I2NSF
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2019

R. Marin-Lopez
G. Lopez-Millan
University of Murcia
F. Pereniguez-Garcia
University Defense Center

March 11, 2019

Software-Defined Networking (SDN)-based IPsec Flow Protection
draft-ietf-i2nsf-sdn-ipsec-flow-protection-04

Abstract

This document describes how providing IPsec-based flow protection by means of a Software-Defined Network (SDN) controller (aka. Security Controller) and establishes the requirements to support this service. It considers two main well-known scenarios in IPsec: (i) gateway-to-gateway and (ii) host-to-host. The SDN-based service described in this document allows the distribution and monitoring of IPsec information from a Security Controller to one or several flow-based Network Security Function (NSF). The NSFs implement IPsec to protect data traffic between network resources with IPsec.

The document focuses in the NSF Facing Interface by providing models for Configuration and State data model required to allow the Security Controller to configure the IPsec databases (SPD, SAD, PAD) and IKEv2 to establish security associations with a reduced intervention of the network administrator.

Table of Contents

1. Introduction

Software-Defined Networking (SDN) is an architecture that enables users to directly program, orchestrate, control and manage network resources through software. SDN paradigm relocates the control of network resources to a dedicated network element, namely SDN controller. The SDN controller manages and configures the distributed network resources and provides an abstracted view of the network resources to the SDN applications. The SDN application can customize and automate the operations (including management) of the abstracted network resources in a programmable manner via this interface [RFC7149][ITU-T.Y.3300] [ONF-SDN-Architecture][ONF-OpenFlow].

Recently, several network scenarios are considering a centralized way of managing different security aspects. For example, Software-Defined WANs (SD-WAN) advocates to manage IPsec SAs from a centralized point. Therefore, with the growth of SDN-based scenarios where network resources are deployed in an autonomous manner, a mechanism to manage IPsec SAs according to the SDN architecture becomes more relevant. Thus, the SDN-based service described in this document will autonomously deal with IPsec SAs management following a SDN paradigm.

An example of usage can be the notion of Software Defined WAN (SD-WAN), SDN extension providing a software abstraction to create secure network overlays over traditional WAN and branch networks. SD-WAN is based on IPsec as underlying security protocol and aims to provide flexible, automated, fast deployment and on-demand security network services.

IPsec architecture [RFC4301] defines a clear separation between the processing to provide security services to IP packets and the key management procedures to establish the IPsec security associations. In this document, we define a service where the key management procedures can be carried by an external entity: the Security Controller.

First, this document exposes the requirements to support the protection of data flows using IPsec [RFC4301]. We have considered two general cases:

1) IKE case. The Network Security Function (NSF) implements the Internet Key Exchange (IKE) protocol and the IPsec databases: the Security Policy Database (SPD), the Security Association Database

(SAD) and the Peer Authorization Database (PAD). The Security Controller is in charge of provisioning the NSF with the required information to IKE, the SPD and the PAD.

2) IKE-less case. The NSF only implements the IPsec databases (no IKE implementation). The Security Controller will provide the required parameters to create valid entries in the SPD and the SAD into the NSF. Therefore, the NSF will have only support for IPsec while automated key management functionality is moved to the controller.

In both cases, an interface/protocol is required to carry out this provisioning in a secure manner between the Security Controller and the NSF. In particular, IKE case requires the provision of SPD and PAD entries and the IKE credential and information related with the IKE negotiation (e.g. IKE_SA_INIT), and IKE-less case requires the management of SPD and SAD entries. Based on YANG models in [netconf-vpn] and [I-D.tran-ipsecme-yang], RFC 4301 [RFC4301] and RFC 7296 [RFC7296] this document defines the required interfaces with a YANG model for configuration and state data for IKE, PAD, SPD and SAD (see Appendix A, Appendix B and Appendix C).

This document considers two typical scenarios to manage autonomously IPsec SAs: gateway-to-gateway and host-to-host [RFC6071]. The analysis of the host-to-gateway (roadwarrior) scenario is out of scope of this document. In these cases, host or gateways or both may act as NSFs. Finally, it also discusses the situation where two NSFs are under the control of two different Security Controllers.

NOTE: This work pays attention to the challenge "Lack of Mechanism for Dynamic Key Distribution to NSFs" defined in [RFC8192] in the particular case of the establishment and management of IPsec SAs. In fact, this I-D could be considered as a proper use case for this particular challenge in [RFC8192].

2. Requirements Language

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]. When these words appear in lower case, they have their natural language meaning.

3. Terminology

This document uses the terminology described in [RFC7149], [RFC4301], [ITU-T.Y.3300], [ONF-SDN-Architecture], [ONF-OpenFlow],

[ITU-T.X.1252], [ITU-T.X.800] and [I-D.ietf-i2nsf-terminology]. In addition, the following terms are defined below:

- Software-Defined Networking. A set of techniques enabling to directly program, orchestrate, control, and manage network resources, which facilitates the design, delivery and operation of network services in a dynamic and scalable manner [ITU-T.Y.3300].

- Flow/Data Flow. Set of network packets sharing a set of characteristics, for example IP dst/src values or QoS parameters.

- Security Controller. A Controller is a management component that contains control plane functions to manage and facilitate information sharing, as well as execute security functions. In the context of this document, it provides IPsec management information.

- Network Security Function (NSF). Software that provides a set of security-related services.

- Flow-based NSF. A NSF that inspects network flows according to a set of policies intended for enforcing security properties. The NSFs considered in this document falls into this classification.

- Flow-based Protection Policy. The set of rules defining the conditions under which a data flow MUST be protected with IPsec, and the rules that MUST be applied to the specific flow.

- Internet Key Exchange (IKE) v2 Protocol to establish IPsec Security Associations (SAs). It requires information about the required authentication method (i.e. raw RSA/ECDSA keys or X.509 certificates), DH groups, modes and algorithms for IKE SA negotiation, etc.

- Security Policy Database (SPD). It includes information about IPsec policies direction (in, out), local and remote addresses, inbound and outboud SAs, etc.

- Security Associations Database (SAD). It includes information about IPsec SAs, such as SPI, destination addresses, authentication and encryption algorithms and keys to protect IP flows.

- Peer Authorization Database (PAD). It provides the link between the SPD and a security association management protocol such as IKE or the SDN-based solution described in this document.

4. Objectives

- To describe the architecture for the SDN-based IPsec management, which implements a security service to allow the establishment and management of IPsec security associations from a central point, in order to protect specific data flows.

- To define the interfaces required to manage and monitor the IPsec Security Associations in the NSF from a Security Controller. YANG models are defined for configuration and state data for IPsec management.

5. SDN-based IPsec management description

As mentioned in Section 1, two cases are considered:

5.1. IKE case: IKE/IPsec in the NSF

In this case the NSF ships an IKEv2 implementation besides the IPsec support. The Security Controller is in charge of managing and applying SPD and PAD entries (deriving and delivering IKE Credentials such as a pre-shared key, certificates, etc.), and applying other IKE configuration parameters (e.g. IKE_SA_INIT algorithms) to the NSF for the IKE negotiation.

With these entries, the IKEv2 implementation can operate to establish the IPsec SAs. The application (administrator) establishes the IPsec requirements and information about the end points information (through the Client Facing Interface, [RFC8192]), and the Security Controller translates those requirements into IKE, SPD and PAD entries that will be installed into the NSF (through the NSF Facing Interface). With that information, the NSF can just run IKEv2 to establish the required IPsec SA (when the data flow needs protection). Figure 1 shows the different layers and corresponding functionality.

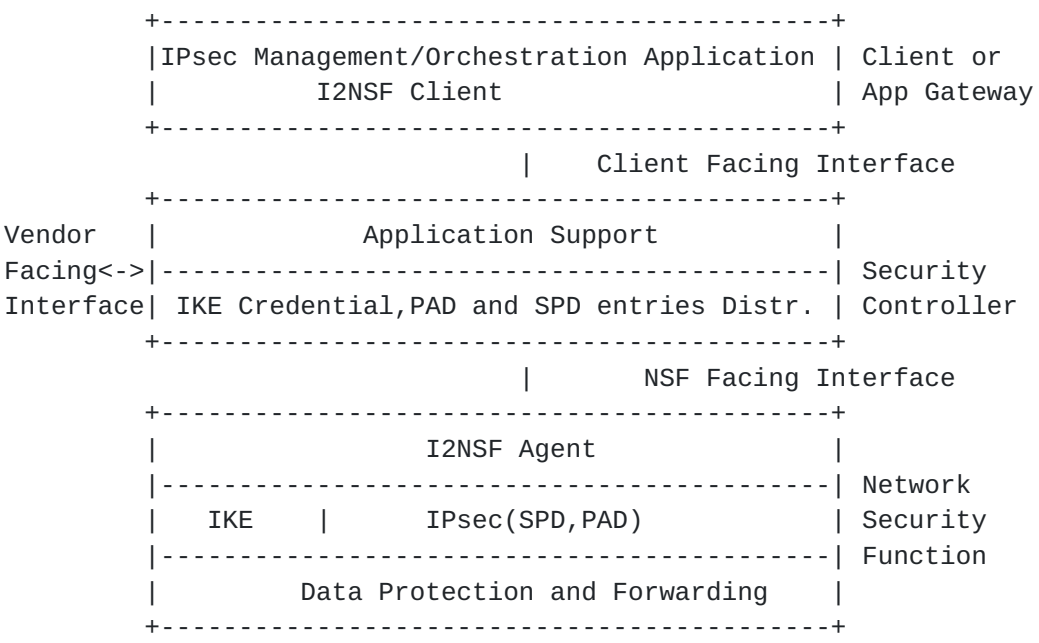

Figure 1: IKE case: IKE/IPsec in the NSF

5.1.1. Interface Requirements for IKE case

SDN-based IPsec flow protection services provide dynamic and flexible management of IPsec SAs in flow-based NSF. In order to support this capability in case IKE case, the following interface requirements are to be met:

- A YANG data model for configuration data for IKEv2, SPD and PAD.

- A YANG data model for state data for IKE, PAD, SPD and SAD (NOTE: the SAD entries are created in runtime by IKEv2.)

- In scenarios where multiple controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface [RFC7426] is required to exchange IPsec-related information. For example, if two gateways need to establish an IPsec SA and both are under the control of two different controllers then both Security Controllers need to exchange information to properly configure their own gateways. That is, the may need to agree on whether IKEv2 authentication will be based on raw public keys or pre-shared keys. In case of using pre-shared keys they will have to agree in the PSK.

5.2. IKE-less case: IPsec (no IKEv2) in the NSF

In this case, the NSF does not deploy IKEv2 and, therefore, the Security Controller has to perform the IKE security functions and management of IPsec SAs by populating and managing the SPD and the SAD.

