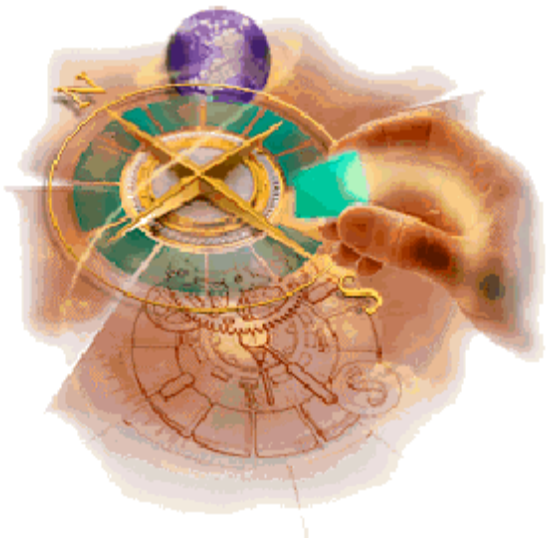
Novell®

ODI IPSec Offloading Support

Proposal

Documentation Version 1.00
July 31, 2000



Disclaimer

Novell, Inc. makes no representations or warranties with respect to the content or use of this manual, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without obligation to notify any person or entity of such changes.

This product may require export authorization from the U.S. Department of Commerce prior to exporting from the U.S. or Canada.

U.S. Patent Nos 5,553,139; 5,553,143, 5,677,851; 5,758,069; 5,784,560; 5,818,936; 5,864,865; 5,903,650; 5,905,860; 5,910,803 and other Patents Pending.

trademarks

Novell and NetWare are registered trademarks of Novell, Inc. in the United States and other countries.

All other company and product names are trademarks or registered trademarks of their respective owners.

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.

www.novell.com

Contents

1 Introduction

Many of the new Network Interface Cards (NICs) have the capability of performing crypto operations. Providing the IP protocol stack with the ability to offload crypto operations, NICs can free up cycles from the host CPU and increase system performance. This document describes the proposed ODI IPSec Offloading support. If approved these changes would eventually become part of the next C ODI Specification revision.

The following is a list of issues that must be understood when reviewing this proposal:

- The support described here does not imply that any or all of the IPSec functions are completely removed from the host.
- Support for ODI IPSec Offloading relies on the ability to pass auxiliary data between modules in the data path as defined by the ODI ECB Extensions document: *ODI Specification Supplement: ECB Extensions*.
- ODI IPSec Offloading will only be provided for the NetWare5.x platform.
- IHVs with hardware supporting IPSec Offloading will need to make the appropriate changes to their drivers.
- IP Protocol Stack will need to be updated to use the IPSec hardware offloading capabilities of the hardware.
- An understanding of IP Security as specified in the following RFCs and drafts published by the Security Working Group of the Internet Engineering Task Force (IETF):
  - Security Architecture for the Internet Protocol (RFC 2401)
  - IP Authentication Header (RFC 2402)
  - The use of HMAC-MD5-96 within ESP and AH (RFC 2403)
  - The use of HMAC-SHA-1-96 within ESP and AH (RFC 2404)
  - HMAC-MD5 IP Authentication with Replay Prevention (RFC 2085)
  - IP Encapsulating Security Payload (ESP) (RFC 2406)
  - The ESP CBC-Mode Cipher Algorithms (RFC2451)
  - The NULL Encryption Algorithm and its Use with IPSec (RFC 2410)

2 IP Security

This section defines the ODI interface between the IP protocol stack and a driver/hardware that supports IPSec offloading.

ODI defines a mechanism to make use of the hardware functions to support IPSec for normal packet processing. The IPSec enabled IP protocol stack bound to a driver that supports offloading of IPSec crypto operations performs the entire packet processing required for IPSec with the exception of functions provided by the driver/hardware. The stack provides all of the necessary information to the driver so that the driver/hardware can assist with the necessary IPSec functions for packet transmission and reception processing.

The Management ECB and Auxiliary Data Blocks used for enabling IP Security (IPSec) functions in the driver/hardware are described in this section.

It should be noted that having the driver perform the security tasks utilizing the host processor does not provide the gains that are hoped for by using hardware accelerators. MLIDs/HSMs should always provide accurate capabilities information based on the actual hardware support being utilized.

