



MosChip Security Processor MCS1000

Software Manual IPSec Processing

Version 1.0

October 10, 2003

For more information contact the Technical Marketing Group:

info@moschip.com (408) 737-7141 x124

www.moschip.com

All information in this document is believed to be accurate as of the publish date.

VPN^{ow}™ is a registered trademark of Artec Group. ARM is a registered trademark of ARM Limited. All other brands or product names are the property of their respective holders.

MosChip Semiconductor products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of MosChip Semiconductor, Ltd.

MosChip Semiconductor believes the information in this document to be accurate and reliable. However, it is subject to change without notice. No responsibility is assumed by MosChip Semiconductor for its use, nor for infringement of patent or other rights of third parties. No part of this document may be reproduced, or transmitted in any form or by any means without prior consent of MosChip Semiconductor, Ltd.

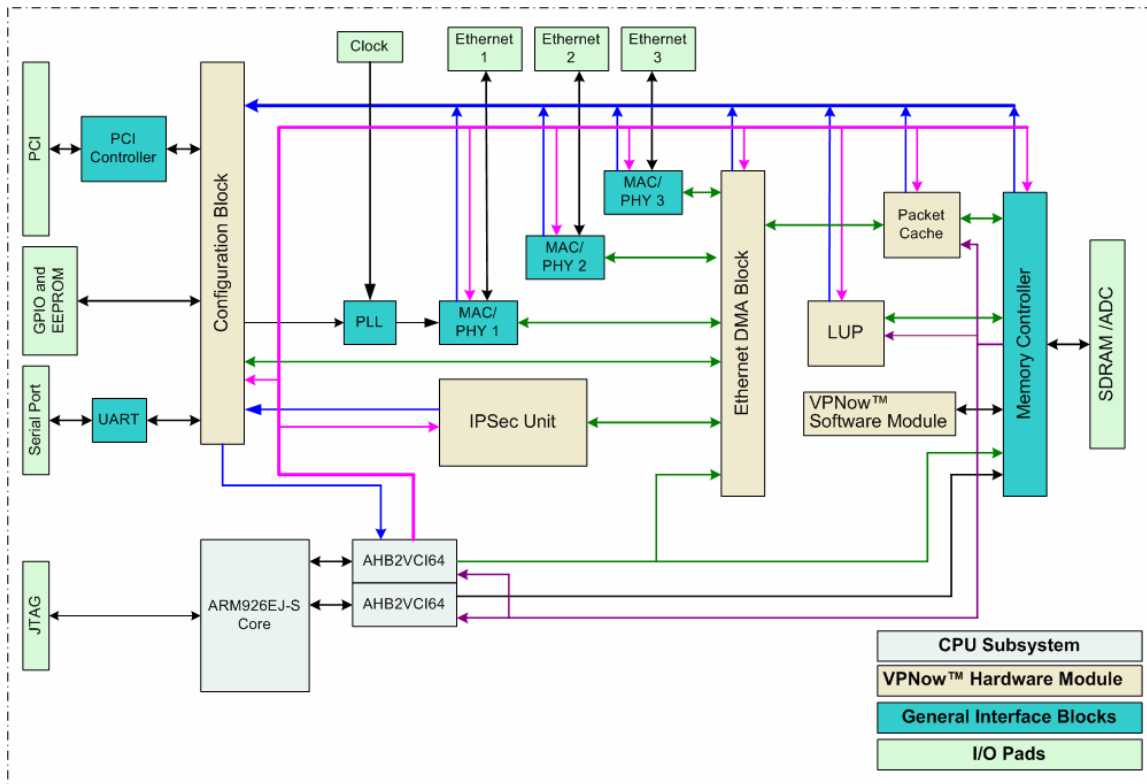
Copyright © 2003 MosChip Semiconductor All Rights Reserved.

1	Preface.....	1
2	Overview: IPsec processing in VPNNow™ Hardware Module.....	1
3	Preparing VPNNow™ Hardware Module for Use.....	4
3.1	Starting IPsec processing.....	4
3.2	Setting-up IP processing in VPNNow™ Module.....	4
3.2.1	Interface Settings.....	5
3.2.2	IPsec settings.....	8
3.3	Supported Cryptographic and Authentication Algorithms.....	12
3.4	Managing the Security Policy Database and Security Association Database... ..	15
3.4.1	Database command header.....	15
3.4.2	Selectors.....	17
3.4.3	SAD Entry.....	21
3.4.4	SPD entry.....	25
3.4.5	Commands for managing SPD and SAD.....	27
3.4.6	Example.....	46
4	Using VPNNow™ Hardware Module.....	49
4.1	Enabling IPsec processing.....	49
4.2	Reading VPNNow™ Hardware Module settings.....	49
4.2.1	Reading status.....	49
4.2.2	Reading interface settings.....	50
4.2.3	Reading IPsec settings.....	50
4.3	Log.....	51
4.4	Messages from VPNNow™ Hardware Module.....	54
4.5	About VPNNow™ Hardware Module.....	58
4.6	Stopping IPsec processing.....	59
4.6.1	Drop all packets being processed in the VPNNow™ Module at the moment. 59	
5	Appendix A: Constants.....	60
6	Appendix B: Commands.....	62
7	Appendix C: Messages.....	64
8	References.....	67

1 Preface

Any discussion of the IPsec processing within the MCS1000 should be referenced to the architecture of the MCS1000. See Figure. MCS1000 Architecture.

Figure. MCS1000 Architecture



The focus of this document is on the VPNNow™ Hardware Module (VHM) and the software functions that operate within this module. The VHM consists of a Configuration Block, IPsec Unit, Ethernet DMA Block, LUP and Packet Cache. The IPsec Unit consists of many blocks, these blocks are the DES, 3DES, and AES cryptography blocks; MD5 and SHA authentication blocks; and I/O blocks.

2 Overview: IPsec processing in VPNNow™ Hardware Module

The IPsec standard is designed to provide cryptographic-based security for IPv4 and IPv6 traffic [RFC_ARC]. The VPNNow™ Hardware Module provides encryption and authentication services at the data link layer. IPsec processing is inserted into the IP stack at the level between IP and Media Access layers of the network protocol stack. In the case of inbound traffic the VPNNow™ Hardware Module receives the packet from the

Media Access layer, finds the IPSec headers and performs the cryptographic processing. The packet is then delivered to the TCP/IP stack. In the case of outbound traffic the VPNow™ Hardware Module receives the packet from the TCP/IP stack, adds the IPSec headers and does the cryptographic processing. After the cryptographic calculations are performed the packet will be delivered for transmission. This mode is commonly referred to as a Bump In The Stack (BITS) implementation. [RFC_ARC] From the host's point of view there is no change to the existing IP stack. The host system prepares the IP packet for transmission as though there is no IPSec processing to take place. The VPNow™ Hardware Module only secures packets and does not run any routing algorithm.

The protection offered is based on the definitions provided by a Security Policy Database (SPD). The SPD is established and maintained by the host system. The security policy defines the services and manner of the IP packet processing. The SPD is consulted during the inspection of each IP packet, which has entered VPNow™ Hardware Module. Packets are matched against entries in the SPD. Each packet received can be treated in three different ways: a) afforded IPSec security services, b) discarded, or c) allowed to bypass IPSec processing. The local policy database created within the VPNow™ Hardware Module must be generated from an external source, such as a host processor or downloaded via the Ethernet.

The IPSec standard defines two protocols to provide traffic security -- Authentication Header (AH) [RFC_AH] and Encapsulating Security Payload (ESP) [RFC_ESP]. These protocols may be applied alone or in combination with each other to provide a desired set of security services in IPv4 and IPv6. Each protocol supports two modes of use: transport mode and tunnel mode. In transport mode the protocols provide protection primarily for upper layer protocols and this mode of IPSec is used when security is desired between two parties, which are also the communication endpoints. When both AH and ESP are used in transport mode, ESP should be applied first. The tunnel mode of IPSec is used when security is desired for packets which ultimate destination does not match the security termination point. In this case the IPSec protocols are applied to tunneled IP packets. The VPNow™ Hardware Module supports transport and tunnel modes.

A Security Association (SA) is created to afford protection to the traffic stream. The SA is used to specify the transforms that are used to provide capabilities of IPSec protocols and methods. The transforms define the transformation applied to the data to secure it: the algorithm, the keys, and any algorithmic-specific information. Each SA is uniquely identified by a three fields consisting of a Security Parameter Index (SPI), an IP Destination Address, and a security protocol (AH or ESP) identifier. In each IPSec implementation there is a Security Association Database (SAD), in which each entry defines the parameters associated with one SA. Each SA has an entry in the SAD. To secure bi-directional communication, two SAs are required. Outbound SAs are used to secure outgoing packets and inbound SAs are used to process inbound packets with IPSec headers. These SAs can be provided either through the use of an IKE (Internet Key Exchange), keyed in manually, or exchanged by some other mechanism. Sometimes it is

necessary to employ multiple SAs to implement the required security policy. The term "SA bundle" is used in this case.

In order to accomplish IPsec processing VPNNow™ Hardware Module follows the following steps:

- Maintains the interface between IPsec processing and the host system in order to facilitate the best performance
- Establishes and maintains the local SPD and SAD databases based on information provided by the host system or external devices
- Processes inbound traffic
- Processes outbound traffic
- Reassembles received fragments
- Maintains processing of the cryptographic accelerator

When a protected packet arrives the VPNNow™ Hardware Module examines the packet. If it is a fragment, then no further processing is performed until all fragments are assembled. The full IP packet is scanned in order to locate any IPsec headers. The headers are peeled off one-by-one using the SAs provided by the host system. For each inbound IP packet to which IPsec processing will be applied an SA look-up in the SAD is made. If no match is found in the SAD database an error condition is generated for transmission to the administrator (missing SA).

Each outbound packet is compared against the SPD to determine what processing is required for the packet. If IPsec processing is to be applied to the packet the SA entry will contain the rules and policies required for security processing. If an SA is not found and automated key exchange is not supported, then the VPNNow™ Hardware Module generates an error condition. This error condition (missing SA) is transmitted to the IPsec administrator. If the length of the protected IP packet, after adding IPsec headers, exceeds the maximum packet size that a network can transmit (Maximum Transmission Unit MTU), then VPNNow™ Hardware Module divides the packet into fragments. As the VPNNow™ Hardware Module is unaware of the path taken by a packet as it travels through a network, the host system has to supply the device with information about transmission capabilities of the interfaces.

The VPNNow™ Hardware Module supports encryption functions DES, 3DES and AES (with 128, 192 and 256 bit keys) in the CBC mode and hash functions SHA1 and MD5 in the HMAC mode. The IPsec Unit contains has an operator, which performs the necessary calculations for the IPsec. The block of cryptographic operators operates on the stream of the packet data supplemented with IPsec headers (AH, ESP). Header processing and construction is done before inserting it into the data stream.

3 Preparing VPNow™ Hardware Module for Use.

The MCS1000 and the host system communicate via messages where the Message ID specifies the command and the data field points to the data added to the message.

The protocol for the MCS1000 and the host system to communicate is described in this section.

To prepare the VPNow™ Hardware Module to function properly the system administrator should:

- Start IPsec processing.
- Set configuration parameters.
- Provide Module with policy rules.
- Enable IPsec processing.

The commands and data structures required in preparing and implementing the IPsec processing will be discussed in the following sections.

3.1 Starting IPsec processing

When reset is inactive and the MCS1000 commences operation the IPsec processing is not started and no protection is afforded to the IP traffic. To start IPsec processing a message with the command `VPNOW_START_IPSEC` should be sent to the MCS1000.

The command `VPNOW_START_IPSEC` causes the MCS1000 to reply with a message specifying the command `VPNOW_IPSEC_STARTED` if the IPsec processing was started successfully or with the command `VPNOW_IPSEC_NOT_STARTED` if an error (lack of memory, an internal error) occurred during the start-up procedure.

Though the IPsec processing is started the MCS1000 is not able to protect IP traffic before some configuration parameters have been set and policy databases SPD and SAD formed.

3.2 Setting-up IP processing in VPNow™ Module

Default configuration values are set during the start-up of the VPNow™ Hardware Module, however, additional configuration parameters can be loaded to adapt the operation of the VPNow™ Hardware Module. With the help of various configuration commands the operation of the module can be fine tuned to define the policy database that will be used to protect the traffic through the module.

Although the VPNow™ Hardware Module is configured with default configuration settings these settings may need to be modified to match the requirements specified by the system administrator. Setting-up IP processing should be done in two steps:

- Providing VPNow™ Hardware Module with IP addresses set for interfaces and with some other interface-specific information.
- Providing VPNow™ Hardware Module with information specifying functionality of IPsec processing.

There are commands to adjust the VPNow™ Hardware Module in order to adapt it to the requirements of the host system. The commands `VPNOW_SET_IP_CONF` and `VPNOW_SET_IPSEC_CONF` are used to set-up the VPNow™ Hardware Module. To read current settings of the module, the commands `VPNOW_GET_IP_CONF` and `VPNOW_GET_IPSEC_CONF` should be used.

To read or set the IPsec module configuration values a message with a pointer to the proper data is used. The Message ID specifies the configuration command. How the configuration data should be organized depends on the command and is described in following sections.

3.2.1 Interface Settings

The MCS1000 has three Ethernet interfaces. To set an IP address and other network specific parameters for an interface, the command `VPNOW_SET_IP_CONF` with the following structure is used:

```
struct T_VPNOW_IP_CONF {
    unsigned long int    Size : 16,
                       Version : 16;
    unsigned long int    Msg;
    unsigned long int    Eth : 1,
                       IPv4_DF_flag : 2,
                       IPv6 : 1,
                       MTU : 16;
    unsigned long int    Ip4;
    unsigned long int    Ip6[4];
};
```

Size

Value in this field specifies the length of configuration data (in bytes).

Version

The value in this field specifies the version of the configuration structure to ensure that there is uniformity in the data organization mode.

Current value is `VPNOW_IP_CONF_STRUCT_VER`.

Msg

If the structure is used with the command `VPNOW_ANSW_TO_SET_IP_CONF` then the value in the field *Msg* determines the result of the configuration command.

Eth

Specifies the Ethernet interface this configuration is for.

Possible values:

`VPNOW_ETH_0`

`VPNOW_ETH_1`

`VPNOW_ETH_2`

IPv4_DF_flag

A control flag bit in the IPv4 packet header ([RFC_IPv4]) specifies if the fragmentation of this packet is allowed or not. If the Don't Fragment flag (DF) bit is set, then fragmentation of this packet is not permitted.

The value in the field *IPv4_DF_flag* specifies, how to set Do Not Fragment flag in IPv4 mode tunneled packet encapsulating header. The system administrator must configure how to treat the DF bit for every interface. This is dependent on whether the Path MTU discovery is turned on or not for specified interface ([RFC_PMTU]).

Possible values for this field:

<code>VPNOW_IPSEC_COPY_DF_FLAG</code>	copy from inner IP header if IPv4 mode or clear if inner packet is in IPv6 mode
<code>VPNOW_IPSEC_CLEAR_DF_FLAG</code>	clear DF flag
<code>VPNOW_IPSEC_SET_DF_FLAG</code>	set DF flag

By default module sets this flag to `VPNOW_IPSEC_COPY_DF_FLAG`.

IPv6

Determines if IP packets in IPv6 mode are permitted through interface specified in *Eth*.

Possible values:

<code>VPNOW_IPV6_ENABLED</code>	IPv6 mode traffic is enabled through this interface
<code>VPNOW_IPV6_DISABLED</code>	IPv6 mode traffic is disabled through this interface

The current version of the VPNow™ Hardware Module does not have IPv6 support, so this field is set to `VPNOW_IPV6_DISABLED`.

