Network Working Group
Internet Draft
Intended status: Standard Track
Expires: September 18, 2016

K. Tran
Ericsson
H. Wang
V. Nagaraj
X. Chen
Huawei Technologies

March 18, 2016

Yang Data Model for Internet Protocol Security (IPsec)
draft-tran-ipsecme-yang-01.txt

Abstract

This document defines a YANG data model that can be used to configure and manage Internet Protocol Security (IPsec). The model covers the IPsec protocol operational state, remote procedural calls, and event notifications data.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on November 18, 2016.

Table of Contents

1. Introduction

Internet Protocol Security (IPsec) is a suite of protocols that provides security to internet communications at the IP layer. This document defines a YANG data model that can be used to configure and manage the IPsec protocol including Encapsulating Security Payload (ESP), Authentication Header (AH), Internet Key Exchange version 1 (IKEv1), and Internet Key Exchange version 2 (IKEv2) components.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s) indicates a compliance requirement statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the explicit compliance requirements of this RFC.

3. IPsec Configuration and Operation Model Overview

This section will give the relationship of AH/ESP/SA with IPsec, and IKEv2 with IPsec.

Figure 1 shows the protocols of (AH and ESP) associated with IPsec.

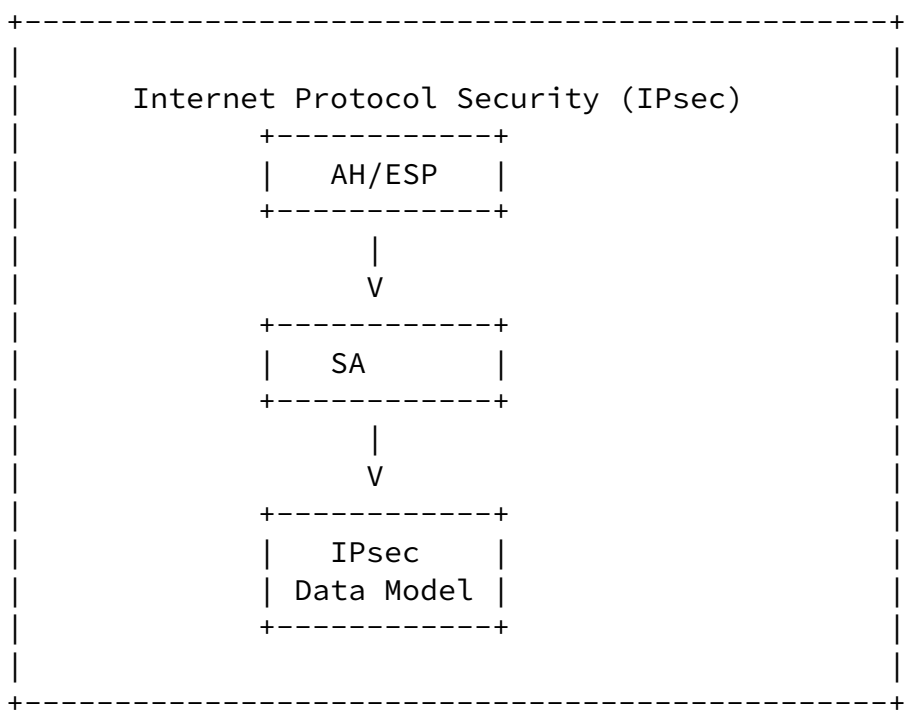

Figure 1. Relationship between AH/ESP/SA and IPsec

3.1. IPsec Configuration Data Model

The IPsec data model provides the appropriate leaves for configuring the IPsec protocol. The IPsec YANG data model shall contain AH, ESP, Security Policy (SP), Security Policy Database (SPD), Security Association (SA), Security Association Database (SAD), and Peer Authorization Database (PAD) components.

module: ietf-ipsec

```
+--rw ipsec {ipsec}?
   |  +--rw sad {ipsec-sad}?
   |  |  +--rw sad-entries* [spi direction]
   |  |     +--rw spi                                uint32
   |  |     +--rw anti-replay-window?                uint16
   |  |     +--rw ip-comp?                           empty
   |  |     +--rw local-peer
   |  |     |  +--rw (ip-address)?
   |  |     |     +--:(ipv4-address)
   |  |     |     |  +--rw ipv4-address?   inet:ipv4-address
   |  |     |     +--:(ipv6-address)
   |  |     |        +--rw ipv6-address?   inet:ipv6-address
   |  |     +--rw remote-peer
   |  |     |  +--rw (ip-address)?
   |  |     |     +--:(ipv4-address)
   |  |     |     |  +--rw ipv4-address?   inet:ipv4-address
```

```
|  |     |     +--:(ipv6-address)
|  |     |        +--rw ipv6-address?   inet:ipv6-address
|  |     +--rw sa-mode?                          ipsec-mode
|  |     +--rw security-protocol?                ipsec-protocol
|  |     +--rw sequence-number?                  uint64
|  |     +--rw sequence-number-overflow-flag?    boolean
|  |     +--rw path-mtu?                         uint16
|  |     +--rw life-time
|  |     |  +--rw life-time-in-seconds?          uint32
|  |     |  +--rw remain-life-time-in-seconds?   uint32
|  |     |  +--rw life-time-in-byte?             uint32
|  |     |  +--rw remain-life-time-in-byte?      uint32
|  |     +--rw upper-protocol?                   string
|  |     +--rw direction                         ipsec-traffic-direction
|  |     +--rw source-address
|  |     |  +--rw (ip-address)?
|  |     |  |  +--:(ipv4-address)
|  |     |  |  |  +--rw ipv4-address?   inet:ipv4-address
|  |     |  |  +--:(ipv6-address)
|  |     |  |     +--rw ipv6-address?   inet:ipv6-address
|  |     |  +--rw port-number?    uint32
|  |     +--rw destination-address
```



```
|  |     |  +--rw (ip-address)?
|  |     |  |  +--:(ipv4-address)
|  |     |  |  |  +--rw ipv4-address?   inet:ipv4-address
|  |     |  |  +--:(ipv6-address)
|  |     |  |     +--rw ipv6-address?   inet:ipv6-address
|  |     |  +--rw port-number?    uint32
|  |     +--rw nat-traversal-flag?               boolean
|  |     +--rw ah
|  |     |  +--rw (authentication-algorithm)?
|  |     |     +--:(hmac-aes-xcbc)
|  |     |     |  +--rw hmac-aes-xcbc
|  |     |     |     +--rw key-str?   union
|  |     |     +--:(hmac-md5-96)
|  |     |     |  +--rw hmac-md5-96
|  |     |     |     +--rw key-str?   union
|  |     |     +--:(hmac-sha1-96)
|  |     |     |  +--rw hmac-sha1-96
|  |     |     |     +--rw key-str?   union
|  |     |     +--:(key-string)
|  |     |        +--rw key-string
```

```
|  |     |           +--rw key-str?   union
|  |     +--rw esp
|  |        +--rw authentication
|  |        |  +--rw (authentication-algorithm)?
|  |        |     +--:(hmac-aes-xcbc)
|  |        |     |  +--rw hmac-aes-xcbc
|  |        |     |     +--rw key-str?   union
|  |        |     +--:(hmac-md5-96)
|  |        |     |  +--rw hmac-md5-96
|  |        |     |     +--rw key-str?   union
|  |        |     +--:(hmac-sha1-96)
|  |        |     |  +--rw hmac-sha1-96
|  |        |     |     +--rw key-str?   union
|  |        |     +--:(key-string)
|  |        |        +--rw key-string
|  |        |           +--rw key-str?   union
|  |        +--rw encryption
|  |           +--rw (encryption-algorithm)?
|  |              +--:(des3-cbc)
|  |              |  +--rw des3-cbd
|  |              |     +--rw key-str?   union
|  |              +--:(aes-128-cbc)
|  |              |  +--rw aes-128-cbc
|  |              |     +--rw key-str?   union
|  |              +--:(aes-192-cbc)
|  |              |  +--rw aes-192-cbc
|  |              |     +--rw key-str?   union
|  |              +--:(aes-256-cbc)
|  |              |  +--rw aes-256-cbc
```



```
|  |              |     +--rw key-str?   union
|  |              +--:(des-cbc)
|  |              |  +--rw des-cbc
|  |              |     +--rw key-str?   union
|  |              +--:(key-string)
|  |                 +--rw key-string
|  |                    +--rw key-str?   union
|  +--rw proposal {ipsec-proposal}?
|  |  +--rw ipsec-proposal* [name]
|  |     +--rw name        string
|  |     +--rw ah?         ike-integrity-algorithm-t
|  |     +--rw esp
```

```
|  |     |  +--rw authentication?   ike-integrity-algorithm-t
|  |     |  +--rw encryption?       ike-encryption-algorithm-t
|  |     +--rw ip-comp?    empty
|  |     +--rw lifetime
|  |        +--rw kbytes?    uint32
|  |        +--rw seconds?   uint32
|  +--rw spd {ipsec-spd}?
|  |  +--rw spd-entries* [name]
|  |     +--rw name                          string
|  |     +--rw description?                  string
|  |     +--rw anti-replay-window?           uint32
|  |     +--rw perfect-forward-secrecy
|  |     |  +--rw dh-group?   diffie-hellman-group-t
|  |     +--rw seq* [seq-id]
|  |        +--rw seq-id         uint32
|  |        +--rw description?   string
|  |        +--rw proposal?      leafref
|  +--rw pad
|     +--rw pad-entries* [pad-type pad-id]
|        +--rw (identity)?
|        |  +--:(ipv4-address)
|        |  |  +--rw ipv4-address?            inet:ipv4-address
|        |  +--:(ipv6-address)
|        |  |  +--rw ipv6-address?            inet:ipv6-address
|        |  +--:(fqdn-string)
|        |  |  +--rw fqdn-string?             inet:domain-name
|        |  +--:(rfc822-address-string)
|        |  |  +--rw rfc822-address-string?   string
|        |  +--:(dnX509)
|        |     +--rw dnX509?                  string
|        +--rw pad-id                   uint32
|        +--rw pad-type                 pad-type-t
|        +--rw ike-peer-name?           string
|        +--rw peer-authentication
|           +--rw algorithm?       ike-integrity-algorithm-t
|           +--rw preshared-key?   empty
|           +--rw rsa-signature?   empty
```


3.2. IKEv1 Configuration Data Model

This section will present the YANG data model for IKEv1.

```
+--rw ikev1 {ikev1}?
|  +--rw proposal* [name]
|  |  +--rw name               string
|  |  +--rw description?       string
|  |  +--rw dh-group           diffie-hellman-group-t
|  |  +--rw encryption
|  |  |  +--rw algorithm?   ike-encryption-algorithm-t
|  |  +--rw lifetime           uint32
|  |  +--rw authentication
|  |     +--rw algorithm?       ike-integrity-algorithm-t
|  |     +--rw preshared-key?   empty
|  |     +--rw rsa-signature?   empty
|  +--rw keepalive?   empty
|  +--rw policy* [name]
|     +--rw name                                string
|     +--rw mode
|     |  +--rw aggressive?   empty
|     |  +--rw main?         empty
|     +--rw connection-type                     connection-type-t
|     +--rw pre-shared-key?                     union
|     +--rw validate-certificate-identity?   empty
|     +--rw seq* [seq-id]
|     |  +--rw seq-id      uint32
|     |  +--rw proposal?   leafref
|     +--rw identity
|        +--rw local
|        |  +--rw (identity)?
|        |     +--:(ipv4-address)
|        |     |  +--rw ipv4-address?            inet:ipv4-address
|        |     +--:(ipv6-address)
|        |     |  +--rw ipv6-address?            inet:ipv6-address
|        |     +--:(fqdn-string)
|        |     |  +--rw fqdn-string?             inet:domain-name
|        |     +--:(rfc822-address-string)
|        |     |  +--rw rfc822-address-string?   string
|        |     +--:(dnX509)
|        |        +--rw dnX509?                  string
|        +--rw remote
|           +--rw (identity)?
|              +--:(ipv4-address)
|              |  +--rw ipv4-address?            inet:ipv4-address
|              +--:(ipv6-address)
|              |  +--rw ipv6-address?            inet:ipv6-address
|              +--:(fqdn-string)
|              |  +--rw fqdn-string?             inet:domain-name
```

```
|                +--:(rfc822-address-string)
|                |  +--rw rfc822-address-string?   string
|                +--:(dnX509)
|                   +--rw dnX509?                  String
```

3.3. IKEv2 Configuration Data Model

This section will present the YANG data model for IKEv2 as per RFC-7296 [RFC7296].

