Security Application Note

Neil Gammage

Technical Marketing

Freescale Canada

Release 2.1

Date	12/13/01
Product	C-5 NP/C-5e NP
Relevant version	All
Description	Describes security implementations with the C-5 NP
Distribution	Customer
Author	Neil Gammage

Release Record

Release	Date	Reason for Release
Draft 1	9 November 2001	Initial Release for review
Draft 2	12 November 2001	Securalink product information added
1.0	21 November 2001	Initial general release
2.0	11 December 2001	Cavium and Freescale MCP190 product information added
2.1	13 December 2001	MCP190 information corrected. Layer N information added

CONTENTS

FIGURES

TABLES

1 Introduction

In network communications, the term ‘Security’ covers the three related issues of ensuring the authenticity, integrity and confidentiality of data transmissions.

- Authenticity is ensured by authentication mechanisms, which provide the means for a data receiver to verify that a message was sent by the node that claims to have sent it
- Integrity is ensured by the same authentication mechanisms, which also provide the means for a data receiver to verify that data has been received exactly as it was sent, and has not been modified in transit
- Confidentiality is ensured by encryption mechanisms, which provide the means to prevent unauthorised receivers of data from being able to read and use it.

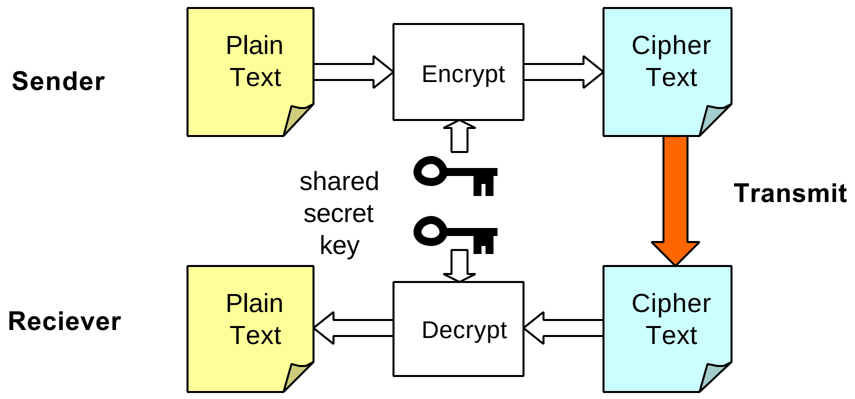
The authentication and encryption mechanisms at the heart of security are provided by security protocols which are used in combination with suites of encryption and authentication algorithms. The three major security protocols are IPSec (Internet Protocol Security), SSL (Secure Sockets Layer) and TLS (Transport Layer Security). The last two are closely related and are often referred to jointly as SSL/TLS. The algorithms used by these protocols include bulk encryption algorithms, public key encryption algorithms, and message authentication algorithms. This document describes the encryption and authentication algorithms, the security protocols, and the implementation of security functions on the C-5 NP.

2 Security Algorithms

2.1 Symmetric Key Encryption

Symmetric key encryption algorithms use a shared secret key to encrypt and decrypt a message. Each algorithm defines an *encrypt* function and a *decrypt* function which both use the key so that a message encrypted with a specified key can only be successfully decrypted using the same key as was used for encryption, as shown in the figure below:

Figure 2.1 – Symmetric Key Encryption and Decryption



Symmetric key algorithms include

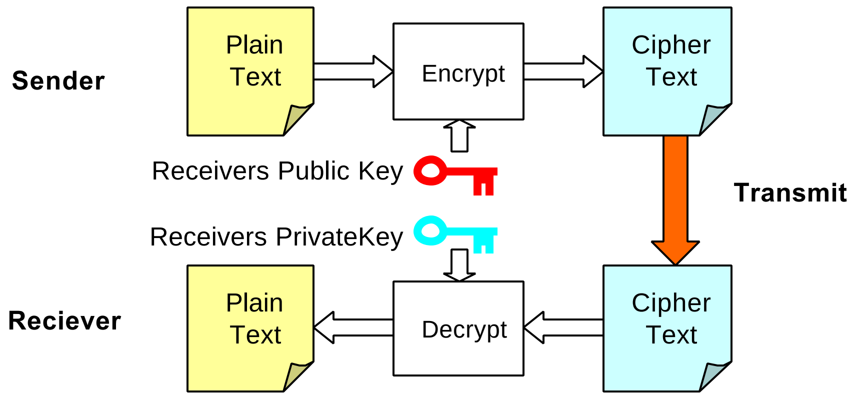
- DES (Digital Encryption Standard) The best known algorithm, which uses a 56 bit key. DES works by chopping the plain text into 64 bit blocks and using the key to generate 64 bit blocks of cipher text.
- 3DES (Triple DES). Similar to DES but uses a triple length 168 bit key and is therefore far harder to break than DES.
- AES (Advanced Encryption Standard) A new US standard cipher with key lengths of 128, 192 and 256 bits, intended to replace DES.
- RC-2 (Rivest Cipher 2) a proprietary RSA cipher.
- RC4/ARC4 (Rivest Cipher 4/Alleged RC4) RC4 was developed by RSA as a proprietary cipher, but the algorithm was cracked, and as ARC4 it is now fairly widely used
- RC5 and RC6 (Rivest Ciphers 5 and 6) are other later proprietary RSA ciphers.
- SKIPJACK is a US government algorithm implemented in hardware by the Clipper chip.
- SAFER (Secure And Fast Encryption Routine) uses a 64 bit key and is designed for use in systems with limited computing capability, like smart cards.

2.2 Public Key Encryption

A problem with symmetric key algorithms is that the key has to be agreed by communication between the sender and the receiver. If a third party were to intercept the key in transit, they would then be able to decrypt any encrypted messages that they intercepted. Public key algorithms address this problem.

Public key encryption algorithms use two keys: one to encrypt and the other to decrypt a message. Each algorithm defines an *encrypt* function and a *decrypt* function. The encrypt function uses the first, *public* key to encrypt the message and the decrypt function uses the second, *private* key to decrypt the message. A message encrypted by a sender with the receivers public key can only be successfully decrypted using the receivers private key, as shown in the figure below:

Figure 2.2 – Public Key Encryption and Decryption



The public key can be freely published; without the unpublished private key it is (nearly) impossible for a malicious third party to decrypt an intercepted message. For this reason, Public Key cryptography is used when exchanging private keys to be used with a private key algorithm such as DES. They are not used for bulk encryption because of the high computational requirements.

Public Key algorithms include

- RSA, invented by Rivest, Shamir and Adelman. The most frequently used public key algorithm. This cipher is based on the computational difficulty of determining the prime factors of very large numbers, where typically the prime factors are chosen to be at least 1024 bits long. It works by choosing numbers A, B, and N, where N is the product of two large primes, such that $x = x^{AB} \mathrm{mod} N$ for any x.
- ECC (Elliptical Curve Cryptography). Relatively new and not frequently implemented.

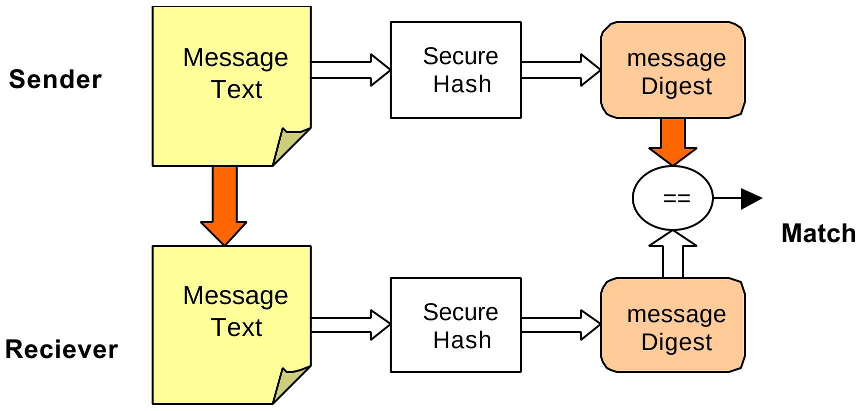
2.3 Message Integrity and Message Authentication

The purpose of message integrity algorithms is to enable a receiver of a message to check the integrity of the message – to verify that it has not been tampered with in transit.