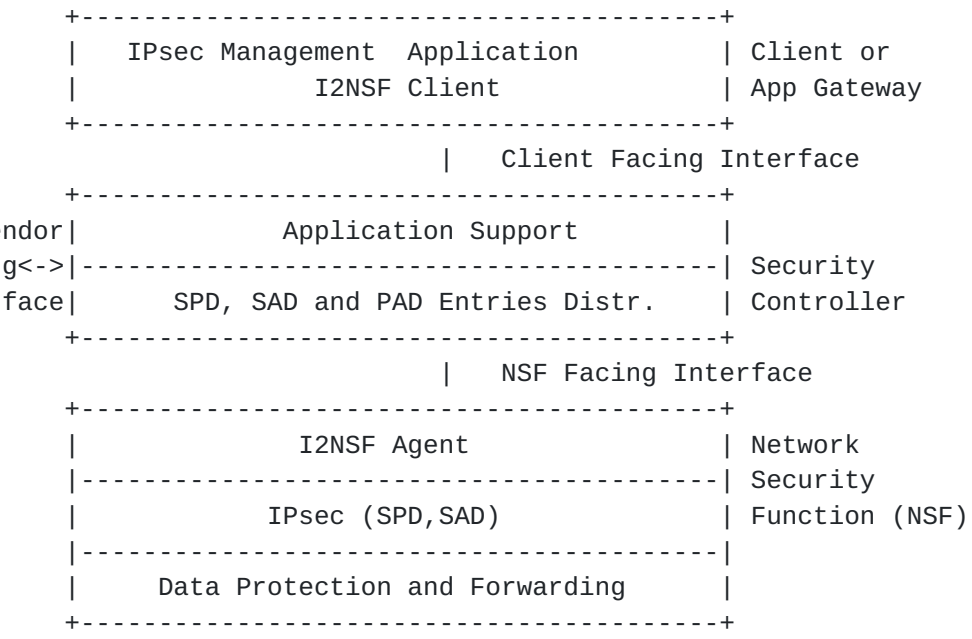

Figure 2: IKE-less case: IPsec (no IKE) in the NSF

As shown in Figure 2, applications for flow protection run on the top of the Security Controller. When an administrator enforces flow-based protection policies through the Client Facing Interface, the Security Controller translates those requirements into SPD and SAD entries, which are installed in the NSF. PAD entries are not required since there is no IKEv2 in the NSF.

5.2.1. Interface Requirements for IKE-less case

In order to support the IKE-less case, the following requirements are to be met:

- A YANG data model for configuration data for SPD and SAD.
- A YANG data model for state data for SPD and SAD.
- In scenarios where multiple controllers are implicated, SDN-based IPsec management services may require a mechanism to discover which Security Controller is managing a specific NSF. Moreover, an east-west interface [RFC7426] is required to exchange IPsec-

related information. NOTE: A possible east-west protocol for this IKE-less case could be IKEv2. However, this needs to be explore since the IKEv2 peers would be the Security Controllers.

Specifically, the IKE-less case assumes that the SDN controller has to perform some security functions that IKEv2 typically does, namely (non-exhaustive):

- IV generation.
- prevent counter resets for same key.
- Generation of pseudo-random cryptographic keys for the IPsec SAs.
- Rekey of the IPsec SAs based on notification from the NSF (i.e. expire).
- Generation of the IPsec SAs when required based on notifications (i.e. sadb_acquire).
- NAT Traversal discovery and management.

Additionally to these functions, another set of tasks must be performed by the Controller (non-exhaustive list):

- SPI random generation.
- Cryptographic algorithm/s selection.
- Usage of extended sequence numbers.
- Establishment of proper traffic selectors.

5.3. IKE case vs IKE-less case

IKE case MAY be easier to deploy than IKE-less case because current gateways typically have an IKEv2/IPsec implementation. Moreover hosts can install easily an IKE implementation. As downside, the NSF needs more resources to hold IKEv2. Moreover, the IKEv2 implementation needs to implement an interface so that the I2NSF Agent can interact with them.

Alternatively, IKE-less case allows lighter NSFs (no IKEv2 implementation), which benefits the deployment in constrained NSFs. Moreover, IKEv2 does not need to be performed in gateway-to-gateway and host-to-host scenarios under the same Security Controller (see Section 7.1). On the contrary, the overload of creating fresh IPsec SAs is shifted to the Security Controller since IKEv2 is not in the

NSF. As a consequence, this may result in a more complex implementation in the controller side. This overload may create some scalability issues when the number of NSFs is high.

In general, literature around SDN-based network management using a centralized SDN controller is aware about scalability issues and solutions have been already provided (e.g. hierarchical SDN controllers; having multiple replicated SDN controllers, etc). In the context of IPsec management, one straight way to reduce the overhead and the potential scalability issue in the Security Controller is to apply IKE case, described in this document, since the IPsec SAs are managed between NSFs without the involvement of the Security Controller at all, except by the initial IKE configuration provided by the Security Controller. Other option with IKE-less is to use techniques already seen in SDN world such as, for example, hierarchical SDN controllers. Other solutions, such as Controller-IKE [I-D.carrel-ipsecme-controller-ike], have proposed that NSFs provide their DH public keys to the Security Controller, so that the Security Controller distributes all public keys to all peers. All peers can calculate a unique pairwise secret for each other peer and there is no inter-NSF messages. A re-key mechanism is further described in [I-D.carrel-ipsecme-controller-ike].

In terms of security, IKE case provides better security properties than IKE-less case, as we discuss in section Section 8. The main reason is that the Security Controller is not able to observe any session keys generated for the IPsec SAs because IKEv2 is in charge of negotiating the IPsec SAs.

5.3.1. Rekeying process

For IKE case, the rekeying process is carried out by IKEv2, following the information defined in the SPD and SAD.

For IKE-less case, the Security Controller needs to take care of the rekeying process. When the IPsec SA is going to expire (e.g. IPsec SA soft lifetime), it has to create a new IPsec SA and remove the old one. This rekeying process starts when the Security Controller receives a sadb_expire notification or it decides so, based on lifetime state data obtained from the NSF.

To explain the rekeying process between two IPsec peers A and B, let assume that SPIa1 identifies the inbound SA in A and SPIb1 the inbound SA in B.

1. The Security Controller chooses two random values as SPI for the new inbound SAs: for example, SPIa2 for A and SPIb2 for B. These numbers MUST not be in conflict with any IPsec SA in A or B.

Then, the Security Controller creates an inbound SA with SPIa2 in A and another inbound SA in B with SPIb2. It can send this information simultaneously to A and B.

2. Once the Security Controller receives confirmation from A and B, inbound SA are correctly installed. Then it proceeds to send in parallel to A and B the outbound SAs: it sends the outbound SA to A with SPIb2 and the outbound SA to B with SPIa2. At this point the new IPsec SA is ready.

3. Once the Security Controller receives confirmation from A and B, that the outbound SAs have been installed, the Security Controller deletes the old IPsec SAs from A (inbound SPIa1 and outbound SPIb1) and B (outbound SPIa1 and inbound SPIb1) in parallel. It is worth noting that if the IPsec implementation can itself detect traffic on the new IPsec SA, and it can delete the old IPsec SA itself without instruction from the Security Controller, then this step 3 is not required.

5.3.2. NSF state loss

If one of the NSF restarts, it will lose the IPsec state (affected NSF). By default, the Security Controller can assume that all the state has been lost and therefore it will have to send IKEv2, SPD and PAD information to the NSF in IKE case, and SPD and SAD information in IKE-less case.

In both cases, the Security Controller is aware of the affected NSF (e.g. the NETCONF/TCP connection is broken with the affected NSF, the Security Controller is receiving sadb_bad-spi notification from a particular NSF, etc.). Moreover, the Security Controller has a register about all the NSFs that have IPsec SAs with the affected NSF. Therefore, it knows the affected IPsec SAs.

In IKE case, the Security Controller will configure the affected NSF with the new IKEv2, SPD and PAD information. It has also to send new parameters (e.g. a new fresh PSK for authentication) to the NSFs which have IKEv2 SAs and IPsec SAs with the affected NSF. It can also instruct the affected NSF to send IKEv2 INITIAL_CONTACT. Finally, the Security Controller will instruct the affected NSF to start the IKEv2 negotiation with the new configuration.

In IKE-less case, if the Security Controller detects that a NSF has lost the IPsec SAs (e.g. it reboots) it will delete the old IPsec SAs of the non-failed nodes established with the failed node (step 1). This prevents the non-failed nodes from leaking plaintext. If the failed node comes to live, the Security Controller will configure the new inbound IPsec SAs between the failed node and all the nodes the

failed was talking to (step 2). After these inbound IPsec SAs have been established, the Security Controller can configure the outbound IPsec SAs (step 3).

Nevertheless other more optimized options can be considered (e.g. making IKEv2 configuration permanent between reboots).

5.3.3. NAT Traversal

In IKE case, IKEv2 already owns a mechanism to detect whether some of the peers or both are located behind a NAT. If there is a NAT network configured between two peers, it is required to activate the usage of UDP or TCP/TLS encapsulation of ESP packets ([RFC3948], [RFC8229]). Note that the usage of TRANSPORT mode when NAT is required is forbidden in this specification.

On the contrary, IKE-less case does not have any protocol in the NSFs to detect whether they are located behind a NAT or not. However, the SDN paradigm generally assumes the Security Controller has a view of the network it controls. This view is built either requesting information to the NSFs under its control, or because these NSFs inform to the Security Controller. Based on this information, the Security Controller can guess if there is a NAT configured between two hosts, and apply the required policies to both NSFs besides activating the usage of UDP or TCP/TLS encapsulation of ESP packets ([RFC3948], [RFC8229]).

For example, the Security Controller could directly request the NSF for specific data such as networking configuration, NAT support, etc. Protocols such as NETCONF or SNMP can be used here. For example, RFC 7317 [RFC7317] provides a YANG data model for system management or [I-D.ietf-opsawg-nat-yang] a data model for NAT management. The Security Controller can use this NETCONF module with a gateway to collect NAT information or even configure a NAT. In any case, if this NETCONF module is not available and the Security Controller cannot know if a host is behind a NAT or not, then IKE case should be the right choice and not the IKE-less.

6. YANG configuration data models

In order to support IKE case and IKE-less case we have modelled the different parameters and values that must be configured to manage IPsec SAs. Specifically, IKE requires modeling IKEv2, SPD and PAD while IKE-less case requires configuration models for the SPD and SAD. We have defined three models: ietf-ipsec-common (Appendix A), ietf-ipsec-ike (Appendix B, IKE case), ietf-ipsec-ikeless (Appendix C, IKE-less case). Since the model ietf-ipsec-common has only typedef and groupings common to the other modules, in the

following we only show a simplified view of the ietf-ipsec-ike and ietf-ipsec-ikeless models.

6.1. IKE case model

The model related to IKEv2 has been extracted from reading IKEv2 standard in [RFC7296], and observing some open source implementations, such as Strongswan or Libreswan.