The IKEv2 data model provides the appropriate leaves for configuring the IKEv2 protocol. The IKEv2 YANG data model has the following structure:

```
+--rw ikev2 {ikev2}?
|  +--rw ike-global-configuration {ikev2-global}?
|  |  +--rw (df-flag)?
|  |  |  +--:(set)
|  |  |  |  +--rw set?                        empty
|  |  |  +--:(clear)
|  |  |  |  +--rw clear?                      empty
|  |  |  +--:(copy)
|  |  |     +--rw copy?                       empty
|  |  +--rw stateful-frag-check?      boolean
|  |  +--rw life-time-kb?             uint32
|  |  +--rw life-time-second?         uint32
|  |  +--rw (anti-replay)?
|  |  |  +--:(enable)
|  |  |  |  +--rw enable?                     empty
|  |  |  |  +--rw (anti-replay-windows-size)?
|  |  |  |     +--:(size-32)
|  |  |  |     +--:(size-64)
|  |  |  |     +--:(size-128)
|  |  |  |     +--:(size-256)
|  |  |  |     +--:(size-512)
|  |  |  |     +--:(size-1024)
|  |  |  +--:(disable)
|  |  |     +--rw disable?                    empty
|  |  +--rw inbound-dscp?             uint16
|  |  +--rw outbound-dscp?            uint16
|  |  +--rw local-name?               string
|  |  +--rw nat-keepalive-interval?   uint16
|  |  +--rw dpd-interval?             uint16
|  +--rw ike-peer {ikev2-peer}?
|  |  +--rw ike-peer-entries* [peer-name]
|  |     +--rw peer-name              string
|  |     +--rw ike-proposal-number?   ike-proposal-number-ref
|  |     +--rw PresharedKey?          string
|  |     +--rw nat-traversal?         boolean
|  |     +--rw (local-id-type)?
```

```
|  |     |  +--:(ip)
|  |     |  |  +--rw ip?                    empty
```



```
|  |     |  +--:(fqdn)
|  |     |  |  +--rw fqdn?                  empty
|  |     |  +--:(dn)
|  |     |  |  +--rw dn?                    empty
|  |     |  +--:(user_fqdn)
|  |     |     +--rw user_fqdn?             empty
|  |     +--rw local-id?              string
|  |     +--rw remote-id?             string
|  |     +--rw low-remote-address?    inet:ip-address
|  |     +--rw high-remote-address?   inet:ip-address
|  |     +--rw certificate?           string
|  |     +--rw auth-address-begin?    inet:ip-address
|  |     +--rw auth-address-end?      inet:ip-address
|  +--rw proposal* [name] {ikev2-proposal}?
|  |  +--rw name                      string
|  |  +--rw description?              string
|  |  +--rw dh-group                  diffie-hellman-group-t
|  |  +--rw encryption
|  |  |  +--rw algorithm?   ike-encryption-algorithm-t
|  |  +--rw pseudo-random-function    pseudo-random-function-t
|  |  +--rw authentication
|  |     +--rw algorithm?   ike-integrity-algorithm-t
|  +--rw policy* [name] {ikev2-policy}?
|     +--rw name                                string
|     +--rw authentication
|     |  +--rw preshared-key?   empty
|     |  +--rw rsa-signature?   empty
|     +--rw lifetime                            uint32
|     +--rw address-allocation
|     |  +--rw aaa?   empty
|     +--rw connection-type                     connection-type-t
|     +--rw pre-shared-key?                     union
|     +--rw validate-certificate-identity?      empty
|     +--rw seq* [seq-id]
|     |  +--rw seq-id      uint32
|     |  +--rw proposal?   leafref
|     +--rw identity
|     |  +--rw local
|     |  |  +--rw (identity)?
|     |  |     +--:(ipv4-address)
```

```
|     |  |     |  +--rw ipv4-address?           inet:ipv4-address
|     |  |     +--:(ipv6-address)
|     |  |     |  +--rw ipv6-address?           inet:ipv6-address
|     |  |     +--:(fqdn-string)
|     |  |     |  +--rw fqdn-string?            inet:domain-name
|     |  |     +--:(rfc822-address-string)
|     |  |     |  +--rw rfc822-address-string?  string
|     |  |     +--:(dnX509)
|     |  |        +--rw dnX509?                 string
```



```
|     |  +--rw remote
|     |     +--rw (identity)?
|     |        +--:(ipv4-address)
|     |        |  +--rw ipv4-address?           inet:ipv4-address
|     |        +--:(ipv6-address)
|     |        |  +--rw ipv6-address?           inet:ipv6-address
|     |        +--:(fqdn-string)
|     |        |  +--rw fqdn-string?            inet:domain-name
|     |        +--:(rfc822-address-string)
|     |        |  +--rw rfc822-address-string?  string
|     |        +--:(dnX509)
|     |           +--rw dnX509?                 string
|     +--rw description?                    String
```

3.4. IPsec Operation Data Model

The IPsec data model provides the appropriate leaves for operational states of the IPsec protocol. The IPsec YANG data model has the following structure:

```
+--ro ipsec-state {ipsec-state}?
|  +--ro policy* {ipsec-policy-state}?
|  |  +--ro name?                        string
|  |  +--ro anti-replay-window?          uint32
|  |  +--ro perfect-forward-secrecy?     diffie-hellman-group-t
|  |  +--ro seq*
|  |     +--ro seq-id?           uint32
|  |     +--ro proposal-name?    string
|  +--ro proposal* {ipsec-proposal-state}?
|  |  +--ro name?        string
|  |  +--ro ah?          ike-integrity-algorithm-t
|  |  +--ro esp
|  |  |  +--ro authentication?   ike-integrity-algorithm-t
|  |  |  +--ro encryption?       ike-encryption-algorithm-t
|  |  +--ro ip-comp?     empty
|  |  +--ro lifetime
|  |     +--ro kbytes?    uint32
|  |     +--ro seconds?   uint32
|  +--ro hold-down?    uint32 {ipsec-alarms-state}?
|  +--ro sa* {ipsec-sa-state}?
|  |  +--ro name?                         string
|  |  +--ro anti-replay-window?           uint16
|  |  +--ro ip-comp?                      empty
```

```
   |  |  +--ro spi?                          uint32 {ipsec-sa-ah-state}?
   |  |  +--ro description?                  string {ipsec-sa-ah-state}?
   |  |  +--ro authentication-algorithm?   ike-integrity-algorithm-t {ipsec-sa-
state}?
   |  |  +--ro encryption-algorithm?       ike-encryption-algorithm-t {ipsec-sa
state}?
   |  +--ro redundancy {ipsec-redundancy}?
      |     +--ro inter-chassis?   Empty
```


3.5. IKEv1 Operation Data Model

The IKEv1 data model provides the appropriate leaves for operational states of the IKEv1 protocol. The IKEv1 YANG data model has the following structure:

```
+--ro ike-state {ikev1-state}?
|  +--ro proposal* {ike-proposal-state}?
|  |  +--ro name?              string
|  |  +--ro lifetime?          uint32
|  |  +--ro encryption?        ike-encryption-algorithm-t
|  |  +--ro dh-group?          diffie-hellman-group-t
|  |  +--ro authentication?    ike-integrity-algorithm-t
|  +--ro policy* {ike-policy-state}?
|     +--ro name?               string
|     +--ro description?        string
|     +--ro mode?               enumeration
|     +--ro connection-type?    connection-type-t
|     +--ro local-identity?     inet:ipv4-address-no-zone
|     +--ro remote-identity?    inet:ipv4-address-no-zone
|     +--ro pre-shared-key?     string
|     +--ro seq?                uint32
|     +--ro proposal?           String
```

3.6. IKEv2 Operation Data Model

The IKEv2 data model provides the appropriate leaves for operational sattes of the IKEv2 protocol. The IKEv2 YANG data model has the following structure:

```
+--ro ikev2-state {ikev2-state}?
|  +--ro proposal* {ikev2-proposal-state}?
|  |  +--ro name?                      string
|  |  +--ro pseudo-random-function?   pseudo-random-function-t
|  |  +--ro authentication?           ike-integrity-algorithm-t
|  |  +--ro encryption?               ike-encryption-algorithm-t
|  |  +--ro dh-group                  diffie-hellman-group-t
|  +--ro policy* {ike-policy-state}?
|     +--ro name?              string
|     +--ro description?       string
```

```
|     +--ro mode?               enumeration
|     +--ro connection-type?    connection-type-t
|     +--ro local-identity?     inet:ipv4-address-no-zone
|     +--ro remote-identity?    inet:ipv4-address-no-zone
|     +--ro pre-shared-key?     string
|     +--ro seq?                uint32
|     +--ro proposal?           String
```


3.7. IPsec SAD Operational Data Model

The IPsec SAD(Security Association Database) container maintains information related to the IPSEC SAs established in a system. This is a run-time data structure that is created upon the first SA being established. The key for fetching SA in this database is the triplet: SPI, Protocol and Destination address of the SA to be fetched form the SA database.

The SAD entries also contain information about the IPSEC tunnel like direction, SA-type (manual or VPN SA), sequence number, anti-replay

window size, protocol mode, ipsec algorithm info, life time in Seconds/Bytes etc, NAT traversal info, path-mtu, dscp etc.

```
   +--ro sad {sad}?
   |  +--ro sad-entries* [spi security-protocol direction]
   |     +--ro spi                                ipsec-spi
   |     +--ro security-protocol                  ipsec-protocol
   |     +--ro direction                          ipsec-traffic-direction
   |     +--ro sa-type?                           enumeration
   |     +--ro sequence-number?                   uint64
   |     +--ro sequence-number-overflow-flag?     boolean
   |     +--ro anti-replay-enable-flag?           boolean
   |     +--ro anti-replay-window-size?           uint64
   |     +--ro ah-auth-algorithm?                 ipsec-authentication-algorithm
ah-authentication}?
   |     +--ro esp-integrity-algorithm?           ipsec-authentication-algorithm
esp-integrity}?
   |     +--ro esp-encrypt-algorithm?             ipsec-encryption-algorithm {ips
encrypt}?
   |     +--ro life-time
   |     |  +--ro life-time-in-seconds?            uint32
   |     |  +--ro remain-life-time-in-seconds?     uint32
   |     |  +--ro life-time-in-byte?               uint32
   |     |  +--ro remain-life-time-in-byte?        uint32
   |     +--ro protocol-mode?                     ipsec-mode
   |     +--ro tunnel-mode-process-info
   |     |  +--ro local-address?                string {ipsec-tunnel}?
   |     |  +--ro remote-address?               string {ipsec-tunnel}?
   |     |  +--ro bypass-df?                    enumeration {ipsec-tunnel}?
   |     |  +--ro dscp-flag?                    boolean {ipsec-tunnel}?
   |     |  +--ro stateful-frag-check-flag?     boolean {ipsec-tunnel}?
   |     +--ro dscp*                              uint8
   |     +--ro path-mtu?                          uint16
   |     +--ro nat-traversal-flag?                boolean
```


3.8. IPsec SPD Operational Data Model

The IPSEC SPD(Security Policy Database) container maintains policy information related to the IPSEC SAs established in a system. This is a run-time data structure that is created when the first IPSEC

policy is created.

The SPD entries also contain information about the traffic selectors, protect action (permit, deny), protocol information etc as shown below. Based on these information the IPSEC module processes the outbound and inbound traffic.

```
+--ro spd {spd}?
|  +--ro spd-entries*
|     +--ro name*
|     |  +--ro name-type?     ipsec-spd-name
|     |  +--ro name-string?   string
|     |  +--ro name-binary?   binary
|     +--ro pfp-flag?            boolean
|     +--ro traffic-selector*
|     |  +--ro local-address-low?     inet:ip-address {ipsec-local-address-r
|     |  +--ro local-address-high?    inet:ip-address {ipsec-local-address-r
|     |  +--ro remote-address-low?    inet:ip-address {ipsec-remote-address-
|     |  +--ro remote-address-high?   inet:ip-address {ipsec-remote-address-
|     |  +--ro next-protocol-low?     uint16 {ipsec-next-protocol-range}?
|     |  +--ro next-protocol-high?    uint16 {ipsec-next-protocol-range}?
|     |  +--ro local-port-low?        inet:port-number {ipsec-local-port-ran
|     |  +--ro local-port-high?       inet:port-number {ipsec-local-port-ran
|     |  +--ro remote-port-high?      inet:port-number {ipsec-remote-port-ra
|     |  +--ro remote-port-low?       inet:port-number {ipsec-remote-port-ra
|     +--ro operation?           ipsec-spd-operation
|     +--ro protect-operation
|        +--ro spd-ipsec-mode?           ipsec-mode
|        +--ro esn-flag?                 boolean
|        +--ro spd-ipsec-protocol?       ipsec-protocol
|        +--ro tunnel-mode-additional
|        |  +--ro local-address?              string {ipsec-tunnel}?
|        |  +--ro remote-address?             string {ipsec-tunnel}?
|        |  +--ro bypass-df?                  enumeration {ipsec-tunnel}?
|        |  +--ro dscp-flag?                  boolean {ipsec-tunnel}?
|        |  +--ro stateful-frag-check-flag?   boolean {ipsec-tunnel}?
|        +--ro spd-algorithm*
|           +--ro ah-auth-algorithm?         ipsec-authentication-algorithm
ah-authentication}?
|           +--ro esp-integrity-algorithm?   ipsec-authentication-algorithm
esp-integrity}?
```

```
   |           +--ro esp-encrypt-algorithm?     ipsec-encryption-algorithm {ips
encrypt}?
```

3.9. IPsec Global Statistics Operational Data Model

The IPSEC Global Statistics container is used to maintain information related to all the IPSEC tunnels established in the system. These could be related to IPv4 IPSEC tunnels or IPv6 IPSEC tunnels.

The information maintained includes: traffic sent/received on an IPSEC tunnel like number of outbound/inbound packets, number of outbound/inbound bytes, number of packets dropped, number of replayed packets, number of packet authentication failures, number of packets dropped due to queue full, number of packets dropped due to deny policy, number of packet dropped due to being malformed, number of packets dropped due to being too large.

```
+--ro ipsec-global-statistics {ipsec-global-stats}?
   +--ro ipv4
   |  +--ro inbound-packets?         uint64 {ipsec-stat}?
   |  +--ro outbound-packets?        uint64 {ipsec-stat}?
   |  +--ro inbound-bytes?           uint64 {ipsec-stat}?
   |  +--ro outbound-bytes?          uint64 {ipsec-stat}?
   |  +--ro inbound-drop-packets?    uint64 {ipsec-stat}?
   |  +--ro outbound-drop-packets?   uint64 {ipsec-stat}?
   |  +--ro dropped-packet-detail {ipsec-stat}?
   |     +--ro sa-non-exist?   uint64
   |     +--ro queue-full?     uint64
   |     +--ro auth-failure?   uint64
   |     +--ro malform?        uint64
   |     +--ro replay?         uint64
   |     +--ro large-packet?   uint64
   |     +--ro invalid-sa?     uint64
   |     +--ro policy-deny?    uint64
   |     +--ro other-reason?   uint64
   +--ro ipv6
   |  +--ro inbound-packets?         uint64 {ipsec-stat}?
   |  +--ro outbound-packets?        uint64 {ipsec-stat}?
   |  +--ro inbound-bytes?           uint64 {ipsec-stat}?
   |  +--ro outbound-bytes?          uint64 {ipsec-stat}?
   |  +--ro inbound-drop-packets?    uint64 {ipsec-stat}?
   |  +--ro outbound-drop-packets?   uint64 {ipsec-stat}?
   |  +--ro dropped-packet-detail {ipsec-stat}?
   |     +--ro sa-non-exist?   uint64
   |     +--ro queue-full?     uint64
   |     +--ro auth-failure?   uint64
   |     +--ro malform?        uint64
   |     +--ro replay?         uint64
```

```
|     +--ro large-packet?   uint64
|     +--ro invalid-sa?     uint64
|     +--ro policy-deny?    uint64
|     +--ro other-reason?   uint64
+--ro global
   +--ro inbound-packets?         uint64 {ipsec-stat}?
   +--ro outbound-packets?        uint64 {ipsec-stat}?
   +--ro inbound-bytes?           uint64 {ipsec-stat}?
   +--ro outbound-bytes?          uint64 {ipsec-stat}?
   +--ro inbound-drop-packets?    uint64 {ipsec-stat}?
   +--ro outbound-drop-packets?   uint64 {ipsec-stat}?
   +--ro dropped-packet-detail {ipsec-stat}?
      +--ro sa-non-exist?   uint64
      +--ro queue-full?     uint64
      +--ro auth-failure?   uint64
      +--ro malform?        uint64
      +--ro replay?         uint64
      +--ro large-packet?   uint64
      +--ro invalid-sa?     uint64
      +--ro policy-deny?    uint64
      +--ro other-reason?   uint64
```