Message integrity algorithms are based on secure hashing algorithms that process a message to produce a message digest of 100 bits or so. The secure hash function is so chosen as to render it improbable that a corrupted message would produce the same message digest as the original message.

Secure hash functions are used to check the integrity of a message, as shown in the figure below. In operation the sender runs the secure hash function over the body of the message to generate the message digest, and then transmits both the message and the digest to the receiver. The receiver runs the same hash function over the received message an generates a message digest and compares it to the one generated by the sender. If they are the same, the message was received exactly as it was transmitted.

Figure 2.3 – Message Intregrity Checking

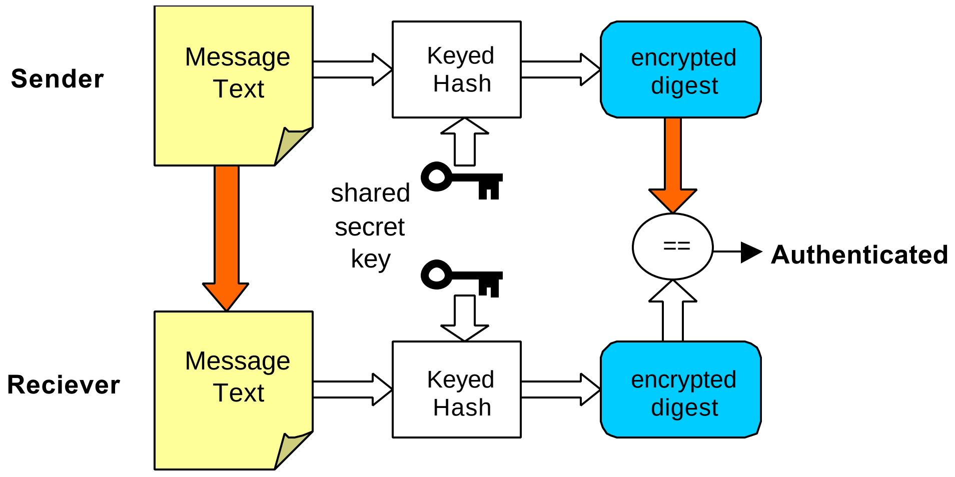


The two most used secure hash algorithms are

- MD5 (Message Digest 5) invented by Rivest. This pads the message out to an integral number of 512 byte blocks, then processes it block by block to generate a 128 bit message digest.
- SHA-1 (Secure Hash Algorithm 1), an FIPS standard. This also processes the message in 512 byte blocks but generates a 160 bit digest. It is considered somewhat more secure than MD5, but is more computationally intensive.
- RIPEMD (RACE Integrity Primitives Evaluation Message Digest), a 128 or160 bit secure hash defined as part of a European Union RACE program.

Message authentication algorithms incorporate a secure hash algorithm, used in conjunction with a symmetric key cryptographic algorithm to permit a message receiver to authenticate the message sender as well as to check the integrity of a message. The operation of message authetication algorithms is ahown in the following figure.

Figure 2.4 – Message Authentication using Symmetric Key Cryptography



The sender creates a message digest using an encryption function possibly in conjunction with a secure hash function in conjunction with a secret key, then sends both the original text and the encrypted message digest to the receiver. The receiver runs the encrypt/hash function using the same secret key to create an encrypted message digest, and compares it with the digest it received. If the two digests are the same, the receiver knows that the message was received as it was sent and the same secret key was used to create the digest, thus authenticating the message.

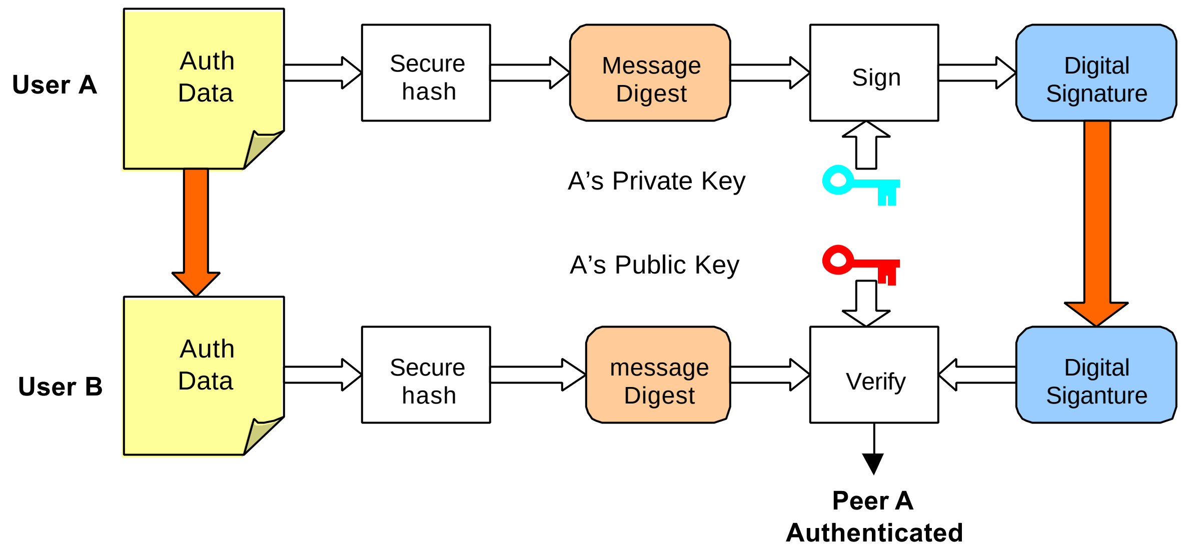
Examples of symmetric key message authentication algorithms are

- CBC-MAC (Cipher Block Chaining – Message Authentication Code). Divides the message into blocks, encrypts each block with DES, then uses the result as the initial value for encrypting the next block. The result of encrypting the last block is the encrypted message digest.
- HMAC (key Hash Message Authentication Code). Divides the message into blocks, appends a secret key to each block, then runs a secure hash algorithm to generate an encrypted message digest. The two main variants are
    - HMAC – MD5. Uses MD5 as the secure hash function, and like MD5, generates a 128 bit encrypted message digest.
    - HMAC – SHA-1. Uses SHA-1 as the secure hash function, and Like SHA-1, generates a 160 bit encrypted message digest.

2.4 Identity Authentication

The purpose of identity authentication is to enable each party connected by a secure communication channel to verify the identity of the other party. Public Key algorithms are an ideal basis for this purpose. In section 2.2, the public key system was used to ensure that only the intended recipient of an encrypted message is able to decrypt it. If the use of keys is reversed, and the senders private key is used for encryption and the public key for decryption, then if a receiver is able to successfully decrypt a message sent by an alleged sender, then the sender must be authentic because the encrypted message could only have been created by someone in possession of the senders private key.

Figure 2.5 – Identity Authentication using Digital Signatures



Private key identity authentication mechanisms use a secure hash function in conjunction with a public key encryption algorithm to produce an encrypted hash. The operation of a public key based identity authentication algorithm is shown in the figure above.

To authenticate its identity to B, A generates a block of authentication data, which may include information that A requires to transmit to B, as well as randomly generated data. A produces a message digest of this data, then signs (encrypts) the digest using the private key to produce a digital signature. Both the original message and the digital signature are transmitted to User B. B uses the same hash function as was used by A to generate a message digest of the received message, then submits it together with the received digital signature to a Verify function which uses A's public key. If it succeeds, the Verify function authenticates A to B. A similar exchange in the reverse direction can be used to authenticate the identity of B to A.