2.1 IPSec Management ECB

This section describes the IPSec Management ECB functions defined for use with IPSec. The IPSec enabled protocol stack maintains Security Associations (SA) and communicates changes to the driver using the IPSec Management ECB.

The following functions will be supported using the IPSec Management ECB.

- Adding Security Associations
- Getting Security Capabilities
- Getting Security Statistics
- Removing Security Associations
- Setting Security Capabilities

The ODI Specification may define additional IPSec Management ECB functions for other driver/hardware features as it becomes necessary and available.

The IPSec Management ECB consists of a header followed by function specific data. The header is defined as:

```
typedef struct _IPSEC_ECB_HEADER
{
        struct _IPSEC_ECB_ HEADER    *nextLink;
        struct _IPSEC_ECB_HEADER     *previousLink;
        UINT16                       status;
        void                         (ESR)(struct _IPSEC_ECB_*);
        UINT16                       stackID;
        PROT_ID                      protocolID;
        UINT32                       boardNumber;
        UINT32                       version;
        UINT32                       function;

        union
        {
                IPSEC_SA_SET         securityAssociationSet;     /* Add SA */
                IPSEC_HW_CAP         getHWCapabilitiesSet;       /* Get Hardware Capabilities */
                UINT32               setCapabilitiesMask;        /* enable Capabilities */
```

```
        void            SAhandle;           / * remove SA */
    } funcData;

} IPSEC_ECB_HEADER;
```

nextLink — On entry this field shall be initialized to zero. It is returned as zero.

previousLink — On entry this field shall be initialized to zero. It is returned as zero.

Status — On entry this field shall be initialized to zero.
On exit, this field shall contain the completion code:

ODISTAT_SUCCESSFUL
: The Management ECB was delivered and acted on.

ODISTAT_BAD_PARAMETER
: The Management ECB was not delivered or acted on.

ODISTAT_BAD_COMMAND
: The function specified is not supported.

ODISTAT_OUT_OF_RESOURCES
: Unable to add the SA

ODISTAT_ITEM_NOT_PRESENT
: The feature specified to be enabled is not supported.

ESR — Pointer to the Event Service Routine (ESR) to call after the requested function has been completed. This field may be set to NULL if an ESR is not needed.

stackID — The protocol stack ID of the IPSec stack assigned by the LSL.

ProtocolID — This field shall be set to ‘IPSEC’ (left justified and NULL extended).

boardNumber — The board number the function is to be performed for.

version — The version of the IPSec Management ECB. The version is defined by IPSEC_MANAGEMENT_ECB_VERSION.

function — The IPSec function to perform. Refer to the individual functions for the valid functional data values associated with this field. Valid values for this field are:

IPSEC_ADD_SECURITY_ASSOCIATION
: This function is used to add one or more Security Association(s) (SA).

IPSEC_GET_SECURITY_CAPABILITIES
: This function is used to get the security capabilities of the driver/hardware.

IPSEC_REMOVE_SECURITY_ASSOCIATION
: This function is used to remove a security association.

IPSEC_SET_SECURITY_CAPABILITIES
: This function is used to enable/disable security capabilities of driver/hardware.

FuncData The function specific payload as defined in the referenced sections:

securityAssociationSet	Refer to section 2.1.1 for the definition and usage of this field.
getHWCapabilitiesSet	Refer to section 2.1.2 for the definition and usage of this field.
SAhandle	Refer to section 2.1.3 for the definition and usage of this field.
SetHWCapabilitiesSet	Refer to section 2.1.4 for the definition and usage of this field.

2.1.1 Add Security Association

This function (IPSEC_ADD_SECURITY_ASSOCIATION) is used to add one or more Security Association(s). The Security Association contains the address, algorithms, keys and other necessary information needed to protect the data. Security Associations and their Security Parameter Index (SPIs) are maintained by the key-management protocol of IPSec.

The securityAssociationSet field is used to provide the needed information to add one or more Security Association(s). The format of the data passed in the IPSec Management ECB to the driver is defined as follows:

```
typedef struct _IPSEC_SA_SET_
        {
                UINT32          verIPSEC_SA_SET;
                UINT32          numSAs;
                UINT32          setAttributes;
                IPSEC_SA        securityAssociation[1];
        } IPSEC_SA_SET;
```

verIPSEC_SA_SET	Version of the IPSEC_SA_SET structure.
numSAs	Number of SAs in this set. There must be at least one.
setAttributes	Attributes applied to all SAs in this set . IPSEC_SET_ATT_GRPHINT_HARDCACHE All SAs in this ECB should be cached together on the hardware if possible. This group of SA's has a high probability of being used often.
securityAssociation	An array of IPSEC_SA structures defining the SAs being added. The 2nd IPSEC_SA follows immediately after the 1st.