3.10. RPC Operation

This section defines a list of RPC support for IPsec protocol.

```
rpcs:
   +---x clear-ipsec-group     {clear-ipsec-group}?
   |  +--ro input
   |     +--ro alarm-hold-down?     uint8
   |     +--ro ipsec-policy-name?   leafref
   +---x clear-ike-group       {clear-ike-group}?
   |  +--ro input
   |     +--ro proposal?   leafref
   +---x clear-ikev2-group     {clear-ikev2-group}?
   |  +--ro input
   |     +--ro proposal?   leafref
   +---x reset-ipv4            {reset-ipv4}?
   |  +--ro input
   |  |  +--ro ipv4?   empty
   |  +--ro output
   |     +--ro status?   string
   +---x reset-ipv6            {reset-ipv6}?
   |  +--ro input
   |  |  +--ro ipv6?   empty
   |  +--ro output
   |     +--ro status?   string
   +---x reset-global          {reset-global}?
      +--ro input
      |  +--ro ipv6?   empty
      +--ro output
         +--ro status?   string
```

3.11. Notifications

This model defines a list of notifications to inform client of important events detected during the protocol operation. These events include events related to changes in the operational state of an IKE SA, IPsec SA, Statistics etc.

```
notifications:
   +---n dpd-failure
   |  +--ro peer-id?   string
   +---n peer-authentication-failure     {peer-authentication-failure}?
   |  +--ro peer-id?   string
   +---n ike-reauth-failure              {ike-reauth-failure}?
   |  +--ro peer-id?   string
   +---n ike-rekey-failure               {ike-rekey-failure}?
   |  +--ro peer-id?     string
   |  +--ro old-i-spi?   uint64
   |  +--ro old-r-spi?   uint64
   +---n ipsec-rekey-failure             {ipsec-rekey-failure}?
      +--ro peer-id?            string
      +--ro old-inbound-spi?    ipsec-spi
      +--ro old-outbound-spi?   ipsec-spi
```

4. IPsec YANG Module

This section will present the YANG data model for IPsec, IKEv1, and IKEv2.