Two variants of this technique are in use :-

- The use of standard public key cryptography (specifically the RSA algorithm) in conjunction with either MD5 or SHA-1. User A produces a message digest, then encrypts it using the RSA encrypt function as the Sign function, and sends the original data and the digital signature to B. B produces a message digest of the received data, then submits that together with the received digital signature to the Verify function. The verify function uses the RSA decrypt function to decrypt the digital signature to produce a message digest and compares it with the message digest produced locally. User A is authenticated if the two message digests are identical.
- DSS (Digital Signature Standard) promulgated by FIPS operates in a similar fashion. It uses DSA (Digital Signature Algorithm), a specialised public key based digital signature algorithm, as its Sign and Verify functions, and uses SHA-1 as the secure hash algorithm.

2.5 Digital Certificates

As part of the identity authentication process, the parties to the connection may exchange digital certificates with each other. A digital certificate is a data record which is created by a trusted certificate issuing authority which may contain

- The certificate holder's (i.e. the certificate sender's) public key
- The certificate serial number
- The certificate validity dates
- The holder's identity (DN, IP address)
- The issuer's identity (DN)
- The issuer's digital signature. This is a piece of data that has been 'signed' using the issuer's private key, as described in section 2.4.

The receiving party maintains a list of certificate issuing authorities (CAs). Each entry contains

- The CA's identity (DN)
- The CA's public key
- The CA's digital signature

The receiving party checks that

1. The issuer is on the trusted CA list
2. The certificate is currently valid
3. The issuer's public key validates the digital signature in the certificate.

If all tests pass, the receiving party is now in possession of the other party's public key and knows that it has not be stolen by a third party.

2.6 Key Exchange Algorithms

Before the end points of a secure communication channel using symmetric key encryption and/or authentication can begin to exchange secure messages, they must first agree on the key(s) to be used in the session. This key agreement process is known as key exchange. It is important that unauthorised parties should not be able to see the keys being exchanges, and for that reason key exchange messages are encrypted using public key encryption algorithms.

A variety of key exchange algorithms are in use. All define a sequence of encrypted messages exchanged between the two users, and operations to be performed on those messages, which enable them to agree on the encryption keys that will be used for the session.

Frequently used key exchange algorithms are

- RSA key exchange. A key exchange algorithm based on the RSA public key algorithm.
- D-H, invented by Diffie and Hellman. A key exchange algorithm based on the infeasibility of computing x^{AB} given only x^A and x^B. The A and B are randomly generated numbers, and any system that implements DH must therefore include a good random number generator.
- Oakley, an extended version of D-H standardised by IETF
- KEA (Key Exchange Algorithm) defined by NSA and based on D-H and SKIPJACK

3 IPSec

Internet Protocol Security is an extension of IP defined by IETF to provide security for all internet traffic. A series of RFCs define the information to be added to IPv4 and IPv6 packets to ensure data security as well as specifying how encryption and authentication algorithms are to be used.

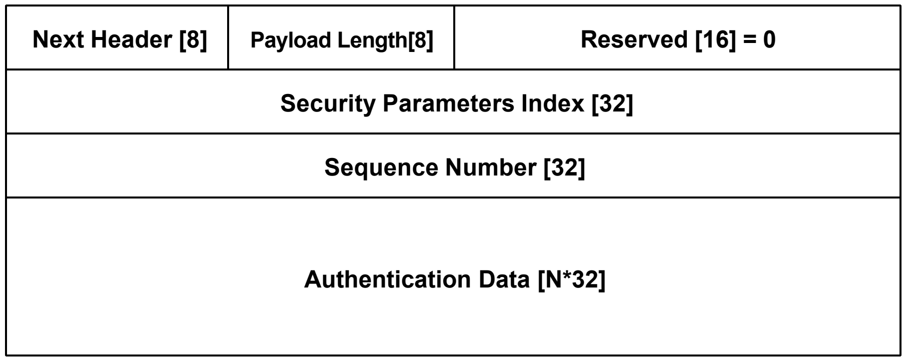
The architecture of IPSec, defined in [IPSEC], is based on setting up *security associations* between senders and receivers of data, which define the algorithms to be used for message encryption and authentication and the security keys to be used with these algorithms. IPSec defines the protocols which may be used for security, and also defines the mechanisms and protocols which may be used to establish security associations.

Security is provided by the IPSec Authentication Header (AH) and the Encapsulating Security Payload header (ESP) which are described in the next sections. A packet may include an AH, an ESP, or both, depending on the level of security required.

3.1 The IPSec AH

The Authentication Header described in [AH] is used to permit authentication data to be transmitted with a packet, and provides for authenticity and integrity, but not confidentiality. The authentication data is calculated by the sender and checked by the recipient to verify that the packet was transmitted by the sender, and has been received is as it was transmitted, without modification. The header format is as shown below.

Figure 3.1 – Authentication Header



The fields are

- Next Header. Defines the protocol of the next header in the packet. For IPv4 this will either be a UDP or TCP header, or no header if used in a raw IP packet. For IPv6 this may also define a further extension header after the AH and before the upper layer header.
- Payload Length. The length of the header in 32 bit words, minus 2.
- Reserved. The next 16 bits are reserved and must be set to 0.
- Security Parameters Index (SPI). A randomly chosen number, which together with the destination IP address, identifies the *security association* (SA) to be used for authenticating

the IP packet. The SA is described in section 3.4

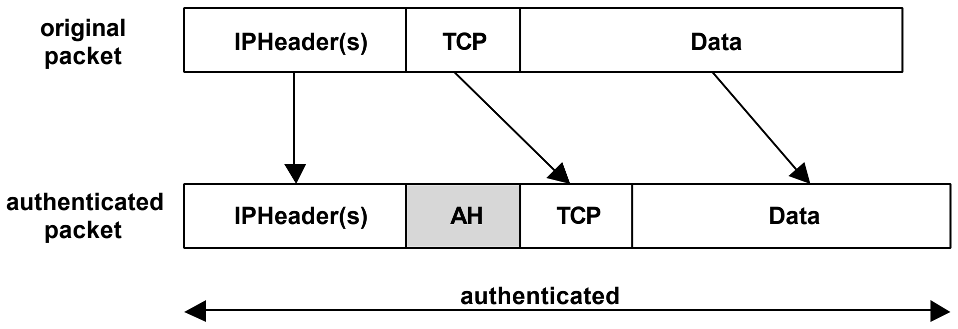
- Sequence Number. A counter incremented by the sender for every packet sent to the same destination. The destination should discard any packet received with a sequence number it has already processed.
- Authentication data. The Integrity Check data used to authenticate the packet. Used as determined by the SPI to authenticate the packet.

The sequence number is incremented by the sender for every packet sent and may be checked by the receiver to detect lost or duplicated packets.

In the case of IPv4, the AH follows the IP header. In the case of IPv6, the AH follows the IP header and all extension headers except the ESP (if present). The authentication data is calculated by applying the authentication algorithm defined by the SPI to the entire packet, including immutable fields in the IP and IPv6 extension headers (e.g. Source and Destination Address fields), as indicated in the figure below.