The securityAssociation field is used to provide the needed information to add a Security Association. The format is defined as follows:

```
typedef struct _IPSEC_SA_
{
        UINT32              verIPSEC_SA;
        void                *SAhandle;
        SA_ATTRIBUTES       attributes;
        UINT32              SPI;
        UINT32              statusSA;

        IP_ADDR             destinationAddr;
        IP_ADDR             sourceAddr;

        UINT32              confAlgorithm;
        UINT32              confKeyLength;

        UINT32              authAlgorithm;
        UINT32              authKeyLength;

        UINT32              keyBufferLen;
        MEON                keyBuffer[1];
} IPSEC_SA;
```

verIPSEC_SA — The IPSEC_SA Structure Version.

SAhandle — Returned by the driver, and is used by IPSec to indicate this SA in future commands.

attributes — This field indicates attributes of the SA.

```
typedef struct _ SA_ATTRIBUTES _
{
        UINT8               securityService;
        UINT8               priority;
        UINT16              flags;
        UINT32              reserved;
} SA_ATTRIBUTES;
```

securityService — This field indicates the type of security service represented by this Security Association. The following are valid for this field.

IPSEC_SERVICE_AH
: This Security Association is for an Authentication Header (i.e., AH).

IPSEC_SERVICE_ESP
: This Security Association if for an Encapsulated Security Payload (i.e., ESP).

priority — This field indicates the relative importance of keeping the SA cached on the adapter hardware:
0 do not care, 1 highest – >127 lowest.

	flags	This field indicates specific attributes that apply to the SA. IPSEC_ATT_BUNDLE_BIT This bit indicates that this SA is part of an SA-bundle with the following SA. SA-bundles will always be added/deleted together, and the last SA in the bundle must not set this bit. IPSEC_ATT_TX_ SA This bit indicates that the Security Association is for data transmission. IPSEC_ATT_RX_SA This bit indicates that the Security Association is for data reception. IPSEC_ATT_PRIMARYUSE_SA This bit indicates that the Security Association is for Primary Use used in normal packet processing. IPSEC_ATT_SECONDUSE_SA This bit indicates that the Security Association is for Secondary Use used in normal packet processing. IPSEC_ATT_HARDCACHE This bit indicates that the Security Association should only be accepted if it will be cached on the hardware.
SPI		Security Parameter Index used in combination with the security service and the IP destination address to uniquely identify the Security Association.
statusSA		The current Status of this SA. Initially set to 0, this field is updated by the adapter to indicate if this SA was accepted or rejected. The actual values for this field are TBD.
destinationAddr		The IP address of the destination host receiving the packet.
sourceAddr		The IP address of the source host sending the packet.
confAlgorithm		This field indicates the Confidentiality algorithm to be used. The following are valid values for this field. IPSEC_ALG_CBC_3DES IPSEC_ALG_CBC_DES IPSEC_ALG_NULL
ConfKeyLength		The length, in bytes, of the Confidentiality key in the keybuffer. It is always the first key in the key buffer.

AuthAlgorithm	This field indicates the Authentication algorithm to be used. The following are valid values for this field. IPSEC_ALG_MD5 IPSEC_ALG_SHA_1 IPSEC_ALG_HMAC_MD5 IPSEC_ALG_HMAC_SHA_1_96 IPSEC_ALG_NULL
AuthKeyLength	The length, in bytes, of the Authentication key. It follows immediately behind the Confidentiality key in the key buffer.
keyBufferLen	The length in bytes of the keybuffer.
keyBuffer	The buffer itself. This space is allocated with sufficient room for all keys defined for this SA.