<CODE BEGINS> file "ietf-ipsec@2016-03-09.yang"

```
module ietf-ipsec {
  namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec";
  prefix "eipsec";

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-yang-types {
    prefix yang;
  }

  organization "Ericsson AB.
                Huawei Technologies India Pvt Ltd.";

  contact "Web:   <http://www.ericsson.com>";

  description
    "This YANG module defines the configuration and operational
     state data for Internet Protocol Security (IPSec) on
     IETF draft.
     Copyright (c) 2016 Ericsson AB.
     All rights reserved.";

  revision 2016-03-09 {
    description
      "Third revision.
```

```
        Fix YANG compiling error because it used internal
        data model econtext and should be removed in the
        draft.
        Fix warnings.
        Run validation on
        http://www.netconfcentral.org/yangdumpresults";
      reference
        "Update since second revision.";
    }
    revision 2015-09-13 {
      description
        "Second revision.";
      reference
        "updates since initial revision.
```



```
        combining:
         draft-tran-ipecme-yang-ipsec-00.
         draft-wang-ipsecme-ike-yang-00.
         draft-wang-ipsecme-ipsec-yang-00.";
    }
    revision 2015-05-14 {
      description
        "Initial revision.";
      reference
        "May 14, 2015  draft-tran-ipecme-yang-ipsec-00.
         May 22, 2015  draft-wang-ipsecme-ike-yang-00.
         June 15, 2015 draft-wang-ipsecme-ipsec-yang-00.";
    }
    /*--------------------*/
    /* Feature            */
    /*--------------------*/

    feature ikev1 {
      description
        "Feature IKEv1";
    }

    feature ike-proposal-state {
      description
        "IKEv2 Proposal Operational State";
    }

    feature ike-policy-state {
```

```
    description
      "IKEv1 Policy Operational State";
  }

  feature ikev1-state {
    description
      "IKEv1 Operational State";
  }

  feature ike-reauth-failure {
    description
      "IKEv1 Reauthorization Failure";
  }

  feature ike-rekey-failure {
    description
      "IKEv1 Rekey Failure";
  }


  feature ikev2 {
```



```
    description
      "Feature IKEv2";

  }

  feature ikev2-global {
    description
      "Feature IKEv2 Global Parameters";

  }

  feature ikev2-peer {
    description
      "Feature IKEv2 Peer";

  }

  feature ikev2-proposal {
    description
      "Feature IKEv2 Proposal";
```

```
    }

    feature ikev2-policy {
      description
        "Feature IKEv2 Policy";

    }

    feature ikev2-proposal-state {
      description
        "IKEv2 Proposal Operational State";
    }

    feature ikev2-state {
      description
        "IKEv2 Operational State";
    }

    feature ipsec {
      description
        "Feature IPsec";

    }

    feature ipsec-acl {
      description
        "Feature IPsec ACL";
```



```
    }
    feature ipsec-sad {
      description
        "Feature IPsec SAD";

    }

    feature ipsec-proposal {
      description
        "Feature IPsec Proposal";

    }
```

```
feature ipsec-spd {
  description
    "Feature IPsec SPD";

}

feature ipsec-policy-state {
  description
    "IPsec Policy Operational State";
}

feature ipsec-proposal-state {
  description
    "IPsec Proposal Operational State";
}

feature ipsec-alarms-state {
  description
    "IPsec Alarm State Operational State";
}

feature ipsec-sa-ah-state {
  description
    "IPsec SA AH Operational State";
}

feature ipsec-sa-state {
  description
    "IPsec SA Operational State";
}

feature ipsec-tunnel {
  description
    "IPsec Tunnel";
}
```



```
feature ipsec-local-address-range {
  description
    "IPsec Local Address Range";
}
```

```
feature ipsec-remote-address-range {
  description
    "IPsec Remote Address Range";
}

feature ipsec-next-protocol-range {
  description
    "IPsec Next Protocol Range";
}

feature ipsec-local-port-range {
  description
    "IPsec Local Port Range";
}

feature ipsec-remote-port-range {
  description
    "IPsec Remote Port Range";
}

feature ipsec-ah-authentication {
  description
    "IPsec AH Authentication";
}

feature ipsec-esp-integrity {
  description
    "IPsec ESP Integrity";
}

feature ipsec-esp-encrypt {
  description
    "IPsec ESP encryption";
}

feature ipsec-stat {
  description
    "IPsec Stats";
}

feature ipsec-state {
  description
    "IPsec Operational State";
}
```

```
    feature ipsec-rekey-failure {
      description
        "IPsec Rekey Failure";
    }

    feature ipsec-redundancy {
      description
        "IPsec Redundancy State";
    }

    feature sad {
      description
        "Security Association (SA) Database";
    }

    feature spd {
      description
        "Security Policy Database";
    }

    feature ipsec-global-stats {
      description
        "IPsec Global Stats";
    }

    feature clear-ipsec-group {
      description
        "Clear IPsec group";
    }

    feature clear-ike-group {
      description
        "Clear IKE group";
    }

    feature clear-ikev2-group {
      description
        "Clear IKEv2 group";
    }

    feature reset-ipv4 {
      description
        "Reset IPv4";
    }

    feature reset-ipv6 {
```

```
    description
      "Reset IPv6";
```



```
  }

  feature reset-global {
    description
      "Reset Global";
  }

  feature peer-authentication-failure {
    description
      "Peer Authentication Failure";
  }
  /*--------------------*/
  /* Typedefs           */
  /*--------------------*/

  typedef authentication-method-t {
    type enumeration {
      enum psk {
        value 0;
        description
          "Pre-Sharing Keys.";
      }
      enum certificate {
        value 1;
        description
          "Certificate.";
      }
    }
    description
      "Available authentication methods.";
  }

  /* IKEv2 Exchange Types (ET) */
  typedef ikev2-exchange-type-t {
    type enumeration {
      enum ikev2-et-ike-sa-init {
        value 34;
        description
          "ikev2-et-ike-sa-init - RFC 7296.";
```

```
        }
        enum ikev2-et-ike-auth {
          value 35;
          description
            "ikev2-et-ike-auth - RFC 7296.";
        }
        enum ikev2-et-create-child-sa {
          value 36;
          description
            "ikev2-et-create-child-sa - RFC 7296.";
```



```
        }
        enum ikev2-et-informational {
          value 37;
          description
            "ikev2-et-informational - RFC 7296.";
        }
        enum ikev2-et-ike-session-resume {
          value 38;
          description
            "ikev2-et-ike-session-resume - RFC 7296.";
        }
        enum ikev2-et-gsa-auth {
          value 39;
          description
            "ikev2-et-gsa-auth - RFC 7296.";
        }
        enum ikev2-et-gsa-registration {
          value 40;
          description
            "ikev2-et-gsa-registration - RFC 7296.";
        }
        enum ikev2-et-gsa-rekey {
          value 41;
          description
            "ikev2-et-gsa-rekey - RFC 7296.";
        }
      }
      description
        "IKEv2 Exchange Types (ET).";
    }
```

```
/* Transform Type Values (TTV), RFC 7296 */
typedef transform-type-value-t {
 type enumeration {
   enum ttv-reserved-0 {
     value 0;
     description
       "ttv-reserved-0 - Transform Type Value Reserved "+
       "(RFC 7296).";
   }
   enum ttv-encr {
     value 1;
     description
       "ttv-encr - Transform Type Value 1,
        Encryption Algorithm "+
       "(ENCR) used in IKE and ESP.";
   }
   enum ttv-prf {
     value 2;
```



```
     description
       "ttv-prf - Transform Type Value 2, "+
       "Pseudo-Random Function(PRF) used in IKE.";
   }
   enum ttv-integ {
     value 3;
     description
       "ttv-integ - Transform Type Value 3, Integrity Algorithm"+
       " (INTEG) used in IKE, AH, optional ESP.";
   }
   enum ttv-dh {
     value 4;
     description
       "ttv-dh - Transform Type Value 4, Diffie-Hellman (DH) "+
       "used in IKE, optional AH and ESP.";
   }
   enum ttv-esn {
     value 5;
     description
       "ttv-esn - Transform Type Value 5, Extended Sequence "+
       "Numbers (ESN) used in AH and ESP.";
   }
 }
```

```
      description
        "Transform Type Values (RFC 7296).";
    }

     /* IKEv2 Transform Attribute Types (TAT) */
     typedef ikev2-transform-attribute-type-t {
      type enumeration {
        enum ikev2-tat-reserved-0 {
          value 0;
          description
            "ikev2-tat-reserved-0 - IKEv2 Transform Attribute "+
            "Type Reserved-0 (RFC 7296).";
        }
        enum ikev2-tat-reserved-1 {
          value 1;
          description
            "ikev2-tat-reserved-1 - IKEv2 Transform Attribute "+
            "Type Reserved-1 (RFC 7296).";
        }
        enum ikev2-tat-reserved-13 {
          value 13;
          description
            "ikev2-tat-reserved-13 - IKEv2 Transform Attribute "+
            "Type Reserved-13 (RFC 7296).";
        }
        enum ikev2-tat-key-length {
```



```
          value 41;
          description
            "ikev2-tat-key-length - IKEv2 Transform Attribute "+
            "Type KEY LENGTH (in bits) (RFC 7296).";
        }
      }
      description
        "IKEv2 Transform Attribute Types (TAT) (RFC 7296).";
    }

     /* Transform Type 1 (Encryption Algorithm Transform IDs) */
    typedef ike-encryption-algorithm-t {
      type enumeration {
        enum encr-reserved-0 {
          value 0;
```

```
          description
            "encr-reserved-0 --> RFC_5996.";
        }
        enum encr-des-iv4 {
          value 1;
          description
            "encr-des-iv4 --> RFC_5996.";
        }
        enum encr-des {
          value 2;
          description
            "encr-des --> RFC_5996.";
        }
        enum encr-3des {
          value 3;
          description
            "encr-3des --> RFC_5996.";
        }
        enum encr-rc5 {
          value 4;
          description
            "encr-rc5 --> RFC_5996.";
        }
        enum encr-idea {
          value 5;
          description
            "encr-idea --> RFC_5996.";
        }
        enum encr-cast {
          value 6;
          description
            "encr-cast --> RFC_5996.";
        }
        enum encr-blowfish {
```



```
          value 7;
          description
            "encr-blowfish --> RFC_5996.";
        }
        enum encr-3idea {
          value 8;
          description
```

```
        "encr-3idea --> RFC_5996.";
    }
    enum encr-des-iv32 {
      value 9;
      description
        "encr-des-iv32 --> RFC_5996.";
    }
    enum encr-reserved-10 {
      value 10;
      description
        "encr-reserved-10 --> RFC_5996.";
    }
    enum encr-null {
      value 11;
      description
        "encr-null --> RFC_5996.";
    }
    enum encr-aes-cbc {
      value 12;
      description
        "encr-aes-cbc --> RFC_5996.";
    }
    enum encr-aes-ctr {
      value 13;
      description
        "encr-aes-ctr --> RFC_5996.";
    }
    enum encr-aes-ccm-8 {
      value 14;
      description
        "encr-aes-ccm-8 --> RFC_5996.";
    }
    enum encr-aes-ccm-12 {
      value 15;
      description
        "encr-aes-ccm-12 --> RFC_5996.";
    }
    enum encr-aes-ccm-16 {
      value 16;
      description
        "encr-aes-ccm-16 --> RFC_5996.";
    }
```

```
enum encr-reserved-17 {
  value 17;
  description
    "encr-reserved-17 --> RFC_5996.";
}
enum encr-aes-gcm-8-icv {
  value 18;
  description
    "encr-aes-gcm-8-icv --> RFC_5996.";
}
enum encr-aes-gcm-12-icv {
  value 19;
  description
    "encr-aes-gcm-12-icv --> RFC_5996.";
}
enum encr-aes-gcm-16-icv {
  value 20;
  description
    "encr-aes-gcm-16-icv--> RFC_5996.";
}
enum encr-null-auth-aes-gmac {
  value 21;
  description
    "encr-null-auth-aes-gmac --> RFC_5996.";
}
enum encr-ieee-p1619-xts-aes {
  value 22;
  description
    "encr-ieee-p1619-xts-aes --> Reserved for "+
    "IEEE P1619 XTS-AES.";
}
enum encr-camellia-cbc {
  value 23;
  description
    "encr-camellia-cbc --> RFC_5996.";
}
enum encr-camellia-ctr {
  value 24;
  description
    "encr-camellia-ctr --> RFC_5996.";
}
enum encr-camellia-ccm-8-icv {
  value 25;
  description
    "encr-camellia-ccm-8-icv --> RFC_5996.";
}
enum encr-camellia-ccm-12-icv {
  value 26;
  description
```

```
            "encr-camellia-ccm-12-icv --> RFC_5996.";
          }
          enum encr-camellia-ccm-16-icv {
            value 27;
            description
              "encr-camellia-ccm-16-icv --> RFC_5996.";
          }
          enum encr-aes-cbc-128 {
            value 1024;
            description
              "encr-aes-cbc-128 --> RFC_5996.";
          }
          enum encr-aes-cbc-192 {
            value 1025;
            description
              "encr-aes-cbc-192 --> RFC_5996.";
          }
          enum encr-aes-cbc-256 {
            value 1026;
            description
              "encr-aes-cbc-256 --> RFC_5996.";
          }
          enum encr-blowfish-128 {
            value 1027;
            description
              "encr-blowfish-128 --> RFC_5996.";
          }
          enum encr-blowfish-192 {
            value 1028;
            description
              "encr-blowfish-192 --> RFC_5996.";
          }
          enum encr-blowfish-256 {
            value 1029;
            description
              "encr-blowfish-256 --> RFC_5996.";
          }
          enum encr-blowfish-448 {
            value 1030;
            description
              "encr-blowfish-448 --> RFC_5996.";
          }
```

```
        enum encr-camellia-128 {
          value 1031;
          description
            "encr-camellia-128 --> RFC_5996.";
        }
        enum encr-camellia-192 {
          value 1032;
```



```
          description
            "encr-camellia-192 --> RFC_5996.";
        }
        enum encr-camellia-256 {
          value 1033;
          description
            "encr-camellia-256 --> RFC_5996.";
        }
      }
      description
        "Transform Type 1 - Internet Key Exchange (IKE) "+
        "encryption algorithms.";
    }

    /* Transform Type 2 (Pseudo-Random Function PRF) */
    typedef pseudo-random-function-t {
      type enumeration {
        enum prf-reserved-0 {
          value 0;
          description
            "prf-reserved-0 --> RFC_2104.";
        }
        enum prf-hmac-md5 {
          value 1;
          description
            "prf-hmac-md5 --> RFC_2104.";
        }
        enum prf-hmac-sha1 {
          value 2;
          description
            "prf-hmac-sha1 --> RFC2104.";
        }
        enum prf-hmac-tiger {
          value 3;
```

```
          description
            "prf-hmac-tiger --> RFC2104.";
        }
        enum prf-aes128-xcbc {
          value 4;
          description
            "prf-aes128-xcbc --> RFC_4434.";
        }
        enum prf-hmac-sha2-256 {
          value 5;
          description
            "prf-hmac-sha2-256 --> RFC_4434.";
        }
        enum prf-hmac-sha2-384 {
          value 6;
```



```
          description
            "prf-hmac-sha2-384 --> RFC_4434.";
        }
        enum prf-hmac-sha2-512 {
          value 7;
          description
            "prf-hmac-sha2-512 --> RFC_4434.";
        }
        enum prf-aes128-cmac {
          value 8;
          description
            "prf-aes128-cmac --> RFC_4615.";
        }
      }
      description
        "Available Pseudo-Random Functions (PRF).";
    }

     /* Transform Type 3 (Integrity Algorithm) */
    typedef ike-integrity-algorithm-t {
      type enumeration {
        enum auth-none {
          value 0;
          description
            "auth-none --> RFC_5996.";
        }
```

```
          enum auth-hmac-md5-96 {
            value 1;
            description
              "auth-hmac-md5-96 --> RFC_5996.";
          }
          enum auth-hmac-sha1-96 {
            value 2;
            description
              "auth-hmac-sha1-96 --> RFC_5996.";
          }
          enum auth-des-mac {
            value 3;
            description
              "auth-des-mac --> RFC_5996.";
          }
          enum auth-kpdk-md5 {
            value 4;
            description
              "auth-kpdk-md5 --> RFC_5996.";
          }
          enum auth-aes-xcbc-96 {
            value 5;
            description
```



```
              "auth-aes-xcbc-96 --> RFC_5996.";
          }
          enum auth-hmac-md5-128 {
            value 6;
            description
              "auth-hmac-md5-128 --> RFC_5996.";
          }
          enum auth-hmac-sha1-160 {
            value 7;
            description
              "auth-hmac-sha1-160 --> RFC_5996.";
          }
          enum auth-aes-cmac-96 {
            value 8;
            description
              "auth-aes-cmac-96 --> RFC_5996.";
          }
          enum auth-aes-128-gmac {
```

```
          value 9;
          description
            "auth-aes-128-gmac --> RFC_5996.";
        }
        enum auth-aes-192-gmac {
          value 10;
          description
            "auth-aes-192-gmac --> RFC_5996.";
        }
        enum auth-aes-256-gmac {
          value 11;
          description
            "auth-aes-256-gmac --> RFC_5996.";
        }
        enum auth-hmac-sha2-256-128 {
          value 12;
          description
            "auth-hmac-sha2-256-128 --> RFC_5996.";
        }
        enum auth-hmac-sha2-384-192 {
          value 13;
          description
            "auth-hmac-sha2-384-192 --> RFC_5996.";
        }
        enum auth-hmac-sha2-512-256 {
          value 14;
          description
            "auth-hmac-sha2-512-256 --> RFC_5996.";
        }
        enum auth-hmac-sha2-256-96 {
          value 1024;
```



```
          description
            "auth-hmac-sha2-256-96.";
        }
      }
      description
        "Transform Type 3 - Internet Key Exchange (IKE) "+
        "Integrity Algorithms.";
    }

    /* Transform Type 4 (Diffie-Hellman Group) */
```

```
typedef diffie-hellman-group-t {
  type enumeration {
    enum group-none {
      value 0;
      description
        "group-none --> RFC_5996.";
    }
    enum modp-768-group-1 {
      value 1;
      description
        "modp-768-group-1 --> RFC_5996.";
    }
    enum modp-1024-group-2 {
      value 2;
      description
        "modp-1024-group-2 --> RFC_5996.";
    }
    enum modp-1536-group-5 {
      value 5;
      description
        "modp-1536-group-5 --> RFC_3526.";
    }
    enum modp-2048-group-14 {
      value 14;
      description
        "modp-2048-group-14 --> RFC_3526.";
    }
    enum modp-3072-group-15 {
      value 15;
      description
        "modp-3072-group-15 --> RFC_3526.";
    }
    enum modp-4096-group-16 {
      value 16;
      description
        "modp-4096-group-16 --> RFC_3526.";
    }
    enum modp-6144-group-17 {
      value 17;
```



```
      description
        "modp-6144-group-17 --> RFC_3526.";
```

```
        }
        enum modp-8192-group-18 {
          value 18;
          description
            "modp-8192-group-18 --> RFC_3526.";
        }
        enum recp-256-group-19 {
          value 19;
          description
            "recp-256-group-19 --> RFC_6989. 256-bit"+
            " Random ECP Group.";