MTU

Specifies the first-hop data-link Maximum Transmission Unit for interface specified in the field *Eth*. Value of this field must be from range

VPNOW_MIN_PATH_MTU - VPNOW_MAX_PATH_MTU.

By default MTU is set to VPNOW_MAX_PATH_MTU.

Ip4

Specifies a 32-bit number that represents an Internet Protocol address for interface specified in *Eth*. The VPNow™ Hardware Module uses this value for a source IPv4 address when constructs a tunneled packet.

Ip6[4]

Specifies an IPv6 address (array of 32-bit numbers) for Ethernet interface specified in *Eth*. The VPNow™ Hardware Module uses this value for a source IPv6 address when constructs a tunneled packet.

NOTE:

An IP address must be expressed as a number in the IP's network order. For example an IPv4 address 192.168.100.210 should be expressed as hexadecimal C0A864D2.

To configure the MCS1000's Ethernet interfaces, a message with the command VPNOW_SET_IP_CONF and with data organized with help of structure (defined above) T_VPNOW_IP_CONF should be sent to the MCS1000. Fields *Size*, *Version*, *Eth*, *Ip4*, *IPv4_DF_flag* and *MTU* must be transmitted with the correct values when the command VPNOW_SET_IP_CONF is sent.

The command VPNOW_SET_IP_CONF generates a reply from the VPNow™ Hardware Module. The contents of the reply are specified in the command VPNOW_ANSW_TO_SET_IP_CONF. The data part of the reply includes a pointer to the structure T_VPNOW_IP_CONF sent by the host system. After the command the parameter in the field *Msg* indicates result of configuration as shown below:

VPNOW_ERR_BAD_IP_CONF_STRUCT_VER

Specifies that current version number and version number given with command in the field *Version* do not match.

VPNOW_ERR_BAD_IP_CONF_STRUCT_SIZE

Specifies that size of current structure and size given with command are not equal.

VPNOW_ERR_BAD_IP_CONF_ETH_VAL

Specifies that the VPNow™ module can't set values for Ethernet interface with the number given in the field *Eth*. This answer may occur for two reasons: the VPNow™ module does not support the Ethernet interface with the number given in the field *Eth* or interface is not available at the moment.

VPNOW_ERR_BAD_IP_CONF_MTU_VAL

Specifies that value in the field *MTU* is not from range
VPNOW_MIN_PATH_MTU to VPNOW_MAX_PATH_MTU.

VPNOW_ERR_BAD_IPSEC_CONF_DF_FLAG

Specifies that the value in the field *IPv4_DF_flag* is not in the set
{VPNOW_IPSEC_COPY_DF_FLAG,
VPNOW_IPSEC_CLEAR_DF_FLAG,
VPNOW_IPSEC_SET_DF_FLAG}.

VPNOW_OPERATION_OK

Specifies that the Ethernet interface has been set up successfully.

3.2.2 IPsec settings

To specify how the IPsec processing functions the command
VPNOW_SET_IPSEC_CONF with the following structure is used:

```
struct T_VPNOW_IPSEC_CONF {  
  
    unsigned long int    Size : 16,  
                        Version : 16;  
    unsigned long int    Msg;  
    unsigned long int    ProcMethod : 1,  
                        Log : 4,  
                        DataSize : 8,  
                        IPv6_support : 1,  
                        PadMode : 2;  
    unsigned long int    FragmentWaitTime : 16,  
                        CiphWaitTime : 16;  
  
};
```

Size

Value in this field specifies length of configuration data (in bytes).

Version

The value in this field specifies the version of the configuration structure to ensure that there is uniformity in the data organization mode.

Current value is VPNOW_IPSEC_CONF_STRUCT_VER.

Msg

If the structure is used with the command `VPNOW_ANSW_TO_SET_IPSEC_CONF` then the value in the field *Msg* determines the result of the configuration command.

ProcMethod

Specifies how the IPsec processing will be applied on inbound protected packets. The host system may want to process the protected packets itself.

<code>VPNOW_USE_IPSEC_ALL</code>	send an inbound protected packet to IPsec module
<code>VPNOW_USE_IPSEC_BYPASS</code>	IPsec module must bypass all protected inbound packets

Default value is `VPNOW_USE_IPSEC_ALL`.

Log

Specifies the contents of the log data and the following are valid switches:

<code>VPNOW_IPSEC_LOG_ENABLED</code>	If the corresponding bit is set then the log is enabled and the host will be informed about processed packets and errors.
<code>VPNOW_DEBUG_LOG_ENABLED</code>	If the corresponding bit is set – a data dump is added to the log header if an internal error occurs.
<code>VPNOW_AH_AUDIT_ENABLED</code>	If the corresponding bit is set then AH auditing is enabled.
<code>VPNOW_ESP_AUDIT_ENABLED</code>	If the corresponding bit is set then ESP auditing is enabled.

Default setting for this field is:

```
VPNOW_IPSEC_LOG_ENABLED |
VPNOW_AH_AUDIT_ENABLED |
VPNOW_ESP_AUDIT_ENABLED
```

DataSize

Specifies the number of bytes to be added to the log info in the debug mode. It is used when flag `IPSEC_DEBUG_LOG` is set. The default value is:

```
VPNOW_DEFAULT_DEBUG_DATA_SIZE.
```

Maximum is specified with `VPNOW_MAX_DEBUG_DATA_SIZE`.

IPv6_support

Specifies whether or not the VPNow™ Hardware Module supports IPv6 mode traffic.

Possible values:

VPNOW_IPV6_SUPPORTED	IPv6 mode traffic is supported
VPNOW_IPV6_NOT_SUPPORTED	IPv6 mode traffic is not supported

The current version of the VPNow™ Hardware Module does not have IPv6 support, so this field is set to VPNOW_IPV6_NOT_SUPPORTED.

PadMode

Specifies how data for encryption will be padded to ensure 32-bit unit boundary:

VPNOW_IPSEC_ENCR_PAD_SERIES	end of data will be filled with series of data
VPNOW_IPSEC_ENCR_PAD_ZERO	end of data will be filled with zeroes
VPNOW_IPSEC_ENCR_PAD_RANDOM	end of data will be filled with random data

Default setting for this field is VPNOW_IPSEC_ENCR_PAD_SERIES.

FragmentWaitTime

Specifies maximum time in milliseconds to wait for missing fragments of an IP packet.

The value may not exceed the maximum VPNOW_FRAGMENT_WAIT_MAX_TIME

Default value is VPNOW_FRAGMENT_WAIT_TIME.

CiphWaitTime

Specifies maximum time in milliseconds to wait for crypto operations if a collision in the IPsec Unit appears.

Value must not exceed maximum value VPNOW_CRYPT_WAIT_MAX_TIME

Default value is VPNOW_CRYPT_WAIT_TIME.

To set the IPsec specific parameters for module, a message with the command VPNOW_SET_IPSEC_CONF and with the configuration data organized in the correct structure T_VPNOW_IPSEC_CONF should be sent to the MCS1000.

The command VPNOW_SET_IPSEC_CONF causes the VPNow™ module to reply with a message specifying the command VPNOW_ANSW_TO_SET_IPSEC_CONF. The data part of the message includes a pointer to the structure T_VPNOW_IPSEC_CONF sent by the host system. After the command the parameter in the field *Msg* indicates result of configuration as shown below:

VPNOW_ERR_BAD_IPSEC_CONF_STRUCT_VER

Specifies that the current version number and version number given with command in the field *Version* do not match.

VPNOW_ERR_BAD_IPSEC_CONF_STRUCT_SIZE

Specifies that the size of current structure and the size given with the command are not equal.

VPNOW_ERR_BAD_FRAG_WAIT_TIME

Specifies that the value in the field *FragmentWaitTime* is greater than VPNOW_FRAGMENT_WAIT_MAX_TIME.

VPNOW_ERR_BAD_CIPH_WAIT_TIME

Specifies that the value in the field *CiphWaitTime* is greater than VPNOW_CRYPTO_WAIT_MAX_TIME.

VPNOW_ERR_BAD_IPSEC_CONF_PROC_METHOD

Specifies that the value in the field *ProcMethod* is not from the set {VPNOW_USE_IPSEC_ALL, VPNOW_USE_IPSEC_BYPASS}.

VPNOW_ERR_BAD_PAD_MODE

Specifies that the value in the field *PadMode* is not from the set {VPNOW_IPSEC_ENCR_PAD_SERIES, VPNOW_IPSEC_ENCR_PAD_ZERO, VPNOW_IPSEC_ENCR_PAD_RANDOM}.

VPNOW_OPERATION_OK

Specifies that the command has been processed successfully.

If the value in the field *DataSize*, which specifies the size of the data dump, exceeds the VPNOW_MAX_DEBUG_DATA_SIZE value then the configuration will be set to VPNOW_MAX_DEBUG_DATA_SIZE without any error messages.

3.3 Supported Cryptographic and Authentication Algorithms

There are the mandatory transforms to implement for the IPSec-based protocols. The IPSec Unit is able to process the following cryptographic and authentication algorithms:

Name	IANA nr	VPNOW™ nr	Key length	Sign size
NULL_ENCRYPTION	11	VPNOW_ENCR_NULL		
DES_CBC	2	VPNOW_ENCR_DES	2	
DES3_CBC	3	VPNOW_ENCR_DES3	6	
AES_128	12	VPNOW_ENCR_AES_128	4	
AES_192	12	VPNOW_ENCR_AES_192	6	
AES_256	12	VPNOW_ENCR_AES_256	8	
HMAC_SHA1	3	VPNOW_AUTH_HMAC_SHA1	1, . . . , 5	5
HMAC_SHA2	4	VPNOW_AUTH_HMAC_SHA2	1, . . . , 8	8
HMAC_MD5	2	VPNOW_AUTH_HMAC_MD5	1, . . . , 4	4

The first six rows in the table above specify encryption transforms and the last three with names *HMAC_SHA1*, *HMAC_SHA2*, and *HMAC_MD5* specify authentication algorithms.

Name specifies a name given for algorithm in the VPNOW™ Hardware Module.

IANA nr specifies IANA number given for this algorithm.

VPNOW™ nr specifies an internal number given for algorithm in the VPNOW™ Hardware Module.

Key length specifies length of the key in double words (32-bit units) for current algorithm. For encryption transforms size of a key is fixed. If an authentication algorithm is used for protection size of the key can be set externally. In this case value for the key size must be chosen from the allowed range given in the table.

Sign size specifies the maximum length of the signature (the Integrity Check Value (ICV)) in double words calculated by authentication algorithm. The ICV will be added to the IP packet at the time of IPSec processing and may be truncated (the first 96 bits are used). The actual length for ICV can be set at the time of describing policy rules and corresponding SA(s).

To get the list of crypto algorithms currently supported by the MCS1000 a message with the command `VPNOW_GET_SUPPORTED_ALGORITHMS` should be sent to the MCS1000. Data part of the message should be NULL.

After receiving a message specifying command `VPNOW_GET_SUPPORTED_ALGORITHMS` the IPSec Module replies with the command `VPNOW_ANSW_TO_GET_SUPPORTED_ALGORITHMS`.
When the MCS1000 provides the host system with the supported encryption and authentication algorithms the following structures will be filled. The data part of the message points to the header of the list of algorithm descriptions:

```
struct T_VPNOW_CRYPTO_ALG_LIST {  
  
    unsigned long int    Size : 16,  
                        Version : 16;  
    unsigned long int    Count;  
};
```

Size

The value in this field specifies the number of bytes of data including header and algorithm descriptions.

Version

The value in this field specifies the version of the list structure to ensure that both parts of conversation use the same data organization mode.

Current value is `VPNOW_CRYPTO_ALG_LIST_STRUCT_VER`.

Count

Specifies the number of algorithm descriptions in the list.

The header part is followed by an array of structures describing available crypto algorithms:

```
struct T_VPNOW_CRYPTO_ALG {  
  
    unsigned long int    Size : 16,  
                        Version : 16;  
    unsigned char        Name[VPNOW_ALG_NAME_LEN];  
    unsigned long int    IANA_ID : 5,  
                        VPNOW_ID : 5;  
                        Type : 1,  
                        MinKeyLen : 7,  
                        MaxKeyLen : 8,  
                        SignSize : 6;  
};
```

Size

The value in this field specifies the length of the algorithm description in bytes.

Version

The value in this field specifies the version of the algorithm's description structure to ensure that both parts of the transaction use the same data organisation mode. Current value is `VPNOW_CRYPTO_ALG_STRUCT_VER`.

Name[]

Name for an algorithm (table above). Specifies a character string, `VPNOW_ALG_NAME_LEN` characters in length.

IANA_ID

Specifies IANA number given for this algorithm (table above).

VPNOW_ID

Specifies an internal number for algorithm (table above).

Type

Specifies type of algorithm:

<code>VPNOW_AUTH_ALG</code>	authentication algorithm
<code>VPNOW_ENCR_ALG</code>	encryption algorithm

MinKeyLen

Specifies the minimum size of the key in double words that can be used with the specified authentication algorithm in the MCS1000 IPsec processing.

MaxKeyLen

Specifies the maximal size of the key in double words that can be used with the authentication algorithm in the MCS1000 IPsec processing.

SignSize

Specifies the maximum size of the signature in double words for the authentication algorithms.

3.4 Managing the Security Policy Database and Security Association Database

There are two databases involved in IPsec processing: the Security Policy Database (SPD) and the Security Association Database (SAD). The SPD specifies the policies that determine the offered security for all inbound and outbound IP traffic. The SAD maintains the list of active security associations (SAs) for outbound and inbound processing. To secure IP traffic between two communicating IPsec enabled systems, two SAs are required: one to protect inbound traffic and one to protect outbound traffic. The VPNNow™ Security Module creates a local SPD and SAD from the data provided by the host system. The protection offered is based on requirements defined by the SPD and means specified by the SAD.

Separate inbound and outbound SPD and SAD tables for both IP modes should be maintained for every Ethernet interface of the MCS1000. The SPD can be seen as a set of separate tables.

There are several structures for managing the IPsec policy database. A database command header structure must be completed to specify which table of the policy database will be changed or inspected for each database command.

3.4.1 Database command header

The data part of each database command starts with a database command header described with the following structure:

```
struct T_VPNOW_IPSEC_DBASE_CMND {  
  
    unsigned long int    Size : 16,  
                        Version : 16;  
  
    unsigned long int    Msg;  
    unsigned long int    PolicyIndex;  
    unsigned long int    BundleID;  
    unsigned long int    SaID;  
    unsigned long int    Eth;  
    unsigned long int    Dir;  
    unsigned long int    IpMode;  
  
};
```

Size

The value in this field specifies the number of bytes of data added to the database command.

Version

The value in this field specifies the version of the header structure to ensure that both parts of the transaction use the same data organization mode.

The current value is `VPNOW_IPSEC_DBASE_CMND_STRUCT_VER`.

Msg

Specifies the result of database command.

PolicyIndex

Determines row number in the database (SPD) table specified with values in the fields *Eth*, *Dir*, and *IpMode*. The rules in the SPD are checked in order, so the order in which they are specified is important. The number of the first row in the table is 0.

BundleID

Specifies the ID given by the Database Manager to the current bundle of Security Associations.

SaID

Specifies the ID given by Database Manager to the current Security Association (SA).

Eth

Specifies Ethernet interface number:

`VPNOW_ETH_0`

`VPNOW_ETH_1`

`VPNOW_ETH_2`

Dir

Specifies direction of traffic this command is for:

`VPNOW_DIR_INBOUND`

Specifies inbound traffic.

