Software-Defined Networking (SDN)-based IPsec Flow Protection
draft-ietf-i2nsf-sdn-ipsec-flow-protection-10

Abstract

   This document describes how to provide IPsec-based flow protection
   (integrity and confidentiality) by means of an Interface to Network
   Security Function (I2NSF) controller. It considers two main well-
   known scenarios in IPsec: (i) gateway-to-gateway and (ii) host-to-
   host. The service described in this document allows the
   configuration and monitoring of IPsec Security Associations (SAs)
   from a I2NSF Controller to one or several flow-based Network Security
   Functions (NSFs) that rely on IPsec to protect data traffic.

   The document focuses on the I2NSF NSF-facing interface by providing
   YANG data models for configuring the IPsec databases (SPD, SAD, PAD)
   and IKEv2. This allows IPsec SA establishment with minimal
   intervention by the network administrator. It does not define any
   new protocol.

Status of This Memo

Copyright Notice

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. The SDN paradigm relocates the control
   of network resources to a centralized entity, namely SDN Controller.
   SDN controllers configure and manage distributed network resources
   and provide an abstracted view of the network resources to SDN
   applications. SDN applications 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 now demand a centralized way of
   managing different security aspects. For example, Software-Defined
   WANs (SD-WANs). SD-WANs are an SDN extension providing a software
   abstraction to create secure network overlays over traditional WAN
   and branch networks. SD-WANs utilize IPsec [RFC4301] as an
   underlying security protocol. The goal of SD-WANs is to provide
   flexible and automated deployment from a centralized point to enable
   on-demand network security services such as IPsec Security
   Association (IPsec SA) management. Additionally, Section 4.3.3 in
   [RFC8192] describes another example use case for Cloud Data Center
   Scenario titled "Client-Specific Security Policy in Cloud VPNs". The
   use case in RFC 8192 states that "dynamic key management is critical
   for securing the VPN and the distribution of policies". These VPNs
   can be established using IPsec. The management of IPsec SAs in data
   centers using a centralized entity is a scenario where the current
   specification maybe applicable.

   Therefore, with the growth of SDN-based scenarios where network
   resources are deployed in an autonomous manner, a mechanism to manage
   IPsec SAs from a centralized entity becomes more relevant in the
   industry.

   In response to this need, the Interface to Network Security Functions
   (I2NSF) charter states that the goal of this working group is "to
   define set of software interfaces and data models for controlling and
   monitoring aspects of physical and virtual Network Security
   Functions". As defined in [RFC8192] an NSF is "a function that is

   used to ensure integrity, confidentiality, or availability of network
   communication; to detect unwanted network activity; or to block, or
   at least mitigate, the effects of unwanted activity". This document
   pays special attention to flow-based NSFs that ensure integrity and
   confidentiality by means of IPsec.

   In fact, as Section 3.1.9 in [RFC8192] states "there is a need for a
   controller to create, manage, and distribute various keys to
   distributed NSFs.", however "there is a lack of a standard interface
   to provision and manage security associations". Inspired in the SDN
   paradigm, the I2NSF framework [RFC8329] defines a centralized entity,
   the I2NSF Controller, which manages one or multiple NSFs through a
   I2NSF NSF-Facing interface. In this document we define a service
   allowing the I2NSF Controller to carry out the key management
   procedures. More specifically, we define YANG data models for I2NSF
   NSF-Facing interface that allow the I2NSF Controller to configure and
   monitor IPsec-enabled flow-based NSFs.

   IPsec architecture [RFC4301] defines clear separation between the
   processing to provide security services to IP packets and the key
   management procedures to establish the IPsec Security Associations,
   which allows to centralize the key management procedures in the I2NSF
   Controller. This document considers two typical scenarios to
   autonomously manage IPsec SAs: gateway-to-gateway and host-to-host
   [RFC6071]. In these cases, hosts, gateways or both may act as NSFs.
   Consideration for the host-to-gateway scenario is out of scope.

   For the definition of the YANG data model for I2NSF NSF-Facing
   interface, this document considers two general cases, namely:

   1) IKE case. The NSF implements the Internet Key Exchange version 2
       (IKEv2) protocol and the IPsec databases: the Security Policy
       Database (SPD), the Security Association Database (SAD) and the
       Peer Authorization Database (PAD). The I2NSF Controller is in
       charge of provisioning the NSF with the required information in
       the SPD, PAD (e.g. IKE credential) and IKE protocol itself (e.g.
       parameters for the IKE_SA_INIT negotiation).

   2) IKE-less case. The NSF only implements the IPsec databases (no
       IKE implementation). The I2NSF 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 key management functionality is moved to the I2NSF
       Controller.

   In both cases, a data model for the I2NSF NSF-Facing interface is
   required to carry out this provisioning in a secure manner between
   the I2NSF Controller and the NSF. Using YANG data modelling language

version 1.1 [RFC7950] and based on YANG models defined in
[netconf-vpn], [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). The proposed YANG data
model conforms to the Network Management Datastore Architecture
(NMDA) defined in [RFC8342]. Examples of the usage of these models
can be found in Appendix D, Appendix E and Appendix F.

In summary, the objetives of this I-D are:

o To describe the architecture for the I2NSF-based IPsec management,
   which allows the establishment and management of IPsec security
   associations from the I2NSF Controller in order to protect
   specific data flows between two flow-based NSFs implementing
   IPsec.

o To map this architecture to the I2NSF Framework.

o To define the interfaces required to manage and monitor the IPsec
   SAs in the NSF from a I2NSF Controller. YANG data models are
   defined for configuration and state data for IPsec and IKEv2
   management through the I2NSF NSF-Facing interface. Thus, this I-D
   does not define any new protocol.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "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 [RFC8329], [RFC8192],
[RFC4301],[RFC7296], [RFC6241], [ITU-T.Y.3300]. The following term
is defined in [ITU-T.Y.3300]:

o Software-Defined Networking.

The following terms are in defined in [RFC8192]:

o NSF.

o Flow-based NSF.

The following terms are defined in [RFC4301]:

   o Peer Authorization Database (PAD).

   o Security Associations Database (SAD).

   o Security Policy Database (SPD).

   The following term is defined in [RFC6437]:

   o Flow/traffic flow.

   The following terms is defined in [RFC7296]:

   o Internet Key Exchange version 2 (IKEv2).

   The following terms are defined in [RFC6241]:

   o Configuration data.

   o Configuration datastore.

   o State date.

   o Startup configuration datastore.

   o Running configuration datastore.

4. SDN-based IPsec management description

   As mentioned in Section 1, two cases are considered, depending on
   whether the NSF implements IKEv2 or not: IKE case and IKE-less case.

4.1. IKE case: IKEv2/IPsec in the NSF

   In this case, the NSF implements IPsec with IKEv2 support. The I2NSF
   Controller is in charge of managing and applying IPsec connection
   information (determining which nodes need to start an IKEv2/IPsec
   session, identifying the type of traffic to be protected, deriving
   and delivering IKEv2 Credentials such as a pre-shared key,
   certificates, etc.), and applying other IKEv2 configuration
   parameters (e.g. cryptographic algorithms for establishing an IKEv2
   SA) to the NSF necessary for the IKEv2 negotiation.

   With these entries, the IKEv2 implementation can operate to establish
   the IPsec SAs. The I2NSF User establishes the IPsec requirements and
   information about the end points information (through the I2NSF
   Consumer-Facing Interface, [RFC8329]), and the I2NSF Controller
   translates these requirements into IKEv2, SPD and PAD entries that
   will be installed into the NSF (through the I2NSF NSF-Facing

Interface). With that information, the NSF can just run IKEv2 to
establish the required IPsec SA (when the traffic flow needs
protection). Figure 1 shows the different layers and corresponding
functionality.

```
        +---------------------------------------------+
        |           IPsec Management System           | I2NSF User
        +---------------------------------------------+
                           |
                           |   I2NSF Consumer-Facing
                           |   Interface
        +---------------------------------------------+
        | IKEv2 Configuration, PAD and SPD Entries  | I2NSF
        |               Distribution                | Controller
        +---------------------------------------------+
                           |
                           |   I2NSF NSF-Facing
                           |   Interface
        +---------------------------------------------+
        |   IKEv2  |      IPsec(PAD, SPD)        | Network
        |-------------------------------------------| Security
        |     IPsec Data Protection and Forwarding  | Function
        +---------------------------------------------+
```
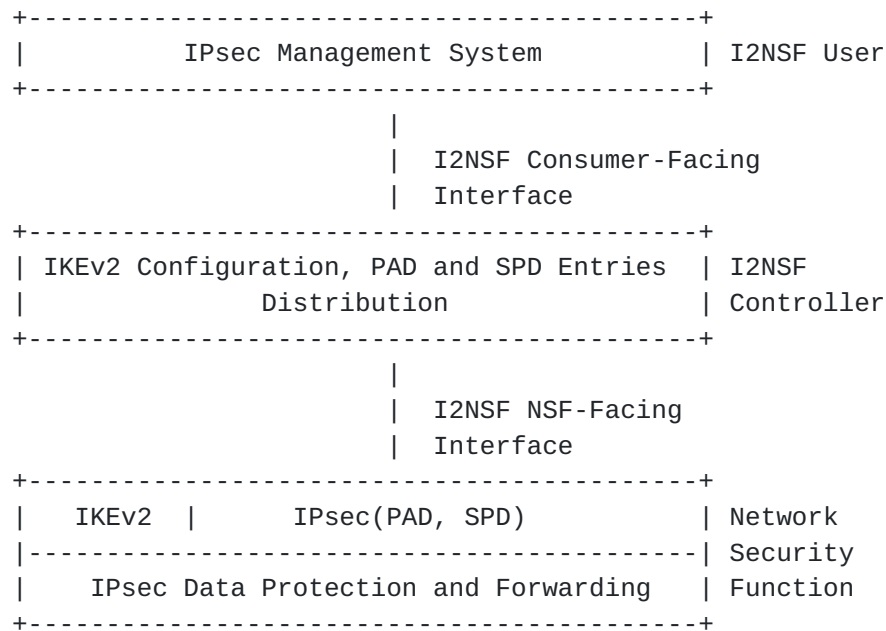
Figure 1: IKE case: IKE/IPsec in the NSF

I2NSF-based IPsec flow protection services provide dynamic and
flexible management of IPsec SAs in flow-based NSFs. In order to
support this capability in the IKE case, a YANG data model for IKEv2,
SPD and PAD configuration data, and for IKEv2 state data MUST be
defined for the I2NSF NSF-Facing Interface.

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

In this case, the NSF does not deploy IKEv2 and, therefore, the I2NSF
Controller has to perform the IKEv2 security functions and management
of IPsec SAs by populating and managing the SPD and the SAD.

```
        +------------------------------------------+
        |         IPsec Management System          | I2NSF User
        +------------------------------------------+
                        |
                        |   I2NSF Consumer-Facing Interface
                        |
        +------------------------------------------+
        |            SPD and SAD Entries           | I2NSF
        |               Distribution               | Controller
        +------------------------------------------+
                        |
                        |   I2NSF NSF-Facing Interface
                        |
        +------------------------------------------+
        |              IPsec (SPD, SAD)            | Network
        |------------------------------------------| Security
        |   IPsec Data Protection and Forwarding   | Function
        +------------------------------------------+
```
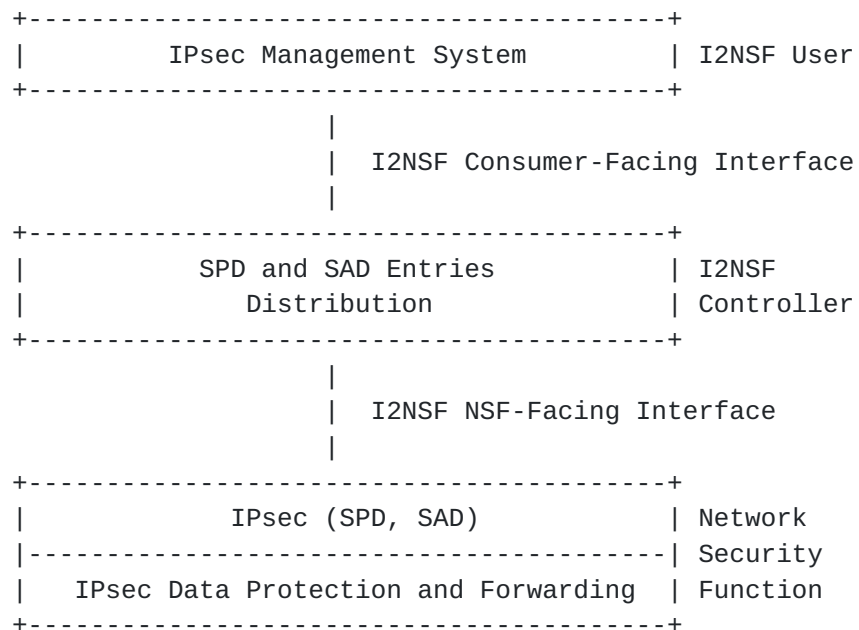
        Figure 2: IKE-less case: IPsec (no IKEv2) in the NSF

   As shown in Figure 2, when an I2NSF User enforces flow-based
   protection policies through the Consumer-Facing Interface, the I2NSF
   Controller translates these 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.

   In order to support the IKE-less case, a YANG data model for SPD and
   SAD configuration data and SAD state data MUST be defined for the
   NSF-Facing Interface.

   Specifically, the IKE-less case assumes that the I2NSF Controller has
   to perform some security functions that IKEv2 typically does, namely
   (non-exhaustive):

   o IV generation.

   o Prevent counter resets for the same key.

   o Generation of pseudo-random cryptographic keys for the IPsec SAs.

   o Generation of the IPsec SAs when required based on notifications
      (i.e. sadb-acquire) from the NSF.

   o Rekey of the IPsec SAs based on notifications from the NSF (i.e.
      expire).

   o NAT Traversal discovery and management.

   Additionally to these functions, another set of tasks must be
   performed by the I2NSF Controller (non-exhaustive list):

   o IPsec SA's SPI random generation.

   o Cryptographic algorithm/s selection.

   o Usage of extended sequence numbers.

   o Establishment of proper traffic selectors.

5. IKE case vs IKE-less case

   In principle, the IKE case is easier to deploy than the IKE-less case
   because current flow-based NSFs (either hosts or gateways) have
   access to IKEv2 implementations. While gateways typically deploy an
   IKEv2/IPsec implementation, hosts can easily install it. As
   downside, the NSF needs more resources to hold IKEv2 such as memory
   for the IKEv2 implementation, and computation, since each IPsec
   security association rekeying MAY involve a Diffie-Hellman exchange.

   Alternatively, IKE-less case benefits the deployment in resource-
   constrained NSFs. Moreover, IKEv2 does not need to be performed in
   gateway-to-gateway and host-to-host scenarios under the same I2NSF
   Controller (see Appendix G.1). On the contrary, the complexity of
   creating and managing IPsec SAs is shifted to the I2NSF Controller
   since IKEv2 is not in the NSF. As a consequence, this may result in
   a more complex implementation in the controller side in comparison
   with IKE case. For example, the I2NSF Controller has to deal with
   the latency existing in the path between the I2NSF Controller and the
   NSF, in order to solve tasks such as rekey, or creation and
   installation of new IPsec SAs. However, this is not specific to this
   contribution but a general aspect in any SDN-based network. In
   summary, this complexity MAY create some scalability and performance
   issues when the number of NSFs is high.

   Nevertheless, literature around SDN-based network management using a
   centralized controller (like the I2NSF Controller) is aware about
   scalability and performance issues and solutions have been already
   provided and discussed (e.g. hierarchical controllers; having
   multiple replicated controllers, dedicated high-speed management
   networks, etc). In the context of I2SNF-based IPsec management, one
   way to reduce the latency and alleviate some performance issues can
   be the installation of the IPsec policies and IPsec SAs at the same
   time (proactive mode, as described in Appendix G.1) instead of
   waiting for notifications (e.g. a notification sadb-acquire when a

new IPsec SA is required) to proceed with the IPsec SA installation
(reactive mode). Another way to reduce the overhead and the
potential scalability and performance issues in the I2NSF Controller
is to apply the IKE case described in this document, since the IPsec
SAs are managed between NSFs without the involvement of the I2NSF
Controller at all, except by the initial configuration (i.e. IKEv2,
PAD and SPD entries) provided by the I2NSF Controller. Other
solutions, such as Controller-IKE
[I-D.carrel-ipsecme-controller-ike], have proposed that NSFs provide
their DH public keys to the I2NSF Controller, so that the I2NSF
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 rekey 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 NSFs generate the session keys and not the I2NSF
Controller.

5.1. Rekeying process

Performing a rekey for IPsec SAs is an important operation during the
IPsec SAs management. With the YANG data models defined in this
document the I2NSF Controller can configure and conduct the rekey
process. Depending on the case, the rekey process is different.

For the IKE case, the rekeying process is carried out by IKEv2,
following the information defined in the SPD and SAD (i.e. based on
the IPsec SA lifetime established by the I2NSF Controller using the
YANG data model defined in this document). Therefore, IPsec
connections will live unless something different is required by the
I2NSF User or the I2NSF Controller detects something wrong.