        }
        enum recp-384-group-20 {
          value 20;
          description
            "recp-384-group-20 --> RFC_6989. 384-bit"+
            " Random ECP Group.";
        }
        enum recp-521-group-21 {
          value 21;
          description
            "recp-521-group-21 --> RFC_6989. 521-bit"+
            " Random ECP Group.";
        }
        enum modp-1024-160-pos-group-22 {
          value 22;
          description
            "modp-1024-160-pos-group-22 --> RFC_6989."+
            " 1024-bit MODP Group with"+
            " 160-bit Prime Order Subgroup (POS).";
        }
        enum modp-2048-224-pos-group-23 {
          value 23;
          description
            "modp-2048-224-pos-group-23 --> RFC_6989."+
            " 2048-bit MODP Group with"+
            " 224-bit Prime Order Subgroup (POS).";
        }
        enum modp-2048-256-pos-group-24 {
          value 24;
          description
            "modp-2048-256-pos-group-24 --> RFC_6989."+
            " 2048-bit MODP Group with"+
            " 256-bit Prime Order Subgroup (POS).";
        }
        enum recp-192-group-25 {
          value 25;
```

```
          description
            "recp-192-group-25 --> RFC_6989."+
            " 192-bit Random ECP Group.";
        }
        enum recp-224-group-26 {
          value 26;
          description
            "recp-224-group-26 --> RFC_6989."+
            " 224-bit Random ECP Group.";
        }
      }
      description
        "Diffie-Hellman Groups (RFC 5996).";
    }


    /* Transform Type 5 (Extended Sequence Numbers
       Transform ESN IDs) */
    typedef extended-sequence-number-t {
      type enumeration {
        enum esn-none {
          value 0;
          description
            "esn-none - Extended Sequence Number None --> RFC_7296.";
        }
        enum esn-1 {
          value 1;
          description
            "esn-1 - Extended Sequence Number --> RFC_7296.";
        }
      }
      description
        "Extended Sequence Number (RFC 7296).";
    }


    typedef connection-type-t {
      type enumeration {
        enum initiator-only {
          value 0;
          description
            "initiator-only: ME will act as initiator for"+
            " bringing up IKEv2"+
            " session with its IKE peer.";
        }
        enum responder-only {
```

```
            value 1;
            description
              "responder-only: ME will act as responder for"+
```



```
              " bringing up IKEv2"+
              " session with its IKE peer.";
          }
          enum both {
            value 2;
            description
              "both: ME can act as initiator or responder.";
          }
        }
        description
          "Connection type for IKE session.";
      }

      typedef transport-protocol-name-t {
        type enumeration {
          enum tcp {
            value 1;
            description
              "Transmission Control Protocol (TCP) Transport Protocol.";
          }
          enum udp {
            value 2;
            description
              "User Datagram Protocol (UDP) Transport Protocol";
          }
          enum sctp {
            value 3;
            description
              "Stream Control Transmission Protocol (SCTP) Transport "+
              "Protocol";
          }
          enum icmp {
            value 4;
            description
              "Internet Control Message Protocol (ICMP) Transport "+
              "Protocol";
          }
        }
```

```
    description
      "Enumeration of well known transport protocols.";
  }

  typedef preshared-key-t {
    type string;
    description
      "Derived string used as Pre-Shared Key.";
  }

  typedef pad-type-t {
```



```
    type enumeration {
      enum dns-name {
        value 1;
        description
          "DNS name (specific or partial)";
      }
      enum distinguished-name {
        value 2;
        description
          "Distinguished Name (complete or sub-tree constrained)";
      }
      enum rfc-822 {
        value 3;
        description
          "RFC 822 email address (complete or partially qualified)";
      }
      enum ipv4-range {
        value 4;
        description
          "IPv4 Address Range";
      }
      enum ipv6-range {
        value 5;
        description
          "IPv6 Address Range";
      }
      enum key-id {
        value 6;
        description
          "Key ID (exact match only)";
```

```
      }
    }
    description
      "PAD Type";
  }


  /*-------------------------------------------------- */
  /* draft-wang-ipsecme-ipsec-yang-00: ietf-ipsec-type */
  /*-------------------------------------------------- */
  typedef ipsec-mode {
    type enumeration {
      enum "transport" {
        description
          "Transport mode";
      }
      enum "tunnel" {
        description
          "Tunnel mode";
```



```
      }
    }
    description
      "type define of ipsec mode";
  }

  typedef ipsec-protocol {
    type enumeration {
      enum "ah" {
        description
          "AH Protocol";
      }
      enum "esp" {
        description
          "ESP Protocol";
      }
    }
    description
      "type define of ipsec security protocol";
  }

  typedef ipsec-spi {
```

```
        type uint32 {
          range "1..max";
        }
        description
          "SPI";
      }

      typedef ipsec-spd-name {
       type enumeration {
        enum id_rfc_822_addr {
            description
              "Fully qualified user name string.";
        }
        enum id_fqdn {
          description
            "Fully qualified DNS name.";
        }
        enum id_der_asn1_dn {
          description
            "X.500 distinguished name.";
        }
        enum id_key {
          description
            "IKEv2 Key ID.";
        }
      }
      description
```



```
        "IPsec SPD name type";
      }

      typedef ipsec-traffic-direction {
        type enumeration {
          enum inbound {
            description
              "Inbound traffic";
          }
          enum outbound {
            description
              "Outbound traffic";
          }
        }
```

```
    description
      "IPsec traffic direction";
  }

  typedef ipsec-spd-operation {
    type enumeration {
      enum protect {
        description
          "PROTECT the traffic with IPsec";
      }
      enum bypass {
        description
          "BYPASS the traffic";
      }
      enum discard {
        description
          "DISCARD the traffic";
      }
    }
    description
      "The operation when traffic matches IPsec security policy";
  }


  /*-------------------------------------------------- */
  /* draft-wang-ipsecme-ipsec-yang-00: ietf-ipsec-crypto */
  /*-------------------------------------------------- */
  typedef ipsec-authentication-algorithm {
    type enumeration {
      enum "null" {
        value 0;
        description
          "null";
      }
      enum "md5" {
```



```
        value 1;
        description
          "MD5 authentication algorithm";
      }
      enum "sha1" {
        value 2;
```

```
        description
          "SHA1 authentication algorithm";
      }
      enum "sha2-256" {
        value 3;
        description
          "SHA2-256 authentication algorithm";
      }
      enum "sha2-384" {
        value 4;
        description
          "SHA2-384 authentication algorithm";
      }
      enum "sha2-512" {
        value 5;
        description
          "SHA2-512 authentication algorithm";
      }
    }
    description
      "typedef for ipsec authentication algorithm";
  }

  typedef ipsec-encryption-algorithm {
    type enumeration {
      enum "null" {
        description
          "null";
      }
      enum "des" {
        description
          "DES encryption algorithm";
      }
      enum "3des" {
        description
          "3DES encryption algorithm";
      }
      enum "aes-128" {
        description
          "AES-128 encryption algorithm";
      }
      enum "aes-192" {
        description
```

```
        "AES-192 encryption algorithm";
    }
    enum "aes-256" {
      description
        "AES-256 encryption algorithm";
    }
  }
  description
   "typedef for ipsec encryption algorithm";
}

/*-------------------------------------------------- */
/* draft-wang-ipsecme-ike-yang-00: ietf-ipsec-type */
/*-------------------------------------------------- */
typedef ike-integrity-algorithm {
  type enumeration {
    enum "hmac-md5-96" {
      description
        "HMAC-MD5-96 Integrity Algorithm";
    }
    enum "hmac-sha1-96" {
      description
        "HMAC-SHA1-96 Integrity Algorithm";
    }
    enum "hmac-sha2-256" {
      description
        "HMAC-SHA2-256 Integrity Algorithm";
    }
    enum "hmac-sha2-384" {
      description
        "HMAC-SHA2-384 Integrity Algorithm";
    }
    enum "hmac-sha2-512" {
      description
        "HMAC-SHA2-512 Integrity Algorithm";
    }
  }
  description
    "typedef for ike integrity algorithm.";
}

typedef ike-encryption-algorithm {
  type enumeration {
    enum "des-cbc" {
      description
        "DES-CBC Encryption algorithm";
    }
    enum "3des-cbc" {
      description
```

```
            "3DES-CBC Encryption algorithm";
        }
        enum "aes-cbc-128" {
          description
            "AES-CBC-128 Encryption algorithm";
        }
        enum "aes-cbc-192" {
          description
            "AES-CBC-192 Encryption algorithm";
        }
        enum "aes-cbc-256" {
          description
            "AES-CBC-256 Encryption algorithm";
        }
      }
      description
        "typedef for ike encryption algorithm.";
    }

    typedef ike-prf-algorithm {
      type enumeration {
        enum "hmac-md5-96" {
          description
            "HMAC-MD5-96 PRF Algorithm";
        }
        enum "hmac-sha1-96" {
          description
            "HMAC-SHA1-96 PRF Algorithm";
        }
        enum "hmac-sha2-256" {
          description
            "HMAC-SHA2-256 PRF Algorithm";
        }
        enum "hmac-sha2-384" {
          description
            "HMAC-SHA2-384 PRF Algorithm";
        }
        enum "hmac-sha2-512" {
          description
            "HMAC-SHA2-512 PRF Algorithm";
        }
```

```
    }
    description
      "typedef for ike prf algorithm.";
  }

  typedef ike-dh-group {
    type enumeration {
      enum "dh-group-none" {
```



```
        description
          "None Diffie-Hellman group";
      }
      enum "dh-group-1" {
        description
          "768 bits Diffie-Hellman group";
      }
      enum "dh-group-2" {
        description
          "1024 bits Diffie-Hellman group";
      }
      enum "dh-group-5" {
        description
          "1536 bits Diffie-Hellman group";
      }
      enum "dh-group-14" {
        description
          "2048 bits Diffie-Hellman group";
      }
    }
    description
      "typedef for ike dh group";
  }


  typedef ike-peer-name-ref {
    type leafref {
      path "/ikev2/ike-peer/ike-peer-entries/peer-name";
    }
    description "reference to ike peer name";
  }

  typedef ike-proposal-number-ref {
```

```
      type leafref {
        path "/ikev2/proposal/name";
      }
      description "reference to ike proposal name";
    }

    typedef ipsec-proposal-name-ref{
      type leafref {
        path "/ipsec/proposal/ipsec-proposal/name";
      }
      description "reference to ike proposal name";
    }

    typedef ike-auth-method {
      type enumeration {
        enum pre-share {
```



```
          description
            "Select pre-shared key message as the
             authentication method";
        }
        enum rsa-digital-signature {
          description
            "Select rsa digital signature as the
             authentication method";
        }
        enum dss-digital-signature {
          description
            "Select dss digital signature as the
             authentication method";
        }
      }
      description "IKE authentication methods";
    }

    /*--------------------*/
    /*   grouping         */
    /*--------------------*/

     /* The following groupings are used in both configuration data
        and operational state data */
    grouping name-grouping {
```

```
    description
      "This grouping provides a leaf identifying the name.";
    leaf name {
      type string;
      description
        "Name of a identifying.";
    }
    leaf description {
      type string;
      description
        "Specify the description.";
    }
  }

  grouping sequence-number-grouping {
    description
      "This grouping provides a leaf identifying
       a sequence number.";
    leaf sequence-number {
      type uint32 {
        range "1..4294967295";
      }
      description
        "Specify the sequence number.";
```



```
    }
  }

  grouping description-grouping {
    description
      "description for free use.";
    leaf description {
      type string;
      description
        "description for free use.";
    }
  }

  grouping traffic-selector-grouping {
    description
      "Traffic selector to be used for SA negotiation.";
    leaf traffic-selector-id {
```

```
        type string;
        mandatory true;
        description
          "Traffic selector identifier.";
      }
      leaf protocol-name {
        type transport-protocol-name-t;
        description
          "Specifies the protocol selector.";
      }
      leaf address-range {
        type string;
        mandatory true;
        description
          "Specifies the IPv4 or IPv6 address range.";
      }
    }



    grouping ike-general-proposal-grouping {
      description
        "IKE proposal.";
      leaf name {
         type string;
         mandatory true;
         description
           "IKE Proposal identify.";
      }
      leaf description {
        type string;
        description
```



```
          "Specify the description.";
      }

      leaf dh-group {
        type diffie-hellman-group-t;
        mandatory true;
        description
          "Specifies a Diffie-Hellman group.";
      }
```

```
      container encryption {
        description
          "Specify IKE Proposal encryption configuration";
        leaf algorithm {
           type ike-encryption-algorithm-t;
           description
             "Specifies an Encryption Algorithm.";
        }
      }
    }

    grouping ike-proposal-grouping {
      description
        "Configure the IKE Proposal";
      uses ike-general-proposal-grouping;

      leaf lifetime {
        type uint32;
         mandatory true;
        description
          "Configure lifetime for IKE SAs
           0: for no timeout.
           300 .. 99999999:  IKE SA lifetime in seconds.";
      }
      container authentication {
        description
          "Specify IKE Proposal authentication configuration";
        leaf algorithm {
          type ike-integrity-algorithm-t;
          description
            "Specify the authentication algorithm";
        }
        leaf preshared-key {
          type empty;
          description
            "Use pre-shared key based authentication";
        }
        leaf rsa-signature {
          type empty;
          description
```



```
            "Use signature based authentication by using
```

```
          PKI certificates";
      }
    }
  }

  grouping ikev2-proposal-grouping {
    description
      "Holds an IKEv2 transform proposal used during "+
      "IKEv2 SA negotiation.  Multiple IKEv2 Transforms "+
      " can be proposed during an IKEv2 session initiation "+
      "in an ordered list.";
    uses ike-general-proposal-grouping;

    leaf pseudo-random-function {
      type pseudo-random-function-t;
      mandatory true;
      description
        "Specifies Pseudo Random Function for IKEv2 key exchange";
    }
    container authentication {
      description
        "Specify IKEv2 Proposal authentication configuration";
      leaf algorithm {
        type ike-integrity-algorithm-t;
        description
          "Specify the authentication algorithm";
      }
    }
  }

  grouping ipsec-proposal-grouping {
    description
      "Configure IPSec Proposal";
    leaf name {
       type string;
       mandatory true;
       description
         "IPSec proposal identifier.";
    }
    leaf ah {
      type ike-integrity-algorithm-t;
      description
        "Configure Authentication Header (AH).";
    }
    container esp {
      description
        "Configure Encapsulating Security Payload (ESP).";
      leaf authentication {
```

```
            type ike-integrity-algorithm-t;
            description
              "Configure ESP authentication";
          }
          leaf encryption {
            type ike-encryption-algorithm-t;
            description
              "Configure ESP encryption";
          }
        }
        leaf ip-comp{
          type empty;
          description
            "Enable IPSec proposal IP-COMP which uses the IP Payload "+
            "compression protocol to compress IP Security (IPSec) "+
            "packets before encryption";
        }
        container lifetime {
          description
            "Configure lifetime for IPSEC SAs";
          leaf kbytes {
            type uint32 {
              range "128..2147483647";
            }
            description
              "Enter lifetime kbytes for IPSEC SAs";
          }
          leaf seconds {
            type uint32 {
              range "300..99999999";
            }
            description
              "Enter lifetime seconds for IPSEC SAs
              0: lifetime of 0 for no timeout
              300..99999999: IPSec SA lifetime in seconds";
          }
        }
      }

      grouping identity-grouping {
        description
          "Identification type. It is an union identity, "+
          "possible type as follows: "+
```

```
        "a) ID_FQDN: A fully-qualified domain name string. "+
        "  An example of a ID_FQDN is, example.com. "+
        "  The string MUST not contain any terminators "+
        "(e.g., NULL, CR, etc.). "+
        "b) ID_RFC822_ADDR: A fully-qualified RFC822 email "+
        "   address string, An example of a ID_RFC822_ADDR is, "+
```



```
        "   jsmith@example.com. The string MUST not contain "+
        "   any terminators. "+
        "c) ID_IPV4_ADDR: A single four (4) octet IPv4 address. "+
        "d) ID_IPV6_ADDR: A single sixteen (16) octet IPv6 address. "+
        "e) DN_X509: Distinguished name in the X.509 tradition.";
      choice identity {
        description
          "Choice of identity.";
        leaf ipv4-address {
          type inet:ipv4-address;
          description
            "Specifies the identity as a single four (4)
             octet IPv4 address.
             An example is, 10.10.10.10. ";
        }
        leaf ipv6-address {
          type inet:ipv6-address;
          description
            "Specifies the identity as a single sixteen (16) "+
            "octet IPv6 address. "+
            "An example is, "+
            "FF01::101, 2001:DB8:0:0:8:800:200C:417A .";
        }
        leaf fqdn-string {
          type inet:domain-name;
          description