`VPNOW_DIR_OUTBOUND`

Specifies outbound traffic.

IpMode

Specifies IP mode of packets this command is for:

VPNOW_IP4_MODE

Specifies IPv4 mode packets.

VPNOW_IP6_MODE

Specifies IPv6 mode packets.

For example header

```
Size = 32
Version = VPNOW_IPSEC_DBASE_CMND_STRUCT_VER
Msg = 0
PolicyIndex = 2
BundleID = 0
SaID = 0
Eth = VPNOW_ETH_1
Dir = VPNOW_DIR_OUTBOUND
IpMode = VPNOW_IP4_MODE
```

Specifies a database command for managing policy table for outbound IPv4 mode traffic traveling through interface 1.

3.4.2 Selectors

The processing mode for each IP packet is determined by matching values in the fields from the IP and transport layer headers of the IP packet against policy rules in the SPD. The key components (referred to as selectors) in a rule are:

- Source IP address
- Destination IP address
- Transport layer protocol
- Source port
- Destination port

Selectors in the SPD and SAD are used to define the set of IP traffic for policy entry and SA and may specify distinct values, ranges of values, or a wild card.

In this way the granularity with which a security service is offered can be controlled. The values for selectors of two SPD or SAD entries may overlap.

The following structure is used to describe selectors for a policy rule and an SA:

```

struct T_VPNOW_IPSEC_SELECTORS {

    unsigned long int    SourceIPAddr1[4];
    unsigned long int    SourceIPAddr2[4];
    unsigned long int    DestIPAddr1[4];
    unsigned long int    DestIPAddr2[4];
    unsigned long int    SourcePort1 : 16,
                        SourcePort2 : 16;
    unsigned long int    DestPort1 : 16,
                        DestPort2 : 16;
    unsigned long int    Protocol : 8,
                        SourceIPAddrFieldType : 3,
                        DestIPAddrFieldType : 3,
                        SourcePortType : 2,
                        DestPortType : 2,
                        ProtocolType : 2;
};
  
```

SourceIPAddr1, DestIPAddr1

Fields *SourceIPAddr1* and *DestIPAddr1* specify an IPv4 or IPv6 address. IP mode of the address depends on the value given in the field *IpMode* of database command header structure. In the case of IPv4 mode only the first 32-bit unit of address-field is used. How to interpret the values in these fields is determined by values in the fields *SourceIPAddrFieldType* and *DestIPAddrFieldType* respectively.

SourceIPAddr2, DestIPAddr2

Fields *SourceIPAddr2* and *DestIPAddr2* may specify upper bound of an IPv4 or IPv6 address range, an IPv4 or IPv6 network mask, or may be not used. IP mode of the address depends on the value given in the field *IpMode* of database command header structure. In the case of IPv4 mode only the first 32-bit unit of address-field is used. How to interpret values these fields is determined by values in the fields *SourceIPAddrFieldType* and *DestIPAddrFieldType* respectively.

SourceIPAddrFieldType, DestIPAddrFieldType

Values in the fields *SourceIPAddrFieldType* and *DestIPAddrFieldType* determine how to interpret values in the fields *DestIPAddr1*, *SourceIPAddr1*, *DestIPAddr2*, and *SourceIPAddr2*. These fields may have values from following set

VPNOW_ANY_VALUE

Denotes that all IP source (or destination) addresses match this SA entry or policy rule. The values in the fields *SourceIPAddr1* and *SourceIPAddr2* (or *DestIPAddr1* and *DestIPAddr2*) will be not used in this case.

VPNOW_IP_ADDR

Type denotes that the value in the field *DestIPAddr1* (or *SourceIPAddr1*) is an IP address and the value in the field *SourceIPAddr2* (or *DestIPAddr2*) will be not used.

VPNOW_IP_ADDR_SUBNET

Type specifies a range of IP addresses, represented by two values. The first value is an IP address given in the field *SourceIPAddr1* (or *DestIPAddr1*). The second is an IP network mask given in the field *SourceIPAddr2* (or *DestIPAddr2*).

VPNOW_IP_ADDR_RANGE

Type specifies a range of IP addresses, represented by two values. The first value given in the field *SourceIPAddr1* (or *DestIPAddr1*) is the beginning IP address (inclusive) and the second value given in the field *SourceIPAddr2* (or *DestIPAddr2*) is the ending IP address (inclusive). All addresses falling between the two specified addresses are considered to be within the list.

NOTE: Ones (1s) in the network mask indicate that the corresponding bit in the address is fixed, while zeros (0s) indicate a "wildcard" bit.

SourcePort1, DestPort1

Fields *SourcePort1* and *DestPort1* specify Internet service port numbers. How to interpret given port numbers depends on the value in the fields *SourcePortType* and *DestPortType* respectively.

SourcePort2, DestPort2

Fields *SourcePort2* and *DestPort2* may specify upper bound of the port range or may not be used. How to interpret values in these fields is determined by the values in the fields *SourcePortType* and *DestPortType* respectively.

SourcePortType, DestPortType

These fields determine how to interpret values in the fields *SourcePort1*, *DestPort1*, *SourcePort2*, and *DestPort2*. Fields *SourcePortType* and *DestPortType* may have values from the following set:

VPNOW_ANY_VALUE

Denotes that all port numbers match this rule. Values of source and destination ports may be inaccessible in the IP packet header because of encryption or fragmentation. The value *VPNOW_ANY_VALUE* indicates this variant also.

VPNOW_VALUE_FROM_FIELD

Indicates that the port value is specified in the field or *SourcePort1* or *DestPort1* respectively.

VPNOW_IP_PORT_RANGE

Type specifies a range of IP port numbers, represented by two values. The first value given in the field *SourcePort1* (or *DestPort1*) is the beginning IP port (inclusive) and the second value given in the field *SourcePort2* (or *DestPort2*) is the ending IP port (inclusive). All port numbers falling between the two specified values are considered to be within the list.

Protocol

This field specifies transport level protocol number ([RFC_AN]). If the value in the *ProtocolType* field is VPNOW_ANY_VALUE then the value in the *Protocol* field is not used.

ProtocolType

This field determines how to interpret the value in the field *Protocol*. Field *ProtocolType* may have values from the following set:

VPNOW_ANY_VALUE

Denotes that all transport level protocols are allowed. The transport layer protocol may be inaccessible because of encryption or fragmentation. The value VPNOW_ANY_VALUE indicates this variant also.

VPNOW_VALUE_FROM_FIELD

Indicates that the transport level protocol is specified in the field *Protocol*.

The system administrator should take into consideration that every table in the SPD specifying rules for traffic with distinct IP mode, direction and interface must be ordered. For the better performance the policy entries should be arranged in (lexicographically) increasing order by the following selector values: *SourceIPAddr1*, *DestIPAddr1*, *SourcePort1*, *DestPort1*, and *Protocol*. That means: the rules should be ordered by the value in the field *SourceIPAddr*. Then the rules with equal *SourceIPAddr* must be rearranged by the value in *DestIPAddr*. Then the rules with equal *SourceIPAddr* and *DestIPAddr* must be rearranged by the value in *SourcePort*, and so on. The value specified as VPNOW_ANY_VALUE (wild card) should be treated as the greatest.

For example a part of an ordered table:

<i>SourceIPAddr1</i>	<i>SourceIPAddr2</i>	<i>DestIPAddr1</i>	<i>DestIPAddr2</i>	<i>SourcePort1</i>	...
192.168.1.0	255.255.255.0	192.168.200.0	255.255.255.0	1000	
192.168.1.0	255.255.255.0	192.168.200.0	255.255.255.0	2001	
192.168.2.1	192.168.2.1	192.168.2.2	192.168.2.2	1234	
192.168.100.0	255.255.255.0	192.168.200.0	255.255.255.0	77	
...	
*	*	192.168.100.210	192.168.100.210	700	

3.4.3 SAD Entry

The Security Association (SA) specifies the type and methods of IPSec processing that is applied:

- IPSec protocol (AH or ESP)
- Protection mode (transport or tunnel)
- Algorithm(s)
- Key(s)
- Granularity of the traffic

SAs are unidirectional and protocol specific, i.e., there should be separate SAs for outbound and inbound processing and there must be an SA for each protocol used for protection.

Following structure describes an SA:

```
struct T_VPNOW_IPSEC_SA_ENTRY {  
  
    unsigned long int    Size : 16,  
                        Version : 16;  
    T_VPNOW_IPSEC_SELECTORS Selectors;  
    unsigned long int    SPI;  
    unsigned long int    DestinationIP [4];  
    unsigned long int    SNC;  
    unsigned long int    TunOptSize;  
    unsigned long int    IPsecProt : 1,  
                        IPsecMode : 2,  
                        TunnelIPmode : 1,  
                        IVmode : 1,  
                        ICVLen : 6,  
                        AuthKeyLen : 8;  
    unsigned long int    AuthAlgType : 8,  
                        EncrAlgType : 8,  
                        PathMTU : 16;  
    unsigned long int    ByteCount;  
    unsigned long int    TunnelTTL;  
    unsigned char        TunnelMAC[8];  
};
```

Size

Value in this field specifies the number of bytes in the structure `T_VPNOW_IPSEC_SA_ENTRY`.

Version

The value in this field specifies the version of the Security Association structure to ensure that both parts of the transaction use the same data organisation mode.

Current value is `VPNOW_IPSEC_SA_ENTRY_STRUCT_VER`.

Selectors

The values in the fields *SourceIPAddr1*, *SourceIPAddr2*, *DestIPAddr1*, *DestIPAddr2*, *SourcePort1*, *SourcePort2*, *DestPort1*, *DestPort2*, and *Protocol* in the structure *Selectors* define the granularity of the current SA. For example, a separate SA can be created for each TCP connection between two IPsec enabled systems, or a single SA protects all the traffic between these systems.

SPI

The *SPI* (Security Parameter Index [RFC_AH], [RFC_ESP]) is a 32-bit value to identify the Security Association (SA) and is included into AH and ESP headers. SPI value 0 is invalid and values 1-255 are reserved to IANA. The system administrator should choose the value for SPI from the range 0x100 – 0xFFFFFFFF.

DestinationIP

An IPv4 or IPv6 address to specify the SA connection endpoint. The value in this field in conjunction with the value in the fields *SPI* and *IPSecProt* uniquely identifies the SA. Whether this field represents an IPv4 or IPv6 address is specified by the value in the field *TunnelIPmode* (in the case of tunnel mode SA) or by the value in the field *IpMode* of the database command header (in the case of transport mode SA). The IPv4 address value will be taken from the first field of the array `DestinationIP[4]`.

TunnelMAC

Specifies a 6-byte hexadecimal address used by the media access control (MAC) layer. The value in this field determines the address for Ethernet interface of the tunnelled packet endpoint.

For example an address 00:12:34:56:78:9A must be expressed as an array with the octets in the same order in which they would appear in the header of an Ethernet frame:

`TunnelMAC [] = {00, 12, 34, 56, 78, 9A}`.

TunnelTTL

Specifies the value to put into the field *Time To Live* in the IPv4 tunnel IP header.

TunOptSize

IPv4 options or IPv6 extended headers (IP mode is specified in *TunnelIPmode*) can be added when a tunnel IP header is created. The value in the field *TunOptSize* specifies the size of options or extended headers in bytes to be added to the tunneled packets. The value 0 specifies that there will be no extensions. If options (or extended headers) should be added; the value in the field *TunOptSize* is greater than 0; then an SA entry must be followed by the data forming options (or extended headers) to be added to the tunnel IP header. The value in the field *Size* must count size of the options also.

IPSecProt

Specifies protocol AH or ESP for IPSec processing:

VPNOW_IPSEC_PROTOCOL_AH

Specifies that an AH header will be created during IPSec processing to authenticate the IP packet.

VPNOW_IPSEC_PROTOCOL_ESP

Specifies that an ESP header will be created during IPSec processing to authenticate and/or encrypt the IP packet.

IPSecMode

Specifies the IPSec processing mode:

VPNOW_IPSEC_MODE_TRANSPORT

Specifies that the IPSec header specified in *IPSecProt* should be built and put between the IP header and the packet payload.

VPNOW_IPSEC_MODE_TUNNEL

Specifies that a new IP header should be created and the IPSec header specified in *IPSecProt* should be built and put between this new IP header and the original IP header.

TunnelIPmode

Specifies which type of tunnel header should be built during IPSec processing given that the field *TunnelIP* is IPv4 or IPv6 address. Possible values are *VPNOW_IP4_MODE* and *VPNOW_IP6_MODE*.

SNC

This 32-bit field specifies the Sequence Number Counter value from the IPSec header AH and ESP ([RFC_AH], [RFC_ESP]) and is used to detect replay attacks by the destination. When the SA is created this field is set to 0 and every time the SA is used the value in this field is increased by 1. So the value in this field specifies how many times this SA has been taken into use to protect outbound packets.

IVmode

Specifies whether or not an initial vector data is included in the given SA:

VPNOW_IV_VALUE_FROM_PACKET

Specifies that IV data for encryption algorithm should be in the start of IP packet payload

VPNOW_IV_VALUE_FROM_SA

Specifies that IV data will be given after the encryption key data.

ICVLen

Specifies length of the signature in 32-bit units that will be added to the IP packet data. Signature computed by algorithm may be truncated to 96 bits.

AuthAlgType

Specifies the authentication algorithm from the following set:

VPNOW_AUTH_HMAC_MD5

VPNOW_AUTH_HMAC_SHA1

VPNOW_AUTH_HMAC_SHA2

VPNOW_AUTH_SHA1

VPNOW_AUTH_MD5

If authentication is not required then it has value
VPNOW_ALGORITHM_NOT_SPECIFIED.

EncrAlgType

Specifies encryption algorithm from the following set:

VPNOW_ENCR_AES_128

VPNOW_ENCR_AES_192

VPNOW_ENCR_AES_256

VPNOW_ENCR_DES

VPNOW_ENCR_DES3

VPNOW_ENCR_NULL

If encryption is not required then the default is
VPNOW_ALGORITHM_NOT_SPECIFIED.

NOTE: fields *AuthAlgType* and *EncrAlgType* cannot both be set to
VPNOW_ALGORITHM_NOT_SPECIFIED at the same time.

AuthKeyLen

Specifies number of double byte words for the key for authentication algorithm specified with field *AuthAlgType*. The value must be from the range s

PathMTU

Specifies the smallest maximum transmission unit (MTU) of any link on the current path between two hosts controlled by the current SA. If the value of the MTU for current path is unknown the value in this field should be set to the first-hop data-link MTU.

Adding IPsec protocols to a packet may lengthen it so that final size exceeds the throughput capabilities of the current path. In this case the IP packet should be fragmented after the IPsec processing. To avoid fragmentation the VPNNow™ Hardware Module:

- Checks all incoming packets with an ICMP message saying a packet is too large and proposing actual value for the MTU ([RFC_ICMPv4], [RFC_PMTU], [RFC_ICMPv6], [RFC_PMTUv6]).
- With help of the information returned in the ICMP message determines SA(s) applied to the original packet.
- Computes the size of the headers that are added by the IPsec processing according to the determined policy rule.
- Decreases the MTU value (reported in the ICMP message) by amount of the IPsec headers overhead. If the actual value for the MTU is not reported in the ICMP message then the current *PathMTU* is decreased by amount of the required IPsec headers, which will be added to the outbound packet.
- Forwards the adjusted MTU to the host system. The transport level protocol should use this new value to compose packets with an optimal size.
- Replaces the value in the field *PathMTU* with the new value.