2.1.2 Get Hardware Capabilities

This function (IPSEC_GET_HW_CAPABILITIES) is used to find out the security offloading capabilities of the hardware. The getHWCapabilities field is used by this function to return the security offloading capabilities of the hardware.

```
typedef struct _ IPSEC_HW_CAP _
{
        UINT32          verIPSEC_HW_CAP;
        UINT32          maxNumSA;
        UINT32          maxHardSA;
        UINT32          currNumSA;
        UINT32          currHardSA;
        UINT32          inboundcapFlags;
        UINT32          inboundAlgorCapMask;
        UINT32          inboundAlgorActiveMask;
        UINT32          outboundcapFlags;
        UINT32          outboundAlgorCapMask;
        UINT32          outboundAlgorActiveMask;
} IPSEC_HW_CAP;
```

verIPSEC_HW_CAP	Version of the IPSEC_HW_CAP structure;
maxNumSA	This field will be set to the max number of SAs that the driver/hardware can efficiently support. This number must not be less than the maxHardSA value.
maxHardSA	This field will be set to the max number of SAs that can be cached on the hardware adapter. The maximum number of SAs that can be handled efficiently should be greater than or equal to the number in this field.
currNumSA	This field indicates the number of SAs currently offloaded.
currHardSA	This field indicates the number of SAs currently cached in hardware.

inboundcapFlags	This field is a bit map field describing the capabilities of outbound services.
inboundAlgorActiveMask	This field is a bit map field indicating which algorithms have been enabled on the adapter.
inboundAlgorCapMask	This field is a bit map field that indicates all of the algorithms that the adapter can offload in the inbound (rx) path.
outboundcapFlags	This field is a bit map field describing the capabilities of outbound services.
outboundAlgorCapMask	This field is a bit map field, and will indicate all of the algorithms that the adapter can offload in the outbound (tx) path.
outboundAlgorActiveMask	This field is a bit map field indicating which algorithms have been enabled on the adapter.

Capability Flag bits:

IPSEC_CAPFLG_AH
: Can perform AH header processing.

IPSEC_CAPFLG_ESP
: Can perform ESP header processing.

IPSEC_CAPFLG_NOT_SEQUENCIAL
: Can perform only AH or ESP but not both on a single packet. By default, complete processing of both headers is defined.

IPSEC_CAPFLG_IP_V4
: IP version 4 support.

IPSEC_CAPFLG_IP_V6
: IP version 6 support.

IPSEC_CAPFLG_TUNNEL_SUPPORT
: Processing of both inner and outer IPSEC headers is supported.

Capabilities Mask bits:
The security capabilities fields are bit map fields. The following bits are defined for the Capabilities mask fields:

IPSEC_CAP_MD5	128 bits
IPSEC_CAP_SHA_1	160 bits
IPSEC_CAP_HMAC_MD5	Support for HMAC MD5, 128 bits
IPSEC_CAP_HMAC_MD5_96	96 bits
IPSEC_CAP_HMAC_SHA_1	Support for HMAC SHA-1, 160 bits
IPSEC_CAP_HMAC_SHA_1_96	96 bits

IPSEC_CAP_CBC_DES_40	Support for DES-CBC 40 bit keying
IPSEC_CAP_CBC_DES_56	Support for DES-CBC 56 bit keying
IPSEC_CAP_CBC_3DES	Support for 3DES-CBC 168 bit keying

2.1.3 Remove Security Association

This function (IPSEC_REMOVE_SECURITY_ASSOCIATION) is used to remove a Security Association. The SAhandle field is used to indicate which Security Association is to be removed. The SAhandle was returned by the driver when the Security Association was added.

SAhandle — The handle returned by the driver in the Add SA command. If the SAhandle is for an SA with the SA-bundle bit set, then all the SAs will be removed in sequence until an SA without the SA-Bundle bit is found. Care must be taken to remove the first SA in a bundle in order to ensure that all SAs in a bundle are removed.

2.1.4 Set Security Capabilities

This function (IPSEC_SET_SECURITY_CAPABILITIES) is used to enable capabilities of the driver/hardware. The IPSEC management ECB setCapabilitiesMask field is used to enable the security features of the driver/hardware.

```
typedef struct _ IPSEC__HW_CAP _SET_
{
        UINT32          verIPSEC_HW_CAP_SET;
        UINT32          inboundCapFlags;
        UINT32          inboundAlgorReqMask;
        UINT32          inboundAlgorActiveMask;
        UINT32          outboundCapFlags;
        UINT32          outboundAlgorReqMask;
        UINT32          outboundAlgorActiveMask;
} IPSEC_HW_CAP_SET;
```

verIPSEC_HW_CAP_SET Version of this structure.