The definition of the PAD model has been extracted from the specification in section 4.4.3 in [RFC4301] (NOTE: We have observed that many implementations integrate PAD configuration as part of the IKEv2 configuration.)

```
module: ietf-ipsec-ike
  +--rw ikev2
     +--rw pad
     |  +--rw pad-entry* [pad-entry-id]
     |     +--rw pad-entry-id                  uint64
     |     +--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
     |     |  +--:(id_key)
     |     |  |  +--rw id_key?                 string
     |     |  +--:(id_null)
     |     |  |  +--rw id_null?                empty
     |     |  +--:(user_fqdn)
     |     |     +--rw user_fqdn?              string
     |     +--rw my-identifier                 string
     |     +--rw pad-auth-protocol?            auth-protocol-type
     |     +--rw auth-method
     |        +--rw auth-m?              auth-method-type
     |        +--rw eap-method
     |        |  +--rw eap-type?   uint8
     |        +--rw pre-shared
     |        |  +--rw secret?   yang:hex-string
     |        +--rw digital-signature
     |           +--rw ds-algorithm?     signature-algorithm-t
```

```
|           +--rw raw-public-key?   yang:hex-string
|           +--rw key-data?         string
|           +--rw key-file?         string
|           +--rw ca-data*          string
|           +--rw ca-file?          string
|           +--rw cert-data?        string
|           +--rw cert-file?        string
|           +--rw crl-data?         string
|           +--rw crl-file?         string
|           +--rw oscp-uri?         inet:uri
|        +--rw ike-conn-entry* [conn-name]
|        |  +--rw conn-name                              string
|        |  +--rw autostartup                            type-autostartup
|        |  +--rw initial-contact?                       boolean
|        |  +--rw version?                               enumeration
|        |  +--rw ike-fragmentation?                     boolean
|        |  +--rw ike-sa-lifetime-hard
|        |  |  +--rw time?      yang:timestamp
|        |  |  +--rw idle?      yang:timestamp
|        |  |  +--rw bytes?     uint32
|        |  |  +--rw packets?   uint32
|        |  +--rw ike-sa-lifetime-soft
|        |  |  +--rw time?      yang:timestamp
|        |  |  +--rw idle?      yang:timestamp
|        |  |  +--rw bytes?     uint32
|        |  |  +--rw packets?   uint32
|        |  |  +--rw action?    ic:lifetime-action
|        |  +--rw ike-sa-authalg*                        ic:integrity-algorithm-t
|        |  +--rw ike-sa-encalg*                         ic:encryption-algorithm-t
|        |  +--rw dh_group                               uint32
|        |  +--rw half-open-ike-sa-timer?                uint32
|        |  +--rw half-open-ike-sa-cookie-threshold?     uint32
|        |  +--rw local
|        |  |  +--rw local-pad-id?   uint64
|        |  +--rw remote
|        |  |  +--rw remote-pad-id?   uint64
|        |  +--rw espencap?                              esp-encap
|        |  +--rw sport?                                 inet:port-number
|        |  +--rw dport?                                 inet:port-number
|        |  +--rw oaddr*                                 inet:ip-address
|        |  +--rw spd
|        |  |  +--rw spd-entry* [spd-entry-id]
|        |  |     +--rw spd-entry-id            uint64
|        |  |     +--rw priority?               uint32
|        |  |     +--rw anti-replay-window?     uint16
|        |  |     +--rw names* [name]
|        |  |     |  +--rw name-type?   ipsec-spd-name
|        |  |     |  +--rw name         string
```

```
      |  |     +--rw condition
      |  |     |  +--rw traffic-selector-list* [ts-number]
      |  |     |     +--rw ts-number               uint32
      |  |     |     +--rw direction?              ipsec-traffic-direction
      |  |     |     +--rw local-subnet?           inet:ip-prefix
      |  |     |     +--rw remote-subnet?          inet:ip-prefix
      |  |     |     +--rw upper-layer-protocol*   ipsec-upper-layer-proto
      |  |     |     +--rw local-ports* [start end]
      |  |     |     |  +--rw start    inet:port-number
      |  |     |     |  +--rw end      inet:port-number
      |  |     |     +--rw remote-ports* [start end]
      |  |     |        +--rw start    inet:port-number
      |  |     |        +--rw end      inet:port-number
      |  |     +--rw processing-info
      |  |     |  +--rw action          ipsec-spd-operation
      |  |     |  +--rw ipsec-sa-cfg
      |  |     |     +--rw pfp-flag?            boolean
      |  |     |     +--rw extSeqNum?           boolean
      |  |     |     +--rw seqOverflow?         boolean
      |  |     |     +--rw statefulfragCheck?   boolean
      |  |     |     +--rw security-protocol?   ipsec-protocol
      |  |     |     +--rw mode?                ipsec-mode
      |  |     |     +--rw ah-algorithms
      |  |     |     |  +--rw ah-algorithm*   integrity-algorithm-t
      |  |     |     |  +--rw trunc-length?   uint32
      |  |     |     +--rw esp-algorithms
      |  |     |     |  +--rw authentication*   integrity-algorithm-t
      |  |     |     |  +--rw encryption*       encryption-algorithm-t
      |  |     |     |  +--rw tfc_pad?          uint32
      |  |     |     +--rw tunnel
      |  |     |        +--rw local?          inet:ip-address
      |  |     |        +--rw remote?         inet:ip-address
      |  |     |        +--rw bypass-df?      boolean
      |  |     |        +--rw bypass-dscp?    boolean
      |  |     |        +--rw dscp-mapping?   yang:hex-string
      |  |     |        +--rw ecn?            boolean
      |  |     +--rw spd-lifetime-soft
      |  |     |  +--rw time?      yang:timestamp
      |  |     |  +--rw idle?      yang:timestamp
      |  |     |  +--rw bytes?     uint32
      |  |     |  +--rw packets?   uint32
      |  |     |  +--rw action?    lifetime-action
      |  |     +--rw spd-lifetime-hard
      |  |     |  +--rw time?      yang:timestamp
      |  |     |  +--rw idle?      yang:timestamp
      |  |     |  +--rw bytes?     uint32
      |  |     |  +--rw packets?   uint32
      |  |     +--ro spd-lifetime-current
```

```
|  |        +--ro time?      yang:timestamp
|  |        +--ro idle?      yang:timestamp
|  |        +--ro bytes?     uint32
|  |        +--ro packets?   uint32
|  +--ro ike-sa-state
|     +--ro uptime
|     |  +--ro running?   yang:date-and-time
|     |  +--ro since?     yang:date-and-time
|     +--ro initiator?             boolean
|     +--ro initiator-ikesa-spi?   uint64
|     +--ro responder-ikesa-spi?   uint64
|     +--ro nat-local?             boolean
|     +--ro nat-remote?            boolean
|     +--ro nat-any?               boolean
|     +--ro espencap?              esp-encap
|     +--ro sport?                 inet:port-number
|     +--ro dport?                 inet:port-number
|     +--ro oaddr*                 inet:ip-address
|     +--ro established?           uint64
|     +--ro rekey-time?            uint64
|     +--ro reauth-time?           uint64
|     +--ro child-sas* []
|        +--ro spis
|           +--ro spi-in?    ic:ipsec-spi
|           +--ro spi-out?   ic:ipsec-spi
+--ro number-ike-sas
   +--ro total?               uint32
   +--ro half-open?           uint32
   +--ro half-open-cookies?   uint32
```

6.2. IKE-less case model

The definition of the SPD model has been mainly extracted from the specification in section 4.4.1 and Appendix D in [RFC4301]. Unlike existing implementations (e.g. XFRM), it is worth mentioning that this model follows [RFC4301] and, consequently, each policy (spd-entry) consists of one or more traffic selectors.

The definition of the SAD model has been extracted from the specification in section 4.4.2 in [RFC4301]. Note that this model not only associates an IPsec SA with its corresponding policy (spd-entry-id) but also indicates the specific traffic selector that caused its establishment. In other words, each traffic selector of a policy (spd-entry) generates a different IPsec SA (sad-entry).

The notifications model has been defined using as reference the PF_KEYv2 standard in [RFC2367].

```
  module: ietf-ipsec-ikeless
    +--rw ietf-ipsec
       +--rw spd
       |  +--rw spd-entry* [spd-entry-id]
       |     +--rw spd-entry-id            uint64
       |     +--rw priority?               uint32
       |     +--rw anti-replay-window?     uint16
       |     +--rw names* [name]
       |     |  +--rw name-type?   ipsec-spd-name
       |     |  +--rw name         string
       |     +--rw condition
       |     |  +--rw traffic-selector-list* [ts-number]
       |     |     +--rw ts-number               uint32
       |     |     +--rw direction?              ipsec-traffic-direction
       |     |     +--rw local-subnet?           inet:ip-prefix
       |     |     +--rw remote-subnet?          inet:ip-prefix
       |     |     +--rw upper-layer-protocol*   ipsec-upper-layer-proto
       |     |     +--rw local-ports* [start end]
       |     |     |  +--rw start    inet:port-number
       |     |     |  +--rw end      inet:port-number
       |     |     +--rw remote-ports* [start end]
       |     |        +--rw start    inet:port-number
       |     |        +--rw end      inet:port-number
       |     +--rw processing-info
       |     |  +--rw action          ipsec-spd-operation
       |     |  +--rw ipsec-sa-cfg
       |     |     +--rw pfp-flag?            boolean
       |     |     +--rw extSeqNum?           boolean
       |     |     +--rw seqOverflow?         boolean
       |     |     +--rw statefulfragCheck?   boolean
       |     |     +--rw security-protocol?   ipsec-protocol
       |     |     +--rw mode?                ipsec-mode
       |     |     +--rw ah-algorithms
       |     |     |  +--rw ah-algorithm*   integrity-algorithm-t
       |     |     |  +--rw trunc-length?   uint32
       |     |     +--rw esp-algorithms
       |     |     |  +--rw authentication*   integrity-algorithm-t
       |     |     |  +--rw encryption*       encryption-algorithm-t
       |     |     |  +--rw tfc_pad?          uint32
       |     |     +--rw tunnel
       |     |        +--rw local?          inet:ip-address
       |     |        +--rw remote?         inet:ip-address
       |     |        +--rw bypass-df?      boolean
       |     |        +--rw bypass-dscp?    boolean
       |     |        +--rw dscp-mapping?   yang:hex-string
       |     |        +--rw ecn?            boolean
       |     +--rw spd-lifetime-soft
       |     |  +--rw time?      yang:timestamp
```

```
        |     |  +--rw idle?      yang:timestamp
        |     |  +--rw bytes?     uint32
        |     |  +--rw packets?   uint32
        |     |  +--rw action?    lifetime-action
        |     +--rw spd-lifetime-hard
        |     |  +--rw time?      yang:timestamp
        |     |  +--rw idle?      yang:timestamp
        |     |  +--rw bytes?     uint32
        |     |  +--rw packets?   uint32
        |     +--ro spd-lifetime-current
        |        +--ro time?      yang:timestamp
        |        +--ro idle?      yang:timestamp
        |        +--ro bytes?     uint32
        |        +--ro packets?   uint32
        +--rw sad
           +--rw sad-entry* [sad-entry-id]
              +--rw sad-entry-id                uint64
              +--rw spi?                        ic:ipsec-spi
              +--rw seq-number?                 uint64
              +--rw seq-number-overflow-flag?   boolean
              +--rw anti-replay-window?         uint16
              +--rw spd-entry-id?               uint64
              +--rw local-subnet?               inet:ip-prefix
              +--rw remote-subnet?              inet:ip-prefix
              +--rw upper-layer-protocol*       ipsec-upper-layer-proto
              +--rw local-ports* [start end]
              |  +--rw start    inet:port-number
              |  +--rw end      inet:port-number
              +--rw remote-ports* [start end]
              |  +--rw start    inet:port-number
              |  +--rw end      inet:port-number
              +--rw security-protocol?          ic:ipsec-protocol
              +--rw sad-lifetime-hard
              |  +--rw time?      yang:timestamp
              |  +--rw idle?      yang:timestamp
              |  +--rw bytes?     uint32
              |  +--rw packets?   uint32
              +--rw sad-lifetime-soft
              |  +--rw time?      yang:timestamp
              |  +--rw idle?      yang:timestamp
              |  +--rw bytes?     uint32
              |  +--rw packets?   uint32
              |  +--rw action?    ic:lifetime-action
              +--rw mode?                       ic:ipsec-mode
              +--rw statefulfragCheck?          boolean
              +--rw dscp?                       yang:hex-string
              +--rw path-mtu?                   uint16
              +--rw tunnel
```

```
            |  +--rw local?          inet:ip-address
            |  +--rw remote?         inet:ip-address
            |  +--rw bypass-df?      boolean
            |  +--rw bypass-dscp?    boolean
            |  +--rw dscp-mapping?   yang:hex-string
            |  +--rw ecn?            boolean
            +--rw espencap?                   esp-encap
            +--rw sport?                      inet:port-number
            +--rw dport?                      inet:port-number
            +--rw oaddr*                      inet:ip-address
            +--ro sad-lifetime-current
            |  +--ro time?      yang:timestamp
            |  +--ro idle?      yang:timestamp
            |  +--ro bytes?     uint32
            |  +--ro packets?   uint32
            +--ro stats
            |  +--ro replay-window?   uint32
            |  +--ro replay?          uint32
            |  +--ro failed?          uint32
            +--ro replay_state
            |  +--ro seq?      uint32
            |  +--ro oseq?     uint32
            |  +--ro bitmap?   uint32
            +--ro replay_state_esn
            |  +--ro bmp-len?         uint32
            |  +--ro oseq?            uint32
            |  +--ro oseq-hi?         uint32
            |  +--ro seq-hi?          uint32
            |  +--ro replay-window?   uint32
            |  +--ro bmp*             uint32
            +--rw ah-sa
            |  +--rw integrity
            |     +--rw integrity-algorithm?   ic:integrity-algorithm-t
            |     +--rw key?                   string
            +--rw esp-sa
               +--rw encryption
               |  +--rw encryption-algorithm?   ic:encryption-algorithm-t
               |  +--rw key?                    yang:hex-string
               |  +--rw iv?                     yang:hex-string
               +--rw integrity
               |  +--rw integrity-algorithm?   ic:integrity-algorithm-t
               |  +--rw key?                   yang:hex-string
               +--rw combined-enc-intr?   boolean

    notifications:
      +---n spdb_expire
      |  +--ro index?   uint64
      +---n sadb_acquire
```

```
      |  +--ro base-list* [version]
      |  |  +--ro version       string
      |  |  +--ro msg_type?     sadb-msg-type
      |  |  +--ro msg_satype?   sadb-msg-satype
      |  |  +--ro msg_seq?      uint32
      |  +--ro local-subnet?           inet:ip-prefix
      |  +--ro remote-subnet?          inet:ip-prefix
      |  +--ro upper-layer-protocol*   ipsec-upper-layer-proto
      |  +--ro local-ports* [start end]
      |  |  +--ro start    inet:port-number
      |  |  +--ro end      inet:port-number
      |  +--ro remote-ports* [start end]
      |     +--ro start    inet:port-number
      |     +--ro end      inet:port-number
      +---n sadb_expire
      |  +--ro base-list* [version]
      |  |  +--ro version       string
      |  |  +--ro msg_type?     sadb-msg-type
      |  |  +--ro msg_satype?   sadb-msg-satype
      |  |  +--ro msg_seq?      uint32
      |  +--ro spi?                         ic:ipsec-spi
      |  +--ro anti-replay-window?          uint16
      |  +--ro encryption-algorithm?        ic:encryption-algorithm-t
      |  +--ro authentication-algorithm?    ic:integrity-algorithm-t
      |  +--ro sad-lifetime-hard
      |  |  +--ro time?      yang:timestamp
      |  |  +--ro idle?      yang:timestamp
      |  |  +--ro bytes?     uint32
      |  |  +--ro packets?   uint32
      |  +--ro sad-lifetime-soft
      |  |  +--ro time?      yang:timestamp
      |  |  +--ro idle?      yang:timestamp
      |  |  +--ro bytes?     uint32
      |  |  +--ro packets?   uint32
      |  +--ro sad-lifetime-current
      |     +--ro time?      yang:timestamp
      |     +--ro idle?      yang:timestamp
      |     +--ro bytes?     uint32
      |     +--ro packets?   uint32
      +---n sadb_bad-spi
         +--ro state    ic:ipsec-spi
```