            "Specifies the identity as a Fully-Qualified
             Domain Name (FQDN) string.
             An example is: example.com.
             The string MUST not contain any terminators
             (e.g., NULL, CR, etc.).";
        }
        leaf rfc822-address-string {
          type string;
          description
```

```
            "Specifies the identity as a fully-qualified RFC822
             email address string.
             An example is, jsmith@example.com.
             The string MUST not contain any terminators
             (e.g., NULL, CR, etc.).";
        }
        leaf dnX509 {
          type string;
          description
            "Specifies the identity as a distinguished name
             in the X.509 tradition.";
        }
      }
    } /* grouping identity-grouping */
```



```
    grouping ike-general-policy-profile-grouping {
      description
        "IKE policy.";
      leaf connection-type {
        type connection-type-t;
        mandatory true;
        description
          "Specify the IKE connection type";
      }
      leaf pre-shared-key {
        type union {
          type string {
            length "16";
          }
          type yang:hex-string {
            length "40";
          }
        }
        description
          "Specify IKE pre-shared-key value";
      }
      leaf validate-certificate-identity {
        type empty;
        description
          "Validate Remote-ID payload against the
          ID's available in the certificate";
```

```
      }
      list seq {
        key seq-id;
        description
          "list of sequence of policy.";
        leaf seq-id {
          type uint32 {
            range "1..429496729";
          }
          description
            "Sequence Number";
        }
        leaf proposal {
          type leafref {
            path "/eipsec:ikev1/eipsec:proposal"+
                 "/eipsec:name";
          }
          description
            "IKE Proposal reference.";
        }
      }
      container identity {
```



```
        description
          "Specify IKE identity value";
        container local {
          description
            "Specify the identity of the local IP Security (IPSec)
            tunnel endpoint in an Internet Key Exchange (IKE)
            policy to use when negotiating IKE request with a
            remote peer.";
          uses identity-grouping;
        }
        container remote {
          description
            "Specify the identity of the remote IP Security (IPSec)
            tunnel endpoint in an
             Internet Key Exchange (IKE) policy to use when
             negotiating IKE request with a remote peer.";
          uses identity-grouping;
        }
      }
```

```
    }

    grouping ike-policy-mode-grouping {
      description
        "IKE Policy Mode";
      container mode {
        description
          "Specify IKE mode configuration";
        leaf aggressive {
          type empty;
          description
            "Set IKE Aggressive mode";
        }
        leaf main {
          type empty;
          description
            "Set IKE Main mode";
        }
      }
    }

    grouping ike-policy-profile-grouping {
      description
        "Configure IKE policy";
      leaf name {
        type string;
        mandatory true;
        description
          "Specify an IKE policy name";
      }
```



```
      uses ike-policy-mode-grouping;
      uses ike-general-policy-profile-grouping;
    }

    grouping ikev2-policy-profile-grouping {
      description
        "Common information for multiple IKE sessions
        to be instantiated on a managed element.;
         One or more Ikev2Session instances might refer
         to this instance.";
      leaf name {
```

```
          type string;
          mandatory true;
          description
            "Value component of the RDN.";
        }
        container authentication {
          description
            "Specify IKE Proposal authentication configuration";
          leaf preshared-key {
            type empty;
            description
              "Use pre-shared key based authentication";
          }
          leaf rsa-signature {
            type empty;
            description
              "Use signature based authentication by using
              PKI certificates";
          }
        }
        leaf lifetime {
          type uint32;
          mandatory true;
          description
            "Configure lifetime for IKE SAs
             0: for no timeout.
             300 .. 99999999:  IKE SA lifetime in seconds.";
        }

        container address-allocation {
          must "../connection-type = 'responder-only'" {
            description
              "address-allocation can be configured only with
              responder-only in ike2 policy";
          }
          leaf aaa {
            type empty;
            description
```



```
              "IRAC address allocation by AAA";
          }
          description
```

```
          "Specify IKE IRAS address allocation option";
      }
      uses ike-general-policy-profile-grouping;

      leaf description {
        type string;
        description
          "Specify the description.";
      }
    }

    grouping ipsec-policy-grouping {
      description
        "Holds configuration information for IPSec policies.";
      leaf name {
        type string;
        mandatory true;
        description
          "IPSec Policy Identification";
      }
      leaf description {
        type string;
        description
          "Specify the description.";
      }

      leaf anti-replay-window {
        type uint32 {
          range "0 | 32..1024";
        }
        description
          "Configure replay window size
          0: to disable anti-replay-window
          32..1024: IPSec anti-replay-window size in multiple of 32";
      }
      container perfect-forward-secrecy {
        description
          "Configure Perfect Forward Secrecy (PFS) for IPSec Policy";
        leaf dh-group {
          type diffie-hellman-group-t;
          description
            "Configure Diffie-Hellman group for
             perfect-forward-secrecy";
        }
      }
      list seq {
```

```
          key seq-id;
          description
            "Specify IPSEC proposal sequence number";
          leaf seq-id {
            type uint32;
            description
              "Sequence ID";
          }
          leaf description {
            type string;
            description
              "Specify the description.";
          }

          leaf proposal {
            type leafref {
              path "/eipsec:ipsec/"+
                   "eipsec:proposal/eipsec:ipsec-proposal/eipsec:name";
            }
            description
              "IKE proposal reference.";
          }
        }
      }

      grouping key-string-grouping {
        description
          "Configure key for authentication algorithm";
        leaf key-str {
          type union {
            type string {
              length "16";
            }
            type yang:hex-string {
              length "40";
            }
          }
          description
            "Key string input is either string value (length of 16)
            or hexadecimal (length of 40)";
        }
      }

      grouping ipsec-sa-ah-grouping {
        description
          "Configure Authentication Header (AH) for
           Security Association (SA)";
```

```
        container ah {
          description
```



```
            "Configure Authentication Header (AH) for SA";
          choice authentication-algorithm {
            description
              "choice for authentication algorithm to set for AH";
            case hmac-aes-xcbc {
              container hmac-aes-xcbc {
                description
                  "Set the authentication algorithm to hmac-aes-xcbc";
                uses key-string-grouping;
              }
            }
            case hmac-md5-96 {
              container hmac-md5-96 {
                description
                  "Set the authentication algorithm to hmac-md5-96";
                uses key-string-grouping;
              }
            }
            case hmac-sha1-96 {
              container hmac-sha1-96 {
                description
                  "Set the authentication algorithm to hmac-sha1-96";
                uses key-string-grouping;
              }
            }
            case key-string {
              container key-string {
                description
                  "Configure key for authentication algorithm";
                uses key-string-grouping;
              }
            }
          }
        }
      }

      grouping ipsec-sa-esp-grouping {
        description
          "Configure IPSec Encapsulation Security Payload (ESP)";
```

```
        container esp {
          description
            "Set IPSec Encapsulation Security Payloer (ESP)";
          container authentication {
            description
              "Configure authentication for IPSec
               Encapsulation Secutiry Payload (ESP)";
            choice authentication-algorithm {
              description
                "choice for authentication algorithm to set";
```



```
              case hmac-aes-xcbc {
                container hmac-aes-xcbc {
                  description
                    "Set the authentication algorithm to hmac-aes-xcbc";
                  uses key-string-grouping;
                }
              }
              case hmac-md5-96 {
                container hmac-md5-96 {
                  description
                    "Set the authentication algorithm to hmac-md5-96";
                  uses key-string-grouping;
                }
              }
              case hmac-sha1-96 {
                container hmac-sha1-96 {
                  description
                    "Set the authentication algorithm to hmac-sha1-96";
                  uses key-string-grouping;
                }
              }
              case key-string {
                container key-string {
                  description
                    "Configure key for authentication algorithm";
                  uses key-string-grouping;
                }
              }
            }
          }
          container encryption {
```

```
          description
            "Configure encryption for IPSec
             Encapsulation Secutiry Payload (ESP)";
          choice encryption-algorithm {
            description
              "type of encryption";
            case des3-cbc {
              container des3-cbd {
                description
                  "Set the encryption algorithm to des3-cbc";
                uses key-string-grouping;
              }
            }
            case aes-128-cbc {
              container aes-128-cbc {
                description
                  "Set the encryption algorithm to aes-128-cbc";
                uses key-string-grouping;
```



```
              }
            }
            case aes-192-cbc {
              container aes-192-cbc {
                description
                  "Set the encryption algorithm to aes-192-cbc";
                uses key-string-grouping;
              }
            }
            case aes-256-cbc {
              container aes-256-cbc {
                description
                  "Set the encryption algorithm to aes-256-cbc";
                uses key-string-grouping;
              }
            }
            case des-cbc {
              container des-cbc {
                description
                  "Set the encryption algorithm to des-cbc";
                uses key-string-grouping;
              }
            }
```

```
            case key-string {
              container key-string {
                description
                  "Configure key for encryption algorithm";
                uses key-string-grouping;
              }
            }
          }
        }
      }

      grouping ipsec-acl-dest-grouping {
        description
          "IPSEC ACL destination.";
        /* For destination */
        choice dest-address {
          description
            "destination address.";
          case dest-ipv4-address {
            leaf destination-ipv4-address {
              type inet:ipv4-address;
              description
                "Destination IPv4 Address A.B.C.D/0..32.";
            }
          }
```



```
          case dest-any {
            leaf dest-any {
              type empty;
              description
                "Match Any Destination IPv4 Address.";
            }
          }
        }
      }

      grouping ipsec-acl-seq-protocol-number-grouping {
        description
          "IPSec ACL Sequence protocol number.";
        leaf number {
          type uint16 {
```

```
          range "0..255";
        }
        description
          "Specify protocol number.";
      }
      choice argument {
        description
          "Source IPv4 address.";
        case source-ipv4-address {
          leaf source-ipv4-address {
            type inet:ipv4-address;
            description
              "Source IPv4 Address A.B.C.D/0..32.";
          }
        }
        case any {
          /* For source */
          leaf source-any {
            type empty;
            description
              "Match Any Source IPv4 Address.";
          }
        }
      }
    }

    grouping ipsec-acl-seq-ip-address-grouping {
      description
        "IPSec ACL Sequence IP Address.";
      leaf source-ipv4-address {
        type inet:ipv4-address;
        description
          "Source is IPv4 Address A.B.C.D/0..32.";
      }
```



```
    }

    grouping ipsec-acl-seq-any-grouping {
      description
        "IPSec ACL Sequence Any.";
      leaf any {
        type empty;
```

```
      description
        "Source is Any.";
    }
  }

  grouping ipsec-acl-seq-tcp-grouping {
    description
      "IPSec ACL Sequence TCP.";
    leaf tcp {
      type empty;
      description
        "Source is TCP protocol.";
    }
  }

  grouping ipsec-acl-seq-udp-grouping {
    description
      "IPSec ACL Sequence for UDP.";
    leaf udp {
      type empty;
      description
        "Source is UDP protocol.";
    }
  }

  grouping ipsec-acl-grouping {
    description
      "IPSec ACL";
    list access-list {
      if-feature ipsec-acl;
      key "name sequence-number";
      uses name-grouping;
      uses sequence-number-grouping;
      description
        "Configure the IPSec access-list.";
      choice protocol {
        description
          "IPSec ACL protocol.";
        case number {
          uses ipsec-acl-seq-protocol-number-grouping;
        }
        case source-ipv4-address {
```

```
        uses ipsec-acl-seq-ip-address-grouping;
      }
      case any {
        uses ipsec-acl-seq-any-grouping;
      }
      case tcp {
        uses ipsec-acl-seq-tcp-grouping;
      }
      case udp {
        uses ipsec-acl-seq-udp-grouping;
      }
    }
    uses ipsec-acl-dest-grouping;
  }
}

grouping ipsec-df-bit-grouping {
  description
    "IPSec Dont Fragment (DF) bit for IP header.";
  container df-bit {
    description
      "Configure Don't Fragment (DF) bit for IP Header.";
    leaf clear {
      type empty;
      description
        "Clear DF bit for outer IP header.";
    }
    leaf propagate {
      type empty;
      description
        "Propagate DF bit for outer IP header.";
    }
    leaf set {
      type empty;
      description
        "Set DF bit for outer IP header.";
    }
  }
}

grouping ipsec-profile-grouping {
  description
    "IPSec profile.";
  list profile {
    key "name";
    uses name-grouping;
    uses ipsec-df-bit-grouping;
    description
      "Configure the IPSec Profile.";
```

```
        leaf mtu {
          type uint32 {
            range "256..1600";
          }
          description
            "Set the MTU.";
        }
        list seq {
          key "sequence-number";
          uses sequence-number-grouping;
          description
            "IPSec Access List sequence number.";
          leaf policy {
            type leafref {
              path "/eipsec:ipsec/eipsec:policy"+
                   "/eipsec:ipsec-policy/eipsec:name";
            }
            description
              "Specify IPSec policy name.";
          }
        }
      }
    }


    grouping ip-address-grouping {
      description
        "IP Address grouping";

      choice ip-address {
        description
          "Choice of IPv4 or IPv6.";
        leaf ipv4-address {
          type inet:ipv4-address;
          description
            "Specifies the identity as a single four (4)
             octet IPv4 address.
             An example is, 10.10.10.10. ";
        }
        leaf ipv6-address {
          type inet:ipv6-address;
          description
```

```
            "Specifies the identity as a single sixteen (16) "+
            "octet IPv6 address. "+
            "An example is, "+
            "FF01::101, 2001:DB8:0:0:8:800:200C:417A .";
        }
      }
    }
```



```
    grouping ipsec-sa-grouping {
      description
        "Configure Security Association (SA)";
      leaf spi {
        type uint32;
        description
          "Specify Security Parameter Index";
      }
      leaf anti-replay-window {
        type uint16 {
          range "0 | 32..1024";
        }
        description
          "Specify replay window size";
      }
      leaf ip-comp {
        type empty;
        description
          "Enables IPCOMP, which uses the IP payload compression
           protocol to compress IP security (IPsec) packets
           before encryption";
      }

      container local-peer {
        description
          "Specify the local peer IP address";
        uses ip-address-grouping;
      }
      container remote-peer {
        description
          "Specify the remote peer IP address";
        uses ip-address-grouping;
      }
```

```
        leaf sa-mode {
          type ipsec-mode;
          description
            "SA Mode: tunnel or transport mode";
        }
        leaf security-protocol {
          type ipsec-protocol;
          description
            "Security protocol of IPsec SA: Either AH or ESP.";
        }
        leaf sequence-number {
          type uint64;
          description
            "Current sequence number of IPsec packet.";
        }
```



```
        leaf sequence-number-overflow-flag {
          type boolean;
          description
            "The flag indicating whether overflow of the sequence
             number counter should prevent transmission of additional
             packets on the SA, or whether rollover is permitted.";
        }
        leaf path-mtu {
          type uint16;
          description
            "maximum size of an IPsec packet that can be transmitted
             without fragmentation";
        }
        container life-time {
          leaf life-time-in-seconds {
            type uint32;
            description
              "SA life time in seconds";
          }
          leaf remain-life-time-in-seconds {
            type uint32;
            description
              "Remain SA life time in seconds";