ByteCount

The VPNNow™ Hardware Module counts the bytes to which the IPsec protection is applied. If value in the field *IPSecProt* specifies AH then all bytes to authenticate will be counted. If ESP protocol is used, bytes to encrypt/decrypt will be counted. The value in this field lets the system administrator to know to how many bytes an SA has been applied since creation. From the host system point of view this is a read only field.

If tunnel mode is required, then the VPNNow™ Hardware Module processing needs to be able to construct the encapsulating IP header for an outbound packet. The system administrator must provide the VPNNow™ Hardware Module with information about the outer IP header style as defined by the fields *TunnelIPmode*, *DestinationIP*, *TunnelMAC*, *TunnelTTL*, and *TunOptSize*.

3.4.4 SPD entry

Every inbound and outbound packet is subject to processing by the VPNNow™ Hardware Module. The details of the processing are described by a rule in the SPD. The rule specifies the policy to be applied to the network traffic and is described with the following structure:

```
struct T_VPNOW_IPSEC_POLICY_ENTRY {  
  
    unsigned long int    Size : 16,  
                        Version : 16;  
    T_VPNOW_IPSEC_SELECTORS Selectors;  
                        unsigned long int    Action : 2,  
                        ICMP_traffic : 1;  
  
};
```

Size

The value in this field specifies the size of the structure in bytes
T_VPNOW_IPSEC_POLICY_ENTRY.

Version

The value in this field specifies the version of the policy rule structure to ensure that both transactions use the same data organisation mode.

Value VPNOW_IPSEC_POLICY_ENTRY_STRUCT_VER.

Selectors

The values in the fields *SourceIPAddr1*, *SourceIPAddr2*, *DestIPAddr1*, *DestIPAddr2*, *SourcePort1*, *SourcePort2*, *DestPort1*, *DestPort2*, and *Protocol* in the structure *Selectors* define the set of IP traffic encompassed by this policy rule. Selectors are used to index into the SPD.

Action

This field determines if the IP packet has to be discarded or bypassed by the IPsec processing or if it is subject to IPsec processing. The *Action* field must be a value from the following list:

VPNOW_POLICY_ACTION_DISCARD

Specifies that all IP packets that have values in their header fields, which match with corresponding values given in a policy rule, are not allowed and will be dropped.

VPNOW_POLICY_ACTION_BYPASS

Specifies that all IP packets that have values in their header fields, which match with corresponding values given in a policy rule, are allowed and the VPNOW™ Hardware Module should forward them.

VPNOW_POLICY_ACTION_IPSEC

Specifies that all IP packets with values in their header fields, that match the corresponding values given in the policy rule, must be processed using the IPsec guidelines.

ICMP_traffic

Value in this field specifies how to manage unprotected ICMP error packets carrying message “Packet Too Big:

VPNOW_ACCEPT_ICMP_TRAFFIC

Specifies that all unprotected ICMP inbound traffic will be forwarded to the host system without IPSec processing.

VPNOW_REJECT_ICMP_TRAFFIC

Specifies that all unprotected ICMP traffic will be ignored, i.e. these IP packets will be dropped.

Allowing unprotected ICMP error messages exposes the nodes to denial of service attacks. On the other hand there may be routers on the path, which do not have IPSec implemented.

3.4.5 Commands for managing SPD and SAD

This section specifies management functionality that is provided by the MCS1000 in order to allow a system administrator to control how IPSec is applied to transmitted or received IP packets. The VPNow™ Hardware Module starts with empty SPD and SAD. Before any traffic can be processed by the IPSec Module policy rules must be created and added to the database. If policy requires protection then SA bundle(s) and IPSec SAs must be created for that policy and will be inserted into the SAD. The order in which policy rules are specified is critical, since the rules are searched one by one to see whether the condition matches the IP packet.

As SAs are unidirectional, both sides of secure communication s create separate SAs used for inbound and outbound processing. The inbound SA at the one communicating device and the outbound SA at the other end have the same keys and other cryptographic parameters.

If the SAs are not established, but policy requires IPSec protection, then packets matching this policy rule will be discarded.

The data part of message carrying the database command always starts with a database command header structure (structure T_VPNOW_IPSEC_DBASE_CMND above) that specifies, which table from the policy database (SPD) will be processed.

The VPNow™ module acknowledges receipt of every database command and the variable in the field *Msg* of the header specifies the result of processing command. The following values in the field *Msg* specify an error condition occurred at the time of the database command header check:

VPNOW_ERR_BAD_DBASE_CMND_STRUCT_VER

Indicates that the version numbers of the structures do not match.

VPNOW_ERR_BAD_DBASE_IP_MODE_VAL

Indicates that value in the header field *IpMode* is not from the set {VPNOW_IP4_MODE, VPNOW_IP6_MODE}.

VPNOW_ERR_BAD_DBASE_ETH_VAL

Indicates that the Ethernet interface with the value given in the header field *Eth* is not available.

VPNOW_ERR_BAD_DBASE_DIR_VAL

Indicates that the value in the header field *Dir* is not from the set {VPNOW_DIR_INBOUND, VPNOW_DIR_OUTBOUND}.

VPNOW_ERR_IP6_MODE_NOT_SUPPORTED

This error occurs if the value in the database command header field *IpMode* is set to VPNOW_IP6_MODE but the processing of IPv6 mode packets is not supported.

VPNOW_ERR_MISSING_DATA

Indicates that there should be more data added to the message than is specified in the field *Size* of the header.

The header part should be followed by data organized according to the structure specified by the database command:

- Security Association structure T_VPNOW_IPSEC_SA_ENTRY
- Structure T_VPNOW_IPSEC_POLICY_ENTRY describing policy rule

Only one editing (or reading) session can be started for a policy rule, a bundle and for an SA at the same time. If the editing (or reading) session of a policy rule, a bundle or an SA is started then this entry is taken as open. Every started session must be closed.

The following commands are available in the VPNOW™ Hardware Module for managing the policy database:

3.4.5.1 VPNOW_ADD_POLICY

This command allows adding a policy rule to the VPNOW™ policy database. The field *PolicyIndex* in the header part contains the row number for this new policy rule. If the number of an existing row is given, then this new row will be inserted into the table before this existing row and rest of table will be shifted down by one, i.e. the existing row in the table with the value given in the field *PolicyIndex* will have number *PolicyIndex* + 1 after processing the command VPNOW_ADD_POLICY and so on. If the value in the

field *PolicyIndex* points to the end of the table (or has a value greater than the last table entry) then the new policy rule will be added at the end of the table. The header portion of the command should be followed by the policy rule data (organized as structure `T_VPNOW_IPSEC_POLICY_ENTRY`). The value in the header field *Size* specifies the size of all the data added to the message, i.e. it is the sum of header and the policy rule entry.

A policy rule stays open after applying the command `VPNOW_ADD_POLICY` and SA bundles can be created for it.

Every database editing session started with the command `VPNOW_ADD_POLICY` should be stopped with the command `VPNOW_CLOSE_POLICY`.

The VPNow™ module acknowledges receipt of the `VPNOW_ADD_POLICY` command using the command `VPNOW_ANSW_TO_ADD_POLICY` with the database command header in the data part. The variable in the field *Msg* of the header specifies the result of processing command:

`VPNOW_OPERATION_OK`

Indicates that the command `VPNOW_ADD_POLICY` has been processed successfully and policy rule has been inserted.

`VPNOW_ERR_POLICY_RULE_NOT_CLOSED`

Specifies that the VPNow™ module couldn't add a new policy rule because the previous editing session has not been completed.
Values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, *BundleID*, and *SaID* determine which row in the SPD is being edited (viewed).

`VPNOW_ERR_IKE_NOT_SUPPORTED`

Automated key exchange is not supported, but the field *Type* in the policy rule data is set to `VPNOW_AUTOMATICALLY_KEYED_POLICY`.

`VPNOW_ERR_OUT_OF_MEMORY`

Indicates that the command failed because of lack of memory.

`VPNOW_ERR_POLICY_ADD_FAILURE`

Indicates that the VPNow™ module couldn't insert this rule into the corresponding SPD table.

`VPNOW_ERR_BAD_POLICY_ENTRY_STRUCT_VER`

Indicates that the version numbers of the structures do not match.

`VPNOW_ERR_BAD_POLICY_ENTRY_STRUCT_SIZE`

Indicates that there is less data added to the database command header than required for correct description of an SPD entry.

VPNOW_ERR_UNSUPPORTED_POLICY_ACTION

Indicates that the value in the field *Action* of the policy rule is not from the set
{VPNOW_POLICY_ACTION_DISCARD,
VPNOW_POLICY_ACTION_BYPASS,
VPNOW_POLICY_ACTION_IPSEC}

For example: to add a policy rule, which specifies that protection must be applied to all IPv4 packets received through interface 0 from the host with address 192.168.100.210 and traveling to the host with address 192.168.100.213. There are all unprotected packets coming from 192.168.100.210 and carrying the ICMP error message “Packet Too Big” allowed. This new rule will be inserted into the row 2 in the specified table.

The command header should look like this:

```
Size = 116
Version = VPNOW_IPSEC_DBASE_CMND_STRUCT_VER
Msg = 0
PolicyIndex = 2
BundleID = 0
SaID = 0
Eth = VPNOW_ETH_1
Dir = VPNOW_DIR_INBOUND
IpMode = VPNOW_IP4_MODE
```

The SPD entry should look like this:

```
Size = 84
Version = VPNOW_IPSEC_POLICY_ENTRY_STRUCT_VER
Selectors.SourceIPAddr1[0] = 0xC0A864D2
Selectors.SourceIPAddr2[0] = 0xC0A864D2
Selectors.DestIPAddr1[0] = 0xC0A864D5
Selectors.DestIPAddr2[0] = 0xC0A864D5
Selectors.SourcePort1 = 0
Selectors.SourcePort2 = 0
Selectors.DestPort1 = 0
Selectors.DestPort2 = 0
Selectors.Protocol = 0
Selectors.SourceIPAddrFieldType = VPNOW_IP_ADDR
Selectors.DestIPAddrFieldType = VPNOW_IP_ADDR
Selectors.SourcePortType = VPNOW_ANY_VALUE
Selectors.DestPortType = VPNOW_ANY_VALUE
Selectors.ProtocolType = VPNOW_ANY_VALUE
Action = VPNOW_POLICY_ACTION_IPSEC
ICMP_traffic = VPNOW_ACCEPT_ICMP_TRAFFIC
```

3.4.5.2 VPNOW_REPLACE_POLICY

This command modifies a policy rule and all the data bound to it in the VPNow™ policy database.

The field *PolicyIndex* in the header part specifies the row number of the policy entry to replace. If the row with number given in the field *PolicyIndex* does not exist then an error message (VPNOW_ERR_POLICY_WRONG_ROW_NR) will be returned. The header portion of the command should be followed by the policy rule data (organized as structure T_VPNOW_IPSEC_POLICY_ENTRY). The value in the header field *Size* specifies the size of all the data added to the message, i.e. it is the sum of header and the policy rule entry.

An existing SPD entry determined by *PolicyIndex* will be replaced with the data added to the command and all SAs bound to it will be removed. A policy rule stays open after applying the command VPNOW_REPLACE_POLICY and SA bundles can be created for it.

Every database editing session started with the command VPNOW_REPLACE_POLICY should be stopped with the command VPNOW_CLOSE_POLICY.

The VPNow™ module acknowledges receipt of the VPNOW_REPLACE_POLICY command using the command VPNOW_ANSW_TO_REPLACE_POLICY with the database command header in the data part. The variable in the field *Msg* of the header specifies the result of processing command:

VPNOW_OPERATION_OK

Indicates that the command VPNOW_REPLACE_POLICY has been processed successfully and the policy rule has been replaced.

VPNOW_ERR_POLICY_RULE_NOT_CLOSED

Specifies that the VPNow™ module couldn't add a new policy rule because the previous editing session has not been completed.

Values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, *BundleID*, and *SaID* determine which row in the SPD is being edited.

VPNOW_ERR_IKE_NOT_SUPPORTED

Automated key exchange is not supported but the field *Type* in the policy rule data is set to VPNOW_AUTOMATICALLY_KEYED_POLICY.

VPNOW_ERR_OUT_OF_MEMORY

Indicates that command failed because of lack of memory.

VPNOW_ERR_POLICY_REPLACE_FAILURE

Indicates that the MCS1000 couldn't insert this rule into the corresponding table.

VPNOW_ERR_BAD_POLICY_ENTRY_STRUCT_VER

Version numbers of the structures do not match.

VPNOW_ERR_BAD_POLICY_ENTRY_STRUCT_SIZE

Indicates that there is less data added to the database command header than required for correct description of an SPD entry.

VPNOW_ERR_UNSUPPORTED_POLICY_ACTION

Indicates that the value in the field *Action* of policy rule is not from the set { VPNOW_POLICY_ACTION_DISCARD, VPNOW_POLICY_ACTION_BYPASS, VPNOW_POLICY_ACTION_IPSEC }

VPNOW_ERR_POLICY_WRONG_ROW_NR

Specifies that the row number given in the field *PolicyIndex* is greater than the number of rows in the corresponding SPD table.

3.4.5.3 VPNOW_REMOVE_POLICY

This command allows the removal of policy rules from the VPNow™ policy database. In the field *PolicyIndex* of the header part should be row number of an existing policy rule. If value in the field *PolicyIndex* points out of table, then error message will be returned.

The VPNow™ module acknowledges receipt of the VPNOW_REMOVE_POLICY command using the command VPNOW_ANSW_TO_REMOVE_POLICY with the database command header in the data part. The variable in the field *Msg* of the header specifies the result of processing command:

VPNOW_OPERATION_OK

Indicates that command VPNOW_REMOVE_POLICY has been processed successfully and policy rule has been removed.

VPNOW_ERR_POLICY_RULE_NOT_CLOSED

Specifies that the VPNow™ module couldn't remove a policy rule because the previous editing session has not been completed. Values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, *BundleID*, and *SaID* determine which row in the SPD is being edited.

VPNOW_ERR_REMOVE_POLICY_FAILURE

Indicates that the VPNow™ module couldn't remove this rule from the corresponding table.

VPNOW_ERR_POLICY_WRONG_ROW_NR

Specifies that row number given in the field *PolicyIndex* is greater than number of rows in the corresponding table.

3.4.5.4 VPNOW_CLOSE_POLICY

This command is used to close a policy rule when editing or viewing the rule has been completed. In the field *PolicyIndex* of the header part should be row number of an open policy rule. If value in the field *PolicyIndex* points out of table then error message will be returned. An open SA and a open bundle should be closed (commands `VPNOW_CLOSE_SA` and `VPNOW_CLOSE_BUNDLE` respectively) before giving the command `VPNOW_CLOSE_POLICY`.

After the processing command the VPNow™ module replies with the message `VPNOW_ANSW_TO_CLOSE_POLICY` having database command header in the data part. The value in the field *Msg* of the header specifies the result of the processing command:

VPNOW_OPERATION_OK

Indicates that command `VPNOW_CLOSE_POLICY` has been processed successfully and policy rule has been closed.

VPNOW_ERR_POLICY_RULE_NOT_OPENED

Indicates that the VPNow™ module couldn't close specified policy rule because this policy rule is not open. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, and *IpMode* remain unchanged if there is no policy rule to close or determine the row that is being edited.

VPNOW_ERR_BUNDLE_NOT_CLOSED

Specifies that editing of this policy rule is in process. Values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, and *BundleID* determine a bundle that is open.

VPNOW_ERR_POLICY_WITHOUT_SA

Specifies that editing of this policy rule is in process. Action of this policy rule determines IPsec processing but there are no SAs added to this rule.

VPNOW_ERR_CLOSE_POLICY_FAILURE

Indicates that VPNow™ couldn't close this rule because of an internal error.