For the IKE-less case, the I2NSF Controller MUST take care of the
rekeying process. When the IPsec SA is going to expire (e.g. IPsec
SA soft lifetime), it MUST create a new IPsec SA and it MAY remove
the old one (if a IPsec SA lifetime has not been defined). This
rekeying process starts when the I2NSF Controller receives a sadb-
expire notification or it decides so, based on lifetime state data
obtained from the NSF. How the I2NSF Controller implements an
algorithm for the rekey process is out of the scope of this document.
Nevertheless, an example of how this rekey could be performed is in
Appendix G.2.

5.2. NSF state loss.

   If one of the NSF restarts, it will lose the IPsec state (affected
   NSF). By default, the I2NSF 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 the IKE case, and SPD and SAD information
   in the IKE-less case.

   In both cases, the I2NSF Controller is aware of the affected NSF
   (e.g. the NETCONF/TCP connection is broken with the affected NSF, the
   I2NSF Controller is receiving sadb-bad-spi notification from a
   particular NSF, etc.). Moreover, the I2NSF Controller keeps a list
   of NSFs that have IPsec SAs with the affected NSF. Therefore, it
   knows the affected IPsec SAs.

   In the IKE case, the I2NSF 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. Finally,
   the I2NSF Controller will instruct the affected NSF to start the
   IKEv2 negotiation with the new configuration.

   Alternatively, IKEv2 configuration MAY be made permanent between NSFs
   reboots without compromising security by means of the startup
   configuration datastore in the NSF. This way, each time a NSF
   reboots it will use that configuration for each rebooting. It would
   imply avoiding to contact with the I2NSF Controller.

   In the IKE-less case, the I2NSF Controller SHOULD delete the old
   IPsec SAs in the non-failed nodes established with the affected NSF.
   Once the affected node restarts, the I2NSF Controller MUST take the
   necessary actions to reestablish IPsec protected communication
   between the failed node and those others having IPsec SAs with the
   affected NSF. How the I2NSF Controller implements an algorithm for
   managing a potential NSF state loss is out of the scope of this
   document. Nevertheless, an example of how this could be performed is
   described in Appendix G.3.

5.3. NAT Traversal

   In the IKE case, IKEv2 already provides 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 for ESP packets ([RFC3948],
   [RFC8229]). Note that the usage of IPsec transport mode when NAT is
   required MUST NOT be used in this specification.

In the IKE-less case, the NSF does not have the assistance of the
IKEv2 implementation to detect if it is located behind a NAT. If the
NSF does not have any other mechanism to detect this situation, the
I2NSF Controller SHOULD implement a mechanism to detect that case.
The SDN paradigm generally assumes the I2NSF Controller has a view of
the network under its control. This view is built either by
requesting information from the NSFs under its control, or by
information pushed from the NSFs to the I2NSF Controller. Based on
this information, the I2NSF Controller MAY 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]). The interface for discovering if
the NSF is behind a NAT is out of scope of this document.

If the I2NSF Controller does not have any mechanism to know whether a
host is behind a NAT or not, then the IKE-case MUST be used and not
the IKE-less case.

5.4. NSF registration and discovery

NSF registration refers to the process of facilitating the I2NSF
Controller information about a valid NSF such as certificate, IP
address, etc. This information is incorporated in a list of NSFs
under its control

The assumption in this document is that, for both cases, before a NSF
can operate in this system, it MUST be registered in the I2NSF
Controller. In this way, when the NSF starts and establishes a
connection to the I2NSF Controller, it knows that the NSF is valid
for joining the system.

Either during this registration process or when the NSF connects with
the I2NSF Controller, the I2NSF Controller MUST discover certain
capabilities of this NSF, such as what is the cryptographic suite
supported, authentication method, the support of the IKE case and/or
the IKE-less case, etc.

The registration and discovery processes are out of the scope of this
document.

6. YANG configuration data models

In order to support the IKE and IKE-less cases we have modeled the
different parameters and values that must be configured to manage
IPsec SAs. Specifically, the IKE case requires modeling IKEv2
configuration parameters, SPD and PAD, while the IKE-less case
requires configuration models for the SPD and SAD. We have defined
three models: ietf-i2nsf-ikec (Appendix A, common to both cases),

ietf-i2nsf-ike (Appendix B, IKE case), ietf-i2nsf-ikeless
(Appendix C, IKE-less case). Since the model ietf-i2nsf-ikec has
only typedef and groupings common to the other modules, we only show
a simplified view of the ietf-i2nsf-ike and ietf-i2nsf-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 [strongswan] or Libreswan
[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).

The data model for the IKE case is defined by YANG model "ietf-i2nsf-
ike". Its structure is depicted in the following diagram, using the
notation syntax for YANG tree diagrams ([RFC8340]).

```
module: ietf-i2nsf-ike
+--rw ipsec-ike
 +--rw pad
 |  +--rw pad-entry* [name]
 |     +--rw name                       string
 |     +--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
 |     |  +--:(gnx509)
 |     |  |  +--rw gnx509?               string
 |     |  +--:(id-key)
 |     |  |  +--rw id-key?               string
 |     |  +--:(id-null)
 |     |     +--rw id-null?              empty
 |     +--rw auth-protocol?              auth-protocol-type
 |     +--rw peer-authentication
 |        +--rw auth-method?        auth-method-type
```

```
|          +--rw eap-method
|          |  +--rw eap-type     uint8
|          +--rw pre-shared
|          |  +--rw secret     yang:hex-string
|          +--rw digital-signature
|             +--rw ds-algorithm?            uint8
|             +--rw (public-key)
|             |  +--:(raw-public-key)
|             |  |  +--rw raw-public-key?   binary
|             |  +--:(cert-data)
|             |     +--rw cert-data?        ct:x509
|             +--rw private-key?            binary
|             +--rw ca-data*                ct:x509
|             +--rw crl-data?               ct:crl
|             +--rw crl-uri?                inet:uri
|             +--rw oscp-uri?               inet:uri
+--rw conn-entry* [name]
|  +--rw name                            string
|  +--rw autostartup?                    autostartup-type
|  +--rw initial-contact?                boolean
|  +--rw version?                        auth-protocol-type
|  +--rw fragmentation?                  boolean
|  +--rw ike-sa-lifetime-soft
|  |  +--rw rekey-time?     uint32
|  |  +--rw reauth-time?    uint32
|  +--rw ike-sa-lifetime-hard
|  |  +--rw over-time?    uint32
|  +--rw authalg*           nsfikec:integrity-algorithm-type
|  +--rw encalg* [id]
|  |  +--rw id               uint8
|  |  +--rw algorithm-type?   nsfikec:encryption-algorithm-type
|  |  +--rw key-length?       uint16
|  +--rw dh-group?                          pfs-group
|  +--rw half-open-ike-sa-timer?            uint32
|  +--rw half-open-ike-sa-cookie-threshold?   uint32
|  +--rw local
|  |  +--rw local-pad-entry-name     string
|  +--rw remote
|  |  +--rw remote-pad-entry-name     string
|  +--rw encapsulation-type
|  |  +--rw espencap?    esp-encap
|  |  +--rw sport?       inet:port-number
|  |  +--rw dport?       inet:port-number
|  |  +--rw oaddr*       inet:ip-address
|  +--rw spd
|  |  +--rw spd-entry* [name]
|  |  +--rw name                  string
|  |  +--rw ipsec-policy-config
```

```
| |     +--rw anti-replay-window?    uint64
| |     +--rw traffic-selector
| |     |  +--rw local-subnet       inet:ip-prefix
| |     |  +--rw remote-subnet      inet:ip-prefix
| |     |  +--rw inner-protocol?    ipsec-inner-protocol
| |     |  +--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-action
| |    |+--rw ipsec-sa-cfg
| |    | +--rw pfp-flag?             boolean
| |    | +--rw ext-seq-num?          boolean
| |    | +--rw seq-overflow?         boolean
| |    | +--rw stateful-frag-check?  boolean
| |    | +--rw mode?                 ipsec-mode
| |    | +--rw protocol-parameters? ipsec-protocol-parameters
| |    |  +--rw esp-algorithms
| |    |  | +--rw integrity* integrity-algorithm-type
| |    |  | +--rw encryption* [id]
| |    |  | | +--rw id                uint8
| |    |  | | +--rw algorithm-type? nsfikec:encryption-algorithm-type
| |    |  | | +--rw key-length?       uint16
| |    |  | +--rw tfc-pad?       boolean
| |    |   +--rw tunnel
| |    |      +--rw local            inet:ip-address
| |    |      +--rw remote           inet:ip-address
| |    |      +--rw df-bit?          enumeration
| |    |      +--rw bypass-dscp?     boolean
| |    |      +--rw dscp-mapping?    yang:hex-string
| |    |      +--rw ecn?             boolean
| |    +--rw spd-mark
| |       +--rw mark?    uint32
| |       +--rw mask?    yang:hex-string
| +--rw child-sa-info
| |  +--rw pfs-groups*              pfs-group
| |  +--rw child-sa-lifetime-soft
| |  |  +--rw time?       uint32
| |  |  +--rw bytes?      uint32
| |  |  +--rw packets?    uint32
| |  |  +--rw idle?       uint32
| |  |  +--rw action?     nsfikec:lifetime-action
| |  +--rw child-sa-lifetime-hard
| |     +--rw time?       uint32
| |     +--rw bytes?      uint32
```

```
   | |      +--rw packets?    uint32
   | |      +--rw idle?       uint32
   | +--ro state
   |    +--ro initiator?              boolean
   |    +--ro initiator-ikesa-spi?    ike-spi
   |    +--ro responder-ikesa-spi?    ike-spi
   |    +--ro nat-local?              boolean
   |    +--ro nat-remote?             boolean
   |    +--ro encapsulation-type
   |    |  +--ro espencap?    esp-encap
   |    |  +--ro sport?       inet:port-number
   |    |  +--ro dport?       inet:port-number
   |    |  +--ro oaddr*       inet:ip-address
   |    +--ro established?            uint64
   |    +--ro current-rekey-time?     uint64
   |    +--ro current-reauth-time?    uint64
   +--ro number-ike-sas
      +--ro total?               uint64
      +--ro half-open?           uint64
      +--ro half-open-cookies?   uint64
```

   The data model consists of a unique "ipsec-ike" container defined as
   follows. Firstly, it contains a "pad" container that serves to
   configure the Peer Authentication Database with authentication
   information about local and remote peers. More precisely, it
   consists of a list of entries, each one indicating the identity,
   authentication method and credentials that will use a particular
   peer.

   Next, we find a list "conn-entry" with information about the
   different IKE connections a peer can maintain with others. Each
   connection entry is composed of a wide number of parameters to
   configure different aspects of a particular IKE connection between
   two peers: local and remote peer authentication information; IKE SA
   configuration (soft and hard lifetimes, cryptographic algorithms,
   etc.); list of IPsec policies describing the type of network traffic
   to be secured (local/remote subnet and ports, etc.) and how must be
   protected (AH/ESP, tunnel/transport, cryptographic algorithms, etc.);
   CHILD SA configuration (soft and hard lifetimes); and, state
   information of the IKE connection (SPIs, usage of NAT, current
   expiration times, etc.).

   Lastly, the "ipsec-ike" container declares a "number-ike-sas"
   container to specify state information reported by the IKE software
   related to the amount of IKE connections established.

Appendix D shows an example of IKE case configuration for a NSF, in
tunnel mode (gateway-to-gateway), with NSFs authentication based on
X.509 certificates.

6.2. IKE-less case model

For this case, the definition of the SPD model has been mainly
extracted from the specification in section 4.4.1 and Appendix D in
  [RFC4301], though with some changes, namely:

o Each IPsec policy (spd-entry) contains one traffic selector,
   instead of a list of them. The reason is that we have observed
   actual kernel implementations only admit a single traffic selector
   per IPsec policy.

o Each IPsec policy contains a identifier (reqid) to relate the
   policy with the IPsec SA. This is common in Linux-based systems.

o Each IPsec policy has only one name and not a list of names.

o Combined algorithms have been removed because encryption
   algorithms MAY include authenticated encryption with associated
   data (AEAD).

o Tunnel information has been extended with information about DSCP
   mapping and ECN bit. The reason is that we have observed real
   kernel implementations accept configuration of these values.

The definition of the SAD model has been mainly extracted from the
specification in section 4.4.2 in [RFC4301] though with some changes,
namely:

o Each IPsec SA (sad-entry) contains one traffic selector, instead
   of a list of them. The reason is that we have observed actual
   kernel implementations only admit a single traffic selector per
   IPsec SA.

o Each IPsec SA contains a identifier (reqid) to relate the IPsec SA
   with the IPsec Policy. The reason is that we have observed real
   kernel implementations allow to include this value.

o Each IPsec SA has also a name in the same way as IPsec policies.

o Combined algorithm has been removed because encryption algorithm
   MAY include authenticated encryption with associated data (AEAD).

o Tunnel information has been extended with information about
   Differentiated Services Code Point (DSCP) mapping and Explicit

Congestion Notificsation (ECN) bit. The reason is that we have
observed actual kernel implementations admit the configurations of
these values.

o Lifetime of the IPsec SAs also include idle time and number of IP
   packets as threshold to trigger the lifetime. The reason is that
   we have observed actual kernel implementations allow to set these
   types of lifetimes.

o Information to configure the type of encapsulation (encapsulation-
   type) for IPsec ESP packets in UDP ([RFC3948]), TCP ([RFC8229]) or
   TLS ([RFC8229]) has been included.

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

The data model for the IKE-less case is defined by YANG model "ietf-
i2nsf-ikeless". Its structure is depicted in the following diagram,
using the notation syntax for YANG tree diagrams ([RFC8340]).

```
module: ietf-i2nsf-ikeless
+--rw ipsec-ikeless
 +--rw spd
 | +--rw spd-entry* [name]
 |    +--rw name                  string
 |    +--rw direction             nsfikec:ipsec-traffic-direction
 |    +--rw reqid?                uint64
 |    +--rw ipsec-policy-config
 |       +--rw anti-replay-window?    uint64
 |       +--rw traffic-selector
 |       |  +--rw local-subnet      inet:ip-prefix
 |       |  +--rw remote-subnet     inet:ip-prefix
 |       |  +--rw inner-protocol?   ipsec-inner-protocol
 |       |  +--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-action
 |       |  +--rw ipsec-sa-cfg
 |       |     +--rw pfp-flag?               boolean
 |       |     +--rw ext-seq-num?            boolean
 |       |     +--rw seq-overflow?           boolean
 |       |     +--rw stateful-frag-check?    boolean
 |       |     +--rw mode?                   ipsec-mode
```

```
     |       |    +--rw protocol-parameters? ipsec-protocol-parameters
     |       |    +--rw esp-algorithms
     |       |    | +--rw integrity* integrity-algorithm-type
     |       |    | +--rw encryption* [id]
     |       |    | |+--rw id                uint8
     |       |    | |+--rw algorithm-type? nsfikec:encryption-algorithm-type
     |       |    | |+--rw key-length?       uint16
     |       |    | +--rw tfc-pad?       boolean
     |       |    +--rw tunnel
     |       |       +--rw local          inet:ip-address
     |       |       +--rw remote         inet:ip-address
     |       |       +--rw df-bit?        enumeration
     |       |       +--rw bypass-dscp?   boolean
     |       |       +--rw dscp-mapping?  yang:hex-string
     |       |       +--rw ecn?           boolean
     |     +--rw spd-mark
     |        +--rw mark?   uint32
     |        +--rw mask?   yang:hex-string
   +--rw sad
    +--rw sad-entry* [name]
     +--rw name                string
     +--rw reqid?              uint64
     +--rw ipsec-sa-config
     | +--rw spi                   uint32
     | +--rw ext-seq-num?          boolean
     | +--rw seq-number-counter?   uint64
     | +--rw seq-overflow?         boolean
     | +--rw anti-replay-window?   uint32
     | +--rw traffic-selector
     | |   +--rw local-subnet      inet:ip-prefix
     | |   +--rw remote-subnet     inet:ip-prefix
     | |   +--rw inner-protocol?   ipsec-inner-protocol
     | |   +--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 protocol-parameters?   nsfikec:ipsec-protocol-parameters
     | +--rw mode?                  nsfikec:ipsec-mode
     | +--rw esp-sa
     | | +--rw encryption
     | | |+--rw encryption-algorithm? nsfikec:encryption-algorithm-type
     | | |+--rw key?                  yang:hex-string
     | | |+--rw iv?                   yang:hex-string
     | | +--rw integrity
     | |   +--rw integrity-algorithm? nsfikec:integrity-algorithm-type
     | |   +--rw key?                    yang:hex-string
```

```
       | +--rw sa-lifetime-hard
       | |  +--rw time?      uint32
       | |  +--rw bytes?     uint32
       | |  +--rw packets?   uint32
       | |  +--rw idle?      uint32
       | +--rw sa-lifetime-soft
       | |  +--rw time?      uint32
       | |  +--rw bytes?     uint32
       | |  +--rw packets?   uint32
       | |  +--rw idle?      uint32
       | |  +--rw action?    nsfikec:lifetime-action
       | +--rw tunnel
       | |  +--rw local           inet:ip-address
       | |  +--rw remote          inet:ip-address
       | |  +--rw df-bit?         enumeration
       | |  +--rw bypass-dscp?    boolean
       | |  +--rw dscp-mapping?   yang:hex-string
       | |  +--rw ecn?            boolean
       | +--rw encapsulation-type
       |    +--rw espencap?   esp-encap
       |    +--rw sport?      inet:port-number
       |    +--rw dport?      inet:port-number
       |    +--rw oaddr*      inet:ip-address
      +--ro ipsec-sa-state
         +--ro sa-lifetime-current
         |  +--ro time?      uint32
         |  +--ro bytes?     uint32
         |  +--ro packets?   uint32
         |  +--ro idle?      uint32
         +--ro replay-stats
            +--ro replay-window?        uint64
            +--ro packet-dropped?       uint64
            +--ro failed?               uint32
            +--ro seq-number-counter?   uint64

   notifications:
     +---n sadb-acquire {ikeless-notification}?
     |  +--ro ipsec-policy-name    string
     |  +--ro traffic-selector
     |     +--ro local-subnet      inet:ip-prefix
     |     +--ro remote-subnet     inet:ip-prefix
     |     +--ro inner-protocol?   ipsec-inner-protocol
     |     +--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 {ikeless-notification}?
      |  +--ro ipsec-sa-name          string
      |  +--ro soft-lifetime-expire?   boolean
      |  +--ro lifetime-current
      |     +--ro time?      uint32
      |     +--ro bytes?     uint32
      |     +--ro packets?   uint32
      |     +--ro idle?      uint32
      +---n sadb-seq-overflow {ikeless-notification}?
      |  +--ro ipsec-sa-name    string
      +---n sadb-bad-spi {ikeless-notification}?
         +--ro spi     uint32
```

The data model consists of a unique "ipsec-ikeless" container which,
in turn, is integrated by two additional containers: "spd" and "sad".
The "spd" container consists of a list of entries that conform the
Security Policy Database. Compared to the IKE case data model, this
part specifies a few additional parameters necessary due to the
absence of an IKE software in the NSF: traffic direction to apply the
IPsec policy, and a value to link an IPsec policy with its associated
IPsec SAs. The "sad" container is a list of entries that conform the
Security Association Database. In general, each entry allows to
specify both configuration information (SPI, traffic selectors,
tunnel/transport mode, cryptographic algorithms and keying material,
soft/hard lifetimes, etc.) as well as state information (time to
expire, replay statistics, etc.) of a concrete IPsec SA.