inboundCapFlags — This field is a bit map field indicating the information about the inbound service ---TBD.

inboundAlgorReqMask — Inbound Algorithm Request Mask is a bit map field that indicates the algorithms that are being requested to be enabled or disabled as indicated in the capFlags field.

inboundAlgorActiveMask — This field is a bit map field indicating which algorithms have been activated.

outboundCapFlags — This field is a bit map field indicating the information about the outbound service ---TBD.

outboundAlgorReqMask — Outbound Algorithm Request Mask is a bit map field that indicates the algorithms that are being requested to be enabled or disabled as indicated in the capFlags field.

outboundAlgorActiveMask This field is a bit map field indicating which algorithms have been activated.

Capability Flag bits:

IPSEC_CAPFLGS_DISABLE
When this bit is set, the Request mask indicates which Algorithms to disable.

Other bits are TBD.

Capabilities Mask bits:
The security capabilities fields are bit map fields. The following bits are defined for the Capabilities mask fields:

IPSEC_CAP_MD5	128 bits
IPSEC_CAP_SHA_1	160 bits
IPSEC_CAP_HMAC_MD5	Support for HMAC MD5, 128 bits
IPSEC_CAP_HMAC_MD5_96	96 bits
IPSEC_CAP_HMAC_SHA_1	Support for HMAC SHA-1, 160 bits
IPSEC_CAP_HMAC_SHA_1_96	96 bits
IPSEC_CAP_CBC_DES_40	Support for DES-CBC 40 bit keying
IPSEC_CAP_CBC_DES_56	Support for DES-CBC 56 bit keying
IPSEC_CAP_CBC_3DES	Support for 3DES-CBC 168 bit keying

2.2 IPSec Auxiliary Data Blocks

2.2.1 Transmit Auxiliary Data

The Auxiliary Data Block used by IPSec during the transmission of data is defined as:

```
typedef struct _IPSEC_TX_AUX_DATA_BLOCK_
{
        AUX_DATA_BLOCK_HDR      hdr;
        void                    outer_AH_SAhandle;
        void                    outer_ESP_SAhandle;
        void                    inner_AH_SAhandle;
        void                    inner_ESP_SAhandle;

} IPSEC_TX_AUX_DATA_BLOCK;
```

hdr.type	The type field of the Auxiliary Data Block is set to IPSEC_TX.
hdr.version	Set to AUX_DATA_BLOCK_HDR_VERSION.

hdr.length	24 bytes; 16 bytes for the auxiliary data block header plus the size of payload in bytes.
hdr.payloadVersion	Set to IPSEC_TX_PAYLOAD_VERSION.
outer_AH_SAhandle	SAhandle for the Outer IPSec AH header, set to NULL if not used/needed.
outer_ESP_SAhandle	SAhandle for the Outer IPSec ESP header, set to NULL if not used/needed.
inner_AH_SAhandle	SAhandle for the Inner IPSec AH header (tunneled header) set to NULL if tunnelling is not used/needed.
inner_ESP_SAhandle	SAhandle for the Inner IPSec ESP header (tunneled header) set to NULL if tunnelling is not used/needed.

2.2.2 Receive Auxiliary Data

The Auxiliary Data Block used by IPSec during the reception of data is defined as:

```
typedef struct _IPSEC_RX_AUX_DATA_BLOCK_
{
        AUX_DATA_BLOCK_HDR      hdr;
        UINT32                  IPSecStatus;
        UINT32                  reserved;

}IPSEC_RX_AUX_DATA_BLOCK;
```

hdr.type	The Type field of the Auxiliary Data Block is set to IPSEC_RX.
hdr.version	Set to AUX_DATA_BLOCK_HDR_VERSION.
hdr.length	24 bytes: 16 bytes for the auxiliary data block header plus the size of payload in bytes.
hdr.payloadVersion	Set to IPSEC_RX_PAYLOAD_VERSION.
IPSecStatus	Status bits that must be set accordingly may include but not be limited to the following: IPSEC_OUTER_AH_CHECKED IPSEC_OUTER_AH_MATCH IPSEC_OUTER_ESP_DECRYPTED IPSEC_OUTER_ESP_AUTH_CHECKED IPSEC_OUTER_ESP_AUTH_MATCHED IPSEC_INNER_AH_CHECKED IPSEC_INNER_AH_MATCH IPSEC_INNER_ESP_DECRYPTED IPSEC_INNER_ESP_AUTH_CHECKED IPSEC_INNER_ESP_AUTH_MATCHED
Reserved	For this version of the spec, this field shall be set to 0.

