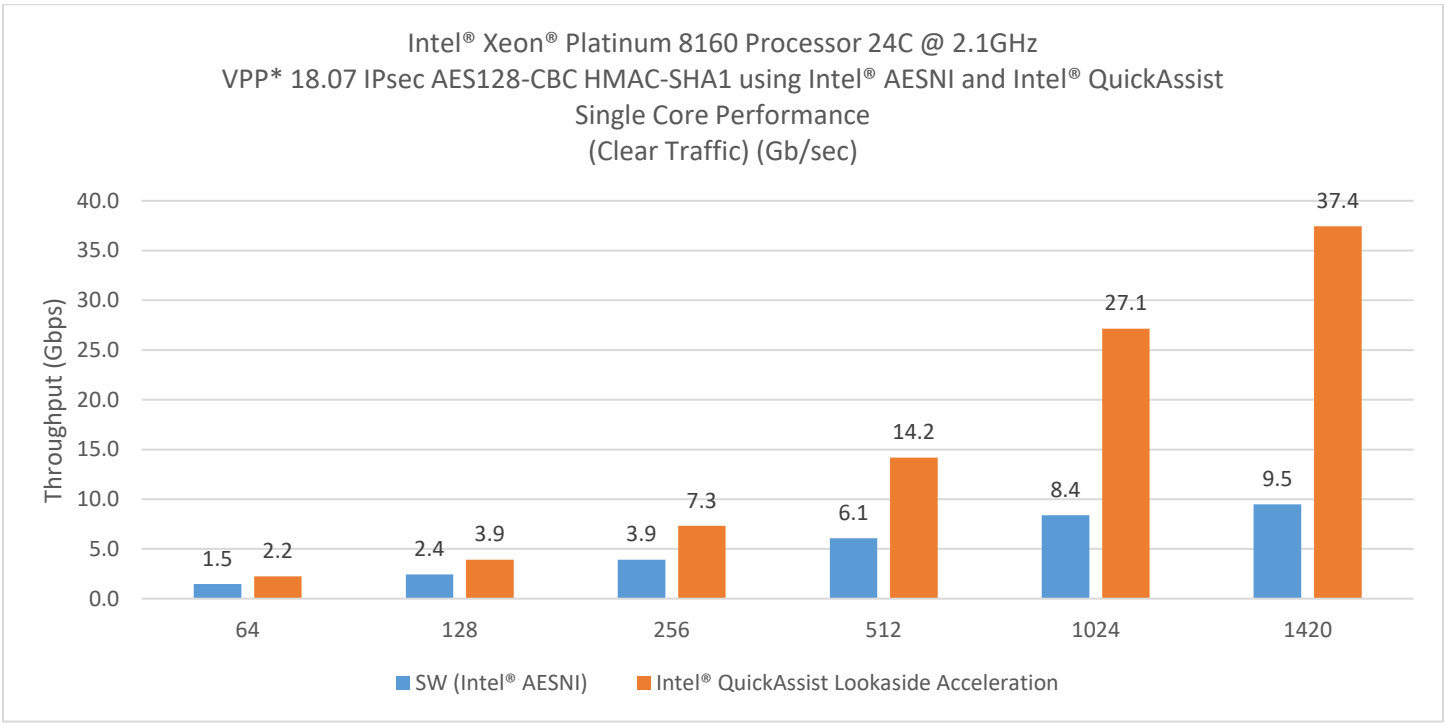


Figure 5 IPsec Forwarding Performance using Intel® AESNI and Intel® QuickAssist Acceleration



7 Conclusion

With the availability of commercial products that support IPsec offload, there are IPsec solutions available that provide a trade-off between the simplicity of deployment with the ubiquity of software implementations and the overall platform level performance achievable with specialized hardware solutions. Software based solutions are recommended as a default starting point. When the CPU cost of processing IPsec traffic is a problem, a hardware offload approach should be considered. Lookaside acceleration provides a reduction in CPU processing cost, especially when the average packet size is large (~1KB+), but often requires the use of extra PCIe lanes to connect the lookaside accelerator to the CPU.