In addition, the module defines a set of notifications to allow the
NSF inform I2NSF controller about relevant events such as IPsec SA
expiration, sequence number overflow or bad SPI in a received packet.

Appendix E shows an example of IKE-less case configuration for a NSF,
in transport mode (host-to-host), with NSFs authentication based on
shared secrets. For the IKE-less case, Appendix F shows examples of
IPsec SA expire, acquire, sequence number overflow and bad SPI
notifications.

7. IANA Considerations

This document registers three URIs in the "ns" subregistry of the
IETF XML Registry [RFC3688]. Following the format in [RFC3688], the
following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikec
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-i2nsf-ike
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.

This document registers three YANG modules in the "YANG Module Names" registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

Name: ietf-i2nsf-ikec
Namespace: urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikec
Prefix: nsfikec
Reference: RFC XXXX

Name: ietf-i2nsf-ike
Namespace: urn:ietf:params:xml:ns:yang:ietf-i2nsf-ike
Prefix: nsfike
Reference: RFC XXXX

Name: ietf-i2nsf-ikeless
Namespace: urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless
Prefix: nsfikels
Reference: RFC XXXX

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 [RFC7426].

On the one hand, it is important to note that there MUST exist a security association between the I2NSF Controller and the NSFs to protect the critical information (cryptographic keys, configuration parameter, etc.) exchanged between these entities.

On the other hand, if encryption is mandatory for all traffic of a NSF, its default policy MUST be to drop (DISCARD) packets to prevent cleartext packet leaks. This default policy MUST be pre-configured in the startup configuration datastore in the NSF before the NSF contacts the I2NSF Controller. Moreover, the startup configuration datastore MUST be also pre-configured with the required ALLOW

policies that allow the NSF to communicate with the I2NSF Controller
once the NSF is deployed. This pre-configuration step is not carried
out by the I2NSF Controller but by some other entity before the NSF
deployment. In this manner, when the NSF starts/reboots, it will
always first apply the configuration in the startup configuration
before contacting the I2NSF Controller.

Finally, we have divided this section in two parts in order to
analyze different security considerations for both cases: NSF with
IKEv2 (IKE case) and NSF without IKEv2 (IKE-less case). In general,
the I2NSF Controller, as typically in the SDN paradigm, is a target
for different type of attacks [SDNSecServ] and [SDNSecurity]. Thus,
the I2NSF Controller is a key entity in the infrastructure and MUST
be protected accordingly. In particular, the I2NSF Controller will
handle cryptographic material thus the attacker may try to access
this information. Although we can assume this attack is not likely
to happen due to the assumed security measurements to protect the
I2NSF Controller, it still deserves some analysis in the hypothetical
case that the attack occurs. The impact is different depending on
the IKE case or IKE-less case.

8.1. IKE case

In the IKE case, the I2NSF Controller sends IKEv2 credentials (PSK,
public/private keys, certificates, etc.) to the NSFs using the
security association between I2NSF Controller and NSFs. The I2NSF
Controller MUST NOT store the IKEv2 credentials after distributing
them. Moreover, the NSFs MUST NOT allow the reading of these values
once they have been applied by the I2NSF Controller (i.e. write only
operations). One option is to always return the same value (i.e. all
0s) if a read operation is carried out.

If the attacker has access to the I2NSF Controller during the period
of time that key material is generated, it might have access to the
key material. Since these values are used during NSF authentication
in IKEv2, it may impersonate the affected NSFs. Several
recommendations are important.

o IKEv2 configurations should adhere to the recommendations in
   [RFC8247].

o If PSK authentication is used in IKEv2, the I2NSF Controller MUST
   remove the PSK immediately after generating and distributing it.

o When public/private keys are used, the I2NSF Controller MAY
   generate both public key and private key. In such a case, the
   I2NSF Controller MUST remove the associated private key
   immediately after distributing them to the NSFs. Alternatively,

the NSF could generate the private key and export only the public
key to the I2NSF Controller.

o If certificates are used, the NSF MAY generate the private key and
   export the public key for certification to the I2NSF Controller.
   How the NSF generates these cryptographic material (public key/
   private keys) and exports the public key, is out of scope of this
   document.

8.2. IKE-less case

In the IKE-less case, the I2NSF Controller sends the IPsec SA
information to the NSF's SAD that includes the private session keys
required for integrity and encryption. The I2NSF Controller MUST NOT
store the keys after distributing them. Moreover, the NSFs receiving
private key material MUST NOT allow the reading of these values by
any other entity (including the I2NSF Controller itself) once they
have been applied (i.e. write only operations) into the NSFs.
Nevertheless, if the attacker has access to the I2NSF Controller
during the period of time that key material is generated, it may
obtain these values. In other words, the attacker might be able to
observe the IPsec traffic and decrypt, or even modify and re-encrypt,
the traffic between peers.

8.3. YANG modules

The YANG modules specified in this document define a schema for data
that is designed to be accessed via network management protocols such
as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer
is the secure transport layer, and the mandatory-to-implement secure
transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer
is HTTPS, and the mandatory-to-implement secure transport is TLS
[RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341]
provides the means to restrict access for particular NETCONF or
RESTCONF users to a preconfigured subset of all available NETCONF or
RESTCONF protocol operations and content.

There are a number of data nodes defined in these YANG modules that
are writable/creatable/deletable (i.e., config true, which is the
default). These data nodes may be considered sensitive or vulnerable
in some network environments. Write operations (e.g., edit-config)
to these data nodes without proper protection can have a negative
effect on network operations. These are the subtrees and data nodes
and their sensitivity/vulnerability:

For the IKE case (ietf-i2nsf-ike):

/ipsec-ike: The entire container in this module is sensitive to
write operations. An attacker may add/modify the credentials
to be used for the authentication (e.g. to impersonate a NSF),
the trust root (e.g. changing the trusted CA certificates), the
cryptographic algorithms (allowing a downgrading attack), the
IPsec policies (e.g. by allowing leaking of data traffic by
changing to a allow policy), and in general changing the IKE SA
conditions and credentials between any NSF.

For the IKE-less case (ietf-i2nsf-ikeless):

/ipsec-ikeless: The entire container in this module is
sensitive to write operations. An attacker may add/modify/
delete any IPsec policies (e.g. by allowing leaking of data
traffic by changing to a allow policy) in the /ipsec-ikeless/
spd container, and add/modify/delete any IPsec SAs between two
NSF by means of /ipsec-ikeless/sad container and, in general
changing any IPsec SAs and IPsec policies between any NSF.

Some of the readable data nodes in this YANG module may be considered
sensitive or vulnerable in some network environments. It is thus
important to control read access (e.g., via get, get-config, or
notification) to these data nodes. These are the subtrees and data
nodes and their sensitivity/vulnerability:

For the IKE case (ietf-i2nsf-ike):

/ipsec-ike/pad: This container includes sensitive information
to read operations. This information should never be returned
to a client. For example, cryptographic material configured in
the NSFs: peer-authentication/pre-shared/secret and peer-
authentication/digital-signature/private-key are already
protected by the NACM extension "default-deny-all" in this
document.

For the IKE-less case (ietf-i2nsf-ikeless):

/ipsec-ikeless/sad/sad-entry/ipsec-sa-config/esp-sa: This
container includes symmetric keys for the IPsec SAs. For
example, encryption/key contains a ESP encryption key value and
encryption/iv contains a initialization vector value.
Similarly, integrity/key has ESP integrity key value. Those
values must not be read by anyone and are protected by the NACM
extension "default-deny-all" in this document.

9. Acknowledgements

Authors want to thank Paul Wouters, Valery Smyslov, Sowmini Varadhan, David Carrel, Yoav Nir, Tero Kivinen, Martin Bjorklund, Graham Bartlett, Sandeep Kampati, Linda Dunbar, Mohit Sethi, Martin Bjorklund, Tom Petch, Christian Hopps, Rob Wilton, Carlos J. Bernardos, Alejandro Perez-Mendez, Alejandro Abad-Carrascosa, Ignacio Martinez, Ruben Ricart and Roman Danyliw for their valuable comments.

10. References

10.1. Normative References

[I-D.draft-ietf-netconf-crypto-types]
           Watsen, K., "YANG Data Types and Groupings for
           Cryptography", draft-ietf-netconf-crypto-types-18 (work in
           progress), August 2020.

[IKEv2-Parameters]
           Internet Assigned Numbers Authority (IANA), "Internet Key
           Exchange Version 2 (IKEv2) Parameters", August 2020.

[ITU-T.X.690]
           "Recommendation ITU-T X.690", August 2015.

[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>.

[RFC2247] Kille, S., Wahl, M., Grimstad, A., Huber, R., and S.
           Sataluri, "Using Domains in LDAP/X.500 Distinguished
           Names", RFC 2247, DOI 10.17487/RFC2247, January 1998,
           <https://www.rfc-editor.org/info/rfc2247>.

[RFC3947] Kivinen, T., Swander, B., Huttunen, A., and V. Volpe,
           "Negotiation of NAT-Traversal in the IKE", RFC 3947,
           DOI 10.17487/RFC3947, January 2005,
           <https://www.rfc-editor.org/info/rfc3947>.

[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>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)",
           RFC 4303, DOI 10.17487/RFC4303, December 2005,
           <https://www.rfc-editor.org/info/rfc4303>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
           Housley, R., and W. Polk, "Internet X.509 Public Key
           Infrastructure Certificate and Certificate Revocation List
           (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
           <https://www.rfc-editor.org/info/rfc5280>.

[RFC5915] Turner, S. and D. Brown, "Elliptic Curve Private Key
           Structure", RFC 5915, DOI 10.17487/RFC5915, June 2010,
           <https://www.rfc-editor.org/info/rfc5915>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for
           the Network Configuration Protocol (NETCONF)", RFC 6020,
           DOI 10.17487/RFC6020, October 2010,
           <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed.,
           and A. Bierman, Ed., "Network Configuration Protocol
           (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,
           <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure
           Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011,
           <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types",
           RFC 6991, DOI 10.17487/RFC6991, July 2013,
           <https://www.rfc-editor.org/info/rfc6991>.

[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>.

[RFC7383] Smyslov, V., "Internet Key Exchange Protocol Version 2
           (IKEv2) Message Fragmentation", RFC 7383,
           DOI 10.17487/RFC7383, November 2014,
           <https://www.rfc-editor.org/info/rfc7383>.

[RFC7427] Kivinen, T. and J. Snyder, "Signature Authentication in
           the Internet Key Exchange Version 2 (IKEv2)", RFC 7427,
           DOI 10.17487/RFC7427, January 2015,
           <https://www.rfc-editor.org/info/rfc7427>.

[RFC7619] Smyslov, V. and P. Wouters, "The NULL Authentication
           Method in the Internet Key Exchange Protocol Version 2
           (IKEv2)", RFC 7619, DOI 10.17487/RFC7619, August 2015,
           <https://www.rfc-editor.org/info/rfc7619>.

   [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language",
              RFC 7950, DOI 10.17487/RFC7950, August 2016,
              <https://www.rfc-editor.org/info/rfc7950>.

   [RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch,
              "PKCS #1: RSA Cryptography Specifications Version 2.2",
              RFC 8017, DOI 10.17487/RFC8017, November 2016,
              <https://www.rfc-editor.org/info/rfc8017>.

   [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,
              <https://www.rfc-editor.org/info/rfc8040>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

   [RFC8221] Wouters, P., Migault, D., Mattsson, J., Nir, Y., and T.
              Kivinen, "Cryptographic Algorithm Implementation
              Requirements and Usage Guidance for Encapsulating Security
              Payload (ESP) and Authentication Header (AH)", RFC 8221,
              DOI 10.17487/RFC8221, October 2017,
              <https://www.rfc-editor.org/info/rfc8221>.

   [RFC8247] Nir, Y., Kivinen, T., Wouters, P., and D. Migault,
              "Algorithm Implementation Requirements and Usage Guidance
              for the Internet Key Exchange Protocol Version 2 (IKEv2)",
              RFC 8247, DOI 10.17487/RFC8247, September 2017,
              <https://www.rfc-editor.org/info/rfc8247>.

   [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams",
              BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018,
              <https://www.rfc-editor.org/info/rfc8340>.

   [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration
              Access Control Model", STD 91, RFC 8341,
              DOI 10.17487/RFC8341, March 2018,
              <https://www.rfc-editor.org/info/rfc8341>.

   [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K.,
              and R. Wilton, "Network Management Datastore Architecture
              (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018,
              <https://www.rfc-editor.org/info/rfc8342>.

   [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018,
              <https://www.rfc-editor.org/info/rfc8446>.

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

   [libreswan]
            The Libreswan Project, "Libreswan VPN software", September
            2020.

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

   [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>.

   [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#),
            DOI 10.17487/RFC3688, January 2004,
            <https://www.rfc-editor.org/info/rfc3688>.

   [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>.

   [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme,
              "IPv6 Flow Label Specification", RFC 6437,
              DOI 10.17487/RFC6437, November 2011,
              <https://www.rfc-editor.org/info/rfc6437>.

   [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>.

   [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>.

   [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>.

   [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>.

   [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>.

   [SDNSecServ]
              Scott-Hayward, S., O'Callaghan, G., and P. Sezer, "SDN
              Security: A Survey", 2013.

   [SDNSecurity]
              Kreutz, D., Ramos, F., and P. Verissimo, "Towards Secure
              and Dependable Software-Defined Networks", 2013.

   [strongswan]
              CESNET, "StrongSwan: the OpenSource IPsec-based VPN
              Solution", September 2020.

Appendix A. Common YANG model for IKE and IKE-less cases

   This Appendix is Normative.

   This YANG module has normative references to [RFC3947], [RFC4301],
   [RFC4303], [RFC8174], [RFC8221] and [IKEv2-Parameters].

   This YANG module has informative references to [RFC3948] and
   [RFC8229].