2.2.3 Secondary Use Auxiliary Data

The Auxiliary Data Block used by IPSec during the Secondary Use operation is defined as:

```
typedef struct _IPSEC_SECOND_USE_AUX_DATA_BLOCK_
{
        AUX_DATA_BLOCK_HDR      hdr;
        Void                    outer_AH_SAhandle;
        void                    outer_ESP _SAhandle;
        void                    inner_AH_SAhandle;
        void                    inner_ESP_SAhandle;
        UINT32                  IPSecStatus;
        UINT32                  reserved;

}IPSEC_SECOND_USE_AUX_DATA_BLOCK;
```

hdr.type	The type field of the Auxiliary Data Block is set to IPSEC_SECOND_USE.
hdr.version	Set to AUX_DATA_BLOCK_HDR_VERSION.
hdr.length	32 bytes; 16 bytes for the auxiliary data block header plus the size of payload in bytes.
hdr.payloadVersion	Set to IPSEC_SECOND_USE_PAYLOAD_VERSION.
outer_AH_SAhandle	SAhandle for the Outer IPSec AH header, set to NULL if not used/needed.
outer_ESP_SAhandle	SAhandle for the Outer IPSec ESP header, set to NULL if not used/needed.
inner_AH_SAhandle	SAhandle for the Inner IPSec AH header (tunneled header) set to NULL if tunnelling is not used/needed.
inner_ESP_SAhandle	SAhandle for the Inner IPSec ESP header (tunneled header) set to NULL if tunnelling is not used/needed.
IPSecStatus	Status bits that must be set accordingly may include but not be limited to the following: IPSEC_OUTER_AH_CHECKED IPSEC_OUTER_AH_MATCH IPSEC_OUTER_ESP_DECRYPTED IPSEC_OUTER_ESP_AUTH_CHECKED IPSEC_OUTER_ESP_AUTH_MATCHED IPSEC_INNER_AH_CHECKED IPSEC_INNER_AH_MATCH IPSEC_INNER_ESP_DECRYPTED IPSEC_INNER_ESP_AUTH_CHECKED IPSEC_INNER_ESP_AUTH_MATCHED
Reserved	For this version of the spec, this field shall be set to 0.

2.3 Packet Transmission

When the IP protocol stack makes a transmission request on which it wants IPSec processing to be done by the driver/hardware, it will set the TX_AUX_DATA value in the n nibble of the ECB_StackID field, indicating that there is auxiliary data associated with the ECB. The stack will include within the auxiliary data an auxiliary data block of type IPSEC_TX. The IPSEC_TX auxiliary data block will contain the handle(s) to the Security Sssociation(s) that are to be use when processing the packet.

2.4 Packet Reception

During the processing of an incoming packet, the driver will pass additional information within the ECB using the auxiliary data mechanism. The driver will set the PAE_RX_AUX_DATA bit in the ECB_PreviousLink field and include an IPSEC_RX auxiliary data block in the auxiliary data that follows immediately after the valid data in the ECB fragment list.

2.5 Secondary Packet Reception

If the Adapter was not able to process a given packet on the initial reception, the IPSEC stack may use the NIC to process the packet through its secondary use interface. This is achieved by transmitting the packet, but setting the Auxiliary block type to Secondary Use. The Driver/NIC shall not place the packet on the wire, but will process the packet with the indicated SAs, and set the status in the AUX block. When the IPSEC stack gets the TX_Complete indication, then it can read the ECB and AUX data blocks for the processed packet.