Figure 3.2 – AH Coverage

The choice of



3.2 The IPSec ESP

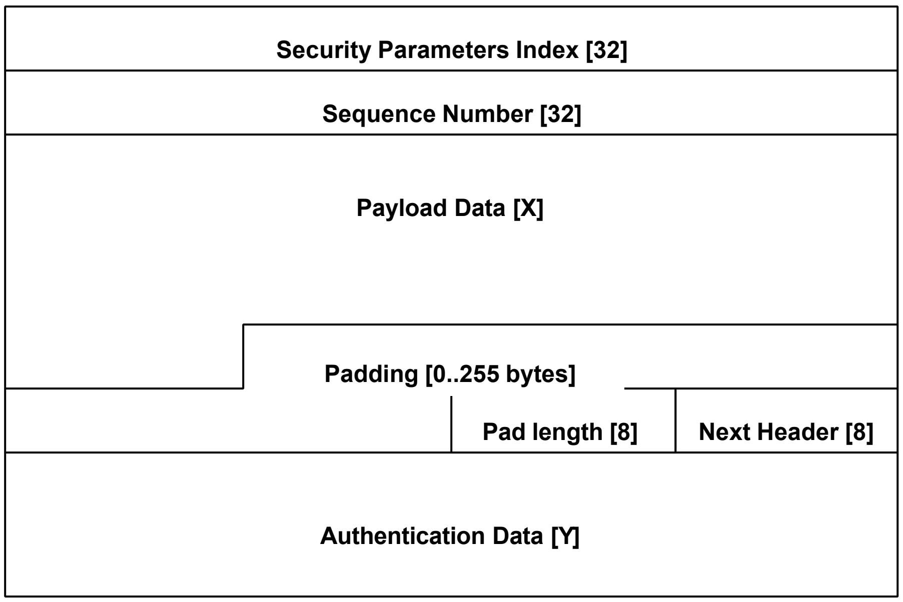
The ESPH described in [ESP] is used to send and received packets with encrypted payloads. Optionally, it also provides similar authentication capabilities to the AH and therefore covers the full set of security concerns – authenticity, integrity and confidentiality. The format of the header is as shown in the figure below.

The fields of the header are

- Security Parameters Index (SPI). A randomly chosen number, which together with the destination IP address, identifies the *security association* (SA) to be used for encrypting and authenticating the IP packet. The SA is described in section 3.4
- Sequence Number. Used as for the AH.
- Payload Data. The encrypted payload along with any data required for decryption (e.g. initialisation data).

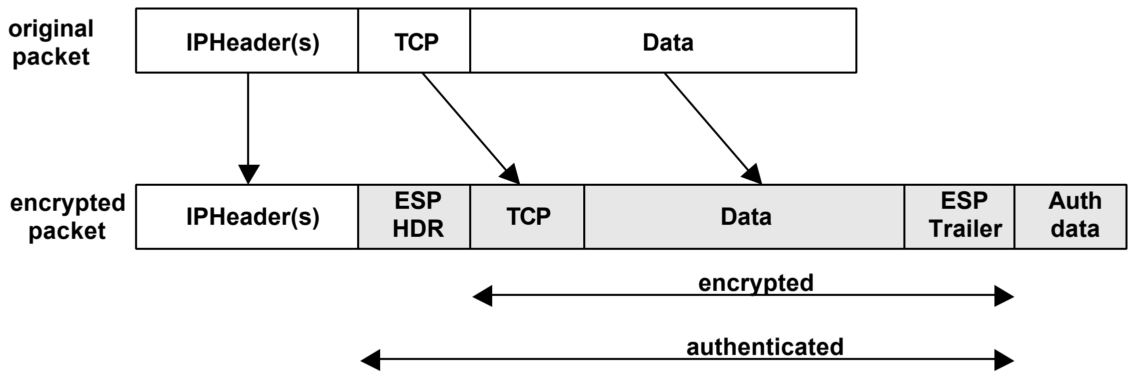
- Padding. Added to the header to ensure that the payload data ends on the appropriate byte boundary. This is required since the encryption algorithms are block ciphers operating on a fixed data block size.
- Pad length. The number of padding bytes added
- Next header. As for the AH.
- Authentication Data. If authentication is applied, is used in the same way to the Authentication Data field in the AH.

Figure 3.3 – Encapsulating Security Payload Header



For IPv4, the ESP follows the IP header, and for IPv6 the ESP follows the IP header and all extension headers. In both cases, all packet data following the IP header (including any upper layer headers) is encrypted, and the authentication data is computed over the entire ESP header (including the enciphered data of the original packet), as shown in the figure below.

Figure 3.4 – ESP Coverage



Because the ESP is able to provide both authentication and encryption, it is far more frequently used than the AH.

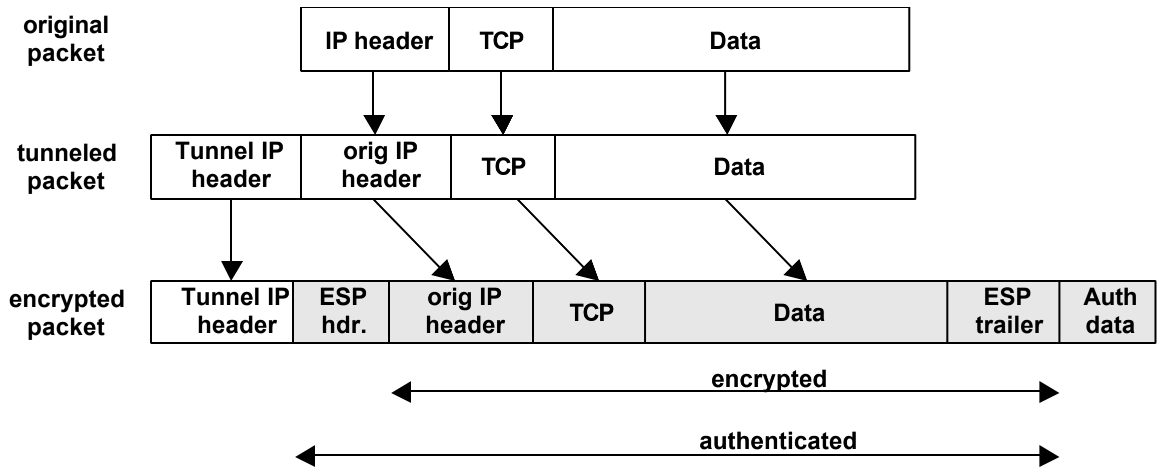
3.3 Transport and Tunnel Modes

IPSec can be used in two modes, *transport mode* and *tunnel mode*. The figures in the previous sections illustrated the AH and the ESP being applied in Transport Mode. In tunnel node, the original packet is tunnelled by encapsulating the original message in a tunnel IP header before applying the AH or ESP.

When using the ESP in transport mode, the IP header is neither encrypted nor authenticated. There is no protection against it being maliciously modified in transit nor against it being viewed and understood by an intermediate node even when an ESP is used. Although such a node will not be able to understand the message itself, it will be able to see which host sent it and which will receive it, and this information can be of significance in itself.

When the ESP is used in Tunnel mode, both encryption and authentication are extended to cover the entire original packet, as shown in the figure below.

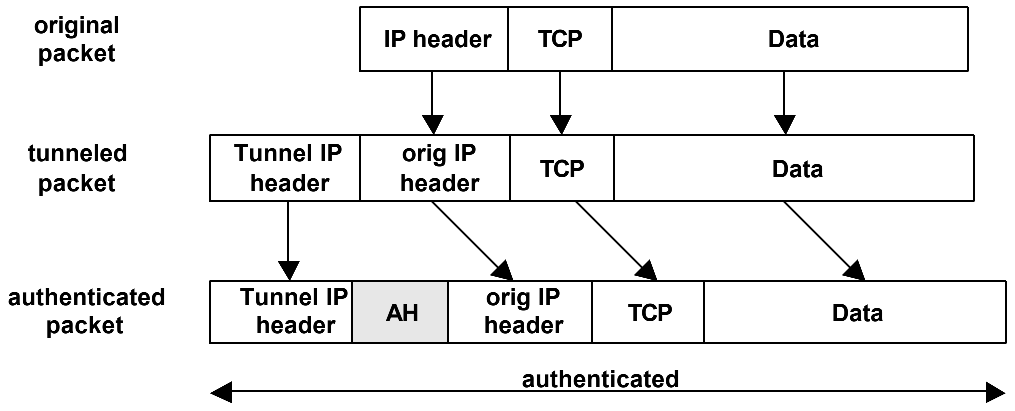
Figure 3.5 – ESP used in Tunnel Mode



Comparing this with figure 3.4, it can be seen that the entire original packet including the original IP header is authenticated and encrypted.