7. Use cases examples

This section explains how different traditional configurations, that is, host-to-host and gateway-to-gateway are deployed using this SDN-based IPsec management service. In turn, these configurations will be typical in modern networks where, for example, virtualization will be key.

7.1. Host-to-host or gateway-to-gateway under the same controller

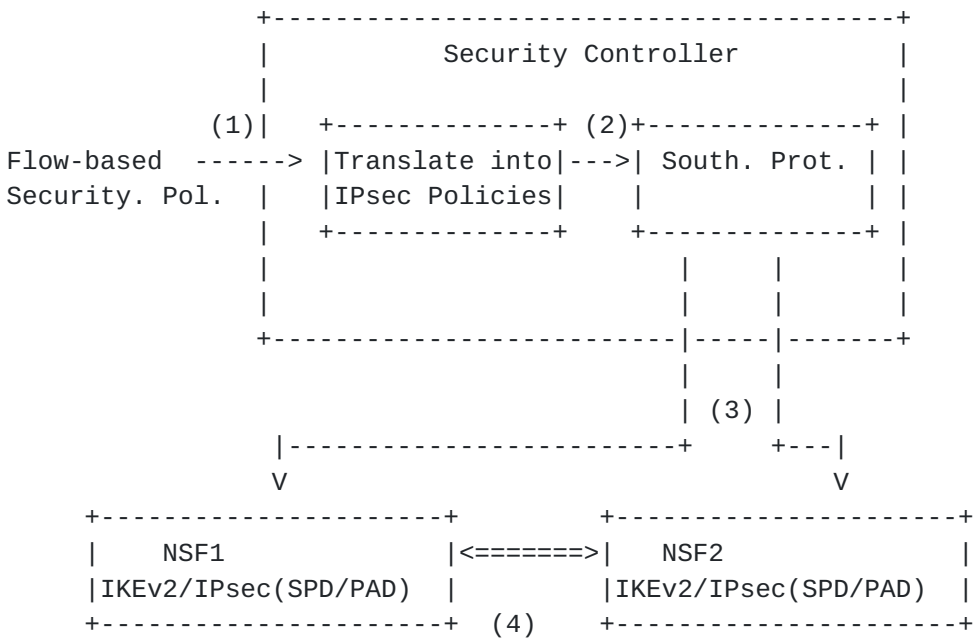

Figure 3: Host-to-host / gateway-to-gateway single controller flow for the IKE case.

Figure 3 describes the case IKE case:

1. The administrator defines general flow-based security policies. The Security Controller looks for the NSFs involved (NSF1 and NSF2).

2. The Security Controller generates IKEv2 credentials for them and translates the policies into SPD and PAD entries.

3. The Security Controller inserts the SPD and PAD entries in both NSF1 and NSF2.

4. The flow is protected with the IPsec SA established with IKEv2.

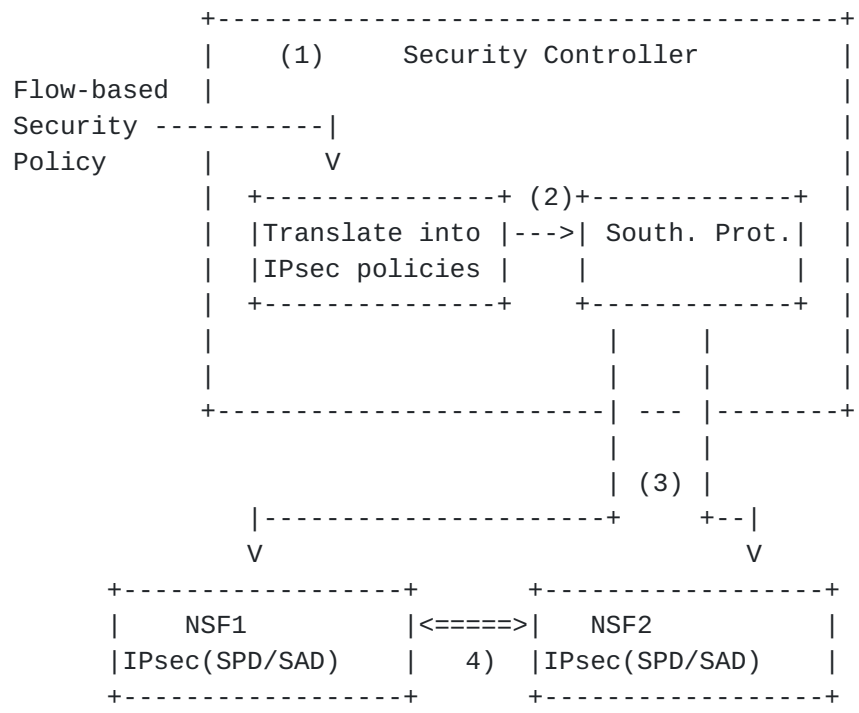

Figure 4: Host-to-host / gateway-to-gateway single controller flow for IKE-less case.

In IKE-less case, flow-based security policies defined by the administrator are translated into IPsec SPD entries and inserted into the corresponding NSFs. Besides, fresh SAD entries will be also generated by the Security Controller and enforced in the NSFs. In this case, the controller does not run any IKEv2 implementation, and it provides the cryptographic material for the IPsec SAs. These keys will be also distributed securely through the southbound interface. Note that this is possible because both NSFs are managed by the same controller.

Figure 4 describes the IKE-less, when a data packet needs to be protected in the path between the NSF1 and NSF2:

1. The administrator establishes the flow-based security policies. The Security Controller looks for the involved NSFs.

2. The Security Controller translates the flow-based security policies into IPsec SPD and SAD entries.

3. The Security Controller inserts the these entries in both NSF1 and NSF2 IPsec databases. It associates a lifetime to the IPsec SAs. When this lifetime expires, the NSF will send a sadb_expire notification to the Security Controller in order to start the rekeying process.

4. The flow is protected with the IPsec SA established by the Security Controller.

Both NSFs could be two hosts that exchange traffic and require to establish an end-to-end security association to protect their communications (host-to-host) or two gateways (gateway-to-gateway), for example, within an enterprise that needs to protect the traffic between, for example, the networks of two branch offices.

Applicability of these configurations appear in current and new networking scenarios. For example, SD-WAN technologies are providing dynamic and on-demand VPN connections between branch offices, or between branches and SaaS cloud services. Beside, IaaS services providing virtualization environments are deployments solutions based on IPsec to provide secure channels between virtual instances (host-to-host) and providing VPN solutions for virtualized networks (gateway-to-gateway).

In general (for IKE and IKE-less case), this system has various advantages:

1. It allows to create IPsec SAs among two NSFs, with only the application of more general flow-based security policies at the application layer. Thus, administrators can manage all security associations in a centralized point with an abstracted view of the network.

2. All NSFs deployed after the application of the new policies are NOT manually configured, therefore allowing its deployment in an automated manner.

7.2. Host-to-host or gateway-to-gateway under different security controllers

It is also possible that two NSFs (i.e. NSF1 and NSF2) are under the control of two different Security Controllers. This may happen, for example, when two organizations, namely Enterprise A and Enterprise B, have their headquarters interconnected through a WAN connection and they both have deployed a SDN-based architecture to provide connectivity to all their clients.