         <CODE BEGINS> file "ietf-i2nsf-ikec@2020-10-21.yang"

         module ietf-i2nsf-ikec {
             yang-version 1.1;
             namespace "urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikec";
             prefix "nsfikec";

             import ietf-inet-types {
                 prefix inet;
                 reference "RFC 6991: Common YANG Data Types";
             }

             import ietf-yang-types {
                 prefix yang;
                 reference "RFC 6991: Common YANG Data Types";
             }

             organization "IETF I2NSF Working Group";

             contact
             "WG Web: <https://datatracker.ietf.org/wg/i2nsf/>
              WG List: <mailto:i2nsf@ietf.org>

             Author: Rafael Marin-Lopez
                     <mailto:rafa@um.es>

             Author: Gabriel Lopez-Millan
                     <mailto:gabilm@um.es>

             Author: Fernando Pereniguez-Garcia
                     <mailto:fernando.pereniguez@cud.upct.es>
             ";

             description
                 "Common Data model for the IKE and IKE-less cases
                  defined by the SDN-based IPsec flow protection service.

```
        revision "2020-10-21" {
            description "Initial version.";
            reference "RFC XXXX: Software-Defined Networking
            (SDN)-based IPsec Flow Protection.";
        }

        typedef encryption-algorithm-type {
            type uint16;
            description
                "The encryption algorithm is specified with a 16-bit
                number extracted from IANA Registry. The acceptable
                values MUST follow the requirement levels for
                encryption algorithms for ESP and IKEv2.";
            reference
                 "IANA Registry- Transform Type 1 - Encryption
                  Algorithm Transform IDs. RFC 8221 - Cryptographic
                  Algorithm Implementation Requirements and Usage
                  Guidance for Encapsulating Security Payload (ESP)
                  and Authentication Header (AH) and RFC 8247 -
                  Algorithm Implementation Requirements and Usage
                  Guidance for the Internet Key Exchange Protocol
                  Version 2 (IKEv2).";
        }

        typedef integrity-algorithm-type {
            type uint16;
            description
                "The integrity algorithm is specified with a 16-bit
                number extracted from IANA Registry.
```

```
                    The acceptable values MUST follow the requirement
                    levels for encryption algorithms for ESP and IKEv2.";
              reference
                  "IANA Registry- Transform Type 3 - Integrity
                   Algorithm Transform IDs. RFC 8221 - Cryptographic
                   Algorithm Implementation Requirements and Usage
                   Guidance for Encapsulating Security Payload (ESP)
                   and Authentication Header (AH) and RFC 8247 -
                   Algorithm Implementation Requirements and Usage
                   Guidance for the Internet Key Exchange Protocol
                   Version 2 (IKEv2).";
        }

        typedef ipsec-mode {
            type enumeration {
                enum transport {
                    description
                        "IPsec transport mode. No Network Address
                         Translation (NAT) support.";
                }
                enum tunnel {
                    description "IPsec tunnel mode.";
                }
            }
            description
                "Type definition of IPsec mode: transport or
                 tunnel.";
            reference
                "Section 3.2 in RFC 4301.";
        }

        typedef esp-encap {
            type enumeration {
                enum espintcp {
                    description
                        "ESP in TCP encapsulation.";
                    reference
                        "RFC 8229 - TCP Encapsulation of IKE and
                         IPsec Packets.";
                }
                enum espintls {
                    description
                        "ESP in TCP encapsulation using TLS.";
                    reference
                        "RFC 8229 - TCP Encapsulation of IKE and
                         IPsec Packets.";
                }
                enum espinudp {
```

```
                        description
                            "ESP in UDP encapsulation.";
                        reference
                            "RFC 3948 - UDP Encapsulation of IPsec ESP
                            Packets.";
                    }
                    enum none {
                        description
                            "NOT ESP encapsulation.";
                    }
                }
                description
                    "Types of ESP encapsulation when Network Address
                     Translation (NAT) is present between two NSFs.";
                reference
                    "RFC 8229 - TCP Encapsulation of IKE and IPsec
                     Packets and RFC 3948 - UDP Encapsulation of IPsec
                     ESP Packets.";
            }

            typedef ipsec-protocol-parameters {
                type enumeration {
                    enum esp { description "IPsec ESP protocol."; }
                }
                description
                    "Only the Encapsulation Security Protocol (ESP) is
                     supported but it could be extended in the future.";
                reference
                    "RFC 4303- IP Encapsulating Security Payload
                    (ESP).";

            }

            typedef lifetime-action {
                type enumeration {
                    enum terminate-clear {
                        description
                            "Terminates the IPsec SA and allows the
                             packets through.";
                    }
                    enum terminate-hold {
                        description
                            "Terminates the IPsec SA and drops the
                             packets.";
                    }
                    enum replace  {
                        description
                            "Replaces the IPsec SA with a new one:
```

```
                                rekey. ";
                        }
                }
                description
                    "When the lifetime of an IPsec SA expires an action
                     needs to be performed over the IPsec SA that
                     reached the lifetime. There are three posible
                     options: terminate-clear, terminate-hold and
                     replace.";
                reference
                    "Section 4.5 in RFC 4301.";
            }

            typedef ipsec-traffic-direction {
                type enumeration {
                    enum inbound {
                        description "Inbound traffic.";
                    }
                    enum outbound {
                        description "Outbound traffic.";
                    }
                }
                description
                    "IPsec traffic direction is defined in two
                     directions: inbound and outbound. From a NSF
                     perspective inbound means the traffic that enters
                     the NSF and outbound is the traffic that is sent
                     from the NSF.";
                reference
                    "Section 5 in RFC 4301.";
            }

            typedef ipsec-spd-action {
                type enumeration {
                    enum protect {
                        description
                            "PROTECT the traffic with IPsec.";
                    }
                    enum bypass {
                        description
                            "BYPASS the traffic. The packet is forwarded
                             without IPsec protection.";
                    }
                    enum discard {
                        description
                            "DISCARD the traffic. The IP packet is
                             discarded.";
                    }
```

```
            }
            description
                "The action when traffic matches an IPsec security
                 policy. According to RFC 4301 there are three
                 possible values: BYPASS, PROTECT AND DISCARD";
            reference
                "Section 4.4.1 in RFC 4301.";
        }

        typedef ipsec-inner-protocol {
            type union {
                type uint8;
                type enumeration {
                    enum any {
                        value 256;
                        description
                            "Any IP protocol number value.";
                    }
                }
            }
            default any;
            description
                "IPsec protection can be applied to specific IP
                 traffic and layer 4 traffic (TCP, UDP, SCTP, etc.)
                 or ANY protocol in the IP packet payload. We
                 specify the IP protocol number with an uint8 or
                 ANY defining an enumerate with value 256 to
                 indicate the protocol number.";
            reference
                "Section 4.4.1.1 in RFC 4301.
                 IANA Registry - Protocol Numbers.";
        }

        grouping encap {
            description
                "This group of nodes allows to define the type of
                 encapsulation in case NAT traversal is
                 required and port information.";
            leaf espencap {
                type esp-encap;
                default none;
                description
                    "ESP in TCP, ESP in UDP or ESP in TLS.";
            }
            leaf sport {
                type inet:port-number;
                default 4500;
                description
```

```
                      "Encapsulation source port.";
            }
            leaf dport {
                type inet:port-number;
                default 4500;
                description
                    "Encapsulation destination port.";
            }

            leaf-list oaddr {
                type inet:ip-address;
                description
                    "If required, this is the original address that
                     was used before NAT was applied over the Packet.
                     ";
            }
            reference
                "RFC 3947 and RFC 8229.";
        }

        grouping lifetime {
            description
                "Different lifetime values limited to an IPsec SA.";
            leaf time {
                type uint32;
                default 0;
                description
                    "Time in seconds since the IPsec SA was added.
                     For example, if this value is 180 seconds it
                     means the IPsec SA expires in 180 seconds since
                     it was added. The value 0 implies infinite.";
            }
            leaf bytes {
                type uint32;
                default 0;
                description
                    "If the IPsec SA processes the number of bytes
                     expressed in this leaf, the IPsec SA expires and
                     should be rekeyed. The value 0 implies
                     infinite.";
            }
            leaf packets {
                type uint32;
                default 0;
                description
                    "If the IPsec SA processes the number of packets
                     expressed in this leaf, the IPsec SA expires and
                     should be rekeyed. The value 0 implies
```

```
                    infinite.";
            }
            leaf idle {
                type uint32;
                default 0;
                description
                    "When a NSF stores an IPsec SA, it
                     consumes system resources. In an idle NSF this
                     is a waste of resources. If the IPsec SA is idle
                     during this number of seconds the IPsec SA
                     should be removed. The value 0 implies
                     infinite.";
            }
            reference
                "Section 4.4.2.1 in RFC 4301.";

        }

        grouping port-range  {
            description
                "This grouping defines a port range, such as
                 expressed in RFC 4301. For example: 1500 (Start
                 Port Number)-1600 (End Port Number).
                 A port range is used in the Traffic Selector.";

            leaf start {
                type inet:port-number;
                description "Start port number.";
            }
            leaf end {
                type inet:port-number;
                description
                    "End port number. The assigned value must be
                     equal or greater than the start port number.
                     To express a single port, set the same value
                     as start and end.";
            }
            reference "Section 4.4.1.2 in RFC 4301.";
        }

        grouping tunnel-grouping {
            description
                "The parameters required to define the IP tunnel
                 endpoints when IPsec SA requires tunnel mode. The
                 tunnel is defined by two endpoints: the local IP
                 address and the remote IP address.";

            leaf local {
```

```
                    type inet:ip-address;
                    mandatory true;
                    description
                        "Local IP address' tunnel endpoint.";
                }
                leaf remote {
                    type inet:ip-address;
                    mandatory true;
                    description
                        "Remote IP address' tunnel endpoint.";
                }
                leaf df-bit {
                    type enumeration {
                        enum clear {
                            description
                                "Disable the DF (Don't Fragment) bit
                                 from the outer header. This is the
                                 default value.";
                        }
                        enum set {
                            description
                                "Enable the DF bit in the outer header.";
                        }
                        enum copy {
                            description
                                "Copy the DF bit to the outer header.";
                        }
                    }
                    default clear;
                    description
                        "Allow configuring the DF bit when encapsulating
                         tunnel mode IPsec traffic. RFC 4301 describes
                         three options to handle the DF bit during
                         tunnel encapsulation: clear, set and copy from
                         the inner IP header.";
                    reference
                        "Section 8.1 in RFC 4301.";
                }
                leaf bypass-dscp {
                    type boolean;
                    default true;
                    description
                        "If DSCP (Differentiated Services Code Point)
                         values in the inner header have to be used to
                         select one IPsec SA among several that match
                         the traffic selectors for an outbound packet";
                    reference
                        "Section 4.4.2.1. in RFC 4301.";
```

```
            }
            leaf dscp-mapping {
                type yang:hex-string;
                default "00:00:00:00:00:00";
                description
                    "DSCP values allowed for packets carried over
                     this IPsec SA.";
                reference
                    "Section 4.4.2.1. in RFC 4301.";
            }
            leaf ecn {
                type boolean;
                default false;
                description
                    "Explicit Congestion Notification (ECN). If true
                     copy CE bits to inner header.";
                reference
                    "Section 5.1.2 and Annex C in RFC 4301.";
            }
        }

        grouping selector-grouping {
            description
                "This grouping contains the definition of a Traffic
                 Selector, which is used in the IPsec policies and
                 IPsec SAs.";

            leaf local-subnet {
                type inet:ip-prefix;
                mandatory true;
                description
                    "Local IP address subnet.";
            }
            leaf remote-subnet {
                type inet:ip-prefix;
                mandatory true;
                description
                    "Remote IP address subnet.";
            }
            leaf inner-protocol {
                type ipsec-inner-protocol;
                default any;
                description
                    "Inner Protocol that is going to be
                     protected with IPsec.";
            }
            list local-ports {
                key "start end";
```

```
                uses port-range;
                description
                   "List of local ports. When the inner
                    protocol is ICMP this 16 bit value
                    represents code and type.
                    If this list is not defined
                    it is assumed that start and
                    end are 0 by default (any port).";
            }
            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 represents
                     code and type.If this list is not defined
                      it is assumed that start and end are 0 by
                      default (any port)";
            }
            reference
                "Section 4.4.1.2 in RFC 4301.";
        }

        grouping ipsec-policy-grouping {
            description
                "Holds configuration information for an IPsec SPD
                 entry.";

            leaf anti-replay-window {
                type uint64;
                default 32;
                description
                    "A 64-bit counter used to determine whether an
                     inbound ESP packet is a replay.";
                reference
                    "Section 4.4.2.1 in RFC 4301.";
            }
            container traffic-selector {
                description
                    "Packets are selected for
                     processing actions based on the IP and inner
                     protocol header information, selectors,
                     matched against entries in the SPD.";
                uses selector-grouping;
                reference
                    "Section 4.4.4.1 in RFC 4301.";
            }
            container processing-info {
```

```
                    description
                        "SPD processing. If the required processing
                         action is protect, it contains the required
                         information to process the packet.";
                    leaf action {
                        type ipsec-spd-action;
                        default discard;
                        description
                            "If bypass or discard, container
                            ipsec-sa-cfg is empty.";
                    }
                    container ipsec-sa-cfg {
                        when "../action = 'protect'";
                        description
                            "IPsec SA configuration included in the SPD
                            entry.";
                        leaf pfp-flag {
                            type boolean;
                            default false;
                            description
                                "Each selector has a Populate From
                                 Packet (PFP) flag. If asserted for a
                                 given selector X, the flag indicates
                                 that the IPsec SA to be created should
                                 take its value (local IP address,
                                 remote IP address, Next Layer
                                 Protocol, etc.) for X from the value
                                 in the packet. Otherwise, the IPsec SA
                                 should take its value(s) for X from
                                 the value(s) in the SPD entry.";
                        }
                        leaf ext-seq-num {
                            type boolean;
                            default false;
                            description
                                "True if this IPsec SA is using extended
                                 sequence numbers. True 64 bit counter,
                                 False 32 bit.";
                        }
                        leaf seq-overflow {
                            type boolean;
                            default false;
                            description
                                "The flag indicating whether
                                 overflow of the sequence number
                                 counter should prevent transmission
                                 of additional packets on the IPsec
                                 SA (false) and, therefore needs to
```

```
                            be rekeyed, or whether rollover is
                            permitted (true). If Authenticated
                            Encryption with Associated Data
                            (AEAD) is used this flag MUST be
                            false.";
                    }
                    leaf stateful-frag-check {
                        type boolean;
                        default false;
                        description
                            "Indicates whether (true) or not (false)
                             stateful fragment checking applies to
                             the IPsec SA to be created.";
                    }
                    leaf mode {
                        type ipsec-mode;
                        default transport;
                        description
                            "IPsec SA has to be processed in
                             transport or tunnel mode.";
                    }
                    leaf protocol-parameters {
                        type ipsec-protocol-parameters;
                        default esp;
                        description
                            "Security protocol of the IPsec SA:
                            Only ESP is supported but it could be
                            extended in the future.";
                    }
                    container esp-algorithms {
                        when "../protocol-parameters = 'esp'";
                        description
                            "Configuration of Encapsulating
                            Security Payload (ESP) parameters and
                            algorithms.";

                        leaf-list integrity {
                            type integrity-algorithm-type;
                            default 0;
                            ordered-by user;
                            description
                                "Configuration of ESP authentication
                                based on the specified integrity
                                algorithm. With AEAD algorithms,
                                the integrity node is not
                                used.";
                            reference
                                "Section 3.2 in RFC 4303.";
```

```
                    }
                    list encryption {
                        key id;
                        ordered-by user;
                        leaf id {
                            type uint8;
                            description
                             "The index of list with the
                              different encryption algorithms and
                              its key-length (if required).";
                        }
                        leaf algorithm-type {
                         type nsfikec:encryption-algorithm-type;
                         default 20;
                         description
                           "Default value 20 (ENCR_AES_GCM_16)";
                        }
                        leaf key-length {
                            type uint16;
                            default 128;
                            description
                                "By default key length is 128
                                 bits";
                        }
                        description
                            "Encryption or AEAD algorithm for the
                             IPsec SAs. This list is ordered
                             following from the higher priority to
                             lower priority. First node of the
                             list will be the algorithm with
                             higher priority. In case the list
                             is empty, then
                             no encryption algorithm
                             is applied (NULL).";
                        reference
                            "Section 3.2 in RFC 4303.";
                    }

                    leaf tfc-pad {
                        type boolean;
                        default false;
                        description
                            "If Traffic Flow Confidentiality
                             (TFC) padding for ESP encryption
                             can be used (true) or not (false)";
                        reference
                            "Section 2.7 in RFC 4303.";
                    }
```

```
                        reference
                            "RFC 4303.";
                    }
                    container tunnel {
                        when "../mode = 'tunnel'";
                        uses tunnel-grouping;
                        description
                            "IPsec tunnel endpoints definition.";
                    }
                }
                reference
                    "Section 4.4.1.2 in RFC 4301.";
            }
            container spd-mark {
                    description
                        "The Mark to set for the IPsec SA of this
                         connection. This option is only available
                         on linux NETKEY/XFRM kernels. It can be
                         used with iptables to create custom
                         iptables rules using CONNMARK. It can also
                         be used with Virtual Tunnel Interfaces
                         (VTI) to direct marked traffic to
                         specific vtiXX devices.";
                    leaf mark {
                        type uint32;
                        default 0;
                        description
                            "Mark used to match XFRM policies and
                             states.";
                    }
                    leaf mask {
                        type yang:hex-string;
                        default 00:00:00:00;
                        description
                            "Mask used to match XFRM policies and
                             states.";
                    }
            }
        }
    }

    <CODE ENDS>
```

Appendix B. YANG model for IKE case

   This Appendix is Normative.

   This YANG module has normative references to [RFC2247], [RFC5280],
   [RFC4301], [RFC5280], [RFC5915], [RFC6991], [RFC7296], [RFC7383],
   [RFC7427], [RFC7619], [RFC8017], [RFC8174], [RFC8341], [ITU-T.X.690],
   [I-D.draft-ietf-netconf-crypto-types] and [IKEv2-Parameters].