The process when an AH is used in tunnel mode is shown in the figure below:

Figure 3.6 – AH used in Tunnel Mode

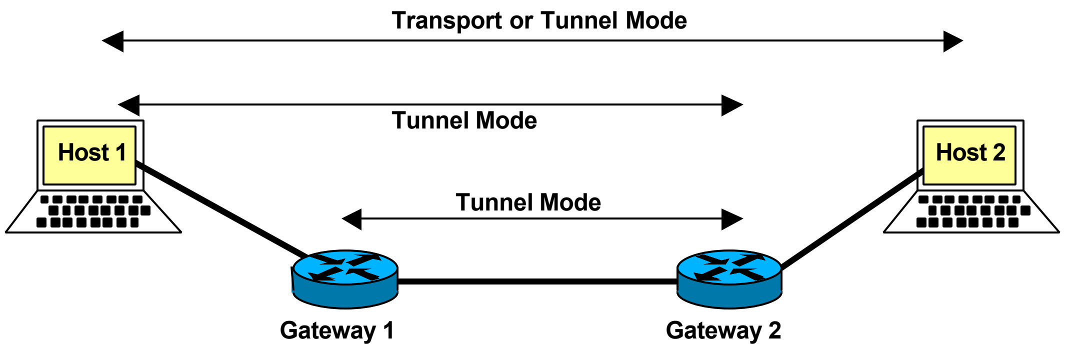


As can be seen, use of the AH in tunnel mode provides no additional coverage than that afforded in transport mode.

Transport mode can only be applied between two host systems – for example a workstation client and a network server. However, tunnel mode can be applied not only between hosts but also between intermediate systems acting as *security gateways*. These might be devices such as firewalls or routers or specialised nodes providing a security service for the traffic flowing through them.

As shown in the figure below, transport mode may only be used between Host 1 and Host 2. Tunnel mode may be used between two Hosts, between a Host and a Gateway, or between the two Gateways.

Figure 3.7 – Transport and Tunnel Mode



Because of the additional coverage when using the ESP and the ability to operate between security gateways, tunnel mode is far more frequently used than transport mode.

3.4 Security Associations

A security association is a connection between an originator and receiver of encrypted or authenticated data. SAs are unidirectional, so that if a bi-directional flow of secure data is required between two nodes, two SAs must be set up, one for each direction. The SA determines whether an AH or ESP (or both) should be used, and defines the encryption and authentication algorithms and the encryption and authentication keys to be used to secure traffic. The cipher algorithms which may be used with the ESP are defined in [DES] and [CBC]. The message authentication algorithms which are used with the ESP and AH are defined in [HMAC], [MD5] and [SHA-1].

Every IPSec enabled node maintains two databases, a *security policy database* (SPD) and a *security association database* (SAD).

The SPD is used for outgoing packets, and is used to determine whether packets are to be processed by IPSec, and if so, how. The SPD is searched using a key which may include the source and destination IP address of the packet, the source and destination port numbers, and the higher layer protocol (UDP/TCP) in use. Wildcard and number ranges are permitted in all fields of the key. If a matching entry is found in the database, the data returned determines

- Whether an AH or ESP (or both) should be used
- Whether transport or tunnel mode should be used
- Which authentication and/or encryption algorithms are to be used.

The entry also contains a pointer to an entry in the SAD if one exists. If an SA does not exist, the SPD data contains all of the information required to set up a new one.

The SAD entries contain all of the information required to construct the AH or ESP header and encrypt and/or authenticate the packet. Each entry includes

- AH Authentication algorithm identifier and key (for AH SAs)
- ESP Authentication algorithm identifier and key (for ESP SAs)
- ESP Encryption algorithm identifier and key (for ESP SAs)
- Lifetime
- Sequence number counter
- Anti-replay window

The last two fields are used if the sequence number in the AH or ESP is to be used to detect lost or duplicated packets. The Lifetime is used to automatically limit the time that an SA can exist. When the lifetime expires, a new SA must be set up if necessary.

The SA data is used to encrypt and/or authentic the AH and/or ESP on outgoing packets. At the receiving IPSec node, the SA is lookup up using a key consisting of the Security Parameters Index (SPI) carried in the AH or ESP and the destination IP address of a packet. This provides the receiving node with all of the information required to decrypt and/or authenticate the packet.

3.5 Key Management

The Internet Security Association and Key Management Protocol defined in [ISAKMP] and [DOI] is a process for the creation of new security associations as required. The process is triggered by an attempt to send a secure packet between a sender and receiver who do not have an appropriate SA set up. ISAKMP defines a two stage process for setting up an SA

1. An authenticated connection is established between the sender and the receiver.
2. Key data is exchanged over the secure connection

ISAKMP does not specify the key exchange protocol to be used, but does recommend the use of Internet Key Exchange protocol, defined in [IKE] for the purpose.

The IKE connection may be authenticated in a variety of ways

- Pre-sharing authentication keys and using MAC or HMAC as an authentication algorithm
- Exchanging data including public keys and using the RSA public key algorithm.
- Exchanging digital signatures using either the RSA public key algorithm or DSS/DSA

Once established, the Oakley (modified DH) protocol is used to exchange key data. At the completion of the key exchange, both parties to the SA will have agreed on the encryption and/or authentication algorithms and keys to be used.

4 SSL/TLS

Secure Sockets Layer is a protocol stack developed by Netscape for secure transmission of web pages. Transport Layer Security is an IETF protocol based on SSL, defined in [TLS].

SSL is built on a variety of security technologies, including

- Symmetric key algorithms such as DES, 3DES, and ARC-4 for bulk data encryption
- Message authentication algorithms MD5 and SHA-1
- User authentication algorithms such as RSA and DSA
- Key exchange algorithms such RSA key exchange and KEA

As can be seen there is a great deal of similarity between IPSec and SSL/TLS in the algorithms required to support them. Like IPSec, SSL defines both a secure message exchange protocol equivalent to the ESP protocol (the SSL record protocol), and a protocol similar to ISAKMP/IKE which is used to establish security associations and exchange session keys (the SSL handshake protocol).

4.1 SSL Cipher Suites

The two handshake protocols supported by SSL are one based on the RSA key exchange algorithm and one based on the KEA algorithm. Two sets of cipher suits are used with SSL. The first set uses the RSA key exchange algorithm in the SSL handshake.

Table 4.1 – SSL RSA Key Exchange based Cipher Suites

Encryption	Key Length	Authentication	Notes
3DES	168 bit key	SHA-1	Strongest cipher suite
(A)RC2	128 bit key	MD5	Fastest cipher suite
(A)RC4	128 bit key	MD5	Fastest cipher suite
DES	56 bit key	SHA-1	Weaker than ARC or 3DES
(A)RC2	40 bit key [1]	MD5	Exportable. Weak cipher suite
(A)RC4	40 bit key [1]	MD5	Exportable. Weak cipher suite
No encryption	-	MD5	Only used if client and server cannot agree on an encryption algorithm

[1] Note. The 40 bit key is padded out to 128 bits but only the first 40 bits have cryptographic significance.

The second set of encryption and message authentication algorithms are the FORTEZZA cipher suites. FORTEZZA is an encryption system used by the US government for sensitive but unclassified information. The KEA key exchange algorithm is used in the SSL handshake when these suits are in use.