          }
          leaf life-time-in-byte {
            type uint32;
```

```
        description
          "SA life time in bytes";
      }
      leaf remain-life-time-in-byte {
        type uint32;
        description
          "Remain SA life time in bytes";
      }
      description
        "SA life time information";
    }
    leaf upper-protocol {
      type string;
      description
        "Upper-layer protocol to be used";
    }
    leaf direction {
      type ipsec-traffic-direction;
      description
        "It indicates whether the SA is inbound SA or
         out bound SA.";
    }
    container source-address {
```



```
      description
        "Specify the source IP address and
         port of protected traffic";
      uses ip-address-grouping;
      leaf port-number {
        type uint32;
        description
          "port of protected traffic";
      }
    }
    container destination-address {
      description
        "Specify the destination IP address and
         port of protected traffic";
      uses ip-address-grouping;
      leaf port-number {
        type uint32;
        description
```

```
            "port of protected traffic";
        }
      }
      leaf nat-traversal-flag {
        type boolean;
        description
          "Whether the SA is used to protect traffic that needs
           nat traversal";
      }
      uses ipsec-sa-ah-grouping;
      uses ipsec-sa-esp-grouping;
    }


    /* draft-wang-ipsecme-ike-yang-00 */
    grouping ipsec-common-configuration {
      choice df-flag {
        default copy;
        case set {
          leaf set {
            type empty;
            description
              "Set the df bit when encapsulate IPsec tunnel.";
          }
        }
        case clear {
          leaf clear {
            type empty;
            description
              "Clear the df bit when encapsulate IPsec tunnel.";
          }
```



```
        }
        case copy {
          leaf copy {
            type empty;
            description
              "Copy the inner IP header df bit.";
          }
        }
        description
          "It indicates how to process the df bit when encapsulate
```

```
          IPsec tunnel.";
      }
      leaf stateful-frag-check {
        type boolean;
        default false;
        description "Whether stateful fragment checking applies.";
      }
      leaf life-time-kb {
        type uint32;
        units "KB";
        default 2000000;
        description "IPsec SA Life time in KB.";
      }
      leaf life-time-second {
        type uint32;
        units "Second";
        default 18400;
        description "IPsec SA Life time in Seconds";
      }
      choice anti-replay {
        default enable;
        case enable {
          leaf enable {
            type empty;
            description "Enable Anti-replay";
          }
          choice anti-replay-windows-size {
            case size-32;
            case size-64;
            case size-128;
            case size-256;
            case size-512;
            case size-1024;
            default size-1024;
            description "It indicate the size of anti-replay window";
          }
        }
        case disable {
          leaf disable {
```



```
            type empty;
            description "Disable Anti-replay";
```

```
        }
      }
      description "Whether enable or disable anti-replay";
    }
    leaf inbound-dscp {
      type uint16 {
        range "0..63";
      }
      default 0;
      description "Inbound DSCP value";
    }
    leaf outbound-dscp {
      type uint16 {
        range "0..63";
      }
      default 0;
      description "Outbound DSCP value";
    }
    description "Common IPsec configurations";
  }

  /*--------------------*/
  /* Configuration Data */
  /*--------------------*/
  container ikev1 {
    if-feature ikev1;
    description
      "Configuration IPSec IKEv1";
    /* The following is for <configure> */
    list proposal {
       key "name";
       uses ike-proposal-grouping;
       description
         "Configure IKE proposal";
    }
    leaf keepalive {
      type empty;
      description
        "Enables sending Dead Peer Detection (DPD) messages "+
        "to Internet Key Exchange (IKE) peers.";
    }
    list policy {
      key "name";
      uses ike-policy-profile-grouping;
      description
        "Configure IKE Policy Profile.";
    }
```

```
    }

    container ikev2 {
      if-feature ikev2;
      description
        "Configuration IPSec IKEv2";
      /* The following is for <configure> */
      /* draft-wang-ipsecme-ike-yang-00 */
      container ike-global-configuration {
        if-feature ikev2-global;
        description "Global IKE configurations";
        uses ipsec-common-configuration;
        leaf local-name {
          type string;
          description
            "Global local name configuration, if it is not configed,
             ip address will be used as default. If configing special
             local name for special peer, it will overwrite the global
             name configuration when negotion with that peer.";
        }
        leaf nat-keepalive-interval {
          type uint16 {
            range "5..300";
          }
          units "Seconds";
          default 20;
          description "Global nat keepalive interval";
        }
        leaf dpd-interval {
          type uint16 {
            range "10..3600";
          }
          units "Seconds";
          default 30;
          description "Global DPD interval";
        }
      }
      container ike-peer {
        if-feature ikev2-peer;
        description "IKE peer information";
        list ike-peer-entries {
          key "peer-name";
          description "IKE peer information";
          leaf peer-name {
            type string;
            mandatory true;
```

```
            description "Name of IKE peer";
          }
          leaf ike-proposal-number {
```



```
            type ike-proposal-number-ref;
            description "IKE proposal number referenced by IKE peer";
          }
          leaf PresharedKey {
            type string;
            description "Preshare key";
          }
          leaf nat-traversal {
            type boolean;
            default false;
            description "Enable/Disable nat traversal";
          }
          choice local-id-type {
            default ip;
            case ip {
              leaf ip {
                type empty;
                description "IP address";
              }
            }
            case fqdn {
              leaf fqdn {
                type empty;
                description "Fully Qualifed Domain name ";
              }
            }
            case dn {
              leaf dn {
                type empty;
                description "Domain name";
              }
            }
            case user_fqdn {
              leaf user_fqdn {
                type empty;
                description "User FQDN";
              }
            }
```

```
            description "Local ID type";
          }
          leaf local-id {
            type string;
            description
              "Local ID Name. When IP is used as local ID type,
               it is ignored. If it is not configurated,
               global local name will be used.";
          }
          leaf remote-id {
            type "string";
```



```
            description "ID of IKE peer";
          }
          leaf low-remote-address {
            type inet:ip-address;
            description "Low range of remote address";
          }
          leaf high-remote-address {
            type inet:ip-address;
            description "High range of remote address";
          }
          leaf certificate {
            type string;
            description "Certificate file name";
          }
          leaf auth-address-begin {
            type inet:ip-address;
            description
              "The begin range of authenticated peer address";
          }
          leaf auth-address-end {
            type inet:ip-address;
            description
              "The end range of authenticated peer address";
          }
        }
      }//End of IKEPeerEntries


      list proposal {
         if-feature ikev2-proposal;
```

```
          key "name";
          uses ikev2-proposal-grouping;
          description
            "Configure IKEv2 proposal";
        }
        list policy {
          if-feature ikev2-policy;
          key "name";
          uses ikev2-policy-profile-grouping;
          description
            "IKEv2 Policy Profile";
        }
      }

      container ipsec {
        if-feature ipsec;
        description
          "Configuration IPsec";
        container sad {
```



```
          if-feature ipsec-sad;
          description
            "Configure the IPSec Security Association Database (SAD)";
          list sad-entries {
            key "spi direction";
            description
              "Configure IPsec Security Association Database(SAD)";
            uses ipsec-sa-grouping;
          }
        }
        container proposal {
          if-feature ipsec-proposal;
          description
            "IPSec Proposal Profile";
          list ipsec-proposal {
            key "name";
            uses ipsec-proposal-grouping;
            description
              "Configure the IP Security (IPSec) proposal";
          }
        }
        container spd {
```

```
        if-feature ipsec-spd;
        description
          "Configure the Security Policy Database (SPD)";
        list spd-entries {
          key "name";
          ordered-by user;
          uses ipsec-policy-grouping;
          description
            "Specify an IPSec policy name";
        }
      }
      container pad {
        description
          "Configure Peer Authorization Database (PAD)";
        list pad-entries {
          key "pad-type pad-id";
          ordered-by user;
          uses identity-grouping;
          description
            "Peer Authorization Database (PAD)";
          leaf pad-id {
            type uint32;
             description
               "PAD identity";
          }
          leaf pad-type {
            type pad-type-t;
```



```
            description
              " PAD type";
          }
          leaf ike-peer-name {
            type string;
            description
              "IKE Peer Name";
          }
          container peer-authentication {
            description
              "Specify IKE peer authentication configuration";
            leaf algorithm {
              type ike-integrity-algorithm-t;
              description
```

```
                "Specify the authentication algorithm";
            }
            leaf preshared-key {
              type empty;
              description
                "Use pre-shared key based authentication";
            }
            leaf rsa-signature {
              type empty;
              description
                "Use signature based authentication by using
                 PKI certificates";
            }
          }
        }
      }
    }


    /*---------------------------*/
    /* Operational State Data   */
    /*---------------------------*/
    grouping ike-proposal-state-components {
      description
        "IKE Proposal operational state";
      list proposal {
        if-feature ike-proposal-state;
        description
          "Operational data for IKE Proposal";
        leaf name {
          type string {
            length "1..50";
          }
          description
```



```
            "Name of the IKE proposal.";
        }
        leaf lifetime {
          type uint32;
          units "seconds";
          description
```

```
            "lifetime";
        }
        leaf encryption {
          type ike-encryption-algorithm-t;
          description
            "Encryption algorithm";
        }
        leaf dh-group {
          type diffie-hellman-group-t;
          description
            "Diffie-Hellman group.";
        }
        leaf authentication {
          type ike-integrity-algorithm-t;
          description
            "authentication";
        }
      }
    }

    grouping ike-policy-state-grouping {
      description
        "IKE Policy State.";
      list policy {
        if-feature ike-policy-state;
        description
          "Operational data for IKE policy";
        leaf name {
          type string {
            length "1..50";
          }
          description
            "Name of the IKE Policy.";
        }
        leaf description {
          type string;
          description
            "Description for IKE Policy.";
        }
        leaf mode {
          type enumeration {
            enum aggressive {
              description
```

```
            "Aggressive mode.";
        }
        enum main {
          description
            "Main mode.";
        }
      }
      description
        "IKE policy mode.";
    }
    leaf connection-type {
      type connection-type-t;
      description
        "IKE policy connection type.";
    }
    leaf local-identity {
      type inet:ipv4-address-no-zone;
      description
        "IP address of the local identity.";
    }
    leaf remote-identity {
      type inet:ipv4-address-no-zone;
      description
        "IP address of the remote identity.";
    }
    leaf pre-shared-key {
      type string;
      description
        "Pre-shared key";
    }
    leaf seq {
      type uint32;
      description
        "sequence number";
    }
    leaf proposal {
      type string;
      description
        "proposal name";
    }
  }
}

grouping ikev2-proposal-state-components {
  description
    "IKEv2 Operational state";
  list proposal {
    if-feature ikev2-proposal-state;
    description
```

```
          "IKEv2 proposal operational data";
        leaf name {
          type string;
          description
            "Name of IKEv2 Proposal.";
        }
        leaf pseudo-random-function {
          type pseudo-random-function-t;
          description
            "Pseudo Random Function for IKEv2.";
        }
        leaf authentication {
          type ike-integrity-algorithm-t;
          description
            "authentication";
        }
        leaf encryption {
          type ike-encryption-algorithm-t;
          description
            "Encryption algorithm";
        }
        leaf dh-group {
          type diffie-hellman-group-t;
          mandatory true;
          description
            "Diffie-Hellman group.";
        }
      }
    }

    grouping ipsec-policy-state-grouping {
      description
        "IPSec operational state";
      list policy {
        if-feature ipsec-policy-state;
        description
          "IPSec policy operational data";
        leaf name {
          type string;
          description
            "IPSec Policy name.";
```

```
        }
        leaf anti-replay-window {
          type uint32;
          description
            "replay window size";
        }
        leaf perfect-forward-secrecy {
          type diffie-hellman-group-t;
```



```
          description
            "Diffie-Hellman group for perfect-forward-secrecy";
        }
        list seq {
          description
            "Sequence number";
          leaf seq-id {
            type uint32;
            description
              "Sequence number";
          }
          leaf proposal-name {
            type string;
            description
              "IPSec proposal name";
          }
        }
      }
    }
    grouping ipsec-proposal-state-grouping {
      description
        "IPSec proposal operational data";
      list proposal {
        if-feature ipsec-proposal-state;
        description
          "IPSec proposal operational data";
        leaf name {
          type string;
          description
            "IPSec Proposal name";
        }
        leaf ah {
          type ike-integrity-algorithm-t;
```

```
          description
            "Authentication Header (AH).";
        }
        container esp {
          description
            "Encapsulating Security Payload (ESP).";
          leaf authentication {
            type ike-integrity-algorithm-t;
            description
              "ESP authentication";
          }
          leaf encryption {
            type ike-encryption-algorithm-t;
            description
              "ESP encryption";
          }
```



```
        }
        leaf ip-comp{
          type empty;
          description
            "IPSec proposal IP-COMP which uses the IP Payload "+
            "compression protocol to compress IP Security (IPSec) "+
            "packets before encryption";
        }
        container lifetime {
          description
            "lifetime for IPSEC SAs";
          leaf kbytes {
            type uint32;
            description
              "lifetime kbytes for IPSEC SAs";

          }
          leaf seconds {
            type uint32;
            description
              "lifetime seconds for IPSEC SAs";
          }
        }
      }
    }
```

```
grouping ipsec-alarms-state-grouping {
  description
    "IPSec alarms operational data";
  leaf hold-down {
    if-feature ipsec-alarms-state;
    type uint32;
    description
      "Hold-down value";
  }
}

grouping ipsec-sa-ah-state-grouping {
  description
    "IPSec SA's AH operational data";

  leaf spi {
    if-feature ipsec-sa-ah-state;
    type uint32;
    description
      "Security Parameter Index (SPI) value";
  }
  leaf description {
    if-feature ipsec-sa-ah-state;
```



```
    type string;
    description
      "the description.";
  }
  leaf authentication-algorithm {
    if-feature ipsec-sa-ah-state;
    type ike-integrity-algorithm-t;
    description
      "Authentication algorithm";
  }
  leaf encryption-algorithm {
    if-feature ipsec-sa-ah-state;
    type ike-encryption-algorithm-t;
    description
      "Encryption algorithm";
  }
}
```

```
grouping ipsec-sa-state-grouping {
  description
    "IPSec Security Association Operational data";
  list sa {
    if-feature ipsec-sa-state;
    description
      "IPSec SA operational data";
    leaf name {
      type string;
      description
        "Specify IPSec Security Association (SA) name";
    }
    leaf anti-replay-window {
      type uint16;
      description
        "replay window size";
    }
    leaf ip-comp {
      type empty;
      description
        "Enables IPCOMP, which uses the IP payload compression
         protocol to compress IP security (IPsec) packets before
         encryption";
    }
    uses ipsec-sa-ah-state-grouping;
  }
}

/* draft-wang-ipsecme-ipsec-yang-00 */
grouping ipsec-tunnel-mode-info {
  description
```



```
    "common infomations when using IPsec tunnel mode";
  leaf local-address {
    if-feature ipsec-tunnel;
    type string;
    description
      "Local address of IPsec tunnel mode";
  }
  leaf remote-address {
    if-feature ipsec-tunnel;
```

```
        type string;
        description
          "Remote address of IPsec tunnel mode";
      }
      leaf bypass-df {
        if-feature ipsec-tunnel;
        type enumeration {
          enum "set" {