3 IPSEC.H

This section defines IPSEC.H that is included by protocol stack and drivers for ODI IPSec Offloading Support.

```
/*****************************************************************
*
•   * IPSec Management ECB structure and defines.
*
*****************************************************************/
#define  IPSEC_MANAGEMENT_ECB_VERSION          1

typedef struct _IPSEC_ECB_HEADER
{
        struct _IPSEC_ECB_ HEADER    *nextLink;
        struct _IPSEC_ECB_HEADER     *previousLink;
        UINT16                       status;
        void                         (ESR)(struct _IPSEC_ECB_*);
        UINT16                       stackID;
        PROT_ID                      protocolID;
        UINT32                       boardNumber;
        UINT32                       version;
        UINT32                       function;

        union
        {
                IPSEC_SA_SET         securityAssociationSet;     /* Add SA */
                IPSEC_HW_CAP         getHWCapabilitiesSet;       /* Get Hardware Capabilities */
                UINT32               setCapabilitiesMask;        /* enable Capabilities */
                void                 SAhandle;                   / * remove SA */
        } funcData;

} IPSEC_ECB_HEADER;


/* Defines for IPSec Management ECB functions */
#define  IPSEC_ADD_SECURITY_ASSOCIATION              1
#define  IPSEC_GET_SECURITY_CAPABILITIES             2
#define  IPSEC_REMOVE_SECURITY_ASSOCIATION           3
#define  IPSEC_SET_SECURITY_CAPABILITIES             4

/*****************************************************************
*
* Defines and structures for adding security associations
*
*****************************************************************
typedef struct _IPSEC_SA_SET_
{
        UINT32              verIPSEC_SA_SET;
        UINT32              numSAs;
        UINT32              setAttributes;
        IPSEC_SA            securityAssociation;
} IPSEC_SA_SET;
```

```
/* SA_SET: Set Attributes*/
#define  IPSEC_SET_ATT_GRPHINT_HARDCACHE                1


typedef struct _IPSEC_SA_
{
        UINT32                  verIPSEC_SA;
        void                    *SAhandle;
        SA_ATTRIBUTES           attributes;
        UINT32                  SPI;
        UINT32                  statusSA;

        IP_ADDR                 destinationAddr;
        IP_ADDR                 sourceAddr;

        UINT32                  confAlgorithm;
        UINT32                  confKeyLength;

        UINT32                  authAlgorithm
        UINT32                  authKeyLength;

        UINT32                  keyBufferLen;
        MEON                    keyBuffer[1];
} IPSEC_SA;

typedef struct _ SA_ATTRIBUTES _
{
        UINT8                   securityService;
        UINT8                   priority;
        UINT16                  flags;
        UINT32                  reserved;
} SA_ATTRIBUTES;


/* SA_ATTRIBUTES: Security Services  */
#define  IPSEC_SERVICE_AH                               0x0001
#define  IPSEC_SERVICE_ESP                              0x0002


/* SA_ATTRIBUTES: Flag          */
#define  IPSEC_ATT_BUNDLE_BIT                           0x0001
#define  IPSEC_ATT _TX_ SA                              0x0002
#define  IPSEC_ATT _RX_SA                               0x0004
#define  IPSEC_ATT _PRIMARY_USE_SA                      0x0008
#define  IPSEC_ATT _SECOUND_USE_SA                      0x0010
#define  IPSEC_ATT _HARDCACHE                           0x0020


/* Confidentiality Algorithm ID defines - */
#define  IPSEC_ALG_NULL                         0x0000
#define  IPSEC_ALG_CBC_DES                      0x0001
#define  IPSEC_ALG_CBC_3DES                     0x0002
```

```
/* Authentication Algorithm ID defines */
/* (use of IPSEC_ALG_NULL for Authentication is also valid ) */
#define  IPSEC_ALG_MD5                      0x0001
#define  IPSEC_ALG_SHA_1                    0x0002
#define  IPSEC_ALG_HMAC_MD5                 0x0003
#define  IPSEC_ALG_HMAC_MD5_96              0x0004
#define  IPSEC_ALG_HMAC_SHA_1               0x0005
#define  IPSEC_ALG_HMAC_SHA_1_96            0x0006

/*****************************************************************
*
*       Get Hardware Capabilities Structure Definition.
*
*****************************************************************/
typedef struct _ IPSEC_HW_CAP _
{
        UINT32          verIPSEC_HW_CAP;
        UINT32          maxNumSA;
        UINT32          maxHardSA;
        UINT32          currNumSA;