Table 4.2 – SSL FORTEZZA Cipher Suites

Encryption	Key Length	Authentication	Notes
(A)RC4	128 bit key	SHA-1	Strong cipher suite
SKIPJACK	80 bit key	SHA-1	Weaker cipher suite
No encryption	-	SHA-1	Only used if client and server cannot agree on an encryption algorithm

4.2 The SSL Handshake

The SSL handshake is used in the same way as ISAKMP in IPSec. That is, it first exchanges data including digital certificates between the client and server which enable the client to authenticate the identity of the server, and optionally to permit the server to authenticate the identity of the client. The second part of the handshake uses a key exchange algorithm to create the session keys which will be used to encrypt and decrypt subsequent SSL messages.

5 IPSec vs SSL

The major difference between the two is the protocol layer at which they operate. IPSec is an enhancement to IP operating at the IP Datagram layer, and therefore does not recognize the concept of a TCP session or an end user. SSL/TLS operates on top of TCP/IP and is designed to provide security on a per TCP session and per user basis. Secondary difference is that whereas IPSec is based on peer – peer protocols, SSL is based on client – server protocols.

It may be asked why we need two sets of security protocols at all. The reason is the usage for which each protocol suite was intended.

IPSec was intended as a security protocol for all internet traffic between any two nodes in the network. This makes it ideal for applications such as VPN, where traffic between two enterprises of between remote nodes and an enterprise need to be secured. This is the major use for IPSec Tunnel mode implemented on Security Gateways.

SSL is intended to secure individual web interactions on a per user basis. As such it is defined exclusively as a host to host protocol – there can be no such thing as an SSL security gateway.

IPSec security associations are relatively long lived and are set up and terminated infrequently. In an IPSec implementation the rate at which secure data can be transmitted (ideally line rate) is far more important than the time taken to execute an IKE handshake.

By contrast, SSL sessions by definition carry the traffic of a single user and are relatively short lived, existing for access to a secure server typically for only part of a browser session. As a result at the server end of an SSL connection, the rate at which the SSL handshake protocol can be operated is a major issue for SSL implementations.

.

6 Security Co-Processors

Since both public key and bulk encryption algorithms are computationally intensive, hardware assist chips are frequently used in servers and security gateways to permit encryption and decryption at line rate. When used with the C-5 NP, a security assist chip would operate as an external co-processor, connected to the C-5 NP through appropriate interfaces.

This section surveys the security accelerator products currently available. Note that some also provide compression algorithms to support protocols like PPP. Also the devices that provide public key support also include a random number generator for key generation.

The vendors and products are listed in the following three tables, which also provides performance numbers where these are known.

Table 6.1 – Security Accelerators: Encryption and Message Authentication

		Encryption				Authentication		
		DES	3DES	AES	ARC4	HMAC	MD-5	SHA-1
Freescale	MPC180	✓	15 Mbps [1]	-	20 Mbps	✓	✓	✓
	MPC190	✓	558 Mbps [5]	-	107 Mbps	✓	✓	✓
Hifn	HIPP 7814	✓	200 Mbps [1]	200 Mbps [2]	✓	✓	✓	✓
	HIPP 7854	✓	500 Mbps [1]	500 Mbps [2]	✓	✓	✓	✓
	HIPP 8154	✓	2 Gbps [1]	2 Gbps [2]	✓	✓	✓	✓
Broadcom	BCM5820	✓	310 Mbps [1]	-	200 Mbps	✓	✓	✓
	BCM5821	✓	470 Mbps [1]	-	600 Mbps	✓	✓	✓
	BCM5840	✓	2.4 Gbps [1]	-	-	✓	✓	✓
Securalink	PCC2010 [4]	133 Mbps	133 Mbps	-	-	✓	✓	✓
	PCC2020	1.8 Gbps	622 Mbps	-	800 Mbps	✓	✓	✓
	PCC-ISES [4]	355 Mbps	128 Mbps	-	-	✓	✓	✓
Cavium	CN1120i	✓	1.3 Gbps	✓	✓	✓	✓	✓
	CN1340i	✓	5 Gbps	✓	✓	✓	✓	✓
	CN1120s	✓	1.0 Gbps	✓	✓	✓	✓	✓
	CN1340s	✓	2.4 Gbps	✓	✓	✓	✓	✓
NetOctave	NSP2000	-	1 Gbps [3]	-	1 Gbps [3]	✓	✓	✓
	NSP3200	-	2.4 Gbps	2.4 Gbps	-	✓	✓	✓
	NSP4200	-	10 Gbps	10 Gbps	-	✓	✓	✓
Corrent	CR7000	-	-	-	-	-	-	-
	CR7020	✓	2.4 Gbps	✓	✓	✓	✓	✓
	CR7120	✓	2.4 Gbps	✓	-	✓	✓	✓

		Encryption				Authentication		
		DES	3DES	AES	ARC4	HMAC	MD-5	SHA-1
Analog	ADSP-2141	✓	155 Mbps [1]	-	-	✓	✓	✓
Layer N	Ultralock	✓	✓	✓	✓	✓	✓	✓

Note [1] 3DES encryption plus HMAC – SHA-1 authentication.

[2] AES encryption plus HMAC – SHA-1 authentication.

[3] 3DES or ARC4 encryption plus HMAC – MD5 or HMAC – SHA-1 authentication

[4] Also implement SAFER encryption and RIPEMD message authentication

[5] 3DES encryption plus HMAC – MD5 authentication

Table 6.2 – Security Accelerators: Public Key

		Algorithm				Other	
		RSA	D-H	DSA	ECC	RNG	Compression
Freescale	MPC180	10 ps	10 ps	-	30 ps	✓	-
	MPC190	500 ps	520 ps [3]	-	1000 ps [3]	✓	-
Hifn	HIPP 7814	120 ps	120 ps	-	-	✓	LZS, MPPC
	HIPP 7854	300 ps	300 ps	-	-	✓	LZS, MPPC
	HIPP 8154	1500 ps	1500 ps	-	-	✓	LZS, MPPC
Broadcom	BCM5820	800 ps	1250 ps	-	-	✓	-
	BCM5821	4000 ps	3000 ps	-	-	✓	-
	BCM5840	-	-	-	-	-	-
Securalink	PCC2010	?	?	?	?	✓	-
	PCC2020	?	?	?	?	✓	-
	PCC-ISES	?	✓	✓	?	✓	-
Cavium	CN1120i	✓	2500 ps	-	-	✓	-
	CN1340i	✓	10000 ps	-	-	✓	-
	CN1120s	8000 ps	-	-	-	✓	-
	CN1340s	16000 ps	-	-	-	✓	-
NetOctave	NSP2000 [1]	2050 ps	-	✓	-	3 Mbps	-
	NSP3200	-	-	-	-	-	-
	NSP4200	-	-	-	-	-	-
Corrent	CR7000 [2]	2500 ps	1500 ps	-	-	✓	-
	CR7020 [2]	2500 ps	1500 ps	-	-	✓	-
	CR7120	2300 ps	2300 ps	-	-	✓	-
Analog	ADSP-2141	35 ps	35 ps	25 ps	-	✓	-

		Algorithm				Other	
		RSA	D-H	DSA	ECC	RNG	Compression
Layer N	Ultralock	✓	-	-	-	✓	-

Note.

All performance numbers are for IKE handshakes per second except as noted below

[1] RSA decryptions per second

[2] SSL handshakes per second.

[3] IKE handshakes per second.