            description
              "Set the df bit";
          }
          enum "clear" {
            description
              "Clear the df bit";
          }
          enum "copy" {
            description
              "Copy the df bit from inner header";
          }
        }
        description
          "This flag indicates how to process tunnel mode df flag";
      }
      leaf dscp-flag {
        if-feature ipsec-tunnel;
        type boolean;
        description
          "This flag indicate whether bypass DSCP or map to
           unprotected DSCP values (array) if needed to
           restrict bypass of DSCP values.";
      }
      leaf stateful-frag-check-flag {
        if-feature ipsec-tunnel;
        type boolean;
        description
          "This flag indicates whether stateful fragment checking
           will be used.";
      }
    }
    grouping traffic-selector {
```



```
      description
```

```
    "IPsec traffic selector information";
  leaf local-address-low {
    if-feature ipsec-local-address-range;
    type inet:ip-address;
    description
      "Low range of local address";
  }
  leaf local-address-high {
    if-feature ipsec-local-address-range;
    type inet:ip-address;
    description
      "High range of local address";
  }
  leaf remote-address-low {
    if-feature ipsec-remote-address-range;
    type inet:ip-address;
    description
      "Low range of remote address";
  }
  leaf remote-address-high {
    if-feature ipsec-remote-address-range;
    type inet:ip-address;
    description
      "High range of remote address";
  }
  leaf next-protocol-low {
    if-feature ipsec-next-protocol-range;
    type uint16;
    description
      "Low range of next protocol";
  }
  leaf next-protocol-high {
    if-feature ipsec-next-protocol-range;
    type uint16;
    description
      "High range of next protocol";
  }
  leaf local-port-low {
    if-feature ipsec-local-port-range;
    type inet:port-number;
    description
      "Low range of local port";
  }
  leaf local-port-high {
    if-feature ipsec-local-port-range;
    type inet:port-number;
    description
      "High range of local port";
```

```
      }
      leaf remote-port-high {
        if-feature ipsec-remote-port-range;
        type inet:port-number;
        description
          "Low range of remote port";
      }
      leaf remote-port-low {
        if-feature ipsec-remote-port-range;
        type inet:port-number;
        description
          "High range of remote port";
      }
    }
    grouping ipsec-algorithm-info {
      description
        "IPsec algorithm information used by SPD and SAD";
      leaf ah-auth-algorithm {
        if-feature ipsec-ah-authentication;
        type ipsec-authentication-algorithm;
        description
          "Authentication algorithm used by AH";
      }
      leaf esp-integrity-algorithm {
        if-feature ipsec-esp-integrity;
        type ipsec-authentication-algorithm;
        description
          "Integrity algorithm used by ESP";
      }
      leaf esp-encrypt-algorithm {
        if-feature ipsec-esp-encrypt;
        type ipsec-encryption-algorithm;
        description
          "Encryption algorithm used by ESP";
      }
    }
    grouping ipsec-stat {
      leaf inbound-packets {
        if-feature ipsec-stat;
        type uint64;
        config false;
        description "Inbound Packet count";
      }
```

```
        leaf outbound-packets {
          if-feature ipsec-stat;
          type uint64;
          config false;
          description "Outbound Packet count";
        }
```



```
        leaf inbound-bytes {
          if-feature ipsec-stat;
          type uint64;
          config false;
          description "Inbound Packet bytes";
        }
        leaf outbound-bytes {
          if-feature ipsec-stat;
          type uint64;
          config false;
          description "Outbound Packet bytes";
        }
        leaf inbound-drop-packets {
          if-feature ipsec-stat;
          type uint64;
          config false;
          description "Inbound dropped packets count";
        }
        leaf outbound-drop-packets {
          if-feature ipsec-stat;
          type uint64;
          config false;
          description "Outbound dropped packets count";
        }
        container dropped-packet-detail {
          if-feature ipsec-stat;
          description "The detail information of dropped packets";
          leaf sa-non-exist {
            type uint64;
            config false;
            description
              "The dropped packets counts caused by SA non-exist.";
          }
          leaf queue-full {
            type uint64;
```

```
          config false;
          description
            "The dropped packets counts caused by full processing
             queue";
        }
        leaf auth-failure {
          type uint64;
          config false;
          description
            "The dropped packets counts caused by authentication
             failure";
        }
        leaf malform {
          type uint64;
```



```
          config false;
          description "The dropped packets counts of malform";
        }
        leaf replay {
          type uint64;
          config false;
          description "The dropped packets counts of replay";
        }
        leaf large-packet {
          type uint64;
          config false;
          description "The dropped packets counts of too large";
        }
        leaf invalid-sa {
          type uint64;
          config false;
          description "The dropped packets counts of invalid SA";
        }
        leaf policy-deny {
          type uint64;
          config false;
          description
            "The dropped packets counts of denyed by policy";
        }
        leaf other-reason {
          type uint64;
          config false;
```

```
          description
            "The dropped packets counts of other reason";
        }
      }
      description "IPsec statistics information";
    }


    container ike-state {
      if-feature ikev1-state;
      config "false";
      uses ike-proposal-state-components;
      uses ike-policy-state-grouping;
      description
        "Contain the operational data for IKE.";
    }
    container ikev2-state {
      if-feature ikev2-state;
      config "false";
      uses ikev2-proposal-state-components;
      uses ike-policy-state-grouping;
      description
```



```
        "Contain the operational data for IKEv2.";
    }
    container ipsec-state {
      if-feature ipsec-state;
      config "false";
      uses ipsec-policy-state-grouping;
      uses ipsec-proposal-state-grouping;
      uses ipsec-alarms-state-grouping;
      uses ipsec-sa-state-grouping;
      container redundancy {
        if-feature ipsec-redundancy;
        description
          "Configure redundancy for IPSec";
        leaf inter-chassis {
          type empty;
          description
            "Set redundancy at chassis level";
        }
      }
```

```
    description
      "Contain the operational data for IPSec.";
  }

  /* draft-wang-ipsecme-ipsec-yang-00 */
  container sad {
    if-feature sad;
    config false;
    description
      "The IPsec SA database";
    list sad-entries {
      key "spi security-protocol direction";
      description
        "The SA entries information";
      leaf spi {
        type ipsec-spi;
        description
          "Security parameter index of SA entry.";
      }
      leaf security-protocol {
        type ipsec-protocol;
        description
          "Security protocol of IPsec SA.";
      }
      leaf direction {
        type ipsec-traffic-direction;
        description
          "It indicates whether the SA is inbound SA or
           out bound SA.";
```



```
      }
      leaf sa-type {
        type enumeration {
          enum "manual" {
            description
              "Manual IPsec SA";
          }
          enum "isakmp" {
            description
              "ISAKMP IPsec SA";
          }
```

```
        }
        description
          "It indicates whether the SA is created by manual
           or by dynamic protocol.";
      }
      leaf sequence-number {
        type uint64;
        description
          "Current sequence number of IPsec packet.";
      }
      leaf sequence-number-overflow-flag {
        type boolean;
        description
          "The flag indicating whether overflow of the sequence
           number counter should prevent transmission of additional
           packets on the SA, or whether rollover is permitted.";
      }
      leaf anti-replay-enable-flag {
        type boolean;
        description
          "It indicates whether anti-replay is enable or disable.";
      }
      leaf anti-replay-window-size {
        type uint64;
        description
          "The size of anti-replay window.";
      }
      uses ipsec-algorithm-info;
      container life-time {
        leaf life-time-in-seconds {
          type uint32;
          description
            "SA life time in seconds";
        }
        leaf remain-life-time-in-seconds {
          type uint32;
          description
            "Remain SA life time in seconds";
```



```
        }
        leaf life-time-in-byte {
          type uint32;
```

```
        description
          "SA life time in bytes";
      }
      leaf remain-life-time-in-byte {
        type uint32;
        description
          "Remain SA life time in bytes";
      }
      description
        "SA life time information";
    }
    leaf protocol-mode {
      type ipsec-mode;
      description
        "It indicates whether tunnel mode or transport mode
         will be used.";
    }
    container tunnel-mode-process-info {
      when "../protocol-mode = 'tunnel'" {
        description
          "External information of SA when SA works in
           tunnel mode.";
      }
      uses ipsec-tunnel-mode-info;
      description
        "External information of SA when SA works in
         tunnel mode.";
    }
    leaf-list dscp {
      type uint8 {
        range "0..63";
      }
      description
        "When traffic matchs SPD, the DSCP values used to
         filter traffic";
    }
    leaf path-mtu {
      type uint16;
      description
        "Path MTU valie";
    }
    leaf nat-traversal-flag {
      type boolean;
      description
        "Whether the SA is used to protect traffic that needs
         nat traversal";
```

```
        }
      }
    }
    container spd {
      if-feature spd;
      config false;
      description
        "IPsec security policy database information";
      list spd-entries {
        description
          "IPsec SPD entry information";
        list name {
          description
            "SPD name information.";
          leaf name-type {
            type ipsec-spd-name;
            description
              "SPD name type.";
          }
          leaf name-string {
            when "../name-type = 'id_rfc_822_addr' or ../name-type =
                  'id_fqdn'" {
              description
                "when name type is id_rfc_822_addr or id_fqdn, the
                 name are saved in string";
            }
            type string;
            description
              "SPD name content";
          }
          leaf name-binary {
            when "../name-type = 'id_der_asn1_dn' or ../name-type =
                  'id_key'" {
              description
                "when name type is id_der_asn1_dn or id_key, the name
                 are saved in binary";
            }
            type binary;
            description
              "SPD name content";
          }
        }
        leaf pfp-flag {
          type boolean;
          description
            "populate from packet flag";
        }
```

```
          list traffic-selector {
            min-elements 1;
```



```
            uses traffic-selector;
            description
              "Traffic selectors of SAD entry";
          }
          leaf operation {
            type ipsec-spd-operation;
            description
              "It indicates how to process the traffic when it matches
               the security policy.";
          }
          container protect-operation {
            when "../operation = 'protect'" {
              description
                "How to protect the traffic when the SPD operation
                 is protect";
            }
            leaf spd-ipsec-mode {
              type ipsec-mode;
              description
                "It indicates which mode is chosen when the traffic need
                be protected by IPsec.";
            }
            leaf esn-flag {
              type boolean;
              description
                "It indicates whether ESN is used.";
            }
            leaf spd-ipsec-protocol {
              type ipsec-protocol;
              description
                "It indicates which protocol (AH or ESP) is chosen.";
            }
            container tunnel-mode-additional {
              when "../spd-ipsec-mode = 'tunnel'" {
                description
                  "Additional informations when choose tunnel mode";
              }
              uses ipsec-tunnel-mode-info;
              description
```

```
            "When use tunnel mode, the additional information of
             SPD.";
          }
          list spd-algorithm {
            min-elements 1;
            uses ipsec-algorithm-info;
            description
              "Algorithms defined in SPD, ordered by decreasing
               priority.";
          }
```



```
          description
            "How to protect the traffic when the SPD operation is
             protect";
        }
      }
    }

    container ipsec-global-statistics {
      if-feature ipsec-global-stats;
      config false;
      description "IPsec global statistics";
      container ipv4 {
        description "IPsec statistics of IPv4";
        uses ipsec-stat;
      }
      container ipv6 {
        description "IPsec statistics of IPv6";
        uses ipsec-stat;
      }
      container global {
        description "IPsec statistics of global";
        uses ipsec-stat;
      }
    }


    /*---------------------*/
    /* RPC                 */
    /*---------------------*/
    rpc clear-ipsec-group {
      if-feature clear-ipsec-group;
```

```
      description
        "RPC for clear ipsec states";
      input {
        leaf alarm-hold-down {
          type uint8;
          description
            "IPSec alarm hold-down";
        }
        leaf ipsec-policy-name {
          type leafref {
            path "/eipsec:ipsec/eipsec:spd/"+
                 "eipsec:spd-entries/eipsec:name";
          }
          description
            "IPSec Policy name.";
        }
      }
    }
```



```
    rpc clear-ike-group {
      if-feature clear-ike-group;
      description
        "RPC for clear IKE states";
      input {
        leaf proposal {
          type leafref {
            path "/eipsec:ikev1/eipsec:proposal/"+
                 "eipsec:name";
          }
          description
            "IPSec IKE Proposal name.";
        }
      }
    }

    rpc clear-ikev2-group {
      if-feature clear-ikev2-group;
      description
        "RPC for clear IKEv2 states";
      input {
        leaf proposal {
```

```
        type leafref {
          path "/eipsec:ikev2/eipsec:proposal/"+
               "eipsec:name";
        }
        description
          "IPSec IKEv2 Proposal name.";
      }
    }
  }

  /* draft-wang-ipsecme-ipsec-yang-00 */
  rpc reset-ipv4 {
    if-feature reset-ipv4;
    description "Reset IPsec IPv4 statistics";
    input {
      leaf ipv4 {
        type empty;
        description "Reset IPsec IPv4 statistics";
      }
    }
    output {
      leaf status {
        type string;
        description "Operation status";
      }
    }
```



```
  }
  rpc reset-ipv6 {
    if-feature reset-ipv6;
    description "Reset IPsec IPv6 statistics";
    input {
      leaf ipv6 {
        type empty;
        description "Reset IPsec IPv6 statistics";
      }
    }
    output {
      leaf status {
        type string;
        description "Operation status";
      }
```

```
    }
  }
  rpc reset-global {
    if-feature reset-global;
    description "Reset IPsec global statistics";
    input {
      leaf ipv6 {
        type empty;
        description "Reset IPsec global statistics";
      }
    }
    output {
      leaf status {
        type string;
        description "Operation status";
      }
    }
  }

  notification dpd-failure{
    description "IKE peer DPD detect failure";
    leaf peer-id {
      type string;
      description "Peer ID";
    }
  }

  notification peer-authentication-failure {
    if-feature peer-authentication-failure;
    description "Peer authentication fail when negotication";
    leaf peer-id {
      type string;
      description "The ID of remote peer";
    }
```



```
  }

  notification ike-reauth-failure {
    if-feature ike-reauth-failure;
    description "IKE peer reauthentication fail";
    leaf peer-id {
      type string;
```

```
        description "The ID of remote peer";
      }
    }

    notification ike-rekey-failure {
      if-feature ike-rekey-failure;
      description "IKE SA rekey failure";
      leaf peer-id {
        type string;
        description "The ID of remote peer";
      }
      leaf old-i-spi {
        type uint64;
        description "old SPI";
      }
      leaf old-r-spi {
        type uint64;
        description "old SPI";
      }
    }

    notification ipsec-rekey-failure {
      if-feature ipsec-rekey-failure;
      description "IPsec SA rekey failure";
      leaf peer-id {
        type string;
        description "The ID of remote peer";
      }
      leaf old-inbound-spi {
        type ipsec-spi;
        description "old inbound SPI";
      }
      leaf old-outbound-spi {
        type ipsec-spi;
        description "old outbound SPI";
      }
    }
  } /* module ericsson-ipsec */
  <CODE ENDS>
```

5. Security Considerations

The configuration, state, and action data defined in this document are designed to be accessed via the NETCONF protocol [RFC6241]. The data model by itself does not create any security implications. The security considerations for the NETCONF protocol are applicable. The NETCONF protocol used for sending the data supports authentication and encryption.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., Kivinen, T., "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, October 2014.

[RFC6071] Frankel, S., Krishnan, S., "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", February 2011.

6.2. Informative References

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.

Authors' Addresses

Khanh Tran
Ericsson
300 Holger Way
San Jose, CA 95134
USA
Email: khanh.x.tran@ericsson.com

Honglei Wang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China
Email: stonewater.wang@huawei.com

Vijay Kumar Nagaraj
Huawei Technologies
Huawei Technologies India Pvt Ltd
Bangalore 560008
India
Email: vijay.kn@huawei.com

Xia Chen
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China
Email: xiachen@huawei.com