   This YANG module has informative references to [RFC8229].

```
     <CODE BEGINS> file "ietf-i2nsf-ike@2020-10-21.yang"

     module ietf-i2nsf-ike {
         yang-version 1.1;
         namespace "urn:ietf:params:xml:ns:yang:ietf-i2nsf-ike";
         prefix "nsfike";

         import ietf-inet-types {
             prefix inet;
             reference "RFC 6991: Common YANG Data Types";
         }

         import ietf-yang-types {
             prefix yang;
             reference "RFC 6991: Common YANG Data Types";
         }

         import ietf-crypto-types {
             prefix ct;
             reference "RFC XXXX: YANG Data Types and Groupings
                        for Cryptography.";
         }

         import ietf-i2nsf-ikec {
             prefix nsfikec;
             reference
                 "Common Data model for SDN-based IPsec
                  configuration.";
         }

         import ietf-netconf-acm {
             prefix nacm;
             reference
                 "RFC 8341: Network Configuration Access Control
                  Model.";
```

```
        }

        organization "IETF I2NSF Working Group";

        contact
        "WG Web:  <https://datatracker.ietf.org/wg/i2nsf/>
         WG List: <mailto:i2nsf@ietf.org>

        Author: Rafael Marin-Lopez
                <mailto:rafa@um.es>

        Author: Gabriel Lopez-Millan
                <mailto:gabilm@um.es>

        Author: Fernando Pereniguez-Garcia
                <mailto:fernando.pereniguez@cud.upct.es>
        ";

        description

        "This module contains IPsec IKE case model for the SDN-based
         IPsec flow protection service. An NSF will implement this
         module.

        Copyright (c) 2020 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.

        The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
        'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
        'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this
        document are to be interpreted as described in BCP 14
        (RFC 2119) (RFC 8174) when, and only when, they appear
        in all capitals, as shown here.";

        revision "2020-10-21" {
            description "Initial version.";
            reference "RFC XXXX: Software-Defined Networking
            (SDN)-based IPsec Flow Protection.";
```

```
            }

            typedef ike-spi {
                type uint64 { range "0..max"; }
                description
                    "Security Parameter Index (SPI)'s IKE SA.";
                reference
                    "Section 2.6 in RFC 7296.";
            }

            typedef autostartup-type {
                type enumeration {
                    enum add {
                        description
                            "IKE/IPsec configuration is only loaded into
                             IKE implementation but IKE/IPsec SA is not
                             started.";
                    }
                    enum on-demand {
                        description
                            "IKE/IPsec configuration is loaded
                             into IKE implementation. The IPsec policies
                             are transferred to the NSF's kernel but the
                             IPsec SAs are not established immediately.
                             The IKE implementation will negotiate the
                             IPsec SAs when the NSF's kernel requests it
                             (i.e. through an ACQUIRE notification).";
                    }
                    enum start {
                        description "IKE/IPsec configuration is loaded
                             and transferred to the NSF's kernel, and the
                             IKEv2 based IPsec SAs are established
                             immediately without waiting any packet.";
                    }
                }
                description
                    "Different policies to set IPsec SA configuration
                     into NSF's kernel when IKEv2 implementation has
                     started.";
            }

            typedef pfs-group {
                type uint16;
                description
                    "DH groups for IKE and IPsec SA rekey.";
                reference
                    "Section 3.3.2 in RFC 7296. Transform Type 4 -
                     Diffie-Hellman Group Transform IDs in IANA Registry
```

```
                        - Internet Key Exchange Version 2 (IKEv2)
                         Parameters.";
            }

            typedef auth-protocol-type {
                type enumeration {
                    enum ikev2 {
                        value 2;
                        description
                            "IKEv2 authentication protocol. It is the
                             only defined right now. An enum is used for
                             further extensibility.";
                    }
                }
                description
                    "IKE authentication protocol version specified in the
                     Peer Authorization Database (PAD). It is defined as
                     enumerate to allow new IKE versions in the
                     future.";
                reference
                    "RFC 7296.";
            }

            typedef auth-method-type {
                type enumeration {
                    enum pre-shared {
                        description
                            "Select pre-shared key as the
                            authentication method.";
                        reference
                            "RFC 7296.";
                    }
                    enum eap {
                        description
                            "Select EAP as the authentication method.";
                        reference
                            "RFC 7296.";
                    }
                    enum digital-signature {
                        description
                            "Select digital signature method.";
                        reference
                            "RFC 7296 and RFC 7427.";
                    }
                    enum null {
                        description
                            "Null authentication.";
                        reference
```

```
                            "RFC 7619.";
                    }

            }
            description
                "Peer authentication method specified in the Peer
                 Authorization Database (PAD).";
        }

        container ipsec-ike {
            description
                "IKE configuration for a NSF. It includes PAD
                 parameters, IKE connections information and state
                 data.";

            container pad {
                description
                   "Configuration of Peer Authorization Database
                    (PAD). The PAD contains information about IKE
                    peer (local and remote). Therefore, the Security
                    Controller also stores authentication
                    information for this NSF and can include
                    several entries for the local NSF not only
                    remote peers. Storing local and remote
                    information makes possible to specify that this
                    NSF with identity A will use some particular
                    authentication with remote NSF with identity B
                    and what are the authentication mechanisms
                    allowed to B.";
                list pad-entry {
                    key "name";
                    ordered-by user;
                    description
                        "Peer Authorization Database (PAD) entry. It
                         is a list of PAD entries ordered by the
                         I2NSF Controller.";
                    leaf name {
                        type string;
                        description
                            "PAD unique name to identify this
                             entry.";
                    }
                    choice identity {
                        mandatory true;
                        description
                            "A particular IKE peer will be
                             identified by one of these identities.
                             This peer can be a remote peer or local
```

```
                            peer (this NSF).";
                reference
                    "Section 4.4.3.1 in RFC 4301.";
                case ipv4-address{
                    leaf ipv4-address {
                        type inet:ipv4-address;
                        description
                            "Specifies the identity as a
                             single four (4) octet.";
                    }
                }
                case ipv6-address{
                    leaf ipv6-address {
                        type inet:ipv6-address;
                        description
                            "Specifies the identity as a
                             single sixteen (16) octet IPv6
                             address. An example is
                             2001:DB8:0:0:8:800:200C:417A.";
                    }
                }
                case fqdn-string {
                    leaf fqdn-string {
                        type inet:domain-name;
                        description
                            "Specifies the identity as a
                             Fully-QualifiedDomain Name
                             (FQDN) string. An example is:
                             example.com. The string MUST
                             NOT contain any terminators
                             (e.g., NULL, CR, etc.).";
                    }
                }
                case rfc822-address-string {
                    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.).";
                        reference
                            "RFC 822.";
                    }
                }
```

```
                            case dnx509 {
                                leaf dnx509 {
                                    type string;
                                    description
                                        "Specifies the identity as a
                                         ASN.1 X.500 Distinguished
                                         Name. An example is
                                         C=US,O=Example
                                         Organisation,CN=John Smith.";
                                    reference
                                        "RFC 2247.";
                                }
                            }
                            case gnx509 {
                                leaf gnx509 {
                                    type string;
                                    description
                                        "ASN.1 X.509 GeneralName. RFC
                                         5280.";
                                }
                            }
                            case id-key {
                                leaf id-key {
                                    type string;
                                    description
                                        "Opaque octet stream that may be
                                         used to pass vendor-specific
                                         information for proprietary
                                         types of identification.";
                                    reference
                                        "Section 3.5 in RFC 7296.";
                                }
                            }
                            case id-null {
                                leaf id-null {
                                    type empty;
                                    description
                                        "ID_NULL identification used
                                         when IKE identification payload
                                         is not used." ;
                                    reference
                                        "RFC 7619.";
                                }
                            }
                        }
                      leaf auth-protocol {
                          type auth-protocol-type;
                          default ikev2;
```

```
                          description
                              "Only IKEv2 is supported right now but
                               other authentication protocols may be
                               supported in the future.";
                      }
                      container peer-authentication {
                          description
                              "This container allows the Security
                               Controller to configure the
                               authentication method (pre-shared key,
                               eap, digitial-signature, null) that
                               will use a particular peer and the
                               credentials, which will depend on the
                               selected authentication method.";
                          leaf auth-method {
                              type auth-method-type;
                              default pre-shared;
                              description
                                  "Type of authentication method
                                   (pre-shared, eap, digital signature,
                                    null).";
                              reference
                                  "Section 2.15 in RFC 7296.";
                          }
                          container eap-method {
                              when "../auth-method = 'eap'";
                               leaf eap-type {
                                   type uint8;
                                   mandatory true;
                                   description
                                       "EAP method type. This
                                        information provides the
                                        particular EAP method to be
                                        used. Depending on the EAP
                                        method, pre-shared keys or
                                        certificates may be used.";
                              }
                               description
                                  "EAP method description used when
                                   authentication method is 'eap'.";
                              reference
                                  "Section 2.16 in RFC 7296.";
                          }
                          container pre-shared {
                              when
                                  "../auth-method[.='pre-shared' or
                                   .='eap']";
                              leaf secret {
```

```
                        nacm:default-deny-all;
                        type yang:hex-string;
                        mandatory true;
                        description
                            "Pre-shared secret value. The
                             NSF has to prevent read access
                             to this value for security
                             reasons.";
                    }
                    description
                        "Shared secret value for PSK or
                         EAP method authentication based on
                         PSK.";
                }
                container digital-signature {
                    when
                     "../auth-method[.='digital-signature'
                    or .='eap']";
                    leaf ds-algorithm {
                        type uint8;
                        default 1;
                        description
                            "The digital signature
                            algorithm is specified with a
                            value extracted from the IANA
                            Registry. Depending on the
                            algorithm, the following leafs
                            must contain information. For
                            example if digital signature
                            involves a certificate then leaf
                            'cert-data' and 'private-key'
                            will contain this information.";
                        reference
                            "IKEv2 Authentication Method -
                             IANA Registry - Internet Key
                             Exchange Version 2 (IKEv2)
                             Parameters.";
                    }

                    choice public-key {
                        mandatory true;
                        leaf raw-public-key {
                            type binary;
                            description
                              "A binary that contains the
                               value of the public key.  The
                               interpretation of the content
                               is defined by the digital
```

```
                                    signature algorithm. For
                                    example, an RSA key is
                                    represented as RSAPublicKey as
                                    defined in RFC 8017, and an
                                    Elliptic Curve Cryptography
                                    (ECC) key is represented
                                    using the 'publicKey'
                                    described in RFC 5915.";
                              reference
                                 "RFC XXXX: YANG Data Types and
                                  Groupings for Cryptography.";
                           }

                           leaf cert-data {
                              type ct:x509;
                              description
                                 "X.509 certificate data -
                                  PEM4. If raw-public-key
                                  is defined this leaf is
                                  empty.";
                              reference
                                 "RFC XXXX: YANG Data Types and
                                  Groupings for Cryptography.";
                           }

                           description
                              "If the I2NSF Controller
                               knows that the NSF
                               already owns a private key
                               associated to this public key
                               (the NSF generated the pair
                               public key/private key out of
                               band), it will only configure
                               one of the leaf of this
                               choice but not the leaf
                               private-key. The NSF, based on
                               the public key value, can know
                               the private key to be used.";
                        }
                        leaf private-key {
                           nacm:default-deny-all;
                           type binary;
                           description
                              "A binary that contains the
                               value of the private key. The
                               interpretation of the content
                               is defined by the digital
                               signature algorithm. For
```

```
                            example, an RSA key is
                            represented as RSAPrivateKey as
                            defined in RFC 8017, and an
                            Elliptic Curve Cryptography
                            (ECC) key is represented as
                            ECPrivateKey as defined in RFC
                            5915. This value is set
                            if public-key is defined and
                            I2NSF controller is in charge
                            of configuring the
                            private-key. Otherwise, it is
                            not set and the value is
                            kept in secret.";
                      reference
                            "RFC XXXX: YANG Data Types and
                             Groupings for Cryptography.";
                }
                leaf-list ca-data {
                    type ct:x509;
                    description
                        "List of trusted Certification
                        Authorities (CA) certificates
                        encoded using ASN.1
                        distinguished encoding rules
                        (DER). If it is not defined
                        the default value is empty.";
                    reference
                            "RFC XXXX: YANG Data Types and
                             Groupings for Cryptography.";
                }
                leaf crl-data {
                    type ct:crl;
                    description
                      "A CertificateList structure, as
                        specified in RFC 5280,
                        encoded using ASN.1
                        distinguished encoding rules
                        (DER),as specified in ITU-T
                        X.690. If it is not defined
                        the default value is empty.";
                    reference
                            "RFC XXXX: YANG Data Types and
                             Groupings for Cryptography.";
                }
                leaf crl-uri  {
                    type inet:uri;
                    description
                        "X.509 CRL certificate URI.
```

```
                                If it is not defined
                                the default value is empty.";
                    }
                    leaf oscp-uri {
                        type inet:uri;
                        description
                            "OCSP URI.
                            If it is not defined
                            the default value is empty.";
                    }
                    description
                        "Digital Signature container.";

            } /*container digital-signature*/
        } /*container peer-authentication*/
    }
}

list conn-entry {
    key "name";
    description
        "IKE peer connection information. This list
        contains the IKE connection for this peer
        with other peers. This will be translated in
        real time by IKE Security Associations
        established with these nodes.";
    leaf name {
        type string;
        description
            "Identifier for this connection
             entry.";
    }
    leaf autostartup {
        type autostartup-type;
        default add;
        description
            "By-default: Only add configuration
             without starting the security
             association.";
    }
    leaf initial-contact {
        type boolean;
        default false;
        description
            "The goal of this value is to deactivate the
             usage of INITIAL_CONTACT notification
             (true). If this flag remains to false it
             means the usage of the INITIAL_CONTACT
```

```
                    notification will depend on the IKEv2
                    implementation.";
            }
            leaf version {
                type auth-protocol-type;
                default ikev2;
                description
                    "IKE version. Only version 2 is supported
                    so far.";
            }
            leaf fragmentation {
                type boolean;
                default false;
                description
                    "Whether or not to enable IKE
                     fragmentation as per RFC 7383 (true or
                     false).";
                reference
                    "RFC 7383.";
            }
            container ike-sa-lifetime-soft {
                description
                    "IKE SA lifetime soft. Two lifetime values
                     can be configured: either rekey time of the
                     IKE SA or reauth time of the IKE SA. When
                     the rekey lifetime expires a rekey of the
                     IKE SA starts. When reauth lifetime
                     expires a IKE SA reauthentication starts.";
                leaf rekey-time {
                    type uint32;
                    default 0;
                    description
                        "Time in seconds between each IKE SA
                        rekey.The value 0 means infinite.";
                }
                leaf reauth-time {
                    type uint32;
                    default 0;
                    description
                      "Time in seconds between each IKE SA
                       reauthentication. The value 0 means
                       infinite.";
                }
                reference
                    "Section 2.8 in RFC 7296.";
            }
            container ike-sa-lifetime-hard {
                description
```

```
                        "Hard IKE SA lifetime. When this
                         time is reached the IKE SA is removed.";
                  leaf over-time {
                      type uint32;
                      default 0;
                      description
                          "Time in seconds before the IKE SA is
                           removed. The value 0 means infinite.";
                  }
                  reference
                      "RFC 7296.";
              }
              leaf-list authalg {
                  type nsfikec:integrity-algorithm-type;
                  default 12;
                  ordered-by user;
                  description
                      "Authentication algorithm for establishing
                       the IKE SA. This list is ordered following
                       from the higher priority to lower priority.
                       First node of the list will be the algorithm
                       with higher priority.";
              }

              list encalg {
                  key id;
                  min-elements 1;
                  ordered-by user;
                  leaf id {
                      type uint8;
                      description
                          "The index of the list with the
                           different encryption algorithms and its
                           key-length (if required). E.g. AES-CBC,
                           128 bits";
                  }
                  leaf algorithm-type {
                      type nsfikec:encryption-algorithm-type;
                      default 12;
                      description
                          "Default value 12 (ENCR_AES_CBC)";
                  }
                  leaf key-length {
                      type uint16;
                      default 128;
                      description
                          "By default key length is 128 bits";
                  }
```

```
                        description
                            "Encryption or AEAD algorithm for the IKE
                             SAs. This list is ordered following
                             from the higher priority to lower priority.
                             First node of the list will be the algorithm
                             with higher priority.";
                    }
                    leaf dh-group {
                        type pfs-group;
                        default 14;
                        description
                            "Group number for Diffie-Hellman
                             Exponentiation used during IKE_SA_INIT
                             for the IKE SA key exchange.";
                    }
                    leaf half-open-ike-sa-timer {
                        type uint32;
                        default 0;
                        description
                            "Set the half-open IKE SA timeout
                              duration.";
                        reference
                            "Section 2 in RFC 7296.";
                    }

                    leaf half-open-ike-sa-cookie-threshold {
                        type uint32;
                        default 0;
                        description
                            "Number of half-open IKE SAs that activate
                              the cookie mechanism." ;
                        reference
                            "Section 2.6 in RFC 7296.";
                    }
                    container local {
                        leaf local-pad-entry-name {
                            type string;
                            mandatory true;
                            description
                                "Local peer authentication information.
                                 This node points to a specific entry in
                                 the PAD where the authorization
                                 information about this particular local
                                 peer is stored. It MUST match a
                                 pad-entry-name.";
                        }
                        description
                            "Local peer authentication information.";
```

```
                        }
                        container remote {
                            leaf remote-pad-entry-name {
                                type string;
                                mandatory true;
                                description
                                    "Remote peer authentication information.
                                     This node points to a specific entry in
                                     the PAD where the authorization
                                     information about this particular
                                     remote peer is stored. It MUST match a
                                     pad-entry-name.";
                            }
                            description
                                "Remote peer authentication information.";
                        }
                        container encapsulation-type
                        {
                            uses nsfikec:encap;
                            description
                                "This container carries configuration
                                 information about the source and destination
                                 ports of encapsulation that IKE should use
                                 and the type of encapsulation that
                                 should use when NAT traversal is required.
                                 However, this is just a best effort since
                                 the IKE implementation may need to use a
                                 different encapsulation as
                                 described in RFC 8229.";
                            reference
                                "RFC 8229.";
                        }
                        container spd {
                            description
                                "Configuration of the Security Policy
                                 Database (SPD). This main information is
                                 placed in the grouping
                                 ipsec-policy-grouping.";
                            list spd-entry {
                                key "name";
                                ordered-by user;
                                leaf name {
                                    type string;
                                    description
                                        "SPD entry unique name to identify
                                         the IPsec policy.";
                                }
                                container ipsec-policy-config {
```

```
                          description
                              "This container carries the
                              configuration of a IPsec policy.";
                          uses nsfikec:ipsec-policy-grouping;
                      }
                      description
                          "List of entries which will constitute
                          the representation of the SPD. Since we
                          have IKE in this case, it is only
                          required to send a IPsec policy from
                          this NSF where 'local' is this NSF and
                          'remote' the other NSF. The IKE
                          implementation will install IPsec
                          policies in the NSF's kernel in both
                          directions (inbound and outbound) and
                          their corresponding IPsec SAs based on
                          the information in this SPD entry.";
                  }
                  reference
                      "Section 2.9 in RFC 7296.";
              }
              container child-sa-info {
                  leaf-list pfs-groups {
                      type pfs-group;
                      default 0;
                      ordered-by user;
                      description
                          "If non-zero, it is required perfect
                           forward secrecy when requesting new
                           IPsec SA. The non-zero value is
                           the required group number. This list is
                           ordered following from the higher
                           priority to lower priority. First node
                           of the list will be the algorithm
                           with higher priority.";
                  }
                  container child-sa-lifetime-soft {
                      description
                          "Soft IPsec SA lifetime soft.
                           After the lifetime the action is
                           defined in this container
                           in the leaf action.";
                      uses nsfikec:lifetime;
                      leaf action {
                          type nsfikec:lifetime-action;
                          default replace;
                          description
                              "When the lifetime of an IPsec SA
```

```
                                expires an action needs to be
                                performed over the IPsec SA that
                                reached the lifetime. There are
                                three possible options:
                                terminate-clear, terminate-hold and
                                replace.";
                    reference
                        "Section 4.5 in RFC 4301 and Section 2.8
                         in RFC 7296.";
                    }
                }
                container child-sa-lifetime-hard {
                    description
                        "IPsec SA lifetime hard. The action will
                         be to terminate the IPsec SA.";
                    uses nsfikec:lifetime;
                    reference
                        "Section 2.8 in RFC 7296.";
                }
                description
                    "Specific information for IPsec SAs
                     SAs. It includes PFS group and IPsec SAs
                     rekey lifetimes.";
            }
            container state {
                config false;

                leaf initiator {
                    type boolean;
                    description
                        "It is acting as initiator for this
                         connection.";
                }
                leaf initiator-ikesa-spi {
                    type ike-spi;
                    description
                        "Initiator's IKE SA SPI.";
                }
                leaf responder-ikesa-spi {
                    type ike-spi;
                    description
                        "Responder's IKE SA SPI.";
                }
                leaf nat-local {
                    type boolean;
                    description
                        "True, if local endpoint is behind a
                         NAT.";
```

```
                        }
                        leaf nat-remote {
                            type boolean;
                            description
                                "True, if remote endpoint is behind
                                a NAT.";
                        }

                        container encapsulation-type
                        {
                            uses nsfikec:encap;
                            description
                                "This container provides information
                                about the source and destination
                                ports of encapsulation that IKE is
                                using, and the type of encapsulation
                                when NAT traversal is required.";
                            reference
                                "RFC 8229.";
                        }
                        leaf established {
                            type uint64;
                            description
                                "Seconds since this IKE SA has been
                                 established.";
                        }
                        leaf current-rekey-time {
                            type uint64;
                            description
                                "Seconds before IKE SA must be rekeyed.";
                        }
                        leaf current-reauth-time {
                            type uint64;
                            description
                                "Seconds before IKE SA must be
                                 re-authenticated.";
                        }
                        description
                            "IKE state data for a particular
                             connection.";
                    } /* ike-sa-state */
                } /* ike-conn-entries */

                container number-ike-sas {
                    config false;
                    leaf total {
                        type uint64;
                        description
```

```
                            "Total number of active IKE SAs.";
                    }
                    leaf half-open {
                        type uint64;
                        description
                            "Number of half-open active IKE SAs.";
                    }
                    leaf half-open-cookies {
                        type uint64;
                        description
                            "Number of half open active IKE SAs with
                             cookie activated.";
                    }
                    description
                        "General information about the IKE SAs. In
                        particular, it provides the current number of
                        IKE SAs.";
               }
          }  /* container ipsec-ike */
     }

     <CODE ENDS>
```