Table 6.3 – Security Accelerators: Interfaces

		Protocols		Interfaces	
		IPsec	SSL	Data I/F	Control I/F
Freescale	MPC180	✓	✓	8xx or 82xx local bus	
	MPC190	✓	✓	PCI 32/64 bit 66 MHz	
Hifn	HIPP 7814	✓	✓	PCI 32/64 bit 66 MHz	
	HIPP 7854	✓	✓	Streaming Bus	PCI 32 bit 66 MHz
	HIPP 8154	✓	✓	Streaming Bus	PCI 32 bit 66 MHz
Broadcom	BCM5820	✓	✓	PCI 32/64 bit 33/66 MHz	
	BCM5821	✓	✓	PCI 32/64 bit 33/66 MHz	
	BCM5840	✓	-	PL3	
Securalink	PCC2010	✓	✓	ARM 32 bit	
	PCC2020	✓	✓	PCI 64bit 33/66 MHz	
	PCC-ISES	✓	✓	Streaming Bus	ARM 32 bit
Cavium	CN1120i	✓	-	PCI 64 bit 64 MHz (PCI-X 64 bit 133 MHz available)	
	CN1340i	✓	-	HyperTransport	
	CN1120s	-	✓	PCI 64 bit 64 MHz (PCI-X 64 bit 133 MHz available)	
	CN1340s	-	✓	HyperTransport	
NetOctave	NSP2000	-	✓	PCI 64 bit 66 MHz	
	NSP3200	✓	-	PL3	PCI-X 64 bit 100 MHz
	NSP4200	✓	-	PL4	PCI-X 64 bit 00 MHz
Corrent	CR7000	✓	✓	-	PCI 64 bit 66 MHz
	CR7020	✓	✓	PL3	PCI 64 bit 66 MHz
	CR7120	✓	-	PL3	PCI 32 bit 66 MHz
Analog	ADSP-2141	✓	✓	PCI 16/32 bit 33 MHz	

		Protocols		Interfaces	
		IPsec	SSL	Data I/F	Control I/F
Layer N	Ultarlock	-	✓	GbE	10/100 Base T

<u>Note</u>

Devices designed for high performance application data plane applications typically provide two interfaces, one for the data stream to be encrypted/decrypted and one for control purposes (e.g. managing on chip SADs). Devices designed for host control plane applications only provide one interface for data and control.

Devices are shown as supporting IPSec if they support the major IPSec encryption and authentication algorithms (DES, 3DES, AES and HMAC – MD5, HMAC – SHA-1) and key negotiation algorithms (RSA, D-H). Devices are shown supporting SSL is they support the major SSL encryption and authentication algorithms (DES/3DES + HMAC – SHA-1, ARC4 + HMAC – MD5) and key negotiation algorithms (RSA, RSA key exchange)

7 C-5 NP Security Implementation

Since SSL runs over TCP, it is normally implemented in hosts or the control plane of systems such as routers which might use a C-5 NP in the forwarding plane.

IPSec in Transport Mode is defined as operating host-host, and would therefore be deployed in a similar way to SSL.

IPSec in Tunnel mode, however, is normally implemented in enterprise edge routers acting as IPSec proxies, to support VPN services. This can be expected to be a typical application scenario where a security solution is needed in conjunction with the C-5 NP.

7.1 Device Selection

This section reviews the devices listed in section 6 and their suitability for use with the C-5 NP.

Most devices designed for host use are not that relevant to the C-5 NP implementation. They may appear in designs which also employ the C-5 NP but would normally be used as a component of the control plane rather than forwarding plane implementation. Typically these devices use PCI bus interfaces for both data and control. On the C-5 NP they would have to be interfaced to the XP's PCI bus and would only be capable of providing adequate performance if only a small percentage of the traffic was secure.

The devices that do appear to be appropriate for use with the C-5 NP, particularly in an IPSec gateway role, are those that provide separate streaming data and control interfaces. They are listed by vendor below.

At this time the most appropriate devices to be used with the C-5 NP are the Corrent CR7020 for SSL and CR7120 for IPSec applications.

7.1.1 Hifn

The HIPP7854 and 8154. The 7854 can support about 3 x OC-3, and the 8154 about 2 x Gb Ethernet of IPsec ESP traffic. Both use a configurable streaming data bus and PCI control bus. The full range of interfaces that the streaming bus can be adapted to is TBD.

7.1.2 Broadcom

The BCM5840 is billed as providing OC-48 IPSec ESP performance. It does not include an IKE accelerator; and does not have a separate control interface. Broadcom recommend the use of a separate BCM5820/5821 to support key exchange.

7.1.3 Securalink

The Securealink products all incorporate an ARM processor which can be loaded with new algorithms if required. The PCC-ISES has a streaming interface (nature undefined). Its exact public key capabilities are unclear, except that it is stated to provide D-H key exchange and DSA. Surprisingly, the significantly faster PCC2020 only provides a PCI control/data interface.

7.1.4 NetOctave

The two parts of interest are the NSP3200 OC-48 IPSec accelerator and the NSP4020 OC-192 accelerator. Neither provide support for ISAKMP. The company literature suggests using the NSP2000 in conjunction with the 3020 to provide IKE functionality. However the NSP2000 is only available as a component of a board level solution, and is an SSL oriented device that does not appear to support IKE.

7.1.5 Corrent

Corrent have three parts of interest, the CR7000, 7020 and 7120. The 7020 is intended for SSL acceleration (although it also provides all the algorithms required for IPSec). The 7000 is a subset of the 7020, providing only the key negotiation algorithms. The 7120 is intended for IPSec acceleration. It is IPv4 and IPSec aware, so it can detect an AH or ESP in secure packets in the ingress direction, apply all the necessary processing and deliver a plain text message to the ingress ports of an NP. In the egress direction, it can apply all necessary algorithms under direction from an NP and apply the necessary AH and/or ESP headers to a secure packet.

7.1.6 Cavium

The Cavium NITROX line of SSL and IPSec co-processors are interesting, but are not available with interfaces useful to C-Port. A future NITROX II series of devices will have PL3/PL4 interfaces but availability and performance is not known.

7.1.7 Layer N

A very interesting device providing SSL/TLS processing (but no IPSec support). It includes a TCP packet processor and so would be an excellent choice for performing SSL/TLS security processing in the data plane. No performance details are known, but it is intended for security processing at Gigabit rates. The use of Ethernet rather than PCI as the control interface is unique.

7.2 Interfacing to the C-5 NP

There are a number of ways of connecting a security chip to the C-5 NP. The data connection can either be made to the network side of the C-5 NP or to the FP. Whether connected to the network side or the FP, all of the devices listed above may be used in ‘look-aside’ mode, in which packets are received on normal network ports, then redirected to the security device only if they require security processing.

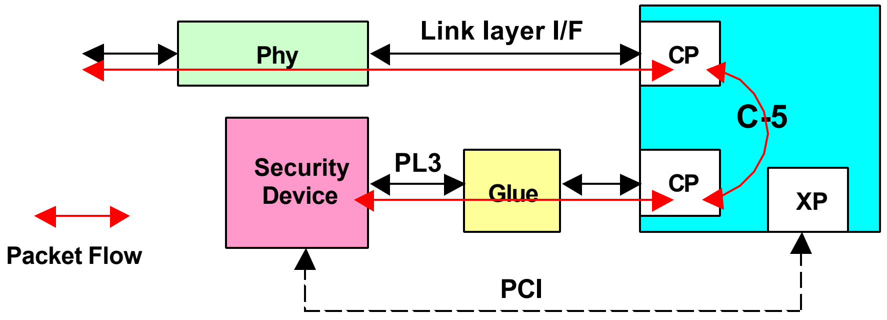
However, some devices connected to the network side may also be used in ‘in-line’ mode, in which all packets on a particular interface flow through the security device, and are processed as appropriate as they pass through it. Devices used in In-line mode must be able to detect ESP and AH headers on ingress packets and be able to apply ESP and AH headers to egress packets. Only the Corrent 7120 and (possibly) Hifn HIPP 8514 and NetOctave NSP3200 appear to have this capability.

In all cases a separate connection will be required to the XP’s PCI interface for control purposes.