3.4.5.5 VPNOW_CLEAR_POLICY_TABLE

This command can remove a policy table from the VPNow™ policy database. The values in the fields *Eth*, *Dir*, and *IpMode* determine the table to be removed.

After processing the command the VPNow™ module replies with the command `VPNOW_ANSW_TO_CLEAR_POLICY_TABLE` that has a database command header in the data part. Value in the field *Msg* of the header specifies result of processing command:

VPNOW_OPERATION_OK

Indicates that command `VPNOW_CLEAR_POLICY_TABLE` has been processed successfully and policy table is empty now.

VPNOW_ERR_CLEAR_POLICY_TABLE_FAILURE

Indicates that the VPNow™ module couldn't remove all rules from the corresponding table.

3.4.5.6 VPNOW_ADD_BUNDLE

In order to bind a policy with the appropriate Security Association the policy rule entry has the list of SA bundle(s). This command allows creation of a new SA bundle in addition to adding to the policy rule in the SPD. The values in the fields *Eth*, *Dir*, *IpMode*, and *PolicyIndex* of command header must exactly specify an open policy rule, i.e. the command `VPNOW_ADD_BUNDLE` must be preceded by the command `VPNOW_ADD_POLICY` or `VPNOW_REPLACE_POLICY`. If a policy rule has many bundles to add, then the rule must be opened only once. Bundles should be added in the order that the host system requires so that they can be located at the time of using the database.

Every database editing session started with the command `VPNOW_ADD_BUNDLE` should be stopped with the command `VPNOW_CLOSE_BUNDLE`.

After processing the command the VPNow™ module replies with the message `VPNOW_ANSW_TO_ADD_BUNDLE` having database command header in the data part. In the field *BundleID* VPNow™ module returns the ID value of the created bundle. This value should be used later with commands in order to form SAs for this bundle. A bundle stays open after applying the command `VPNOW_ADD_BUNDLE` and SA(s) and keys can be added to it. The value in the field *Msg* of the header specifies the result of the processing command:

VPNOW_OPERATION_OK

Indicates that command `VPNOW_ADD_BUNDLE` has been processed successfully and a new bundle has been created and the bundle is now open for receiving information about the Security Associations forming this bundle.

VPNOW_ERR_BUNDLE_ADD_FAILURE

Indicates that the VPNow™ unit couldn't create a new bundle.

VPNOW_ERR_POLICY_RULE_NOT_OPENED

Indicates that the VPNow™ module couldn't add a bundle to the specified policy rule because no policy rule is opened for editing.

VPNOW_ERR_BAD_POLICY_NR

Indicates that the VPNow™ module couldn't add a bundle to the specified policy rule because another policy rule is opened for editing. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, and *IpMode* determine the table and the row that is being edited.

3.4.5.7 VPNOW_CLOSE_BUNDLE

This command completes the editing of an SA bundle and closes the bundle. With the help of command **VPNOW_CLOSE_BUNDLE** the host system lets the VPNow™ Hardware Module know that all of the SAs forming this bundle have been inserted. The values in the fields *Eth*, *Dir*, *IpMode*, *PolicyIndex*, and *BundleID* of the command header must exactly specify an open bundle.

After processing the command the VPNow™ unit replies with the message **VPNOW_ANSW_TO_CLOSE_BUNDLE** having a database command header in the data part. The value in the field *Msg* of the header specifies the result of the processing command:

VPNOW_OPERATION_OK

Indicates that command **VPNOW_CLOSE_BUNDLE** has been processed successfully and a new bundle of SAs is closed.

VPNOW_ERR_POLICY_RULE_NOT_OPENED

Indicates that the VPNow™ module couldn't close a bundle because no policy rule is opened for editing.

VPNOW_ERR_BAD_POLICY_NR

Indicates that the VPNow™ module couldn't close a bundle for a policy rule specified in the field *PolicyIndex* because another policy rule is opened for editing. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, and *IpMode* determine the table and the row that is being edited.

VPNOW_ERR_BUNDLE_NOT_OPENED

Specifies that there is no open bundle for given policy rule.

VPNOW_ERR_BAD_BUNDLE_ID

Specifies that the value in the header part field *BundleID* of the command determines a wrong ID for the bundle, i.e., the bundle, which is being edited has a different ID. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, and *BundleID* determine the bundle that is being edited.

VPNOW_ERR_SA_NOT_CLOSED

Signifies that the VPNow™ Hardware Module couldn't close a bundle because editing of an SA has not been finished.

3.4.5.8 VPNOW_ADD_SA_TO_BUNDLE

This command enables the IPsec module to add a new SA to the bundle. The ordering of IPsec processing is very important. The SAs must be inserted into the bundle in the order they will be processed during the time of applying IPsec rule. The values in the fields *Eth*, *Dir*, *IpMode*, *PolicyIndex*, and *BundleID* of the command header must exactly specify an open policy bundle, i.e. the host system must open a bundle with the command **VPNOW_ADD_BUNDLE** and get the ID for that bundle before starting with command(s) **VPNOW_ADD_SA_TO_BUNDLE**. If a bundle has many SAs to add, then the bundle must be opened only once and every SA should be added separately. The header part of the command should be followed by the SA data. The value in the header field *Size* specifies the size of all of the data added to the message, i.e. it is the sum of sizes of all header parts and SA entries.

After processing the given SA material the VPNow™ unit replies with the command **VPNOW_ANSW_TO_ADD_SA_TO_BUNDLE** having database command header in the data part. In the field *SaID* VPNow™ unit returns the ID value for the created SA entry. This value should be used later with the commands **VPNOW_ADD_KEYS_TO_SA** and **VPNOW_CLOSE_SA** in the command header field *SaID*.

The value in the field *Msg* of the header specifies the result of the processing command:

VPNOW_OPERATION_OK

Indicates that command **VPNOW_ADD_SA_TO_BUNDLE** has been processed successfully and a new SA has been added to the specified bundle.

VPNOW_ERR_POLICY_RULE_NOT_OPENED

Indicates that the VPNow™ module couldn't add a SA because no policy rule is opened for editing.

VPNOW_ERR_BAD_POLICY_NR

Indicates that the VPNow™ module couldn't add an SA to the bundle for a policy rule specified in the field *PolicyIndex* because another policy rule is opened for editing. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, and *IpMode* determine the table and the row that is being edited.

VPNOW_ERR_BUNDLE_NOT_OPENED

Specifies that there is no open bundle for the specified policy rule. To add SAs to the policy rule at first a bundle should be created using the command **VPNOW_ADD_BUNDLE**.

VPNOW_ERR_BAD_BUNDLE_ID

Specifies that the value in the header part field *BundleID* of the command determines a wrong ID for the bundle, i.e., the bundle, which is being edited has a different ID. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, and *BundleID* determine the bundle that is being edited.

VPNOW_ERR_OUT_OF_MEMORY

VPNow™ couldn't add an SA to the bundle because of lack of memory.

VPNOW_ERR_SA_ADD_FAILURE

Indicates that the VPNow™ couldn't add an SA to the bundle because of an internal error.

VPNOW_ERR_BAD_SA_ENTRY_STRUCT_VER

Indicates that the version numbers of structures do not match.

VPNOW_ERR_BAD_SA_ENTRY_STRUCT_SIZE

Indicates that there is less data added to the database command header than required for correct description of an SAD entry.

VPNOW_ERR_TUNNEL_IP_NOT_SPECIFIED

Value in the field *IPSecMode* of the structure **T_VPNOW_IPSEC_SA_ENTRY** specifies **VPNOW_IPSEC_MODE_TUNNEL** but the value in the field *TunnelIP* is zero.

VPNOW_ERR_TUNNEL_MAC_NOT_SPECIFIED

Value in the field *IPSecMode* of the structure **T_VPNOW_IPSEC_SA_ENTRY** specifies **VPNOW_IPSEC_MODE_TUNNEL** but the value in the field *TunnelMAC* is zero.

VPNOW_ERR_IP6_MODE_NOT_ENABLED

The VPNow™ Hardware Module cannot create tunnel header in IPv6 mode, i.e. value in the field *IPSecMode* in the structure *T_VPNOW_IPSEC_SA_ENTRY* specifies *VPNOW_IPSEC_MODE_TUNNEL* and the value in the field *TunnellIPmode* specifies *VPNOW_IP6_MODE* but IPv6 mode traffic is not enabled for specified interface.

VPNOW_ERR_BAD_ICV_LEN

The value in the field *ICVLen* in the structure *T_VPNOW_IPSEC_SA_ENTRY* is 0 or is greater than the VPNow™ cipher unit can produce applying authentication algorithm specified by the field *AuthAlgType*.

VPNOW_ERR_BAD_AUTH_KEY_LEN

The value in the field *AuthKeyLen* in the structure *T_VPNOW_IPSEC_SA_ENTRY* is out of range that VPNow™ cipher unit can support in the case of authentication algorithm specified in the field *AuthAlgType*.

VPNOW_ERR_UNSUPPORTED_AUTH_ALGORITHM

The value in the field *AuthAlgType* in the structure *T_VPNOW_IPSEC_SA_ENTRY* specifies number for authentication algorithm that is not supported by the VPNow™ cipher unit.

VPNOW_ERR_UNSUPPORTED_ENCR_ALGORITHM

The value in the field *EncrAlgType* in the structure *T_VPNOW_IPSEC_SA_ENTRY* specifies number for encryption algorithm that is not supported by the VPNow™ cipher unit.

VPNOW_ERR_UNSUPPORTED_IPSEC_PROTOCOL

Indicates that the value in the field *IPSecProt* in the structure *T_VPNOW_IPSEC_SA_ENTRY* specifies a protocol that is not from the set {*VPNOW_IPSEC_PROTOCOL_AH*, *VPNOW_IPSEC_PROTOCOL_ESP*}

VPNOW_ERR_ALGORITHM_NOT_SPECIFIED

Indicates that algorithm is not specified for given IPsec protocol processing. This situation occurs in two cases:

- The value in the field *IPSecProt* in the structure *T_VPNOW_IPSEC_SA_ENTRY* specifies an IPsec protocol *VPNOW_IPSEC_PROTOCOL_AH* but the value in the field *AuthAlgType* is *VPNOW_ALGORITHM_NOT_SPECIFIED*
- The value in the field *IPSecProt* in the structure *T_VPNOW_IPSEC_SA_ENTRY* specifies an IPsec protocol *VPNOW_IPSEC_PROTOCOL_ESP* but the values in the fields *AuthAlgType* and *EncrAlgType* are both set to *VPNOW_ALGORITHM_NOT_SPECIFIED*

For example: to add an SA to the bundle with ID = 1234 for a policy rule opened in the row 2, a message with the command `VPNOW_ADD_SA_TO_BUNDLE` should be sent to the MCS1000. The data added to the message should be organized as follows.

The command header should look like this:

```
Size = 224
Version = VPNOW_IPSEC_DBASE_CMND_STRUCT_VER
Msg = 0
PolicyIndex = 2
BundleID = 1234
SaID = 0
Eth = VPNOW_ETH_1
Dir = VPNOW_DIR_OUTBOUND
IpMode = VPNOW_IP4_MODE
```

The SA entry following the header should look like this:

```
Size = 192
Version = VPNOW_IPSEC_SA_ENTRY_STRUCT_VER
Selectors.SourceIPAddr1[0] = 0xC0A864D2
Selectors.SourceIPAddr2[0] = 0xC0A864D2
Selectors.DestIPAddr1[0] = 0xC0A864D5
Selectors.DestIPAddr2[0] = 0xC0A864D5
Selectors.SourcePort1 = 0
Selectors.SourcePort2 = 0
Selectors.DestPort1 = 0
Selectors.DestPort2 = 0
Selectors.Protocol = 0
Selectors.SourceIPAddrFieldType = VPNOW_IP_ADDR
Selectors.DestIPAddrFieldType = VPNOW_IP_ADDR
Selectors.SourcePortType = VPNOW_ANY_VALUE
Selectors.DestPortType = VPNOW_ANY_VALUE
Selectors.ProtocolType = VPNOW_ANY_VALUE

SPI = 300
DestinationIP [0] = 0
SNC = 0
TunnelOpt[0] = 0
IPSecProt = VPNOW_IPSEC_PROTOCOL_ESP
IPSecMode = VPNOW_IPSEC_MODE_TRANSPORT
TunnelIPmode = 0
IVmode = VPNOW_IV_VALUE_FROM_PACKET
ICVLen = 3
```

```
AuthKeyLen = 4
AuthAlgType = VPNOW_AUTH_HMAC_MD5
EncrAlgType = VPNOW_ENCR_DES3
PathMTU = 1480
ByteCount = 0
TunnelMAC[] = {0, 0, 0, 0, 0, 0}
TunnelTTL = 0
```

The description given above specifies that all IPv4 packets going out through interface 1 from the host with address 192.168.100.210 to the host with address 192.168.100.213 must be encrypted and authenticated with help of the IPSec header ESP.

3.4.5.9 VPNOW_CLOSE_SA

This command closes the editing of an SA. With the help of the command `VPNOW_CLOSE_SA` the host system lets VPNow™ Hardware Module know that all data forming this SA has been inserted. The values in the fields *Eth*, *Dir*, *IpMode*, *PolicyIndex*, *BundleID*, and *SalD* of the command header must exactly specify an open SA entry.

After processing the command the VPNow™ module replies with the command `VPNOW_ANSW_TO_CLOSE_SA` having a database command header in the data part. The value in the field *Msg* of the header specifies the result of the processing command:

VPNOW_OPERATION_OK

Indicates that the command `VPNOW_CLOSE_SA` has been processed successfully and a new SA has been closed.

VPNOW_ERR_POLICY_RULE_NOT_OPENED

Indicates that the VPNow™ module couldn't close an SA because no policy rule has been opened for editing.

VPNOW_ERR_BAD_POLICY_NR

Indicates that the VPNow™ module couldn't close an SA because the policy rule specified in the field *PolicyIndex* is not opened for editing. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, and *IpMode* determine the table and the row that is being edited.

VPNOW_ERR_BUNDLE_NOT_OPENED

Specifies that there is no open bundle for the specified policy rule.

VPNOW_ERR_BAD_BUNDLE_ID

Specifies that the value in the header part field *BundleID* of the command determines a wrong ID for the bundle, i.e., the bundle, which is being edited has a different ID. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, and *BundleID* determine the bundle that is being edited.

VPNOW_ERR_SA_NOT_OPENED

Specifies that the VPNow™ Hardware Module couldn't close an SA because there is no open SA for given policy rule and bundle.

VPNOW_ERR_BAD_SA_ID

Specifies that the VPNow™ Hardware Module couldn't close the SA because there is no SA with the given ID being edited. The values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, *BundleID*, and *SaID* determine row that is being edited.

VPNOW_ERR_MISSING_KEYS

The policy rule specified in the header part of a command is of the type **VPNOW_MANUALLY_KEYED_POLICY**, but there is no key data for the algorithms, i.e. the command **VPNOW_ADD_KEYS_TO_SA** has not been processed for this SA.

3.4.5.10 VPNOW_ADD_KEYS_TO_SA

This command enables the IPsec module to add key data to an SA entry. The values in the fields *Eth*, *Dir*, *IpMode*, *PolicyIndex*, *BundleID*, and *SaID* of a command header must exactly specify an open SA, i.e. the command **VPNOW_ADD_KEYS_TO_SA** must be preceded by command **VPNOW_ADD_SA_TO_BUNDLE** to get an ID for an SA.