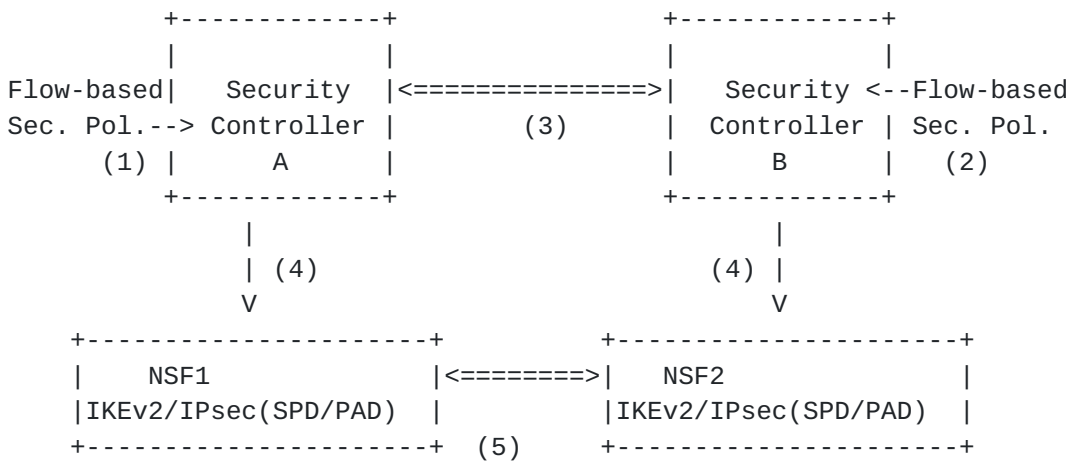

Figure 5: Different security controllers in IKE case

Figure 5 describes IKE case when two security controllers are involved in the process.

1. The A's administrator establishes general Flow-based Security Policies in Security Controller A.

2. The B's administrator establishes general Flow-based Security Policies in Security Controller B.

3. The Security Controller A realizes that protection is required between the NSF1 and NSF2, but the NSF2 is under the control of another Security Controller (Security Controller B), so it starts negotiations with the other controller to agree on the IPsec SPD policies and IKEv2 credentials for their respective NSFs. NOTE: This may require extensions in the East/West interface.

4. Then, both Security Controllers enforce the IKEv2 credentials and related parameters and the SPD and PAD entries in their respective NSFs.

5. The flow is protected with the IPsec SAs established with IKEv2 between both NSFs.

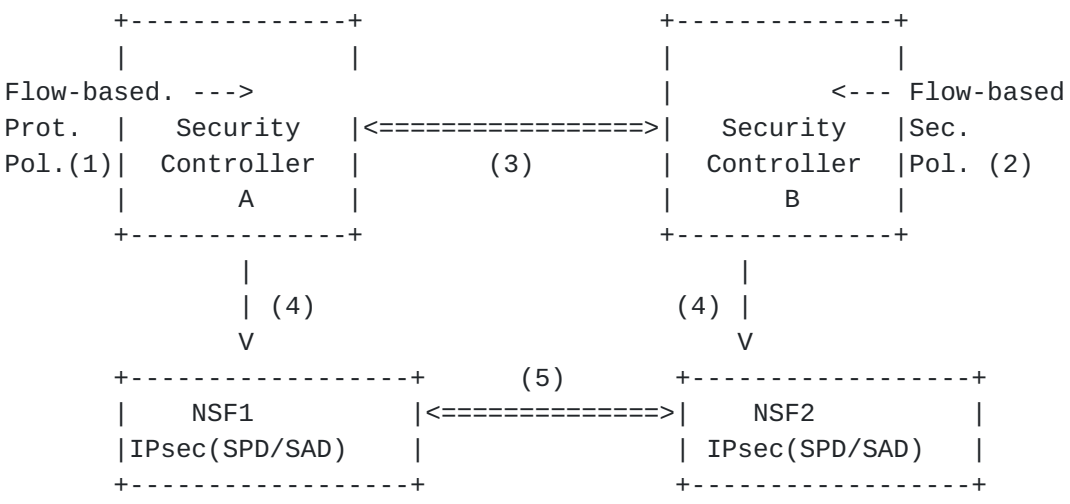

Figure 6: Different security controllers in IKE-less case

Figure 5 describes IKE-less case when two security controllers are involved in the process.

1. The A's administrator establishes general Flow Protection Policies in Security Controller A.

2. The B's administrator establishes general Flow Protection Policies in Security Controller B.

3. The Security Controller A realizes that the flow between NSF1 and NSF2 MUST be protected. Nevertheless, the controller notices that NSF2 is under the control of another Security Controller, so it starts negotiations with the other controller to agree on the IPsec SPD and SAD entries that define the IPsec SAs. NOTE: It would worth evaluating IKEv2 as the protocol for the East/West interface in this case.

4. Once the Security Controllers have agreed on key material and the details of the IPsec SAs, they both enforce this information into their respective NSFs.

5. The flow is protected with the IPsec SAs established by both Security Controllers in their respective NSFs.

8. Security Considerations

First of all, this document shares all the security issues of SDN that are specified in the "Security Considerations" section of [ITU-T.Y.3300] and [RFC8192]. On the one hand, it is important to note that there MUST exit a security association between the Security Controller and the NSFs to protect of the critical information

(cryptographic keys, configuration parameter, etc...) exchanged between these entities. For example, if NETCONF is used as southbound protocol between the Security Controller and the NSFs, it is defined that TLS or SSH security association MUST be established between both entities. On the other hand, we have divided this section in two parts to analyze different security considerations for both cases: NSF with IKEv2 (IKE case) and NSF without IKEv2 (IKE-less case). In general, the Security Controller, as typically in the SDN paradigm, is a target for different type of attacks. As a consequence, the Security Controller is a key entity in the infrastructure and MUST be protected accordingly. In particular, according to this document, the Security Controller will handle cryptographic material so that the attacker may try to access this information. Although, we can assume this attack will not likely to happen due to the assumed security measurements to protect the Security Controller, it deserves some analysis in the hypothetical the attack occurs. The impact is different depending on the IKE case or IKE-less case.

8.1. IKE case

In IKE case, the Security Controller sends IKE credentials (PSK, public/private keys, certificates, etc...) to the NSFs using the security association between Security Controller and NSFs. The general recommendation is that the Security Controller SHOULD NEVER store the IKE credentials after distributing them. Moreover the NSFs MUST NOT allow the reading of these values once they have been applied by the Security Controller (i.e. write only operations). One option is return always the same value (all 0s). If the attacker has access to the Security Controller during the period of time that key material is generated, it may access to these values. Since these values are used during NSF authentication in IKEv2, it may impersonate the affected NSFs. Several recommendations are important. If PSK authentication is used in IKEv2, the Security Controller SHOULD remove the PSK immediately after generating and distributing it. Moreover, the PSK MUST have a proper length (e.g. minimu, 128 bit length) and strength. If raw public keys are used, the Security Controller SHOULD remove the associated private key immediately after generating and distributing them to the NSFs. If certificates are used, the NSF may generate the private key and exports the public key for certification to the Security Controller.

8.2. IKE-less case

In the IKE-less case, the controller sends the IPsec SA information to the SAD that includes the keys for integrity and encryption (when ESP is used). That key material are symmetric keys to protect data traffic. The general recommendation is that the Security Controller

SHOULD NEVER stores the keys after distributing them. Moreover, the NSFs MUST NOT allow the reading of these values once they have been applied by the Security Controller (i.e. write only operations). Nevertheless, if the attacker has access to the Security Controller during the period of time that key material is generated, it may access to these values. In other words, it may have access to the key material used in the distributed IPsec SAs and observe the traffic between peers. In any case, some escenarios with special secure environments (e.g. physically isolated data centers) make this type of attack difficult. Moreover, some scenarios such as IoT networks with constrained devices, where reducing implementation and computation overhead is important, can apply IKE-less case as a tradeoff between security and low overhead at the constrained device, at the cost of assuming the security impact described above.

9. Acknowledgements

Authors want to thank Paul Wouters, Sowmini Varadhan, David Carrel, Yoav Nir, Tero Kivinen, Graham Bartlett, Sandeep Kampati, Linda Dunbar, Carlos J. Bernardos, Alejandro Perez-Mendez, Alejandro Abad-Carrascosa, Ignacio Martinez and Ruben Ricart for their valuable comments.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <https://www.rfc-editor.org/info/rfc7296>.

[RFC8192] Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "Interface to Network Security Functions (I2NSF): Problem Statement and Use Cases", RFC 8192, DOI 10.17487/RFC8192, July 2017, <https://www.rfc-editor.org/info/rfc8192>.

[RFC8329] Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", RFC 8329, DOI 10.17487/RFC8329, February 2018, <https://www.rfc-editor.org/info/rfc8329>.

10.2. Informative References

[I-D.carrel-ipsecme-controller-ike]
Carrel, D. and B. Weiss, "IPsec Key Exchange using a Controller", draft-carrel-ipsecme-controller-ike-01 (work in progress), March 2019.

[I-D.ietf-i2nsf-framework]
Lopez, D., Lopez, E., Dunbar, L., Strassner, J., and R. Kumar, "Framework for Interface to Network Security Functions", draft-ietf-i2nsf-framework-10 (work in progress), November 2017.

[I-D.ietf-i2nsf-problem-and-use-cases]
Hares, S., Lopez, D., Zarny, M., Jacquenet, C., Kumar, R., and J. Jeong, "I2NSF Problem Statement and Use cases", draft-ietf-i2nsf-problem-and-use-cases-16 (work in progress), May 2017.

[I-D.ietf-i2nsf-terminology]
Hares, S., Strassner, J., Lopez, D., Xia, L., and H. Birkholz, "Interface to Network Security Functions (I2NSF) Terminology", draft-ietf-i2nsf-terminology-07 (work in progress), January 2019.

[I-D.ietf-opsawg-nat-yang]
Boucadair, M., Sivakumar, S., Jacquenet, C., Vinapamula, S., and Q. Wu, "A YANG Module for Network Address Translation (NAT) and Network Prefix Translation (NPT)", draft-ietf-opsawg-nat-yang-17 (work in progress), September 2018.

[I-D.jeong-i2nsf-sdn-security-services-05]
Jeong, J., Kim, H., Park, J., Ahn, T., and S. Lee, "Software-Defined Networking Based Security Services using Interface to Network Security Functions", draft-jeong-i2nsf-sdn-security-services-05 (work in progress), July 2016.

[I-D.pfkey-spd]
Sakane, S., "PF_KEY Extensions for IPsec Policy Management in KAME Stack", October 2002.

[I-D.tran-ipsecme-yang]
Tran, K., Wang, H., Nagaraj, V., and X. Chen, "Yang Data Model for Internet Protocol Security (IPsec)", draft-tran-ipsecme-yang-01 (work in progress), June 2015.

[ITU-T.X.1252]
"Baseline Identity Management Terms and Definitions", April 2010.

[ITU-T.X.800]
"Security Architecture for Open Systems Interconnection for CCITT Applications", March 1991.

[ITU-T.Y.3300]
"Recommendation ITU-T Y.3300", June 2014.

[netconf-vpn]
Stefan Wallin, "Tutorial: NETCONF and YANG", January 2014.

[netopeer]
CESNET, CESNET., "NETCONF toolset Netopeer", November 2016.

[ONF-OpenFlow]
ONF, "OpenFlow Switch Specification (Version 1.4.0)", October 2013.

[ONF-SDN-Architecture]
"SDN Architecture", June 2014.

[RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, DOI 10.17487/RFC2367, July 1998, <https://www.rfc-editor.org/info/rfc2367>.