7.2.1 Network side – look-aside mode

In this configuration, the security chip is connected using appropriate glue logic (FPGA and/or off-the-shelf components) to a dedicated CP or more likely CP cluster, as shown in the figure below.

Figure 7.1 – Network Side Connection – Look-aside Mode



Ingress and egress packet flows are shown in red. Ingress traffic to other CP clusters which may be encrypted is checked for the presence of an ESP or AH header. Secure packets are sent to the dedicated CP cluster for decryption/authentication. The packets (or only the packet payload, depending on the packet awareness of the device) are sent to the security device for processing and decrypted authenticated packets are then forwarded as appropriate.

Similarly all packets must be checked against the SPD on egress to determine whether the should have ESP and/or AH processing applied. If they do require processing, they are sent to the security cluster to be passed through the security device.

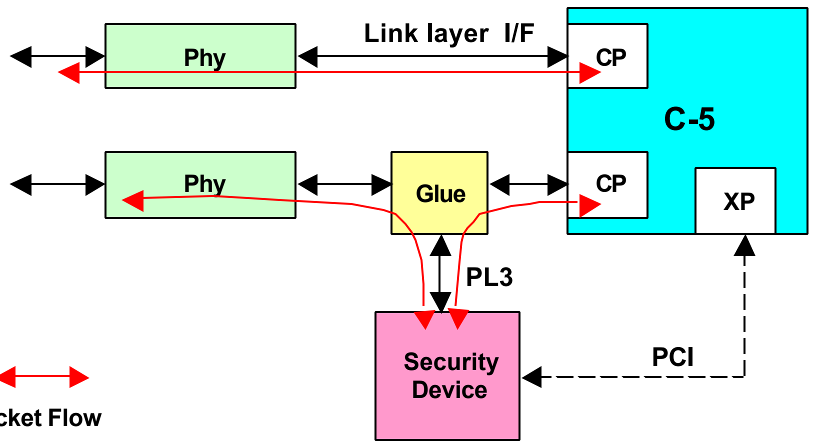
The nature of the glue logic depends on the interface provided by the security chip and the interface chosen at the CP. One possible combination is PL3 at the security device and GMII at the CP. PMC-Sierra have a part (The PM3386) which maps 2 x GMII interfaces to PL3.

7.2.2 Network Side – In-Line Mode

The alternative network side configuration is to place the security device in line with the packet flow. This is only possible if the device is highly packet aware. It must be able on ingress to detect and interpret ESP and AH headers and apply all necessary processing without any participation by the C-5 NP. On egress it must be able to create ESP and AH headers as directed by the C-5 NP, but without any further C-5 NP processing.

The following figure shows how a security device is connected in in-line mode. Again, the red line indicate packet flow.

Figure 7.2 – Network Side Connection – In-line Mode



The figure shows one interface which carries clear traffic and another carrying secure traffic. On the secure interace, ingress packets are diverted by the glue logic to the security device. If it detects an ESP or an AH it carries out all necessary IPSec processing and sends a clear packet to the ingress CP.

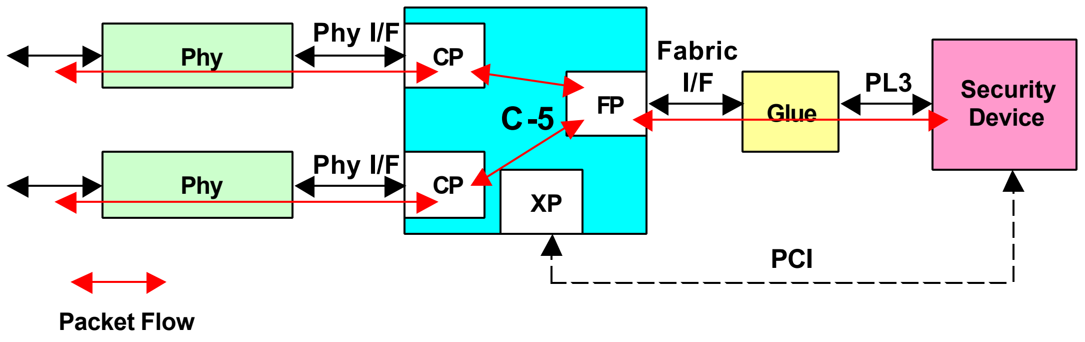
Egress packets on the secure interface are prepended with a header that indicates whether ESP, AH or no processing should be applied, and if so which SA should be used. The security device performs the required processing and send the secure packet out.

The Glue logic is similar to the previous case, but now requires two link interfaces and one PL3. Again the PM3386 could be used if GMII link layer interfaces were used.

7.2.3 Fabric Side – Look-Aside Mode

The last option is to place the security device on the fabric side of the C-5 NP and use it in look-aside mode. This configuration is shown in the figure below. The red lines indicate traffic flow.

Figure 7.3 – Fabric Side Connection – Look-aside Mode



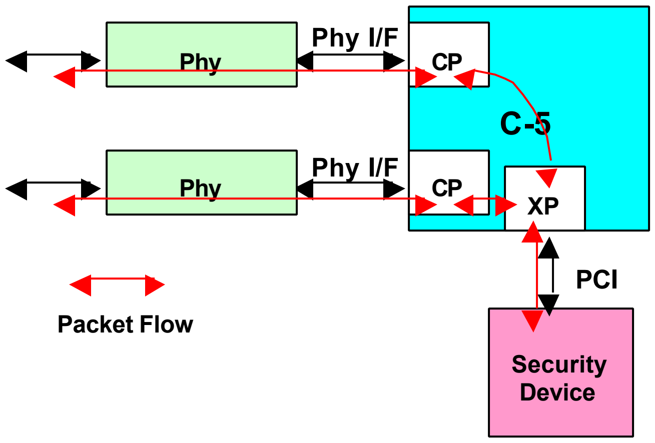
The operation is similar to the network side – look-aside mode case, except that traffic to be encrypted/decrypted is sent to the security device via the fabric port.

The glue logic now is required to bridge the fabric I/F (CSIX, UTOPIA etc.) to PL3. Agere Corporation has developed an FPGA to do exactly that.

7.2.4 PCI Connection – Look-Aside mode

For the sake of completeness, the devices not listed in section 7.1 can only be used in look-aside mode interfaced to the C-5 NP via the XP PCI bus for both control and data purposes, as shown below:

Figure 7.4 – PCI Connection – Look-aside Mode



This would be the method used for connecting the MPC190 to the C-5 NP. Clearly, due to the inherent limitations of PCI and the fact that the interface will also be carrying all host – C-5 NP traffic, this configuration can only be used if the data rate of secure traffic is relatively low.

References

[HMAC]	RFC 2104 ‘HMAC: Keyed-Hashing for Message Authentication’
[IPSEC]	RFC 2401 ‘Security Architecture for the Internet protocol’
[AH]	RFC 2402 ‘IP Authentication Header’
[MD5]	RFC 2403 ‘The Use of HMAC-MD5-96 within ESP and AH’
[SHA-1]	RFC 2404 ‘The Use of HMAC-SHA-1-96 within ESP and AH’
[DES]	RFC 2405 ‘The ESP DES-CBC Cipher Algorithm With Explicit IV’
[ESP]	RFC 2406 ‘IP Encapsulating Security Payload (ESP)’
[DOI]	RFC 2407 ‘The Internet IP Security Domain of Interpretation for ISAKMP’
[ISAKMP]	RFC 2408 ‘Internet Security Associatio n and Key Management Protocol (ISAKMP)’
[IKE]	RFC 2409 ‘The Internet Key Exchange (IKE)’
[TLS]	RFC 2246 ‘The TLS Protocol Version 1.0’
[CBC]	RFC 2451 ‘The ESP CBC-Mode Cipher Algorithms’
[RIPE]	RFC 2851 ‘The Use of HMAC-RIPEMD-160-96 within ESP and AH’