Appendix C. YANG model for IKE-less case

   This Appendix is Normative.

   This YANG module has normative references to [RFC4301], [RFC6991],
   [RFC8174] and [RFC8341].

```
     <CODE BEGINS> file "ietf-i2nsf-ikeless@2020-10-21.yang"

     module ietf-i2nsf-ikeless {

         yang-version 1.1;
         namespace "urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless";

         prefix "nsfikels";

         import ietf-yang-types {
             prefix yang;
             reference "RFC 6991: Common YANG Data Types";
         }
```

```
import ietf-i2nsf-ikec {
    prefix nsfikec;
    reference
        "Common Data model for SDN-based IPsec
         configuration.";
}

import ietf-netconf-acm {
    prefix nacm;
    reference
        "RFC 8341: Network Configuration Access Control
         Model.";
}

organization "IETF I2NSF Working Group";

contact
"WG Web:  <https://datatracker.ietf.org/wg/i2nsf/>
 WG List: <mailto:i2nsf@ietf.org>

Author: Rafael Marin-Lopez
        <mailto:rafa@um.es>

Author: Gabriel Lopez-Millan
        <mailto:gabilm@um.es>

Author: Fernando Pereniguez-Garcia
        <mailto:fernando.pereniguez@cud.upct.es>
";

description
    "Data model for IKE-less case in the SDN-base IPsec flow
     protection service.

     Copyright (c) 2020 IETF Trust and the persons
     identified as authors of the code.  All rights reserved.
     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the
     Simplified BSD License set forth in Section 4.c of the
     IETF Trust's Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX;;
     see the RFC itself for full legal notices.

     The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
     'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
```

```
              'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this
              document are to be interpreted as described in BCP 14
              (RFC 2119) (RFC 8174) when, and only when, they appear
              in all capitals, as shown here.";

      revision "2020-10-21" {
          description "Initial version.";
          reference "RFC XXXX: Software-Defined Networking
          (SDN)-based IPsec Flow Protection.";
      }

      feature ikeless-notification {
          description
              "To ensure broader applicability of this module,
              the notifications are marked as a feature.
              For the implementation of ikeless case,
              the NSF is expected to implement this
              feature.";
      }

      container ipsec-ikeless {
          description
              "Container for configuration of the IKE-less
               case. The container contains two additional
               containers: 'spd' and 'sad'. The first allows the
               I2NSF Controller to configure IPsec policies in
               the Security Policy Database SPD, and the second
               allows to configure IPsec Security Associations
               (IPsec SAs) in the Security Association Database
               (SAD).";
          reference "RFC 4301.";
          container spd {
              description
                  "Configuration of the Security Policy Database
                   (SPD.)";
              reference "Section 4.4.1.2 in RFC 4301.";

              list spd-entry {
                  key "name";
                  ordered-by user;
                  leaf name {
                      type string;
                      description
                          "SPD entry unique name to identify this
                           entry.";
                  }
                  leaf direction {
                      type nsfikec:ipsec-traffic-direction;
```

```
                        mandatory true;
                        description
                            "Inbound traffic or outbound
                             traffic. In the IKE-less case the
                             I2NSF Controller needs to
                             specify the policy direction to be
                             applied in the NSF. In the IKE case
                             this direction does not need to be
                             specified since IKE
                             will determine the direction that
                             IPsec policy will require.";
                    }
                    leaf reqid {
                        type uint64;
                        default 0;
                        description
                            "This value allows to link this
                             IPsec policy with IPsec SAs with the
                             same reqid. It is only required in
                             the IKE-less model since, in the IKE
                             case this link is handled internally
                             by IKE.";
                    }

                    container ipsec-policy-config {
                        description
                            "This container carries the
                             configuration of a IPsec policy.";
                        uses nsfikec:ipsec-policy-grouping;
                    }
                    description
                        "The SPD is represented as a list of SPD
                         entries, where each SPD entry represents an
                         IPsec policy.";
                } /*list spd-entry*/
            } /*container spd*/

            container sad {
                description
                    "Configuration of the IPsec Security Association
                     Database (SAD)";
                reference "Section 4.4.2.1 in RFC 4301.";
                list sad-entry {
                    key "name";
                    ordered-by user;
                    leaf name {
                        type string;
                        description
```

```
                              "SAD entry unique name to identify this
                               entry.";
                     }
                     leaf reqid {
                         type uint64;
                         default 0;
                         description
                             "This value allows to link this
                              IPsec SA with an IPsec policy with
                              the same reqid.";
                     }

                     container ipsec-sa-config {
                         description
                             "This container allows configuring
                             details of an IPsec SA.";
                         leaf spi {
                             type uint32 { range "0..max"; }
                             mandatory true;
                             description
                                 "Security Parameter Index (SPI)'s
                                  IPsec SA.";
                         }
                         leaf ext-seq-num {
                             type boolean;
                             default true;
                             description
                                 "True if this IPsec SA is using
                                  extended sequence numbers. True 64
                                  bit counter, FALSE 32 bit.";
                         }
                         leaf seq-number-counter {
                             type uint64;
                             default 0;
                             description
                                     "A 64-bit counter when this IPsec
                                     SA is using Extended Sequence
                                     Number or 32-bit counter when it
                                     is not. It used to generate the
                                     initial Sequence Number field
                                     in ESP headers.";
                         }
                         leaf seq-overflow {
                             type boolean;
                             default false;
                             description
                                 "The flag indicating whether
                                  overflow of the sequence number
```

```
                          counter should prevent transmission
                          of additional packets on the IPsec
                          SA (false) and, therefore needs to
                          be rekeyed, or whether rollover is
                          permitted (true). If Authenticated
                          Encryption with Associated Data
                          (AEAD) is used this flag MUST BE
                          false.";
            }
            leaf anti-replay-window {
                type uint32;
                default 32;
                description
                    "A 32-bit counter and a bit-map (or
                     equivalent) used to determine
                     whether an inbound ESP packet is a
                     replay. If set to 0 no anti-replay
                     mechanism is performed.";
            }
            container traffic-selector {
                uses nsfikec:selector-grouping;
                description
                    "The IPsec SA traffic selector.";
            }
            leaf protocol-parameters {
                type nsfikec:ipsec-protocol-parameters;
                default esp;
                description
                    "Security protocol of IPsec SA: Only
                     ESP so far.";
            }
            leaf mode {
                type nsfikec:ipsec-mode;
                default transport;
                description
                    "Tunnel or transport mode.";
            }
            container esp-sa {
              when "../protocol-parameters =
             'esp'";
               description
                    "In case the IPsec SA is
                     Encapsulation Security Payload
                     (ESP), it is required to specify
                     encryption and integrity
                     algorithms, and key material.";

              container encryption {
```

```
                    description
                            "Configuration of encryption or
                             AEAD algorithm for IPsec
                             Encapsulation Security Payload
                             (ESP).";

                    leaf encryption-algorithm {
                     type nsfikec:encryption-algorithm-type;
                     default 12;
                     description
                                "Configuration of ESP
                                 encryption. With AEAD
                                 algorithms, the integrity
                                 leaf is not used.";
                    }

                    leaf key {
                            nacm:default-deny-all;
                            type yang:hex-string;
                            description
                                "ESP encryption key value.
                                If this leaf is not defined
                                the key is not defined
                                (e.g. encryption is NULL).
                                The key length is
                                determined by the
                                length of the key set in
                                this leaf. By default is
                                128 bits.";
                    }
                    leaf iv {
                            nacm:default-deny-all;
                            type yang:hex-string;
                            description
                                "ESP encryption IV value. If
                                this leaf is not defined the
                                IV is not defined (e.g.
                                encryption is NULL)";
                    }
                  }

                  container integrity {
                    description
                            "Configuration of integrity for
                             IPsec Encapsulation Security
                             Payload (ESP). This container
                             allows to configure integrity
                             algorithm when no AEAD
```

```
                            algorithms are used, and
                            integrity is required.";
                    leaf integrity-algorithm {
                        type nsfikec:integrity-algorithm-type;
                        default 12;
                        description
                                "Message Authentication Code
                                (MAC) algorithm to provide
                                integrity in ESP
                                (default
                                AUTH_HMAC_SHA2_256_128).
                                With AEAD algorithms,
                                the integrity leaf is not
                                used.";
                    }
                    leaf key {
                            nacm:default-deny-all;
                            type yang:hex-string;
                            description
                                "ESP integrity key value.
                                If this leaf is not defined
                                the key is not defined (e.g.
                                AEAD algorithm is chosen and
                                integrity algorithm is not
                                required). The key length is
                                determined by the length of
                                the key configured.";
                    }
                  }
                } /*container esp-sa*/

                container sa-lifetime-hard {
                    description
                        "IPsec SA hard lifetime. The action
                        associated is terminate and
                        hold.";
                    uses nsfikec:lifetime;
                }
                container sa-lifetime-soft {
                    description
                        "IPsec SA soft lifetime.";
                    uses nsfikec:lifetime;
                    leaf action {
                        type nsfikec:lifetime-action;
                        description
                            "Action lifetime:
                             terminate-clear,
                             terminate-hold or replace.";
```

```
                    }
                }
                container tunnel {
                    when "../mode = 'tunnel'";
                    uses nsfikec:tunnel-grouping;
                    description
                        "Endpoints of the IPsec tunnel.";
                }
                container encapsulation-type
                {
                    uses nsfikec:encap;
                    description
                        "This container carries
                         configuration information about
                         the source and destination ports
                         which will be used for ESP
                         encapsulation that ESP packets the
                         type of encapsulation when NAT
                         traversal is in place.";
                }
            } /*ipsec-sa-config*/

            container ipsec-sa-state {
                config false;
                description
                    "Container describing IPsec SA state
                    data.";
                container sa-lifetime-current {
                    uses nsfikec:lifetime;
                    description
                        "SAD lifetime current.";
                }
                container replay-stats {
                    description
                        "State data about the anti-replay
                         window.";
                    leaf replay-window {
                        type uint64;
                        description
                            "Current state of the replay
                             window.";
                    }
                    leaf packet-dropped {
                        type uint64;
                        description
                            "Packets detected out of the
                             replay window and dropped
                             because they are replay
```

```
                                      packets.";
                            }
                            leaf failed {
                                type uint32;
                                description
                                    "Number of packets detected out
                                     of the replay window.";
                            }
                            leaf seq-number-counter {
                                type uint64;
                                description
                                    "A 64-bit counter when this
                                     IPsec SA is using Extended
                                     Sequence Number or 32-bit
                                     counter when it is not.
                                     Current value of sequence
                                     number.";
                            }
                        } /* container replay-stats*/
                    } /*ipsec-sa-state*/

                    description
                        "List of SAD entries that conforms the SAD.";
                } /*list sad-entry*/
            } /*container sad*/
        }/*container ipsec-ikeless*/

        /* Notifications */
        notification sadb-acquire {
            if-feature ikeless-notification;
            description
                "An IPsec SA is required. The traffic-selector
                 container contains information about the IP packet
                 that triggers the acquire notification.";
            leaf ipsec-policy-name {
                type string;
                mandatory true;
                description
                    "It contains the SPD entry name (unique) of
                     the IPsec policy that hits the IP packet
                     required IPsec SA. It is assumed the
                     I2NSF Controller will have a copy of the
                     information of this policy so it can
                     extract all the information with this
                     unique identifier. The type of IPsec SA is
                     defined in the policy so the Security
                     Controller can also know the type of IPsec
                     SA that must be generated.";
```

```
              }
              container traffic-selector {
                  description
                      "The IP packet that triggered the acquire
                       and requires an IPsec SA. Specifically it
                       will contain the IP source/mask and IP
                       destination/mask; protocol (udp, tcp,
                       etc...); and source and destination
                       ports.";
                  uses nsfikec:selector-grouping;
               }
          }

          notification sadb-expire {
              if-feature ikeless-notification;
              description "An IPsec SA expiration (soft or hard).";
              leaf ipsec-sa-name {
                  type string;
                  mandatory true;
                  description
                      "It contains the SAD entry name (unique) of
                       the IPsec SA that has expired.  It is assumed
                       the I2NSF Controller will have a copy of the
                       IPsec SA information (except the cryptographic
                       material and state data) indexed by this name
                       (unique identifier) so it can know all the
                       information (crypto algorithms, etc.) about
                       the IPsec SA that has expired in order to
                       perform a rekey (soft lifetime) or delete it
                       (hard lifetime) with this unique identifier.";
              }
              leaf soft-lifetime-expire {
                  type boolean;
                  default true;
                  description
                      "If this value is true the lifetime expired is
                       soft. If it is false is hard.";
              }
              container lifetime-current {
                  description
                      "IPsec SA current lifetime. If
                       soft-lifetime-expired is true this container is
                       set with the lifetime information about current
                       soft lifetime.";
                  uses nsfikec:lifetime;
              }
          }
          notification sadb-seq-overflow {
```

```
                if-feature ikeless-notification;
                description "Sequence overflow notification.";
                leaf ipsec-sa-name {
                    type string;
                    mandatory true;
                    description
                        "It contains the SAD entry name (unique) of
                         the IPsec SA that is about to have sequence
                         number overflow and rollover is not permitted.
                         It is assumed the I2NSF Controller will have
                         a copy of the IPsec SA information (except the
                         cryptographic material and state data) indexed
                         by this name (unique identifier) so the it can
                         know all the information (crypto algorithms,
                         etc.) about the IPsec SA that has expired in
                         order to perform a rekey of the IPsec SA.";
                }
            }
            notification sadb-bad-spi {
                if-feature ikeless-notification;
                description
                    "Notify when the NSF receives a packet with an
                     incorrect SPI (i.e. not present in the SAD).";
                leaf spi {
                    type uint32 { range "0..max"; }
                    mandatory true;
                    description
                        "SPI number contained in the erroneous IPsec
                         packet.";
                }
            }
        }

        <CODE ENDS>
```