[RFC3549] Salim, J., Khosravi, H., Kleen, A., and A. Kuznetsov, "Linux Netlink as an IP Services Protocol", RFC 3549, DOI 10.17487/RFC3549, July 2003, <https://www.rfc-editor.org/info/rfc3549>.

[RFC3948] Huttunen, A., Swander, B., Volpe, V., DiBurro, L., and M. Stenberg, "UDP Encapsulation of IPsec ESP Packets", RFC 3948, DOI 10.17487/RFC3948, January 2005, <https://www.rfc-editor.org/info/rfc3948>.

[RFC6071] Frankel, S. and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap", RFC 6071, DOI 10.17487/RFC6071, February 2011, <https://www.rfc-editor.org/info/rfc6071>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <https://www.rfc-editor.org/info/rfc7149>.

[RFC7317] Bierman, A. and M. Bjorklund, "A YANG Data Model for System Management", RFC 7317, DOI 10.17487/RFC7317, August 2014, <https://www.rfc-editor.org/info/rfc7317>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <https://www.rfc-editor.org/info/rfc7426>.

[RFC8229] Pauly, T., Touati, S., and R. Mantha, "TCP Encapsulation of IKE and IPsec Packets", RFC 8229, DOI 10.17487/RFC8229, August 2017, <https://www.rfc-editor.org/info/rfc8229>.

[strongswan] CESNET, CESNET., "StrongSwan: the OpenSource IPsec-based VPN Solution", April 2017.

Appendix A. Appendix A: Common YANG model for IKE and IKEless cases

```
        <CODE BEGINS> file "ietf-ipsec-common@2019-03-11.yang"

        module ietf-ipsec-common{
                yang-version 1.1;
                namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec-common";
                prefix "ipsec-common";

                import ietf-inet-types { prefix inet; }
                import ietf-yang-types { prefix yang; }

                import ietf-crypto-types {
                        prefix ct;
                        reference "draft-ietf-netconf-crypto-types-01: Common
YANG Dta Types for Cryptography";
                }

                organization "IETF I2NSF (Interface to Network Security
Functions) Working Group";

                contact
                " Rafael Marin Lopez
                Dept. Information and Communications Engineering (DIIC)
                Faculty of Computer Science-University of Murcia
                30100 Murcia - Spain
                Telf: +34868888501
                e-mail: rafa@um.es

                Gabriel Lopez Millan
                Dept. Information and Communications Engineering (DIIC)
                Faculty of Computer Science-University of Murcia
                30100 Murcia - Spain
                Tel: +34 868888504
                email: gabilm@um.es

                Fernando Pereniguez Garcia
                Department of Sciences and Informatics
                University Defense Center (CUD), Spanish Air Force Academy,
MDE-UPCT
                30720 San Javier - Spain
                Tel: +34 968189946
                email: fernando.pereniguez@cud.upct.es
                ";

                description "Common Data model for SDN-based IPSec
configuration.";
```

```
                revision "2019-03-11" {
                        description "Revision";
```

```
                        reference "";
                }

                typedef encryption-algorithm-t {
                        type ct:encryption-algorithm-ref;
                        description "typedef";
                }

                typedef integrity-algorithm-t {
                        type ct:mac-algorithm-ref;
                        description
                                "This typedef enables importing modules to
easily define an
                                identityref to the 'asymmetric-key-encryption-
algorithm'
                                base identity.";
                }

                typedef ipsec-mode {
                        type enumeration {
                                enum TRANSPORT { description "Transport mode.
No NAT support."; }
                                enum TUNNEL { description "Tunnel mode"; }
                        }
                        description "Type definition of IPsec mode";
                }

                typedef esp-encap {
                        type enumeration {
                                enum ESPINTCP { description "ESP in TCP
encapulation.";}
                                enum ESPINTLS { description "ESP in TCP
encapsulation using TLS.";}
                                enum ESPINUDP { description "ESP in UDP
encapsulation. RFC 3948 ";}
                                enum NONE { description "NOT ESP
encapsulation" ; }
                        }
                        description "type defining types of ESP encapsulation";
                }

                grouping encap { /* This is defined by XFRM */
                        description "Encapsulation container";
                        leaf espencap { type esp-encap; description "ESP in
TCP, ESP in UDP or ESP in TLS";}
                        leaf sport {type inet:port-number; description
"Encapsulation source port";}
                        leaf dport {type inet:port-number; description
```

```
"Encapsulation destination port"; }
                        leaf-list oaddr {type inet:ip-address; description
"Encapsulation Original Address ";}
                 }

                 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 "0..max"; }
                        description "SPI";
                }

                typedef lifetime-action {
                        type enumeration {
                                enum terminate-clear {description "Terminate
the IPsec SA and allow the packets through";}
                                enum terminate-hold {description "Terminate the
IPsec SA and drop the packets";}
                                enum replace  {description "Replace the IPsec
SA with a new one";}
                        }
                        description "Action when lifetime expiration";
                }

                /*################### SPD basic groupings ####################*/

                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";
                }

                typedef ipsec-upper-layer-proto {
                        type enumeration {
                                enum TCP { description "TCP traffic"; }
                                enum UDP { description "UDP traffic"; }
                                enum SCTP { description "SCTP traffic";}
```

```
                                enum DCCP { description "DCCP traffic";}
                                enum ICMP { description "ICMP traffic";}
                                enum IPv6-ICMP { description "IPv6-ICMP
traffic";}
                                enum GRE {description "GRE traffic";}
                            }
                            description "Next layer proto on top of IP";
                        }
```

```
                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";
                }

                grouping lifetime {
                        description "lifetime current state data";
                        leaf time {type yang:timestamp; default 0; description
"Time since the element is added";}
                        leaf idle {type yang:timestamp; default 0; description
"Time the element is in idle state";}
                        leaf bytes { type uint32; default 0; description
"Lifetime in bytes number";}
                        leaf packets {type uint32; default 0; description
"Lifetime in packets number";}
                }

                /*################### SAD and SPD common basic groupings
####################*/

                grouping port-range  {
                        description "Port range grouping";
                        leaf start { type inet:port-number; description "Start
Port Number"; }
                        leaf end { type inet:port-number; description "End Port
Number"; }
                }

                grouping tunnel-grouping {
                        description "Tunnel mode grouping";
                        leaf local{ type inet:ip-address; description "Local
tunnel endpoint"; }
                        leaf remote{ type inet:ip-address; description "Remote
tunnel enpoint"; }
                        leaf bypass-df { type boolean; description "Bypass DF
bit"; }
                        leaf bypass-dscp { type boolean; description "Bypass
DSCP"; }
                        leaf dscp-mapping { type yang:hex-string; description
"DSCP mapping"; }
```

```
                        leaf ecn { type boolean; description "Bit ECN"; } /*
RFC 4301 ASN1 notation. Annex C*/
                }

                grouping selector-grouping {
                        description "Traffic selector grouping";

                        leaf local-subnet { type inet:ip-prefix; description
"Local IP address subnet"; }
                        leaf remote-subnet { type inet:ip-prefix; description
"Remote IP address subnet"; }

                        leaf-list upper-layer-protocol { type ipsec-upper-
layer-proto; description "List of Upper Layer Protocol";}

                        list local-ports {
                                key "start end";
                                uses port-range;
                                description "List of local ports. When the
upper-layer-protocol is ICMP this 16 bit value respresents code and type as
mentioned in RFC 4301";
```

```
                        }

                        list remote-ports {
                                key "start end";
                                uses port-range;
                                description "List of remote ports. When the
upper-layer-protocol is ICMP this 16 bit value respresents code and type as
mentioned in RFC 4301";
                        }
                }

                /*################### SPD ipsec-policy-grouping
####################*/

                grouping ipsec-policy-grouping {

                        description "Holds configuration information for an
IPSec SPD entry.";

                        leaf spd-entry-id { type uint64; description "SPD entry
id "; }
                        leaf priority {type uint32; default 0; description
"Policy priority";}
                        leaf anti-replay-window { type uint16 { range "0 |
32..1024"; } description "Anti replay window size"; }

                        list names {
                                key "name";
                                leaf name-type { type ipsec-spd-name;
description "SPD name type."; }
                                leaf name { type string; description "Policy
name"; }
                                description "List of policy names";
                        }

                        container condition {
                                description "SPD condition - RFC4301";
                                list traffic-selector-list {
                                        key "ts-number";
                                        leaf ts-number { type uint32;
description "Traffic selector number"; }
                                        leaf direction { type ipsec-traffic-
direction; description "in/out"; }
                                        uses selector-grouping;
                                        ordered-by user;
                                        description "List of traffic
selectors";
                                }
```

```
                        }

                        container processing-info {
                                description "SPD processing - RFC4301";
                                leaf action{ type ipsec-spd-operation;
mandatory true; description "Bypass or discard, container ipsec-sa-cfg is
empty";}

                                container ipsec-sa-cfg {
                                        when "../action = 'PROTECT'";

                                        leaf pfp-flag { type boolean;
description "Each selector has with a pfp flag."; }
                                        leaf extSeqNum { type boolean;
description "TRUE 64 bit counter, FALSE 32 bit"; }
                                        leaf seqOverflow { type boolean;
description "TRUE rekey, FALSE terminare &amp;amp; audit"; }
```

```
                                        leaf statefulfragCheck { type boolean;
description "Indicates whether (TRUE) or not (FALSE) stateful fragment checking
(RFC 4301) applies to the SA to be created."; }
                                        leaf security-protocol { type ipsec-
protocol; description "Security protocol of IPsec SA: Either AH or ESP."; }
                                        leaf mode { type ipsec-mode;
description "transport/tunnel"; }

                                        container ah-algorithms {
                                                when "../security-protocol =
'ah'";
                                                leaf-list ah-algorithm { type
integrity-algorithm-t; description "Configure Authentication Header (AH)."; }
                                                leaf trunc-length { type
uint32; description "Truncation value for AH algorithm"; }
                                                description "AH algoritms ";
                                        }

                                        container esp-algorithms {
                                                when "../security-protocol =
'esp'";
                                                description "Configure
Encapsulating Security Payload (ESP).";
                                                leaf-list authentication { type
integrity-algorithm-t; description "Configure ESP authentication"; }
                                                /* With AEAD algorithms, the
authentication node is not used */
                                                leaf-list encryption { type
encryption-algorithm-t; description "Configure ESP encryption"; }
                                                leaf tfc_pad { type uint32;
default 0; description "TFC padding for ESP encryption"; }
                                        }

                                        container tunnel {
                                                when "../mode = 'TUNNEL'";
                                                uses tunnel-grouping;
                                                description "tunnel grouping
container";
                                        }

                                        description " IPSec SA configuration
container";
                                }
                        }

                        container spd-lifetime-soft {
                                description "SPD lifetime hard state data";
                                uses lifetime;
```

```
                                leaf action {type lifetime-action; description
"Action lifetime";}
                        }

                        container spd-lifetime-hard {
                                description "SPD lifetime hard state data. The
action after the lifetime is to remove the SPD entry.";
                                uses lifetime;
                        }

                        // State data for an IPsec SPD entry
                        container spd-lifetime-current {
                                uses lifetime;
                                config false;
                                description "SPD lifetime current state data";
                        }
                } /* grouping ipsec-policy-grouping */
```

```
        }
    <CODE ENDS>
```