The header part of the command must be followed by the key data. The key data should be organized depending on the values in fields of the structure **T_VPNOW_IPSEC_SA_ENTRY** which describes the current SA entry. If authentication is required, then the number of 32-bit units of the corresponding key data, specified by the value in the field *AuthKeyLen* must be included. If encryption is required, then the number of 32-bit units of the corresponding key data specified by the algorithm description must be included. If IV is given with key data, i.e. the field *IVmode* in the structure **T_VPNOW_IPSEC_SA_ENTRY**, describing SA entry, is set to **VPNOW_IV_VALUE_FROM_SA**, then IV data should follow the encryption key data. If both authentication and encryption are required, then the key data must be organized as follows:

- Authentication key
- Encryption key
- IV (if *IVmode* **VPNOW_IV_VALUE_FROM_SA** is set)

The value in the header field *Size* specifies the size of all the data added to the message, i.e. it is the sum of all sizes of the header and key data.

After processing the command the VPNow™ module replies with the command `VPNOW_ANSW_TO_ADD_SA_TO_BUNDLE` having a database command header in the data part.

The value in the field *Msg* of the header specifies the result of the processing command:

VPNOW_OPERATION_OK

Indicates that the command `VPNOW_ADD_KEYS_TO_SA` has been processed successfully and a new SA has the appropriate key data.

VPNOW_ERR_POLICY_RULE_NOT_OPENED

Indicates that the VPNow™ module couldn't add keys to an SA because no policy rule has been opened for editing.

VPNOW_ERR_BAD_POLICY_NR

Indicates that the VPNow™ module couldn't add keys to an SA because the policy rule specified in the field *PolicyIndex* is not opened for editing. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, and *IpMode* determine the table and the row that is being edited.

VPNOW_ERR_BUNDLE_NOT_OPENED

Specifies that there is no open bundle for the specified policy rule.

VPNOW_ERR_BAD_BUNDLE_ID

Specifies that the value in the header part field *BundleID* of the command determines a wrong ID for the bundle, i.e., the bundle, which is being edited has a different ID. On return the values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, and *BundleID* determine the bundle that is being edited.

VPNOW_ERR_SA_NOT_OPENED

Specifies that the VPNow™ Hardware Module couldn't add keys to an SA because there is no open SA for given policy rule and bundle.

VPNOW_ERR_BAD_SA_ID

Specifies that the VPNow™ Hardware Module couldn't add keys to the SA because there is no SA with the given ID being edited. The values in the fields *PolicyIndex*, *Eth*, *Dir*, *IpMode*, *BundleID*, and *SaID* determine row that is being edited.

VPNOW_ERR_MISSING_DATA

Specifies that there is not enough key data added to the header part of the command.

For example, to add keys to the SA with ID = 5678 belonging to the bundle with ID = 1234 pointed by the policy entry from the row 2, a message with the command `VPNOW_ADD_KEYS_TO_SA` should be sent to the MCS1000. The data added to the message should be arranged as follows:

The command header should look like this:

```
Size = 72
Version = VPNOW_IPSEC_DBASE_CMND_STRUCT_VER
Msg = 0
PolicyIndex = 2
BundleID = 1234
SaID = 5678
Eth = VPNOW_ETH_1
Dir = VPNOW_DIR_OUTBOUND
IpMode = VPNOW_IP4_MODE
```

Keys should be given after the database command header:

```
0x0123456789ABCDEF0123456789ABCDEF
0x0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF
```

3.4.5.11 VPNOW_GET_POLICY

This command enables the system to view the policy rules and all related data from the VPNow™ policy database. The values in the fields *Eth*, *Dir*, *IpMode*, and *PolicyIndex*, of the command header must exactly specify the policy rule, which is the subject of interest. If the row with the number given in the field *PolicyIndex* does not exist, then an error message will be returned. Viewing of the policy rule data is not enabled when the previous editing or viewing of the policy has not been finished.

After processing the command the VPNow™ module replies with the command `VPNOW_ANSW_TO_GET_POLICY` having a database command header followed by a policy rule data in the data part organized as the structure `T_VPNOW_IPSEC_POLICY_ENTRY`. The value in the header field *Size* specifies the size of all of the data added to the message, i.e. it is the sum of the sizes of all of the header parts and policy rules entry. If an error occurs, then the data added to the message consists of only the command header part.

The value in the field *Msg* of the header specifies the result of processing the command:

VPNOW_OPERATION_OK

Indicates that the command `VPNOW_GET_POLICY` has been processed successfully. The database command header is followed by the data describing the policy rule.

VPNOW_ERR_POLICY_RULE_NOT_CLOSED

Specifies that the VPNow™ module couldn't get a policy rule because the previous editing or viewing session has not been completed. Values in the fields *PolicyIndex*, *Eth*, *Dir*, and *IpMod* determine which row in the SPD is being edited or viewed.

VPNOW_GET_POLICY_FAILURE

The VPNow™ Hardware Module database manager couldn't get policy rule data because of an internal error.

VPNOW_ERR_OUT_OF_MEMORY

The MCS1000 couldn't show policy rule because of lack of memory.

3.4.5.12 VPNOW_GET_NEXT_BUNDLE

This command allows the system to open an SA bundle for a policy rule that has already been opened for viewing. The values in the fields *Eth*, *Dir*, *IpMode*, and *PolicyIndex* of the command header must exactly specify the policy rule, which is the subject of interest. The value in the field *BundleID* specifies previous bundle in the list of bundles for this policy rule. The value 0 specifies the first bundle. If a row with number given in the field *PolicyIndex* does not exist, then an error message will be returned. In order to get to the next bundle the ID that corresponds to the policy rule must be open.

After processing the command the VPNow™ module replies with the command **VPNOW_ANSW_TO_GET_NEXT_BUNDLE** having database command header in the data part. The value in the header field *BundleID* specifies an ID for the next bundle. The value 0 indicates the end of the list of bundles for this policy rule. The value in the field *Msg* of the header specifies the result of the processing command:

VPNOW_OPERATION_OK

Indicates that the command **VPNOW_GET_NEXT_BUNDLE** has been processed successfully and an ID for the opened bundle is returned in the field *BundleID*.

VPNOW_ERR_POLICY_RULE_NOT_OPENED

Specifies that no policy rule is open in order to get the bundle ID.

VPNOW_GET_NEXT_BUNDLE_FAILURE

The VPNow™ Hardware Module database manager couldn't get the next bundle for the specified policy rule.

3.4.5.13 VPNOW_GET_NEXT_SA

This command allows the system to view SA data from a bundle that is opened for viewing. The values in the fields *Eth*, *Dir*, *IpMode*, *PolicyIndex*, and *BundleID* of the command header must exactly specify the bundle, which is the subject of interest. The value in the field *SaID* specifies the previous SA in the bundle. The value 0 specifies the first SA. In order to get to the next SA in the bundle the ID the corresponding policy rule and bundle must be open.

After processing the command the VPNow™ module replies with the command VPNOW_ANSW_TO_GET_NEXT_SA having database command header followed by SA data, organized as the structure T_VPNOW_IPSEC_SA_ENTRY, in the data part. If the SA is keyed manually then the SA data will be followed by the key data:

- Database command header
- SA entry
- Authentication key
- Encryption key
- IV (if *IVmode* VPNOW_IV_VALUE_FROM_SA is set)

The value in the header field *Size* specifies the size of all of the data added to the message, i.e. it is the sum of all of the sizes of the header part and the SA entry. If an error occurs, or the end of list has been reached, then the data added to the message consists of only the command header part. The value in the header field *SaID* specifies the ID of the next SA in the bundle. The value 0 indicates the end of list of SAs for this bundle.

The value in the field *Msg* of the header specifies result of processing the command:

VPNOW_OPERATION_OK

Indicates that the command VPNOW_GET_NEXT_SA has been processed successfully and the SA data has been sent to the host system.

VPNOW_ERR_POLICY_RULE_NOT_OPENED

Specifies that there is not an open policy rule. In order to get SA data the commands VPNOW_GET_POLICY and VPNOW_GET_NEXT_BUNDLE should be issued.

VPNOW_ERR_BUNDLE_NOT_OPENED

Specifies that there is not an open bundle. In order to get SA data the command VPNOW_GET_NEXT_BUNDLE should be issued.

VPNOW_ERR_WRONG_BUNDLE_ID

The bundle with ID specified in the header part of the command is not open.

VPNOW_ERR_OUT_OF_MEMORY

The VPNow™ module couldn't show the SA data because of a lack of memory.

VPNOW_ERR_GET_SA_FAILURE

The VPNow™ Hardware Module database manager couldn't get the next SA for the specified policy rule and bundle.

3.4.5.14 VPNOW_GET_SPD_ROW_COUNT

This command allows the system to get the actual size of the SPD table. The values in the fields *Eth*, *Dir*, *IpMode*, and *PolicyIndex* of the command header must exactly specify the table.

After processing the command VPNow™ module replies with the command VPNOW_ANSW_TO_GET_SPD_ROW_COUNT having database command header in the data part. The value in the header field *PolicyIndex* specifies the number of rows in specified table.

The value in the field *Msg* of the header specifies the result of processing the command:

VPNOW_OPERATION_OK

Indicates that the command VPNOW_GET_SPD_ROW_COUNT has been processed successfully and the number of the rows in the specified table is in the field *PolicyIndex* of the header part of the command.

3.4.6 Example

The following steps should be done to replace keys for an existing manually keyed SA from the SAD pointed by policy rule from the row *PolRow* in the SPD:

1. Send a message with the command VPNOW_GET_POLICY to the MCS1000. The field *PolicyIndex* in the database command header should be set to *PolRow*. Wait for the reply command VPNOW_ANSW_TO_GET_POLICY from the MCS1000. Save the returned SPD entry data.
2. Send a message with the command VPNOW_GET_NEXT_BUNDLE to the MCS1000. In the database command header: the field *PolicyIndex* should be set to *PolRow* and the field *BundleID* should specify the ID for the previous bundle (0 if the first bundle). Wait for the reply command VPNOW_ANSW_TO_GET_NEXT_BUNDLE from the MCS1000. Save value for bundle ID. If the value 0 returned for bundle ID (end of bundles) then processing continues from (5).

3. Send a message with the command `VPNOW_GET_NEXT_SA` to the MCS1000. In the database command header: the field *PolicyIndex* should be set to *PolRow* and the field *BundleID* should specify the ID for the bundle saved in (2). The field *SaID* should specify the ID for the previous SA (0 if the first SA). Wait for the reply command `VPNOW_ANSW_TO_GET_NEXT_SA` from the MCS1000. Save the SAD entry and keys. Repeat (3) until the ID returned for SA by the MCS1000 is 0 (end of SAs).
4. Send a message with the command `VPNOW_CLOSE_BUNDLE` to the MCS1000. In the database command header: the field *PolicyIndex* should be set to *PolRow* and the field *BundleID* should specify the ID for the bundle saved in (2). Wait for the reply command `VPNOW_ANSW_TO_CLOSE_BUNDLE` from the MCS1000. Processing continues from (2)
5. Send a message with the command `VPNOW_CLOSE_POLICY` to the MCS1000. The field *PolicyIndex* in the database command header should be set to *PolRow*. Wait for the reply command `VPNOW_ANSW_TO_CLOSE_POLICY` from the MCS1000.
6. Send a message with the command `VPNOW_REPLACE_POLICY` to the MCS1000. The field *PolicyIndex* in the database command header should be set to *PolRow* and the SPD entry saved in (1) should be added to the header. Wait for the reply command `VPNOW_ANSW_TO_REPLACE_POLICY` from the MCS1000.
7. Send a message with the command `VPNOW_ADD_BUNDLE`. In the database command header: the field *PolicyIndex* should be set to *PolRow*. Wait for the reply command `VPNOW_ANSW_TO_ADD_BUNDLE` from the MCS1000. Save the value returned by the MCS1000 for bundle ID.
8. Send a message with the command `VPNOW_ADD_SA_TO_BUNDLE` to the MCS1000. In the database command header: the field *PolicyIndex* should be set to *PolRow* and the field *BundleID* should specify the ID for the bundle saved in (7). The SAD entry saved in (3) should be added to the header. Wait for the reply command `VPNOW_ANSW_TO_ADD_SA_TO_BUNDLE` from the MCS1000. Save the value returned by the MCS1000 for an SA ID. Send a message with the command `VPNOW_ADD_KEYS_TO_SA` to the MCS1000. In the database command header: the field *PolicyIndex* should be set to *PolRow*, the field *BundleID* should specify the ID for a bundle saved in (7), and the field *SaID* should specify the ID saved for an SA in (8). The keys saved in (3) for this SA, or new keys should be added to the header. Wait for the reply command `VPNOW_ANSW_TO_ADD_KEYS_TO_SA` from the MCS1000. Send a message with the command `VPNOW_CLOSE_SA`. In the database command header: the field *PolicyIndex* should be set to *PolRow*, the field *BundleID* should specify the ID for a bundle saved in (7), and the field *SaID* should specify the ID saved for an SA in (8). Wait for the reply command `VPNOW_ANSW_TO_CLOSE_SA` from the MCS1000. Repeat (8) until all SAs and keys (read in (3)) for current bundle have been replaced.

9. Send a message with the command `VPNOW_CLOSE_BUNDLE` to the MCS1000. In the database command header: the field *PolicyIndex* should be set to *PolRow* and the field *BundleID* should specify the ID for the bundle saved in (7). Wait for the reply command `VPNOW_ANSW_TO_CLOSE_BUNDLE` from the MCS1000. Processing continues from (7) if there are more bundles (read in (2)) to replace for current SPD entry *PolRow*.
10. Send a message with the command `VPNOW_CLOSE_POLICY` to the MCS1000. The field *PolicyIndex* in the database command header should be set to *PolRow*. Wait for the reply command `VPNOW_ANSW_TO_CLOSE_POLICY` from the MCS1000.

NOTE:

The database command header fields *Size*, *Version*, *Eth*, *Dir*, and *IpMode* should be set with correct values to specify table from the SPD.

4 Using VPNow™ Hardware Module.

4.1 Enabling IPsec processing

When the VPNow™ Hardware Module processing has been started and provided with policy rules then the prepared security will be afforded only if the IPsec processing is enabled.

IPsec processing in the VPNow™ Hardware Module can be enabled or disabled using the commands: `VPNOW_ENABLE_IPSEC` and `VPNOW_DISABLE_IPSEC`.

To enable IPsec processing, a message with the command `VPNOW_ENABLE_IPSEC` should be sent to the MCS1000.

If IPsec processing is enabled all inbound and outbound IP packets are matched against entries in the SPD, required protection will be applied, and the log messages will be prepared (if log is enabled) and sent to the host system.

To disable IPsec processing, a message with the command `VPNOW_DISABLE_IPSEC` should be sent to MCS1000.

If IPsec processing is disabled all inbound and outbound IP packets will bypass IPsec processing and no log will be prepared. The SPD and SAD will not be cleared and configuration settings stay unchanged.

4.2 Reading VPNow™ Hardware Module settings

To get the current status of the VPNow™ Hardware Module IPsec processing or the current parameters programmed into the VPNow™ Hardware Module the following commands should be used:

4.2.1 Reading status

To read the current status of the VPNow™ Hardware Module IPsec processing the command `VPNOW_GET_IPSEC_STATUS` can be used.

The command `VPNOW_GET_IPSEC_STATUS` causes the VPNow™ module to reply with a message specifying the command:

- `VPNOW_IPSEC_STATUS_ENABLED` if IPsec processing is started and enabled.
- `VPNOW_IPSEC_STATUS_DISABLED` if IPsec processing is started but switched off.
- `VPNOW_IPSEC_STATUS_STOPPED` if IPsec processing is not started.