Appendix D. XML configuration example for IKE case (gateway-to-gateway)

This example shows a XML configuration file sent by the I2NSF
Controller to establish a IPsec Security Association between two NSFs
(see Figure 3) in tunnel mode (gateway-to-gateway) with ESP,
authentication based on X.509 certificates and applying the IKE case.

```
                         +-----------------+
                         | I2NSF Controller |
                         +-----------------+
                    I2NSF NSF-Facing |
                          Interface |
                /----------------+----------------\
               /                                   \
              /                                     \
      +----+  +--------+                      +--------+  +----+
      | h1 |--| nsf_h1 |== IPsec_ESP_Tunnel_mode == | nsf_h2 |--| h2 |
      +----+  +--------+                      +--------+  +----+
         :1          :100                       :200        :1

    (2001:DB8:1:/64)          (2001:DB8:123:/64)       (2001:DB8:2:/64)
```
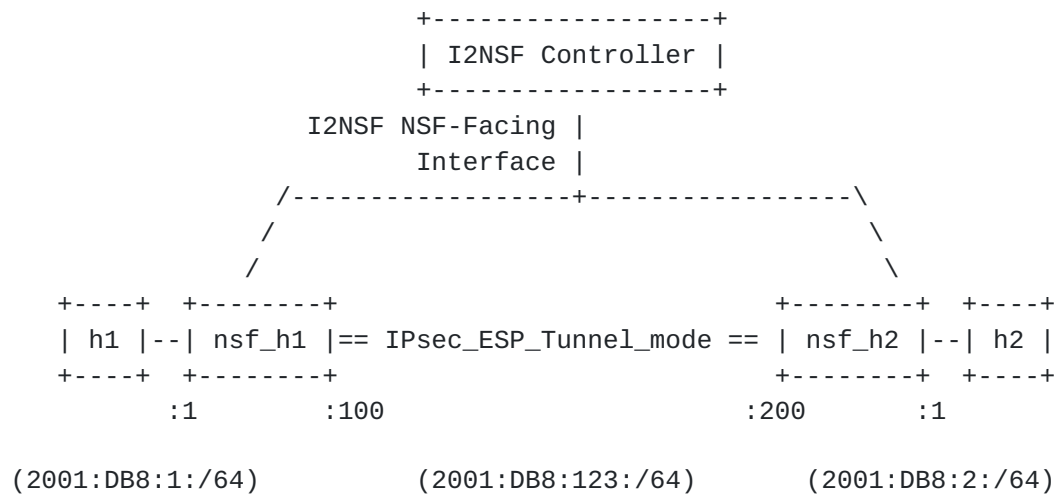
Figure 3: IKE case, tunnel mode , X.509 certificate authentication.

```
<ipsec-ike xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-ike"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <pad>
    <pad-entry>
      <name>nsf_h1_pad</name>
      <ipv6-address>2001:DB8:123::100</ipv6-address>
      <peer-authentication>
         <auth-method>digital-signature</auth-method>
         <digital-signature>
            <cert-data>base64encodedvalue==</cert-data>
            <private-key>base64encodedvalue==</private-key>
            <ca-data>base64encodedvalue==</ca-data>
         </digital-signature>
      </peer-authentication>
    </pad-entry>
    <pad-entry>
      <name>nsf_h2_pad</name>
      <ipv6-address>2001:DB8:123::200</ipv6-address>
      <auth-protocol>ikev2</auth-protocol>
      <peer-authentication>
        <auth-method>digital-signature</auth-method>
        <digital-signature>
          <!-- RSA Digital Signature -->
          <ds-algorithm>1</ds-algorithm>
          <cert-data>base64encodedvalue==</cert-data>
          <ca-data>base64encodedvalue==</ca-data>
        </digital-signature>
      </peer-authentication>
    </pad-entry>
```

```
        </pad>
        <conn-entry>
           <name>nsf_h1-nsf_h2</name>
           <autostartup>start</autostartup>
           <version>ikev2</version>
           <initial-contact>false</initial-contact>
           <fragmentation>true</fragmentation>
           <ike-sa-lifetime-soft>
              <rekey-time>60</rekey-time>
              <reauth-time>120</reauth-time>
           </ike-sa-lifetime-soft>
           <ike-sa-lifetime-hard>
              <over-time>3600</over-time>
           </ike-sa-lifetime-hard>
           <!--AUTH_HMAC_SHA1_160-->
           <authalg>7</authalg>
           <!--ENCR_AES_CBC - 128 bits-->
           <encalg>
              <id>1</id>
           </encalg>
           <!--8192-bit MODP Group-->
           <dh-group>18</dh-group>
           <half-open-ike-sa-timer>30</half-open-ike-sa-timer>
           <half-open-ike-sa-cookie-threshold>
              15
           </half-open-ike-sa-cookie-threshold>
           <local>
               <local-pad-entry-name>nsf_h1_pad</local-pad-entry-name>
           </local>
           <remote>
               <remote-pad-entry-name>nsf_h2_pad</remote-pad-entry-name>
           </remote>
           <spd>
             <spd-entry>
                <name>nsf_h1-nsf_h2</name>
                <ipsec-policy-config>
                  <anti-replay-window>32</anti-replay-window>
                  <traffic-selector>
                     <local-subnet>2001:DB8:1::0/64</local-subnet>
                     <remote-subnet>2001:DB8:2::0/64</remote-subnet>
                     <inner-protocol>any</inner-protocol>
                     <local-ports>
                       <start>0</start>
                       <end>0</end>
                     </local-ports>
                     <remote-ports>
                       <start>0</start>
                       <end>0</end>
```

```
                </remote-ports>
              </traffic-selector>
              <processing-info>
                 <action>protect</action>
                 <ipsec-sa-cfg>
                    <pfp-flag>false</pfp-flag>
                    <ext-seq-num>true</ext-seq-num>
                    <seq-overflow>false</seq-overflow>
                    <stateful-frag-check>false</stateful-frag-check>
                    <mode>tunnel</mode>
                    <protocol-parameters>esp</protocol-parameters>
                    <esp-algorithms>
                       <!-- AUTH_HMAC_SHA1_96 -->
                       <integrity>2</integrity>
                        <encryption>
                            <!-- ENCR_AES_CBC -->
                            <id>1</id>
                            <algorithm-type>12</algorithm-type>
                            <key-length>128</key-length>
                         </encryption>
                         <encryption>
                            <!-- ENCR_3DES-->
                            <id>2</id>
                            <algorithm-type>3</algorithm-type>
                         </encryption>
                       <tfc-pad>false</tfc-pad>
                     </esp-algorithms>
                     <tunnel>
                        <local>2001:DB8:123::100</local>
                        <remote>2001:DB8:123::200</remote>
                        <df-bit>clear</df-bit>
                        <bypass-dscp>true</bypass-dscp>
                        <ecn>false</ecn>
                     </tunnel>
                  </ipsec-sa-cfg>
               </processing-info>
            </ipsec-policy-config>
         </spd-entry>
      </spd>
      <child-sa-info>
         <!--8192-bit MODP Group -->
         <pfs-groups>18</pfs-groups>
         <child-sa-lifetime-soft>
            <bytes>1000000</bytes>
            <packets>1000</packets>
            <time>30</time>
            <idle>60</idle>
            <action>replace</action>
```

```
            </child-sa-lifetime-soft>
            <child-sa-lifetime-hard>
               <bytes>2000000</bytes>
               <packets>2000</packets>
               <time>60</time>
               <idle>120</idle>
            </child-sa-lifetime-hard>
         </child-sa-info>
      </conn-entry>
   </ipsec-ike>
```

Appendix E. XML configuration example for IKE-less case (host-to-host)

   This example shows a XML configuration file sent by the I2NSF
   Controller to establish a IPsec Security Association between two NSFs
   (see Figure 4) in transport mode (host-to-host) with ESP, and
   applying the IKE-less case.

```
                     +------------------+
                     | I2NSF Controller |
                     +------------------+
                 I2NSF NSF-Facing |
                       Interface |
            /-------------------+------------------\
           /                                        \
          /                                          \
     +--------+                                  +--------+
     | nsf_h1 |===== IPsec_ESP_Transport_mode =====| nsf_h2 |
     +--------+                                  +--------+
          :100        (2001:DB8:123:/64)         :200
```

                 Figure 4: IKE-less case, transport mode.

```
   <ipsec-ikeless
     xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless"
     xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
     <spd>
       <spd-entry>
          <name>
             in/trans/2001:DB8:123::200/2001:DB8:123::100
          </name>
          <direction>inbound</direction>
          <reqid>1</reqid>
```

```
            <ipsec-policy-config>
               <traffic-selector>
                 <local-subnet>2001:DB8:123::200/128</local-subnet>
                 <remote-subnet>2001:DB8:123::100/128</remote-subnet>
                 <inner-protocol>any</inner-protocol>
                    <local-ports>
                       <start>0</start>
                       <end>0</end>
                    </local-ports>
                    <remote-ports>
                       <start>0</start>
                       <end>0</end>
                     </remote-ports>
               </traffic-selector>
               <processing-info>
                  <action>protect</action>
                  <ipsec-sa-cfg>
                    <ext-seq-num>true</ext-seq-num>
                    <seq-overflow>true</seq-overflow>
                    <mode>transport</mode>
                    <protocol-parameters>esp</protocol-parameters>
                    <esp-algorithms>
                       <!--AUTH_HMAC_SHA1_96-->
                       <integrity>2</integrity>
                       <!--ENCR_AES_CBC -->
                       <encryption>
                         <id>1</id>
                         <algorithm-type>12</algorithm-type>
                          <key-length>128</key-length>
                       </encryption>
                       <encryption>
                         <id>2</id>
                         <algorithm-type>3</algorithm-type>
                       </encryption>
                    </esp-algorithms>
                  </ipsec-sa-cfg>
               </processing-info>
            </ipsec-policy-config>
          </spd-entry>
          <spd-entry>
            <name>out/trans/2001:DB8:123::100/2001:DB8:123::200</name>
            <direction>outbound</direction>
            <reqid>1</reqid>
            <ipsec-policy-config>
              <traffic-selector>
                 <local-subnet>2001:DB8:123::100/128</local-subnet>
                 <remote-subnet>2001:DB8:123::200/128</remote-subnet>
                 <inner-protocol>any</inner-protocol>
```
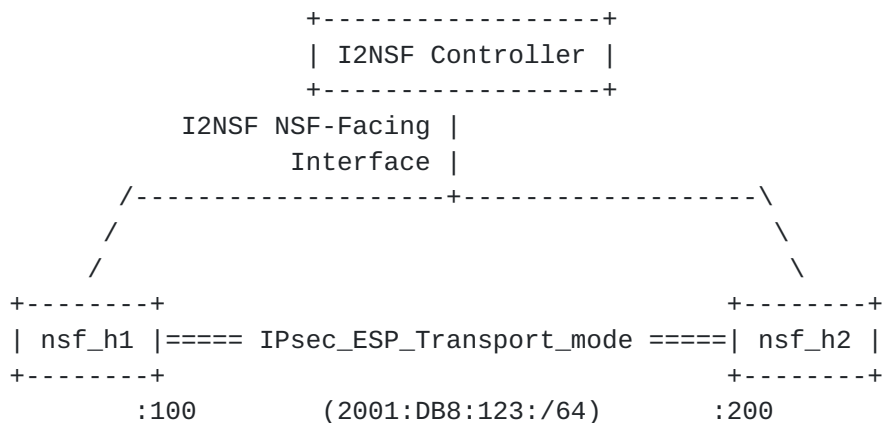
```
                    <local-ports>
                       <start>0</start>
                       <end>0</end>
                    </local-ports>
                    <remote-ports>
                       <start>0</start>
                       <end>0</end>
                    </remote-ports>
                 </traffic-selector>
                 <processing-info>
                    <action>protect</action>
                    <ipsec-sa-cfg>
                       <ext-seq-num>true</ext-seq-num>
                       <seq-overflow>true</seq-overflow>
                       <mode>transport</mode>
                       <protocol-parameters>esp</protocol-parameters>
                       <esp-algorithms>
                          <!-- AUTH_HMAC_SHA1_96 -->
                          <integrity>2</integrity>
                          <!-- ENCR_AES_CBC -->
                          <encryption>
                             <id>1</id>
                             <algorithm-type>12</algorithm-type>
                             <key-length>128</key-length>
                          </encryption>
                          <encryption>
                             <id>2</id>
                             <algorithm-type>3</algorithm-type>
                          </encryption>
                       </esp-algorithms>
                    </ipsec-sa-cfg>
                 </processing-info>
              </ipsec-policy-config>
           </spd-entry>
        </spd>
        <sad>
           <sad-entry>
              <name>out/trans/2001:DB8:123::100/2001:DB8:123::200</name>
              <reqid>1</reqid>
              <ipsec-sa-config>
                 <spi>34501</spi>
                 <ext-seq-num>true</ext-seq-num>
                 <seq-number-counter>100</seq-number-counter>
                 <seq-overflow>true</seq-overflow>
                 <anti-replay-window>32</anti-replay-window>
                 <traffic-selector>
                    <local-subnet>2001:DB8:123::100/128</local-subnet>
                    <remote-subnet>2001:DB8:123::200/128</remote-subnet>
```

```
                     <inner-protocol>any</inner-protocol>
                     <local-ports>
                        <start>0</start>
                        <end>0</end>
                     </local-ports>
                     <remote-ports>
                        <start>0</start>
                        <end>0</end>
                     </remote-ports>
                  </traffic-selector>
                  <protocol-parameters>esp</protocol-parameters>
                  <mode>transport</mode>
                  <esp-sa>
                    <encryption>
                       <!-- //ENCR_AES_CBC -->
                       <encryption-algorithm>12</encryption-algorithm>
                       <key>01:23:45:67:89:AB:CE:DF</key>
                       <iv>01:23:45:67:89:AB:CE:DF</iv>
                    </encryption>
                    <integrity>
                       <!-- //AUTH_HMAC_SHA1_96 -->
                       <integrity-algorithm>2</integrity-algorithm>
                       <key>01:23:45:67:89:AB:CE:DF</key>
                    </integrity>
                  </esp-sa>
              </ipsec-sa-config>
          </sad-entry>
          <sad-entry>
            <name>in/trans/2001:DB8:123::200/2001:DB8:123::100</name>
            <reqid>1</reqid>
            <ipsec-sa-config>
                <spi>34502</spi>
                <ext-seq-num>true</ext-seq-num>
                <seq-number-counter>100</seq-number-counter>
                <seq-overflow>true</seq-overflow>
                <anti-replay-window>32</anti-replay-window>
                <traffic-selector>
                   <local-subnet>2001:DB8:123::200/128</local-subnet>
                   <remote-subnet>2001:DB8:123::100/128</remote-subnet>
                   <inner-protocol>any</inner-protocol>
                   <local-ports>
                      <start>0</start>
                      <end>0</end>
                   </local-ports>
                   <remote-ports>
                      <start>0</start>
                      <end>0</end>
                   </remote-ports>
```

```
                    </traffic-selector>
                    <protocol-parameters>esp</protocol-parameters>
                    <mode>transport</mode>
                    <esp-sa>
                        <encryption>
                            <!-- //ENCR_AES_CBC -->
                            <encryption-algorithm>12</encryption-algorithm>
                            <key>01:23:45:67:89:AB:CE:DF</key>
                            <iv>01:23:45:67:89:AB:CE:DF</iv>
                        </encryption>
                        <integrity>
                            <!-- //AUTH_HMAC_SHA1_96 -->
                            <integrity-algorithm>2</integrity-algorithm>
                            <key>01:23:45:67:89:AB:CE:DF</key>
                        </integrity>
                    </esp-sa>
                    <sa-lifetime-hard>
                        <bytes>2000000</bytes>
                        <packets>2000</packets>
                        <time>60</time>
                        <idle>120</idle>
                    </sa-lifetime-hard>
                    <sa-lifetime-soft>
                        <bytes>1000000</bytes>
                        <packets>1000</packets>
                        <time>30</time>
                        <idle>60</idle>
                        <action>replace</action>
                    </sa-lifetime-soft>
                </ipsec-sa-config>
            </sad-entry>
        </sad>
    </ipsec-ikeless>
```

Appendix F. **XML notification examples**

   Below we show several XML files that represent different types of
   notifications defined in the IKE-less YANG model, which are sent by
   the NSF to the I2NSF Controller. The notifications happen in the
   IKE-less case.

```
   <sadb-expire xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless">
   <ipsec-sa-name>in/trans/2001:DB8:123::200/2001:DB8:123::100
   </ipsec-sa-name>
       <soft-lifetime-expire>true</soft-lifetime-expire>
          <lifetime-current>
             <bytes>1000000</bytes>
             <packets>1000</packets>
             <time>30</time>
             <idle>60</idle>
          </lifetime-current>
   </sadb-expire>
```

                Figure 5: Example of sadb-expire notification.


```
   <sadb-acquire xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless">
       <ipsec-policy-name>in/trans/2001:DB8:123::200/2001:DB8:123::100
       </ipsec-policy-name>
       <traffic-selector>
           <local-subnet>2001:DB8:123::200/128</local-subnet>
           <remote-subnet>2001:DB8:123::100/128</remote-subnet>
           <inner-protocol>any</inner-protocol>
            <local-ports>
                 <start>0</start>
                 <end>0</end>
            </local-ports>
            <remote-ports>
                 <start>0</start>
                 <end>0</end>
            </remote-ports>
       </traffic-selector>
   </sadb-acquire>
```

               Figure 6: Example of sadb-acquire notification.


```
   <sadb-seq-overflow
       xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless">
         <ipsec-sa-name>in/trans/2001:DB8:123::200/2001:DB8:123::100
         </ipsec-sa-name>
   </sadb-seq-overflow>
```

            Figure 7: Example of sadb-seq-overflow notification.

```
   <sadb-bad-spi
          xmlns="urn:ietf:params:xml:ns:yang:ietf-i2nsf-ikeless">
         <spi>666</spi>
   </sadb-bad-spi>
```

Figure 8: Example of sadb-bad-spi notification.