Appendix B. Appendix B: YANG model for IKE case

```
        <CODE BEGINS> file "ietf-ipsec-ike@2019-03-11.yang"

        module ietf-ipsec-ike {
                yang-version 1.1;
                namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec-ike";
                prefix "ipsec-ike";

                import ietf-inet-types { prefix inet; }
                import ietf-yang-types { prefix yang; }

                import ietf-crypto-types {
                        prefix ct;
                        reference "draft-ietf-netconf-crypto-types-01: Common
YANG Data Types for Cryptography";
                }

                import ietf-ipsec-common {
                        prefix ic;
                        reference "Common Data model for SDN-based IPSec
configuration";
                }

                organization "IETF I2NSF (Interface to Network Security
Functions) Working Group";

                contact
                " Rafael Marin Lopez
                Dept. Information and Communications Engineering (DIIC)
                Faculty of Computer Science-University of Murcia
                30100 Murcia - Spain
                Telf: +34868888501
                e-mail: rafa@um.es

                Gabriel Lopez Millan
                Dept. Information and Communications Engineering (DIIC)
                Faculty of Computer Science-University of Murcia
                30100 Murcia - Spain
                Tel: +34 868888504
                email: gabilm@um.es
```

Department of Sciences and Informatics
University Defense Center (CUD), Spanish Air Force Academy, MDE-UPCT
30720 San Javier - Spain
Tel: +34 968189946
email: fernando.pereniguez@cud.upct.es
";

```
description "Data model for IKE case.";

revision "2019-03-11" {
        description "Revision 1.1";
        reference "";
}

typedef type-autostartup {
        type enumeration {
                enum ADD {description "IPsec configuration is
only loaded but not started.";}
                enum ON-DEMAND {description "IPsec
configuration is loaded and transferred to the NSF's kernel";}
                enum START { description "IPsec configuration
is loaded and transferred to the NSF's kernel, and the IKEv2 based IPsec SAs
are established";}
        }
        description "Different policies of when to start an
IKEv2 based IPsec SA";
}

typedef auth-protocol-type {
        type enumeration {
                enum IKEv2 { description "Authentication
protocol based on IKEv2"; }
        }
        description "IKE authentication protocol version";
}

typedef pfs-group {
        type enumeration {
                enum NONE {description "NONE";}
                enum 768-bit-MODP {description "768-bit MODP
Group";}
                enum 1024-bit-MODP {description "1024-bit MODP
Group";}
                enum 1536-bit-MODP {description "1536-bit MODP
Group";}
                enum 2048-bit-MODP {description "2048-bit MODP
Group";}
```

```
                                enum 3072-bit-MODP {description "3072-bit MODP
Group";}
                                enum 4096-bit-MODP {description "4096-bit MODP
Group";}
                                enum 6144-bit-MODP {description "6144-bit MODP
Group";}
                                enum 8192-bit-MODP {description "8192-bit MODP
Group";}
                        }
                        description "PFS group for IPsec rekey";
                }

                /*################### PAD  ####################*/

                typedef auth-method-type {
```

```
                        /* Most implementations also provide XAUTH protocol,
others used are: BLISS, P12, NTLM, PIN */
                        type enumeration {
                                enum pre-shared { description "Select pre-
shared key message as the authentication method"; }
                                enum eap { description "Select EAP as the
authentication method"; }
                                enum digital-signature { description "Select
digital signature method";}
                                enum null {description "null authentication";}
                        }
                        description "Peer authentication method";
                }

                typedef signature-algorithm-t {
                        type ct:signature-algorithm-ref; // We must reference
to "signature-algorithm-ref" but we temporary use hash-algorithm-ref
                        description "This typedef enables referencing to any
digital signature algorithm";
                }

                grouping auth-method-grouping {
                        description "Peer authentication method data";

                        container auth-method {
                                description "Peer authentication method
container";

                                leaf auth-m { type auth-method-type;
description "Type of authentication method (pre-shared, eap, digital signature,
null)"; }

                                container eap-method {
                                        when "../auth-m = 'eap'";
                                        leaf eap-type { type uint8; description
"EAP method type"; }
                                        description "EAP method description
used when auth method is eap";
                                }

                                container pre-shared {
                                        when "../auth-m[.='pre-shared'
or .='eap']";
                                        leaf secret { type yang:hex-string;
description "Pre-shared secret value";}
                                        description "Shared secret value";
                                }
```

```
                                container digital-signature {
                                        when "../auth-m[.='digital-signature'
or .='eap']";
                                        leaf ds-algorithm {type signature-
algorithm-t; description "Name of the digital signature algorithm";}
                                        leaf raw-public-key {type yang:hex-
string; description "RSA raw public key" ;}
                                        leaf key-data { type string;
description "RSA private key data - PEM"; }
                                        leaf key-file { type string;
description "RSA private key file name "; }
                                        leaf-list ca-data { type string;
description "List of trusted CA certs - PEM"; }
                                        leaf ca-file { type string; description
"List of trusted CA certs file"; }
                                        leaf cert-data { type string;
description "X.509 certificate data - PEM4"; }
                                        leaf cert-file { type string;
description "X.509 certificate file"; }
                                        leaf crl-data { type string;
description "X.509 CRL certificate data in base64"; }
                                        leaf crl-file { type string;
description " X.509 CRL certificate file"; }
                                        leaf oscp-uri { type inet:uri;
description "OCSP URI";}
```

```
                                        description "RSA signature container";
                                }
                        }
                }

                grouping identity-grouping {
                        description "Identification type. It is an union
identity";
                        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."; }
                                leaf id_key { type string; description "Key
id"; }
                                leaf id_null { type empty; description "RFC
7619" ; }
                                leaf user_fqdn { type string; description "User
FQDN"; }
                        }
                        leaf my-identifier { type string; mandatory true;
description "id used for authentication"; }
                }

                /*################# end PAD ##################*/


                /*################## IKEv2-grouping ##################*/
                grouping ike-proposal {
                        description "IKEv2 proposal grouping";

                        container ike-sa-lifetime-hard {
                                description "IKE SA lifetime hard";
                                uses ic:lifetime;
                        }
```

```
                        container ike-sa-lifetime-soft {
                                description "IPsec SA lifetime soft";
                                uses ic:lifetime;
                                leaf action {type ic:lifetime-action;
description "Action lifetime";}
                        }

                        leaf-list ike-sa-authalg { type ic:integrity-algorithm-
t; description "Auth algorigthm for IKE SA";}
                        leaf-list ike-sa-encalg { type ic:encryption-algorithm-
t; description "Auth algorigthm for IKE SAs";}
                        leaf dh_group { type uint32; mandatory true;
description "Group number for Diffie Hellman Exponentiation";}
                        leaf half-open-ike-sa-timer { type uint32; description
"Set the half-open IKE SA timeout duration" ; }
                        leaf half-open-ike-sa-cookie-threshold { type uint32;
description "Number of half-open IKE SAs that activate the cookie
mechanism." ; }
                }

                grouping ike-child-sa-info {
                        description "IPsec SA Information";
```

```
                        leaf-list pfs_groups { type pfs-group; description "If
non-zero, require perfect forward secrecy when requesting new SA. The non-zero
value is the required group number"; }

                        container child-sa-lifetime-soft {
                                description "IPsec SA lifetime soft";
                                uses ic:lifetime;
                                leaf action {type ic:lifetime-action;
description "action lifetime";}
                        }

                        container child-sa-lifetime-hard {
                                description "IPsec SA lifetime hard. The action
will be to terminate the IPsec SA.";
                                uses ic:lifetime;
                        }
                }

                /*################### End IKEv2-grouping ###################*/


                container ikev2 {

                        description "Configure the IKEv2 software";

                        container pad {
                                description "Configure Peer Authorization
Database (PAD)";
                                list pad-entry {
                                        key "pad-entry-id";
                                        ordered-by user;
                                        description "Peer Authorization
Database (PAD)";
                                        leaf pad-entry-id { type uint64;
description "SAD index. ";}
                                        uses identity-grouping;
                                        leaf pad-auth-protocol { type auth-
protocol-type; description "IKEv2, etc. ";}
                                        uses auth-method-grouping;
                                }
                        }

                        list ike-conn-entry {
                                key "conn-name";
                                description "IKE peer connection information";
                                leaf conn-name  { type string; mandatory true;
description "Name of IKE connection";}
                                leaf autostartup { type type-autostartup;
mandatory true; description "if True: automatically start tunnel at startup;
```

```
else we do lazy tunnel setup based on trigger from datapath";}
                                leaf initial-contact {type boolean; default
false; description "This IKE SA is the only currently active between the
authenticated identities";}
                                leaf version {
                                        type enumeration {
                                                enum ikev2 {value 2;
description "IKE version 2";}
                                        }
                                        description "IKE version";
                                }

                                leaf ike-fragmentation { type boolean;
description "Whether to use IKEv2 fragmentation as per RFC 7383 (TRUE or
FALSE)"; }
```

```
                                uses ike-proposal;

                                container local {
                                        description "Local peer connection
information";
                                        leaf local-pad-id { type uint64;
description " ";}
                                }

                                container remote {
                                        description "Remote peer connection
information";
                                        leaf remote-pad-id { type uint64;
description " ";}
                                }

                                uses ic:encap;

                                container spd {
                                        description "Configure the Security
Policy Database (SPD)";
                                        list spd-entry {
                                                key "spd-entry-id";
                                                uses ic:ipsec-policy-grouping;
                                                ordered-by user;
                                                description "List of SPD
entries";
                                        }
                                }

                                container ike-sa-state {
                                        container uptime {
                                                description "IKE service
uptime";
                                                leaf running { type yang:date-
and-time;  description "Relative uptime";}
                                                leaf since   { type yang:date-
and-time;  description "Absolute uptime";}
                                        }

                                        leaf initiator { type boolean;
description "It is acting as initiator in this connection";}
                                        leaf initiator-ikesa-spi {type uint64;
description "Initiator's IKE SA SPI";}
                                        leaf responder-ikesa-spi {type uint64;
description "Responsder's IKE SA SPI";}
                                        leaf nat-local {type boolean;
description "YES, if local endpoint is behind a NAT";}
```

```
                                        leaf nat-remote {type boolean;
description "YES, if remote endpoint is behind a NAT";}
                                        leaf nat-any {type boolean; description
"YES, if both local and remote endpoints are behind a NAT";}

                                        uses ic:encap;

                                        leaf established {type uint64;
description "Seconds the IKE SA has been established";}
                                        leaf rekey-time {type uint64;
description "Seconds before IKE SA gets rekeyed";}
                                        leaf reauth-time {type uint64;
description "Seconds before IKE SA gets re-authenticated";}
                                        list child-sas {
                                                container spis{
                                                        description "IPsec SA's
SPI '";
                                                        leaf spi-in {type
ic:ipsec-spi;  description "Security Parameter Index for inbound IPsec SA";}
                                                        leaf spi-out {type
ic:ipsec-spi;  description "Security Parameter Index for the corresponding
outbound IPsec SA";}
```

```
                                                }
                                                description "State data about
IKE CHILD SAs";
                                        }
                                        config false;
                                        description "IKE state data";
                                } /* ike-sa-state */
                        } /* ike-conn-entries */


                        container number-ike-sas{
                                leaf total {type uint32; description "Total
number of IKEv2 SAs";}
                                leaf half-open {type uint32; description
"Number of half-open IKEv2 SAs";}
                                leaf half-open-cookies {type uint32;
description "Number of half open IKE SAs with cookie activated" ;}
                                config false;
                                description "Number of IKE SAs";
                        }
                }  /* container ikev2 */
        }

        <CODE ENDS>
```