The MCS1000 starts in the mode `VPNOW_IPSEC_STATUS_STOPPED`.

4.2.2 Reading interface settings

To read the current parameters programmed for an Ethernet interface in the IPsec Module, a message with the command `VPNOW_GET_IP_CONF` and with the data organized with the help of the structure `T_VPNOW_IP_CONF` (defined in the section Interface settings) should be sent to the MCS1000. The value in the field `Eth` specifies the interface, which settings are to be read.

The command `VPNOW_GET_IP_CONF` causes the VPNow™ module to reply with a message specifying the command `VPNOW_ANSW_TO_GET_IP_CONF`. The data part of the message includes a pointer to the structure `T_VPNOW_IP_CONF` filled with values of the current settings of the interface defined in *Eth*.

The value in the field *Msg* specifies the validity of the added data as shown below:

`VPNOW_ERR_BAD_IP_CONF_STRUCT_VER`

Specifies that the current version number and the version number given with the command in the field *Version* do not match.

`VPNOW_ERR_BAD_IP_CONF_STRUCT_SIZE`

Specifies that the size of the current structure and the size given with the command are not equal.

`VPNOW_ERR_BAD_IP_CONF_ETH_VAL`

Specifies that the MCS1000 cannot return the parameters programmed into the Ethernet interface defined in the field *Eth*. This response may occur in two instances: either the VPNow™ does not support the Ethernet interface given in the field *Eth* or interface is not available at the moment.

`VPNOW_OPERATION_OK`

Specifies that data is correct.

4.2.3 Reading IPsec settings

In order to read the parameters concerned in IPsec processing that were programmed into the IPsec module, a message with the command `VPNOW_GET_IPSEC_CONF` should be sent to the MCS1000. Fields *Size* and *Version* must have the correct values.

The command `VPNOW_GET_IPSEC_CONF` causes the VPNow™ module to reply with a message specifying the command `VPNOW_ANSW_TO_GET_IPSEC_CONF`. The data part of the message includes a pointer to the structure `T_VPNOW_IPSEC_CONF` (defined in the section IPsec settings) filled with values of the current settings.

The value in the field *Msg* specifies the validity of the added data as shown below:

VPNOW_ERR_BAD_IPSEC_CONF_STRUCT_VER

Specifies that the current version number and version number given with command in the field *Version* do not match.

VPNOW_ERR_BAD_IPSEC_CONF_STRUCT_SIZE

Specifies that the size of the current structure and the size given with the command are not equal.

VPNOW_OPERATION_OK

Specifies that the data is correct.

4.3 Log

The VPNow™ Hardware Module supports logging protocol to let the host system to keep track of the device's internal behavior during IPsec processing. If IPsec log is enabled (flag `VPNOW_IPSEC_LOG_ENABLED` is set) then VPNow™ Hardware Module sends a log message to the host system when one of the following events occurs:

- IPsec has been applied on IP packet
- IP packet with corrupted data entered system
- SA is missing for outbound packet
- IP packet with corrupted IPsec header entered system
- Error occurs during IPsec processing

If AH and/or ESP audit is enabled (flags `VPNOW_AH_AUDIT_ENABLED` and/or `VPNOW_ESP_AUDIT_ENABLED` are set) then the VPNow™ Hardware Module sends a log message to the host system when one of the following errors occurs:

- ICV validation failure
- SA is missing for incoming packet specifying IPsec header

Information about event, traffic parameters, and IPsec processing status are added to the Log command.

To send log data the VPNow™ Hardware Module sends a message to the host system with the command `VPNOW_IPSEC_LOG_READY` and with a pointer to the log data organized as following structure specifies:

```
struct T_VPNOW_IPSEC_LOG {  
  
    unsigned long int    Size : 16,  
                        Version : 16;  
    unsigned long int    Eth : 2,  
                        Dir : 1,  
                        IpMode : 1,  
                        Mode : 1,  
                        NextHeader : 8,  
                        Hdrs : 3,  
                        FrameID : 16;  
    unsigned long int    Time;  
    unsigned long int    PolicyIndex;  
    unsigned long int    Message;  
    unsigned long int    SPI[VPNOW_MAX_SPI_COUNT];  
    unsigned long int    SourceIP[4];  
    unsigned long int    DestinationIP[4];  
    unsigned char        FlowLbl[4];  
};
```

Size

The value in this field specifies the length of the log data in bytes.

Version

The value in this field specifies the version of configuration structure to ensure that both parts of conversation use the same data organisation mode.

Current value: VPNOW_LOG_IPSEC_STRUCT_VER.

Time

Specifies when this IP packet entered the VPNow™ Hardware Module. The value in this field is the system time expressed in seconds.

PolicyIndex

Specifies the row number of applied policy rule in the SPD table.

Eth

Specifies the Ethernet interface from the set:

```
{VPNOW_ETH_0,  
VPNOW_ETH_1,  
VPNOW_ETH_2}
```

Dir

Specifies the direction of the traffic:

VPNOW_DIR_INBOUND or VPNOW_DIR_OUTBOUND.

IpMode

Specifies IP mode of the packet:
 VPNOW_IP4_MODE or VPNOW_IP6_MODE.

Mode

Specifies the amount of the IPsec log data:

VPNOW_LOG_WITH_DATA	debug data (start of the IP packet) is added to the end of the log header
VPNOW_LOG_WITHOUT_DATA	only log header

Message

Specifies the reason for this log. The section Messages from VPNow™ Hardware Module describes these messages more precisely.

Hdrs

Specifies the number of successfully processed IPsec headers (AH and ESP). Not all IPsec headers from the packet may be counted: processing of the packet is terminated when an error occurs.

SPI

List of the SPIs from processed IPsec headers. Length of this list is given in the field *Hdrs*. Maximal length of this array is VPNOW_MAX_SPI_COUNT.

SourceIP

Specifies the value from the field *Source Address* in the IP packet header. In the case of IPv4 mode only the first field of the array specifying the address is used.

DestinationIP

Specifies the value from the field *Destination Address* in the IP packet header. In the case of IPv4 mode only the first field of the array specifying the address is used.

FrameId

Specifies the frame identification value from the IPv4 packet header.

FlowLbl

In the case of an IPv6 packet this specifies the value from the field Flow Label in the IP header. In the case of IPv4 not used.

NextHeader

Depending on the value in the field *IpMode* this field specifies the value from the IPv4 header field *Protocol* or from the IPv6 header field *Next Header*.

If the debug mode is enabled (the flag `VPNOW_DEBUG_LOG_ENABLED` is set) then *DataSize* bytes (the amount is specified at the time of the IPsec set-up) from the start of the IP packet are added to the end of the log header.

For example log message with the values in the header fields:

```
NextHeader = 6
Hdrs = 1
FrameId = 3245
Time = 1034851126
PolicyIndex = 2
Message = 37636
SPI[0] = 257
SourceIP[] = {3232261330, 0, 0, 0}
DestinationIP[] = {3232261333, 0, 0, 0}
FlowLbl[] = {0, 0, 0}
```

should be read as:

IPv4 packet with ID = 3245 traveling from 192.168.100.210 to 192.168.100.213 arrived at 10/17/02 13:38:46 through the interface number 0. Protected packet was successfully decrypted by an SA having SPI = 257 and pointed by the policy rule from the row 2. An IP packet carrying a TCP packet was forwarded to the host IP stack. No debug data is added to the log header.

4.4 Messages from VPNow™ Hardware Module

When IPsec processing is enabled then the header part of each IP packet entering MCS1000 is checked to avoid corrupted packets to enter the IPsec processing. If any error occurs during the inspection the packet will be dropped and the log with message from the following set will be prepared and sent to the host system:

VPNOW_IP_ERR_IP4_WRONG_LENGTH

Error occurs when the field *Total Length* in the IPv4 packet header specifies a value greater than the amount that has been entered into the VPNow™ system.

VPNOW_IP_ERR_IP4_TTL_ERROR

Error occurs when the value in the field *Time to Live* in the IPv4 packet header is 0.

VPNOW_IP_ERR_IP4_WRONG_CHECKSUM

Error occurs when the checksum given in the IPv4 packet header field *Header Checksum* and checksum calculated in the VPNow™ system do not match.

VPNOW_IP_ERR_IP4_WRONG_HEADER_LEN

Error occurs when the value in the IPv4 packet header field *IHL* is smaller than actual length of the header.

VPNOW_IP_ERR_IP4_DF_MF_SET

Error occurs when the flags DF and MF are both set in the IPv4 packet header field *Flags*.

VPNOW_IP_ERR_IP4_PROTOCOL_NOT_SET

Error occurs if the value in the IPv4 packet header field *Protocol* is 0.

VPNOW_IP_ERR_IP6_WRONG_JUMBO_PAYLOAD_LENGTH

Error occurs if the value of the given jumbo payload length is smaller than 0xFFFFFFFF.

VPNOW_IP_ERR_IP6_JUMBO_PAYLOAD_OPTION_MISSING

Error occurs if there should be a jumbo payload option in the IPv6 packet.

VPNOW_IP_ERR_IP6_WRONG_PAYLOAD_LENGTH

Error occurs when there is a redundant jumbo option in the IPv6 packet.

VPNOW_IP_ERR_CORRUPTED_FRAME

Error occurs when the value of the data length from the IP header and the length of the available data do not match.

VPNOW_IP_ERR_IP6_JUMBO_WITH_FRAGMENT_HEADER

Error occurs when there is a jumbo option in the fragmented IPv6 packet.

VPNOW_IP_ERR_IP6_FRAGMENT_DATA_ERROR

Error occurs when the IPv6 fragment contains corrupted data.

VPNOW_IP_ERR_WRONG_VERSION

Error occurs when the value in the first field of the IP packet specifies an unsupported version.

VPNOW_IP_ERR_FRAGMENT_DATA_CORRUPTED

The VPNow™ IPsec system received an IP packet fragment, with an incorrect header.

VPNOW_IP_ERR_FRAG_WAIT_TIME_EXPIRED

Error occurs when the IPsec module has been waiting for missing fragments of an IP packet for a time longer than specified in the configuration field *FragmentWaitTime*. All fragments of this packet will be dropped.

VPNOW_IP_ERR_CORRUPTED_ICMP_PACKET

Error occurs when the data in an ICMP packet is corrupted.

If IPsec processing is enabled then each IP packet, which enters VPNow™ Hardware Module is matched against entries in the SPD to get the action for this packet. If the rule is not found or specifies an action that has not been applied to the incoming packet then an error occurs and the packet will be dropped:

VPNOW_IP_ERR_POLICY_NOT_FOUND

Error occurs if there is not a policy rule for the current IP packet in the corresponding table of VPNow™ policy database (SPD).

VPNOW_IP_ERR_UNPROTECTED_INBOUND_PACKET

Error occurs when an unprotected inbound IP packet entered VPNow™ IPsec system, but corresponding policy rule specifies that received packet must be protected.

VPNOW_IP_ERR_PACKET_IS_TO_BE_DISCARDED

This event occurs when a not allowed IP packet entered VPNow™ IPsec system, i.e. corresponding policy rule specifies that received packet must be discarded.

If the header of an IP packet is correct and policy action specifies IPsec then IPsec processing according to security requirements specified by SA(s) starts:

- An incoming packet will be authenticated and/or decrypted. IPsec headers will be removed before forwarding the packet to the TCP/IP stack.
- Required IPsec headers will be added to an outbound packet. The packet will be encrypted and/or authenticated and forwarded to the Media Access layer.

Message VPNOW_IPSEC_SUCCESS occurs when the required IPsec processing has been applied successfully and an IP packet has been forwarded.

There are several auditable events specified in the descriptions of IPsec protocols ([RFC_AH] and [RFC_ESP]) and IPsec processing ([RFC_ARC]), which auditing is implemented in the VPNow™ Hardware Module.

An error due to the data delivered to the device or an internal error may occur and cause the packet to be dropped during IPsec processing in the VPNow™ Hardware Module:

VPNOW_ERR_OUT_OF_MEMORY

Error occurs when there is insufficient memory for processing IPsec.

VPNOW_IPSEC_FAILURE

Error occurs when an IPsec processing internal error occurred.

VPNOW_IPSEC_ERR_DECRYPTION_FAILURE

Error occurs when the IP packet data is useless after the decryption procedure, i.e. the result of decryption is corrupted due to the incorrect decryption method, bad keys or crypto-unit failure.

VPNOW_IPSEC_ERR_TUNNEL_IP_WRONG_MODE

Error occurs when a tunneled IP packet entered VPNow™ IPsec system with an inner IP header version number that does not match value given in the header AH or ESP (or in another IPv6 mode Extension header).

VPNOW_IPSEC_ERR_OUTBOUND_SA_MISSING

Error occurs when the corresponding policy rule requires IPsec processing, but SAs and keys have not been set manually or the VPNow™ Hardware Module could not negotiate SAs.

VPNOW_IPSEC_ERR_INBOUND_SA_MISSING

Error occurs if there is an IPsec header (AH or ESP) in an inbound packet but the corresponding SA is not found in the policy database.

VPNOW_IPSEC_ERR_REQUIRED_AND_APPLIED_IPSEC_DIFFER

After IPsec processing on an IP packet data the VPNow™ Hardware Module checks whether the required IPsec processing has been applied, i.e. verifies that the SA(s) corresponding to the IPsec header(s) found at the time of parsing the received packet(s) match the type and order of the SAs required by the corresponding policy. Error occurs when required and applied protection differ.

VPNOW_IPSEC_ERR_CRYPTO_WAIT_TIME_EXPIRED

Error occurs when the IPsec module has been waiting for cipher unit resources for a time longer than specified in the configuration field *CiphWaitTime*. The packet will be dropped.

VPNOW_IPSEC_ERR_WRONG_SRC_IP_SEL_VALUE

Error occurs when the source IP address selector value check failed after decryption of an inbound protected packet. The packet's source address must match the SA selector value.

VPNOW_IPSEC_ERR_CRYPTO_ICV_CHECK_FAILURE

Error occurs when the Integrity Check Value (ICV) of a protected inbound packet and the signature that is calculated by VPNow™ Cipher Unit do not match.

4.5 About VPNow™ Hardware Module

For using the VPNow™ Hardware Module properly the version control should be checked.

In order to get the version numbers of the current IPsec version and structure versions the command `VPNOW_GET_IPSEC_VERSION_NR` should be used. The data field in the message sent to VPNow™ should be set to NULL. The VPNow™ module replies with the message `VPNOW_ANSW_TO_GET_IPSEC_VERSION_NR` and the data part of the message points to the following structure:

```
struct T_VPNOW_IPSEC_VER_CMND {  
  
    unsigned long int    Size;  
    unsigned long int    IPsecVer;  
    unsigned long int    IpConfStructVer;  
    unsigned long int    IPsecConfStructVer;  
    unsigned long int    DbaseCmndStructVer;  
    unsigned long int    PolicyStructVer;  
    unsigned long int    SaStructVer;  
    unsigned long int    KeyStructVer;  
    unsigned long int    CryptoStructVer;  
    unsigned long int    CryptoListStructVer;  
    unsigned long int    IPsecLogStructVer;  
  
};
```

Size

Value in this field specifies the size of all of the data, i.e. size of structure `T_VPNOW_IPSEC_VER_CMND`.

IPsecVer

Version number of the IPsec module (`VPNOW_IPSEC_VERSION_NUMBER`).

IpConfStructVer

Version number of the structure for the IP configuration
`VPNOW_IP_CONF_STRUCT_VER`

IPSecConfStructVer

Version number of the structure for the IPSec configuration:
VPNOW_IPSEC_CONF_STRUCT_VER.

DbaseCmndStructVer

Version number of the database managing structure:
VPNOW_IPSEC_DBASE_CMND_STRUCT_VER.