Appendix G. Operational use cases examples

G.1. Example of IPsec SA establishment

This appendix exemplifies the applicability of IKE case and IKE-less
case to traditional IPsec configurations, that is, host-to-host and
gateway-to-gateway. The examples we show in the following assume the
existence of two NSFs needing to establish an end-to-end IPsec SA to
protect their communications. Both NSFs could be two hosts that
exchange traffic (host-to-host) or gateways (gateway-to-gateway), for
example, within an enterprise that needs to protect the traffic
between 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. Besides, IaaS services
providing virtualization environments are deployments that often rely
on IPsec to provide secure channels between virtual instances (host-
to-host) and providing VPN solutions for virtualized networks
(gateway-to-gateway).

As we will show in the following, the I2NSF-based IPsec management
system (for IKE and IKE-less cases), exhibits various advantages:

1. It allows to create IPsec SAs among two NSFs, based only on the
    application of general Flow-based Protection Policies at the
    I2NSF User. Thus, administrators can manage all security
    associations in a centralized point with an abstracted view of
    the network.

2. Any NSF deployed in the system does not need manual
    configuration, therefore allowing its deployment in an automated
    manner.

G.1.1. IKE case

```
            +-----------------------------------------+
            |  I2NSF User  (IPsec Management System) |
            +-----------------------------------------+
                    |
              (1)    Flow-based    I2NSF Consumer-Facing
                  Protection Policy      Interface
                    |
            +---------|-------------------------------+
            |         |                               |
            |         |     I2NSF Controller          |
            |         V                               |
            |    +--------------+ (2)+--------------+  |
            |    |Translate into|--->|   NETCONF/   |  |
            |    |IPsec Policies|    |   RESTCONF   |  |
            |    +--------------+    +--------------+  |
            |                        |     |      |
            |                        |     |      |
            +------------------------|-----|------+
                                     |     |
             I2NSF NSF-Facing Interface |     |
                                     | (3) |
              |-------------------------+     +---|
              V                                    V
         +----------------------+      +----------------------+
         |        NSF A         |      |        NSF B         |
         | IKEv2/IPsec(SPD/PAD) |      | IKEv2/IPsec(SPD/PAD) |
         +----------------------+      +----------------------+
```
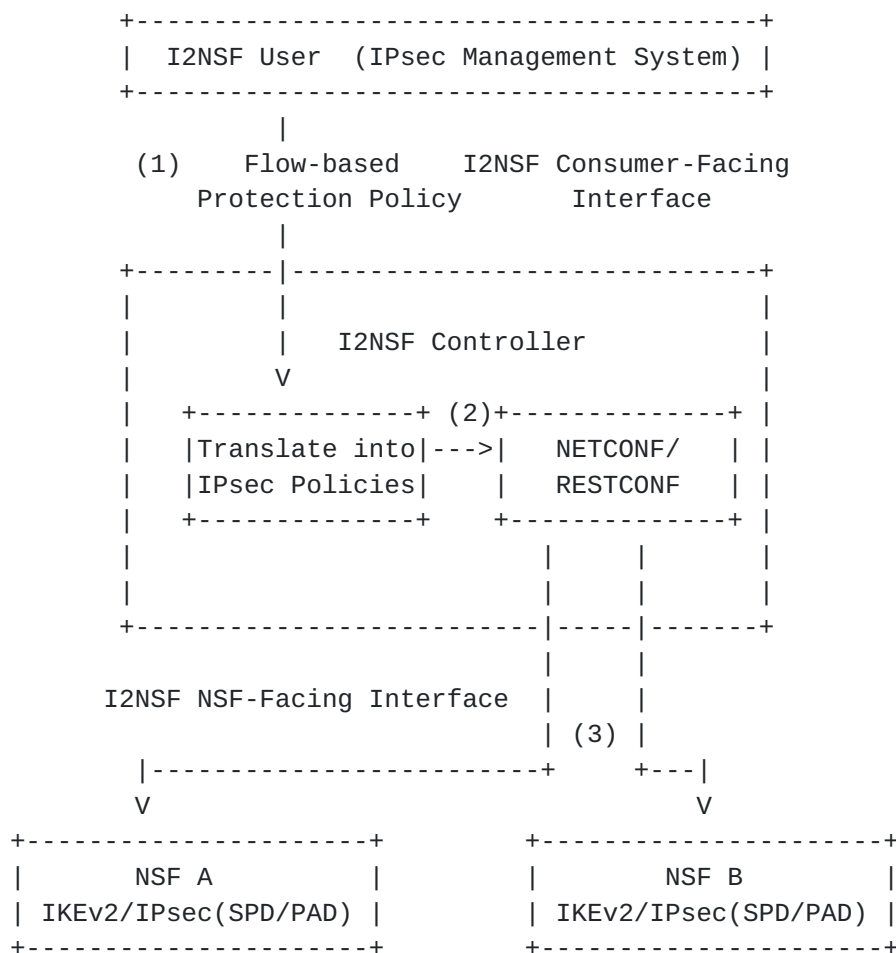
       Figure 9: Host-to-host / gateway-to-gateway for the IKE case.

   Figure 9 describes the application of the IKE case when a data packet
   needs to be protected in the path between the NSF A and NSF B:

   1. The I2NSF User defines a general flow-based protection policy
       (e.g. protect data traffic between NSF A and B). The I2NSF
       Controller looks for the NSFs involved (NSF A and NSF B).

   2. The I2NSF Controller generates IKEv2 credentials for them and
       translates the policies into SPD and PAD entries.

   3. The I2NSF Controller inserts an IKEv2 configuration that includes
       the SPD and PAD entries in both NSF A and NSF B. If some of
       operations with NSF A and NSF B fail the I2NSF Controller will
       stop the process and perform a rollback operation by deleting any
       IKEv2, SPD and PAD configuration that had been successfully
       installed in NSF A or B.

If the previous steps are successful, the flow is protected by means
of the IPsec SA established with IKEv2 between NSF A and NSF B.
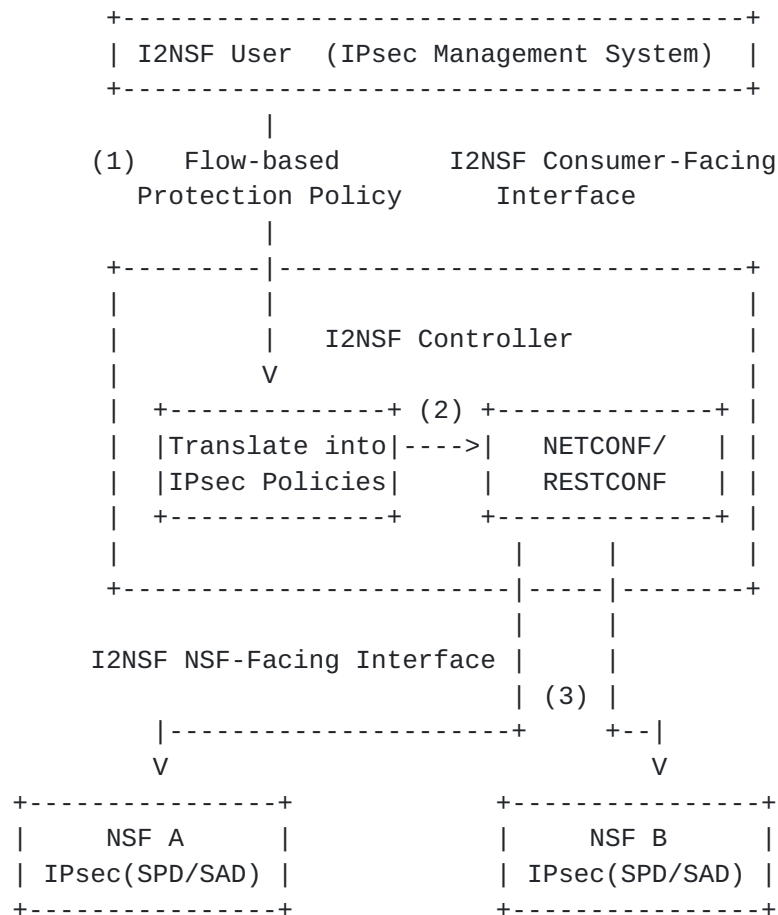
G.1.2. IKE-less case

```
            +----------------------------------------+
            | I2NSF User  (IPsec Management System)   |
            +----------------------------------------+
                      |
            (1)   Flow-based       I2NSF Consumer-Facing
               Protection Policy        Interface
                      |
          +---------|-------------------------------+
          |         |                               |
          |         |      I2NSF Controller          |
          |         V                               |
          |   +--------------+ (2) +--------------+  |
          |   |Translate into|---->|   NETCONF/    | |
          |   |IPsec Policies|     |   RESTCONF    | |
          |   +--------------+     +--------------+  |
          |                          |     |        |
          +--------------------------|-----|--------+
                                     |     |
          I2NSF NSF-Facing Interface |     |
                                     | (3) |
            |----------------------+     +--|
              V                              V
        +----------------+        +----------------+
        |     NSF A       |        |     NSF B       |
        | IPsec(SPD/SAD)  |        | IPsec(SPD/SAD)  |
        +----------------+        +----------------+
```

Figure 10: Host-to-host / gateway-to-gateway for IKE-less case.

Figure 10 describes the application of the IKE-less case when a data
packet needs to be protected in the path between the NSF A and NSF B:

1. The I2NSF User establishes a general Flow-based Protection Policy
    and the I2NSF Controller looks for the involved NSFs.

2. The I2NSF Controller translates the flow-based security policies
    into IPsec SPD and SAD entries.

3. The I2NSF Controller inserts these entries in both NSF A and NSF
    B IPsec databases (SPD and SAD). The following text describes
    how this would happen:

   *   The I2NSF Controller chooses two random values as SPIs: for
       example, SPIa1 for NSF A and SPIb1 for NSF B. These numbers
       MUST NOT be in conflict with any IPsec SA in NSF A or NSF B.
       It also generates fresh cryptographic material for the new
       inbound/outbound IPsec SAs and their parameters.

   *   After that, the I2NSF Controller sends simultaneously the new
       inbound IPsec SA with SPIa1 and new outbound IPsec SA with
       SPIb1 to NSF A; and the new inbound IPsec SA with SPIb1 and
       new outbound IPsec SA with SPIa1 to B, together with the
       corresponding IPsec policies.

   *   Once the I2NSF Controller receives confirmation from NSF A and
       NSF B, it knows that the IPsec SAs are correctly installed and
       ready.

   Other alternative to this operation is: the I2NSF Controller
   sends first the IPsec policies and new inbound IPsec SAs to A and
   B and once it obtains a successful confirmation of these
   operations from NSF A and NSF B, it proceeds with installing to
   the new outbound IPsec SAs. Even though this procedure may
   increase the latency to complete the process, no traffic is sent
   over the network until the IPsec SAs are completely operative.
   In any case other alternatives MAY be possible to implement step
   3.

   4. If some of the operations described above fail (e.g. the NSF A
       reports an error when the I2NSF Controller is trying to install
       the SPD entry, the new inbound or outbound IPsec SAs) the I2NSF
       Controller must perform rollback operations by deleting any new
       inbound or outbound SA and SPD entry that had been successfully
       installed in any of the NSFs (e.g NSF B) and stop the process.
       Note that the I2NSF Controller may retry several times before
       giving up.

   5. Otherwise, if the steps 1 to 3 are successful, the flow between
       NSF A and NSF B is protected by means of the IPsec SAs
       established by the I2NSF Controller. It is worth mentioning that
       the I2NSF Controller associates a lifetime to the new IPsec SAs.
       When this lifetime expires, the NSF will send a sadb-expire
       notification to the I2NSF Controller in order to start the
       rekeying process.

   Instead of installing IPsec policies (in the SPD) and IPsec SAs (in
   the SAD) in step 3 (proactive mode), it is also possible that the
   I2NSF Controller only installs the SPD entries in step 3 (reactive
   mode). In such a case, when a data packet requires to be protected
   with IPsec, the NSF that saw first the data packet will send a sadb-

acquire notification that informs the I2NSF Controller that needs SAD entries with the IPsec SAs to process the data packet. Again, if some of the operations installing the new inbound/outbound IPsec SAs fail, the I2NSF Controller stops the process and performs a rollback operation by deleting any new inbound/outbound SAs that had been successfully installed.

G.2. Example of the rekeying process in IKE-less case

To explain an example of the rekeying process between two IPsec NSFs A and B, let assume that SPIa1 identifies the inbound IPsec SA in A, and SPIb1 the inbound IPsec SA in B. The rekeying process will take the following steps:

1. The I2NSF Controller chooses two random values as SPI for the new inbound IPsec 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 I2NSF Controller creates an inbound IPsec SA with SPIa2 in A and another inbound IPsec SA in B with SPIb2. It can send this information simultaneously to A and B.

2. Once the I2NSF Controller receives confirmation from A and B, the controller knows that the inbound IPsec SAs are correctly installed. Then it proceeds to send in parallel to A and B, the outbound IPsec SAs: the outbound IPsec SA to A with SPIb2, and the outbound IPsec SA to B with SPIa2. At this point the new IPsec SAs are ready.

3. Once the I2NSF Controller receives confirmation from A and B that the outbound IPsec SAs have been installed, the I2NSF Controller, in parallel, deletes the old IPsec SAs from A (inbound SPIa1 and outbound SPIb1) and B (outbound SPIa1 and inbound SPIb1).

If some of the operations in step 1 fail (e.g. the NSF A reports an error when the I2NSF Controller is trying to install a new inbound IPsec SA) the I2NSF Controller must perform rollback operations by removing any new inbound SA that had been successfully installed during step 1.

If step 1 is successful but some of the operations in step 2 fails (e.g. the NSF A reports an error when the I2NSF Controller is trying to install the new outbound IPsec SA), the I2NSF Controller must perform a rollback operation by deleting any new outbound SA that had been successfully installed during step 2 and by deleting the inbound SAs created in step 1.

If the steps 1 and 2 are successful but the step 3 fails, the I2NSF Controller will avoid any rollback of the operations carried out in

step 1 and step 2 since new and valid IPsec SAs were created and are
functional. The I2NSF Controller may reattempt to remove the old
inbound and outbound SAs in NSF A and NSF B several times until it
receives a success or it gives up. In the last case, the old IPsec
SAs will be removed when their corresponding hard lifetime is
reached.

G.3. Example of managing NSF state loss in IKE-less case

In the IKE-less case, if the I2NSF Controller detects that a NSF has
lost the IPsec state, it could follow the next steps:

1. The I2NSF Controller SHOULD delete the old IPsec SAs on the non-
    failed nodes, established with the failed node. This prevents
    the non-failed nodes from leaking plaintext.

2. If the affected node restarts, the I2NSF Controller configures
    the new inbound IPsec SAs between the affected node and all the
    nodes it was talking to.

3. After these inbound IPsec SAs have been established, the I2NSF
    Controller configures the outbound IPsec SAs in parallel.

Step 2 and step 3 can be performed at the same time at the cost of a
potential packet loss. If this is not critical then it is an
optimization since the number of exchanges between I2NSF Controller
and NSFs is lower.

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