Appendix C. Appendix C: YANG model for IKE-less case

```
        <CODE BEGINS> file "ietf-ipsec-ikeless@2019-03-11.yang"

        module ietf-ipsec-ikeless {

                yang-version 1.1;
                namespace "urn:ietf:params:xml:ns:yang:ietf-ipsec-ikeless";

                prefix "ipsec-ikeless";

                import ietf-yang-types { prefix yang; }

                import ietf-ipsec-common {
                        prefix ic;
                        reference "Common Data model for SDN-based IPSec
configuration";
                }
```

```
                organization "IETF I2NSF (Interface to Network Security
Functions) Working Group";

                contact
```

" Rafael Marin Lopez
Dept. Information and Communications Engineering (DIIC)
Faculty of Computer Science-University of Murcia
30100 Murcia - Spain
Telf: +34868888501
e-mail: rafa@um.es

Gabriel Lopez Millan
Dept. Information and Communications Engineering (DIIC)
Faculty of Computer Science-University of Murcia
30100 Murcia - Spain
Tel: +34 868888504
email: gabilm@um.es

Fernando Pereniguez Garcia
Department of Sciences and Informatics
University Defense Center (CUD), Spanish Air Force Academy, MDE-UPCT
30720 San Javier - Spain
Tel: +34 968189946
email: fernando.pereniguez@cud.upct.es
";

```
description "Data model for IKE-less case";

revision "2019-03-11" {
        description "Revision";
        reference "";
}

/*################### SAD grouping ####################*/
grouping ipsec-sa-grouping {
        description "Configure Security Association (SA). Section 4.4.2.1 in RFC 4301";

                leaf sad-entry-id {type uint64; description "This value identifies a specific entry in the SAD";}
                leaf spi { type ic:ipsec-spi;  description "Security Parameter Index. This may not be unique for a particular SA";}
                leaf seq-number { type uint64; description "Current sequence number of IPsec packet."; }
                leaf seq-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-window { type uint16 { range "0 | 32..1024"; } description "Anti replay window size"; }
                leaf spd-entry-id {type uint64; description "This value
```

```
links the SA with the SPD entry";}

                        uses ic:selector-grouping;

                        leaf security-protocol { type ic:ipsec-protocol;
description "Security protocol of IPsec SA: Either AH or ESP."; }

                        container sad-lifetime-hard {
                                description "SAD lifetime hard state data. The
action associated is terminate.";
                                uses ic:lifetime;
                        }
```

```
                        container sad-lifetime-soft {
                                description "SAD lifetime hard state data";
                                uses ic:lifetime;
                                leaf action {type ic:lifetime-action;
description "action lifetime";}
                        }

                        leaf mode { type ic:ipsec-mode; description "SA
Mode"; }
                        leaf statefulfragCheck { type boolean; description
"Indicates whether (TRUE) or not (FALSE) stateful fragment checking (RFC 4301)
applies to this SA."; }

                        leaf dscp { type yang:hex-string; description "DSCP
value"; }
                        leaf path-mtu { type uint16; description "Maximum size
of an IPsec packet that can be transmitted without fragmentation"; }

                        container tunnel {
                                when "../mode = 'TUNNEL'";
                                uses ic:tunnel-grouping;
                                description "Container for tunnel grouping";
                        }

                        uses ic:encap;

                        // STATE DATA for SA
                        container sad-lifetime-current {
                                uses ic:lifetime;
                                config false;
                                description "SAD lifetime current state data";
                        }

                        container stats { // xfrm.h
                                leaf replay-window {type uint32; default 0;
description " "; }
                                leaf replay {type uint32; default 0;
description "packets detected out of the replay window and dropped because they
are replay packets";}
                                leaf failed {type uint32; default 0;
description "packets detected out of the replay window ";}
                                config false;
                                description "SAD statistics";
                        }

                        container replay_state { // xfrm.h
                                leaf seq {type uint32; default 0; description
"input traffic sequence number when anti-replay-window != 0";}
```

```
                                leaf oseq {type uint32; default 0; description
"output traffic sequence number";}
                                leaf bitmap {type uint32; default 0;
description "";}
                                config false;
                                description "Anti-replay Sequence Number
state";
                        }

                        container replay_state_esn { // xfrm.h
                                leaf bmp-len {type uint32; default 0;
description "bitmap length for ESN"; }
                                leaf oseq { type uint32; default 0; description
"output traffic sequence number"; }
                                leaf oseq-hi { type uint32; default 0;
description ""; }
                                leaf seq-hi { type uint32; default 0;
description ""; }
```

```
                                leaf replay-window {type uint32; default 0;
description ""; }
                                leaf-list bmp { type uint32; description
"bitmaps for ESN (depends on bmp-len) "; }
                                config false;
                                description "Anti-replay Extended Sequence
Number (ESN) state";
                        }

                }
                /*################### end SAD grouping ###################*/


                /*################## Register grouping ##################*/
                typedef sadb-msg-type {
                        type enumeration {
                                enum sadb_acquire { description
"SADB_ACQUIRE"; }
                                enum sadb_expire { description "SADB_EXPIRE"; }
                        }
                        description "Notifications (PF_KEY message types) that
must be forwarded by the NSF to the controller in IKE-less case";
                }

                typedef sadb-msg-satype {
                         type enumeration {
                                enum sadb_satype_unspec { description
"SADB_SATYPE_UNSPEC"; }
                                enum sadb_satype_ah { description
"SADB_SATYPE_AH"; }
                                enum sadb_satype_esp { description
"SADB_SATYPE_ESP"; }
                                enum sadb_satype_rsvp { description
"SADB_SATYPE_RSVP"; }
                                enum sadb_satype_ospfv2 { description
"SADB_SATYPE_OSPFv2"; }
                                enum sadb_satype_ripv2 { description
"SADB_SATYPE_RIPv2"; }
                                enum sadb_satype_mip { description
"SADB_SATYPE_MIP"; }
                                enum sadb_satype_max { description
"SADB_SATYPE_MAX"; }
                        }
                        description "PF_KEY Security Association types";
                }

                grouping base-grouping {
                        description "Configuration for the  message header
```

```
format";
                        list base-list {
                                        key "version";
                                        leaf version { type string;
description "Version of PF_KEY (MUST be PF_KEY_V2)"; }
                                        leaf msg_type { type sadb-msg-type;
description "Identifies the type of message"; }
                                        leaf msg_satype { type sadb-msg-
satype; description "Defines the type of Security Association"; }
                                        leaf msg_seq { type uint32;
description "Sequence number of this message."; }
                                        description "Configuration for a
specific message header format";
                        }
                }
                /*################## End Register grouping ##################*/


                /*################### IPsec configuration ###################*/
```

```
                container ietf-ipsec {
                        description "IPsec configuration";

                        container spd {
                                                description "Configure the
Security Policy Database (SPD)";
                                                list spd-entry {
                                                   key "spd-entry-id";
                                                   uses ic:ipsec-policy-
grouping;
                                                   ordered-by user;
                                                   description "List of SPD
entries";
                                                }
                        }

                        container sad {
                                description "Configure the IPSec Security
Association Database (SAD)";

                                list sad-entry {
                                        key "sad-entry-id";

                                        uses ipsec-sa-grouping;

                                        container ah-sa {
                                                when "../security-protocol =
'ah'";
                                                description "Configure
Authentication Header (AH) for SA";
                                                container integrity {
                                                        description "Configure
integrity for IPSec Authentication Header (AH)";
                                                        leaf integrity-
algorithm { type ic:integrity-algorithm-t; description "Configure
Authentication Header (AH)."; }
                                                        leaf key { type string;
description "AH key value";}
                                                }
                                        }

                                        container esp-sa {
                                                when "../security-protocol =
'esp'";
                                                description "Set IPSec
Encapsulation Security Payload (ESP)";

                                                container encryption {
```

```
                                                        description "Configure
encryption for IPSec Encapsulation Secutiry Payload (ESP)";
                                                        leaf encryption-
algorithm { type ic:encryption-algorithm-t; description "Configure ESP
encryption"; }
                                                        leaf key { type
yang:hex-string; description "ESP encryption key value";}
                                                        leaf iv {type yang:hex-
string; description "ESP encryption IV value"; }
                                                }

                                                container integrity {
                                                        description "Configure
authentication for IPSec Encapsulation Secutiry Payload (ESP)";
                                                        leaf integrity-
algorithm { type ic:integrity-algorithm-t; description "Configure
Authentication Header (AH)."; }
                                                        leaf key { type
yang:hex-string; description "ESP integrity key value";}
                                                }
```

```
                                        /* With AEAD algorithms, the
integrity node is not used */

                                        leaf combined-enc-intr { type
boolean; description "ESP combined mode algorithms. The algorithm is specified
in encryption-algorithm";}
                                    }
                                    description "List of SAD entries";
                                }
                        }
                } /* container ietf-ipsec */


                /*################### RPC and Notifications ###################*/

                // These RPCs are needed by a Security Controller in IKEless
case

                notification spdb_expire {
                        description "A SPD entry has expired";
                        leaf index { type uint64; description "SPD index.
RFC4301 does not mention an index however real implementations (e.g. XFRM or
PFKEY_v2 with KAME extensions provide a policy index to refer a policy. "; }
                }

                notification sadb_acquire {
                        description "A IPsec SA is required ";
                        uses base-grouping;
                        uses ic:selector-grouping; // To indicate the concrete
traffic selector of the policy that triggered this acquire.
                }

                notification sadb_expire {
                        description "A IPsec SA expiration (soft or hard)";

                        uses base-grouping;
                        leaf spi { type ic:ipsec-spi;  description "Security
Parameter Index";}
                        leaf anti-replay-window { type uint16 { range "0 |
32..1024"; } description "Anti replay window"; }

                        leaf encryption-algorithm { type ic:encryption-
algorithm-t; description "encryption algorithm of the expired SA"; }
                        leaf authentication-algorithm { type ic:integrity-
algorithm-t; description "authentication algorithm of the expired SA"; }

                        container sad-lifetime-hard {
                                description "SAD lifetime hard state data";
                                uses ic:lifetime;
```

```
                        }
                        container sad-lifetime-soft {
                                description "SAD lifetime soft state data";
                                uses ic:lifetime;
                        }

                        container sad-lifetime-current {
                                description "SAD lifetime current state data";
                                uses ic:lifetime;
                        }
```

```
                }

                notification sadb_bad-spi {
                        description "Notifiy when the NSF receives a packet
with an incorrect SPI (i.e. not present in the SAD)";
                        leaf state { type ic:ipsec-spi; mandatory "true";
description "SPI number contained in the erroneous IPsec packet"; }
                }

        }/*module ietf-ipsec*/

        <CODE ENDS>
```

Authors' Addresses

Rafa Marin-Lopez
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 01
EMail: rafa@um.es

Gabriel Lopez-Millan
University of Murcia
Campus de Espinardo S/N, Faculty of Computer Science
Murcia 30100
Spain

Phone: +34 868 88 85 04
EMail: gabilm@um.es

Fernando Pereniguez-Garcia
University Defense Center
Spanish Air Force Academy, MDE-UPCT
San Javier (Murcia) 30720
Spain

Phone: +34 968 18 99 46
EMail: fernando.pereniguez@cud.upct.es