PolicyStructVer

Version number of the structure for the policy rule:
VPNOW_IPSEC_POLICY_ENTRY_STRUCT_VER.

CryptoStructVer

Version number of the structure for describing the crypto algorithms:
VPNOW_CRYPTO_ALG_STRUCT_VER.

CryptoListStructVer

Version number of the structure specifying the number of crypto algorithms:
VPNOW_CRYPTO_ALG_LIST_STRUCT_VER.

IPSecLogStructVer

Version number of the structure for the IPSec configuration:
VPNOW_LOG_IPSEC_STRUCT_VER.

Values in these fields of the corresponding structures will be compared when VPNow™ Hardware Module receives a message from the host system.

4.6 Stopping IPSec processing

To stop IPSec processing and working of the VPNow™ Hardware Module a message with the command VPNOW_STOP_IPSEC should be sent to the MCS1000.

The command VPNOW_STOP_IPSEC causes the VPNow™ module to

- Stop IPSec processing.

4.6.1 Drop all packets being processed in the VPNow™ Module at the moment.

- Clear SPD and SAD.
- Reply with a message specifying the command VPNOW_IPSEC_STOPPED.

The configuration settings will be not changed.

5 Appendix A: Constants

List of the VPNow™ Hardware Module named constants:

Constant	Value
VPNOW_IPSEC_VERSION_NUMBER	
VPNOW_IP_CONF_STRUCT_VER	
VPNOW_IPSEC_CONF_STRUCT_VER	
VPNOW_CRYPTO_ALG_STRUCT_VER	
VPNOW_CRYPTO_ALG_LIST_STRUCT_VER	
VPNOW_IPSEC_SA_ENTRY_STRUCT_VER	
VPNOW_IPSEC_POLICY_ENTRY_STRUCT_VER	
VPNOW_IPSEC_DBASE_CMND_STRUCT_VER	
VPNOW_LOG_IPSEC_STRUCT_VER	
VPNOW_IPSEC_COPY_DF_FLAG	0
VPNOW_IPSEC_CLEAR_DF_FLAG	1
VPNOW_IPSEC_SET_DF_FLAG	2
VPNOW_MIN_PATH_MTU	68
VPNOW_MAX_PATH_MTU	1480
VPNOW_IPV6_ENABLED	1
VPNOW_IPV6_DISABLED	0
VPNOW_USE_IPSEC_ALL	0
VPNOW_USE_IPSEC_BYPASS	1
VPNOW_IPSEC_LOG_ENABLED	1
VPNOW_AH_AUDIT_ENABLED	4
VPNOW_ESP_AUDIT_ENABLED	8
VPNOW_DEBUG_LOG_ENABLED	2
VPNOW_DEFAULT_DEBUG_DATA_SIZE	64
VPNOW_MAX_DEBUG_DATA_SIZE	128
VPNOW_FRAGMENT_WAIT_MAX_TIME	10000
VPNOW_FRAGMENT_WAIT_TIME	1000
VPNOW_CRYPTO_WAIT_MAX_TIME	10000
VPNOW_CRYPTO_WAIT_TIME	1000
VPNOW_IKE_SUPPORTED	1
VPNOW_IKE_NOT_SUPPORTED	0
VPNOW_IKE_ENABLED	1
VPNOW_IKE_DISABLED	0
VPNOW_IPV6_SUPPORTED	1
VPNOW_IPV6_NOT_SUPPORTED	2
VPNOW_IPSEC_ENCR_PAD_ZERO	0
VPNOW_IPSEC_ENCR_PAD_SERIES	1
VPNOW_IPSEC_ENCR_PAD_RANDOM	2

Constant	Value
VPNOW_AUTH_ALG	0
VPNOW_ENCR_ALG	1
VPNOW_ALG_NAME_LEN	20
VPNOW_MAX_SPI_COUNT	7
VPNOW_LOG_WITH_DATA	1
VPNOW_LOG_WITHOUT_DATA	0
VPNOW_ANY_VALUE	0
VPNOW_IP_ADDR	1
VPNOW_IP_ADDR_SUBNET	2
VPNOW_IP_ADDR_RANGE	3
VPNOW_VALUE_FROM_FIELD	1
VPNOW_PORT_VALUE_RANGE	2
VPNOW_POLICY_ACTION_DISCARD	0
VPNOW_POLICY_ACTION_BYPASS	1
VPNOW_POLICY_ACTION_IPSEC	2
VPNOW_ACCEPT_ICMP_TRAFFIC	1
VPNOW_REJECT_ICMP_TRAFFIC	0
VPNOW_DIR_INBOUND	0
VPNOW_DIR_OUTBOUND	1
VPNOW_ETH_0	0
VPNOW_ETH_1	1
VPNOW_ETH_2	2
VPNOW_IP4_MODE	0
VPNOW_IP6_MODE	1
VPNOW_IPSEC_PROTOCOL_AH	0
VPNOW_IPSEC_PROTOCOL_ESP	1
VPNOW_IPSEC_MODE_TRANSPORT	0
VPNOW_IPSEC_MODE_TUNNEL	1
VPNOW_IV_VALUE_FROM_PACKET	0
VPNOW_IV_VALUE_FROM_SA	1
VPNOW_AUTH_HMAC_MD5	1
VPNOW_AUTH_HMAC_SHA1	2
VPNOW_AUTH_HMAC_SHA2	3
VPNOW_ENCR_AES_128	6
VPNOW_ENCR_AES_192	7
VPNOW_ENCR_AES_256	8
VPNOW_ENCR_NULL	11
VPNOW_ALGORITHM_NOT_SPECIFIED	0

6 Appendix B: Commands

List of the VPNow™ Hardware Module commands:

Command	Value
VPNOW_SET_IP_CONF	41193
VPNOW_GET_IP_CONF	41194
VPNOW_SET_IPSEC_CONF	41195
VPNOW_GET_IPSEC_CONF	41196
VPNOW_GET_IPSEC_VERSION_NR	41197
VPNOW_ANSW_TO_SET_IP_CONF	
VPNOW_ANSW_TO_GET_IP_CONF	
VPNOW_ANSW_TO_SET_IPSEC_CONF	
VPNOW_ANSW_TO_GET_IPSEC_CONF	
VPNOW_ANSW_TO_GET_IPSEC_VERSION_NR	
VPNOW_GET_SUPPORTED_ALGORITHMS	
VPNOW_ANSW_TO_GET_SUPPORTED_ALGORITHMS	
VPNOW_IPSEC_LOG_READY	
VPNOW_START_IPSEC	
VPNOW_IPSEC_STARTED	
VPNOW_IPSEC_NOT_STARTED	
VPNOW_ENABLE_IPSEC	
VPNOW_DISABLE_IPSEC	
VPNOW_GET_IPSEC_STATUS	
VPNOW_IPSEC_STATUS_ENABLED	
VPNOW_IPSEC_STATUS_DISABLED	
VPNOW_IPSEC_STATUS_STOPPED	
VPNOW_STOP_IPSEC	
VPNOW_IPSEC_STOPPED	
VPNOW_ADD_POLICY	
VPNOW_REPLACE_POLICY	
VPNOW_REMOVE_POLICY	
VPNOW_CLEAR_POLICY_DATABASE	
VPNOW_ADD_BUNDLE	
VPNOW_CLOSE_BUNDLE	
VPNOW_ADD_SA_TO_BUNDLE	
VPNOW_ADD_KEYS_TO_SA	
VPNOW_GET_POLICY	

Command	Value
VPNOW_GET_NEXT_BUNDLE	
VPNOW_GET_NEXT_SA	
VPNOW_GET_SPD_ROW_COUNT	
VPNOW_CLOSE_POLICY	
VPNOW_CLOSE_SA	
VPNOW_ANSW_TO_ADD_POLICY	
VPNOW_ANSW_TO_REPLACE_POLICY	
VPNOW_ANSW_TO_REMOVE_POLICY	
VPNOW_ANSW_TO_CLEAR_POLICY_TABLE	
VPNOW_ANSW_TO_ADD_BUNDLE	
VPNOW_ANSW_TO_CLOSE_BUNDLE	
VPNOW_ANSW_TO_ADD_SA_TO_BUNDLE	
VPNOW_ANSW_TO_ADD_KEYS_TO_SA	
VPNOW_ANSW_TO_GET_POLICY	
VPNOW_ANSW_TO_GET_NEXT_BUNDLE	
VPNOW_ANSW_TO_GET_NEXT_SA	
VPNOW_ANSW_TO_GET_SPD_ROW_COUNT	
VPNOW_ANSW_TO_CLOSE_POLICY	
VPNOW_ANSW_TO_CLOSE_SA	

7 Appendix C: Messages

List of the VPNow™ Hardware Module messages:

Message	Value
VPNOW_IPSEC_SUCCESS	
VPNOW_OPERATION_OK	
VPNOW_ERR_OUT_OF_MEMORY	
VPNOW_IPSEC_FAILURE	
VPNOW_ERR_BAD_IP_CONF_STRUCT_VER	
VPNOW_ERR_BAD_IP_CONF_STRUCT_SIZE	
VPNOW_ERR_BAD_IP_CONF_ETH_VAL	
VPNOW_ERR_BAD_IP_CONF_MTU_VAL	
VPNOW_ERR_BAD_IPSEC_CONF_DF_FLAG	
VPNOW_ERR_BAD_IPSEC_CONF_STRUCT_VER	
VPNOW_ERR_BAD_IPSEC_CONF_STRUCT_SIZE	
VPNOW_ERR_BAD_FRAG_WAIT_TIME	
VPNOW_ERR_BAD_CIPH_WAIT_TIME	
VPNOW_ERR_BAD_IPSEC_CONF_PROC_METHOD	
VPNOW_ERR_BAD_PAD_MODE	
VPNOW_ERR_BAD_DBASE_CMND_STRUCT_VER	
VPNOW_ERR_BAD_DBASE_IP_MODE_VAL	
VPNOW_ERR_BAD_DBASE_ETH_VAL	
VPNOW_ERR_BAD_DBASE_DIR_VAL	
VPNOW_ERR_IP6_MODE_NOT_SUPPORTED	
VPNOW_ERR_POLICY_RULE_NOT_CLOSED	
VPNOW_ERR_IKE_NOT_SUPPORTED	
VPNOW_ERR_POLICY_WRONG_ROW_NR	
VPNOW_ERR_POLICY_ADD_FAILURE	
VPNOW_ERR_POLICY_REPLACE_FAILURE	
VPNOW_ERR_BAD_POLICY_ENTRY_STRUCT_VER	
VPNOW_ERR_BAD_POLICY_ENTRY_STRUCT_SIZE	
VPNOW_ERR_MISSING_DATA	
VPNOW_ERR_UNSUPPORTED_POLICY_ACTION	
VPNOW_ERR_REMOVE_POLICY_FAILURE	
VPNOW_ERR_CLEAR_POLICY_TABLE_FAILURE	
VPNOW_ERR_POLICY_RULE_NOT_OPENED	
VPNOW_ERR_BUNDLE_NOT_CLOSED	
VPNOW_ERR_POLICY_WITHOUT_SA	
VPNOW_ERR_CLOSE_POLICY_FAILURE	
VPNOW_ERR_BUNDLE_ADD_FAILURE	

Message	Value
VPNOW_ERR_BAD_POLICY_NR	
VPNOW_ERR_BUNDLE_NOT_OPENED	
VPNOW_ERR_BAD_BUNDLE_ID	
VPNOW_ERR_SA_ADD_FAILURE	
VPNOW_ERR_BAD_SA_ENTRY_STRUCT_VER	
VPNOW_ERR_BAD_SA_ENTRY_STRUCT_SIZE	
VPNOW_ERR_TUNNEL_IP_NOT_SPECIFIED	
VPNOW_ERR_TUNNEL_MAC_NOT_SPECIFIED	
VPNOW_ERR_IP6_MODE_NOT_ENABLED	
VPNOW_ERR_BAD_ICV_LEN	
VPNOW_ERR_BAD_AUTH_KEY_LEN	
VPNOW_ERR_UNSUPPORTED_AUTH_ALGORITHM	
VPNOW_ERR_UNSUPPORTED_ENCR_ALGORITHM	
VPNOW_ERR_UNSUPPORTED_IPSEC_PROTOCOL	
VPNOW_ERR_ALGORITHM_NOT_SPECIFIED	
VPNOW_ERR_SA_NOT_OPENED	
VPNOW_ERR_BAD_SA_ID	
VPNOW_ERR_MISSING_KEYS	
VPNOW_ERR_SA_NOT_CLOSED	
VPNOW_ERR_POLICY_RULE_NOT_CLOSED	
VPNOW_ERR_GET_POLICY_FAILURE	
VPNOW_ERR_GET_NEXT_BUNDLE_FAILURE	
VPNOW_ERR_GET_NEXT_SA_FAILURE	
VPNOW_IP_ERR_POLICY_NOT_FOUND	
VPNOW_IP_ERR_IP4_WRONG_LENGTH	
VPNOW_IP_ERR_IP4_TTL_ERROR	
VPNOW_IP_ERR_IP4_WRONG_CHECKSUM	
VPNOW_IP_ERR_IP4_WRONG_HEADER_LEN	
VPNOW_IP_ERR_IP4_DF_MF_SET	
VPNOW_IP_ERR_IP4_PROTOCOL_NOT_SET	
VPNOW_IP_ERR_IP6_WRONG_JUMBO_PAYLOAD_LENGTH	
VPNOW_IP_ERR_IP6_JUMBO_PAYLOAD_OPTION_MISSING	
VPNOW_IP_ERR_IP6_WRONG_PAYLOAD_LENGTH	
VPNOW_IP_ERR_CORRUPTED_FRAME	
VPNOW_IP_ERR_IP6_JUMBO_WITH_FRAGMENT_HEADER	
VPNOW_IP_ERR_IP6_FRAGMENT_DATA_ERROR	
VPNOW_IP_ERR_WRONG_VERSION	
VPNOW_IP_ERR_UNPROTECTED_INBOUND_PACKET	
VPNOW_IPSEC_ERR_DECRYPTION_FAILURE	
VPNOW_IPSEC_ERR_TUNNEL_IP_WRONG_MODE	
VPNOW_IPSEC_ERR_OUTBOUND_SA_MISSING	
VPNOW_IPSEC_ERR_WRONG_SRC_IP_SEL_VALUE	

Message	Value
VPNOW IPSEC ERR REQUIRED AND APPLIED IPSEC DIFFER	
VPNOW IP ERR FRAG WAIT TIME EXPIRED	
VPNOW IPSEC ERR CRYPTO WAIT TIME EXPIRED	
VPNOW IP ERR FRAGMENT DATA CORRUPTED	
VPNOW IP ERR CORRUPTED ICMP PACKET	
VPNOW IPSEC ERR CRYPTO ICV CHECK FAILURE	

8 References

[RFC_ARC]	rfc2401: Security Architecture for the Internet Protocol
[RFC_AH]	rfc2402: IP Authentication Header
[RFC_ESP]	rfc2406: IP Encapsulating Security Payload (ESP)
[RFC_AN]	rfc1700: Assigned Numbers
[RFC_IPv4]	rfc791: Internet Protocol
[RFC_IPv6]	rfc1883: Internet Protocol, Version 6 (IPv6)
[RFC_ICMPv4]	rfc792: Internet Control Message Protocol
[RFC_ICMPv6]	rfc1885: Internet Control Message Protocol (ICMPv6)
[RFC_PMTU]	rfc1191: Path MTU Discovery
[RFC_PMTUv6]	rfc1981: Path MTU Discovery for IP version 6