        UINT32          currHardSA;
        UINT32          inboundCapFlags;
        UINT32          inboundAlgorCapMask;
        UINT32          inboundAlgorActiveMask;
        UINT32          outboundCapFlags;
        UINT32          outboundAlgorCapMask;
        UINT32          outboundAlgorActiveMask;
} IPSEC_HW_CAP;

/*****************************************************************
*
* Defines for Get Cap Flags bits.
*
*****************************************************************/
#define  IPSEC_CAPFLG_AH                    0x0001
#define  IPSEC_CAPFLG_ESP                   0x0002
#define  IPSEC_CAPFLG_NOT_SEQUENCIAL        0x0004
#define  IPSEC_CAPFLG_IP_V4                 0x0008
#define  IPSEC_CAPFLG_IP_V6                 0x0010
#define  IPSEC_CAPFLG_TUNNEL_SUPPORT        0x0020

/*****************************************************************
*
*       Set Hardware Capabilities Structure Definition.
*
*****************************************************************/
typedef struct _ IPSEC__HW_CAP _SET_
{
        UINT32          verIPSEC_HW_CAP_SET;
        UINT32          inboundCapFlags;
        UINT32          inboundAlgorReqMask;
        UINT32          inboundAlgorActiveMask;
        UINT32          outboundCapFlags;
        UINT32          outboundAlgorReqMask;
        UINT32          outboundAlgorActiveMask;
```

```
} IPSEC_HW_CAP_SET;

/*****************************************************************
*
* Defines for Set Cap Flags bits.
*
*****************************************************************/
#define IPSEC_CAPFLGS_DISABLE        0x8000



/*****************************************************************
*
* Defines for Get and Set Algorithm Mask bits.
*
*****************************************************************/

#define  IPSEC_CAP _MD5                      0x0001
#define  IPSEC_CAP_SHA_1                     0x0002
#define  IPSEC_CAP_HMAC_MD5                  0x0003
#define  IPSEC_CAP_HMAC_MD5_96               0x0004
#define  IPSEC_CAP_HMAC_SHA_1                0x0005
#define  IPSEC_CAP_HMAC_SHA_1_96             0x0006
#define  IPSEC_CAP_CBC_DES_40                0x0007
#define  IPSEC_CAP_CBC_DES_56                0x0008
#define  IPSEC_CAP_CBC_3DES                  0x0009



/*****************************************************************
*
*        Auxiliary data block for packet transmission
*
*****************************************************************/
#define  IPSEC_TX_PAYLOAD_VERSION          1


typedef struct _IPSEC_TX_AUX_DATA_BLOCK_
{
        AUX_DATA_BLOCK_HDR      hdr;
        void                    outer_AH_SAhandle;
        void                    outer_ESP_SAhandle;
        void                    inner_AH_SAhandle;
        void                    inner_ESP_SAhandle;
} IPSEC_TX_AUX_DATA_BLOCK;



/*****************************************************************
*
* Auxiliary data block for packet reception
*
*****************************************************************/
#define  IPSEC_RX_PAYLOAD_VERSION          1
```

```
typedef struct _IPSEC_RX_AUX_DATA_BLOCK_
{
        AUX_DATA_BLOCK_HDR      hdr;
        UINT32                  IPSecStatus;
        UINT32                  reserved;

}IPSEC_RX_AUX_DATA_BLOCK;

/* Possible Status Attributes to be set in Aux Data Block status field */
IPSEC_OUTER_AH_CHECKED                  0x00000001
IPSEC_OUTER_AH_MATCH                    0x00000002
IPSEC_OUTER_ESP_DECRYPTED               0x00000004
IPSEC_OUTER_ESP_AUTH_CHECKED            0x00000008
IPSEC_OUTER_ESP_AUTH_MATCHED            0x00000010
IPSEC_INTER_AH_CHECKED                  0x00000020
IPSEC_INTER_AH_MATCH                    0x00000040
IPSEC_INTER_ESP_DECRYPTED               0x00000080
IPSEC_INTER_ESP_AUTH_CHECKED            0x00000100
IPSEC_INTER_ESP_AUTH_MATCHED            0x00000200



/*****************************************************************
*
* Auxiliary data block for Secondary Use reception
*
*****************************************************************/
#define  IPSEC_SECOND_USE_PAYLOAD_VERSION      1

typedef struct _IPSEC_SECOND_USE_AUX_DATA_BLOCK_
{
        AUX_DATA_BLOCK_HDR      hdr;
        void                    outer_AH_SAhandle;
        void                    outer_ESP_SAhandle;
        void                    inner_AH_SAhandle;
        void                    inner_ESP_SAhandle;
        UINT32                  IPSecStatus;
        UINT32                  reserved;

}IPSEC_SECOND_USE_AUX_DATA_BLOCK;
```