NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2020

K. Watsen
Watsen Networks

March 8, 2020

A YANG Data Model for a Keystore
draft-ietf-netconf-keystore-16

Abstract

This document defines a YANG 1.1 module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

Editorial Note (To be removed by RFC Editor)

This draft contains many placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Artwork in this document contains shorthand references to drafts in progress. Please apply the following replacements:

- "AAAA" --> the assigned RFC value for [I-D.ietf-netconf-crypto-types].
- "XXXX" --> the assigned RFC value for this draft

Artwork in this document contains placeholder values for the date of publication of this draft. Please apply the following replacement:

- "2020-03-08" --> the publication date of this draft

The following Appendix section is to be removed prior to publication:

- Appendix A. Change Log

Status of This Memo

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

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

working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on September 9, 2020.

Table of Contents

1. Introduction

This document defines a YANG 1.1 [RFC7950] module called "ietf-keystore" that enables centralized configuration of both symmetric and asymmetric keys. The secret value for both key types may be encrypted. Asymmetric keys may be associated with certificates. Notifications are sent when certificates are about to expire.

The "ietf-keystore" module defines many "grouping" statements intended for use by other modules that may import it. For instance, there are groupings that defined enabling a key to be either configured locally (within the defining data model) or be a reference to a key in the keystore.

Special consideration has been given for systems that have cryptographic hardware, such as a Trusted Protection Module (TPM). These systems are unique in that the cryptographic hardware hides the secret key values. To support such hardware, symmetric keys may have

the value "hidden-key" and asymmetric keys may have the value "hidden-private-key". While how such keys are created or destroyed is outside the scope of this document, the keystore can contain entries for such keys, enabling them to be reference by other configuration elements.

This document in compliant with Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, keys and associated certificates installed during manufacturing (e.g., for a IDevID [Std-802.1AR-2009] certificate), are expected to appear in <operational> (see Section 4).

It is not required that a system has an operating system level keystore utility to implement this module.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. The Keystore Model

3.1. Tree Diagram

This section provides a tree diagrams [RFC8340] for the "ietf-keystore" module that presents both the protocol-accessible "keystore" as well the all the groupings intended for external usage.

```
module: ietf-keystore
  +--rw keystore
     +--rw asymmetric-keys
     |  +--rw asymmetric-key* [name]
     |     +--rw name                                  string
     |     +--rw algorithm
     |     |       iasa:asymmetric-algorithm-type
     |     +--rw public-key-format                     identityref
     |     +--rw public-key                            binary
     |     +--rw private-key-format?                   identityref
```

```
|     +--rw (private-key-type)
|     |  +--:(private-key)
|     |  |  +--rw private-key?                        binary
|     |  +--:(hidden-private-key)
|     |  |  +--rw hidden-private-key?                 empty
|     |  +--:(encrypted-private-key)
|     |     +--rw encrypted-private-key
|     |        +--rw (key-type)
|     |        |  +--:(symmetric-key-ref)
|     |        |  |  +--rw symmetric-key-ref?    leafref
|     |        |  |           {keystore-supported}?
|     |        |  +--:(asymmetric-key-ref)
|     |        |     +--rw asymmetric-key-ref?   leafref
|     |        |              {keystore-supported}?
|     |        +--rw value?                       binary
|     +--rw certificates
|     |  +--rw certificate* [name]
|     |     +--rw name                       string
|     |     +--rw cert?                      end-entity-cert-cms
```



```
|     |     +---n certificate-expiration
|     |        +-- expiration-date    yang:date-and-time
|     +---x generate-certificate-signing-request
|        +---w input
|        |  +---w subject       binary
|        |  +---w attributes?   binary
|        +--ro output
|           +--ro certificate-signing-request    binary
+--rw symmetric-keys
   +--rw symmetric-key* [name]
      +--rw name                    string
      +--rw algorithm               isa:symmetric-algorithm-type
      +--rw key-format?             identityref
      +--rw (key-type)
         +--:(key)
         |  +--rw key?              binary
         +--:(hidden-key)
         |  +--rw hidden-key?       empty
         +--:(encrypted-key)
            +--rw encrypted-key
               +--rw (key-type)
               |  +--:(symmetric-key-ref)
```

```
                    |  |  +--rw symmetric-key-ref?    leafref
                    |  |          {keystore-supported}?
                    |  +--:(asymmetric-key-ref)
                    |     +--rw asymmetric-key-ref?   leafref
                    |             {keystore-supported}?
                    +--rw value?                      binary

  augment /ct:generate-symmetric-key/ct:input:
    +---w encrypt-with!
       +---w (key-type)
          +--:(symmetric-key-ref)
          |  +---w symmetric-key-ref?
          |          -> /keystore/symmetric-keys/symmetric-key/name
          |          {keystore-supported}?
          +--:(asymmetric-key-ref)
             +---w asymmetric-key-ref?
                     -> /keystore/asymmetric-keys/asymmetric-key/name
                     {keystore-supported}?
  augment /ct:generate-symmetric-key/ct:output/ct:key-type:
    +-- encrypted-key
       +-- (key-type)
       |  +--:(symmetric-key-ref)
       |  |  +-- symmetric-key-ref?
       |  |          -> /keystore/symmetric-keys/symmetric-key/name
       |  |          {keystore-supported}?
       |  +--:(asymmetric-key-ref)
```



```
       |     +-- asymmetric-key-ref?
       |             -> /keystore/asymmetric-keys/asymmetric-key/name
       |             {keystore-supported}?
       +-- value?                      binary
  augment /ct:generate-asymmetric-key/ct:input:
    +---w encrypt-with!
       +---w (key-type)
          +--:(symmetric-key-ref)
          |  +---w symmetric-key-ref?
          |          -> /keystore/symmetric-keys/symmetric-key/name
          |          {keystore-supported}?
          +--:(asymmetric-key-ref)
             +---w asymmetric-key-ref?
                     -> /keystore/asymmetric-keys/asymmetric-key/name
                     {keystore-supported}?
```

```
augment /ct:generate-asymmetric-key/ct:output/ct:private-key-type:
  +-- encrypted-private-key
     +-- (key-type)
     |  +--:(symmetric-key-ref)
     |  |  +-- symmetric-key-ref?
     |  |          -> /keystore/symmetric-keys/symmetric-key/name
     |  |          {keystore-supported}?
     |  +--:(asymmetric-key-ref)
     |     +-- asymmetric-key-ref?
     |             -> /keystore/asymmetric-keys/asymmetric-key/name
     |             {keystore-supported}?
     +-- value?                      binary

grouping key-reference-type-grouping
  +-- (key-type)
     +--:(symmetric-key-ref)
     |  +-- symmetric-key-ref?
     |          -> /keystore/symmetric-keys/symmetric-key/name
     |          {keystore-supported}?
     +--:(asymmetric-key-ref)
        +-- asymmetric-key-ref?
                -> /keystore/asymmetric-keys/asymmetric-key/name
                {keystore-supported}?
grouping encrypted-value-grouping
  +-- (key-type)
  |  +--:(symmetric-key-ref)
  |  |  +-- symmetric-key-ref?
  |  |          -> /keystore/symmetric-keys/symmetric-key/name
  |  |          {keystore-supported}?
  |  +--:(asymmetric-key-ref)
  |     +-- asymmetric-key-ref?
  |             -> /keystore/asymmetric-keys/asymmetric-key/name
  |             {keystore-supported}?
```



```
  +-- value?                      binary
grouping symmetric-key-grouping
  +-- algorithm              isa:symmetric-algorithm-type
  +-- key-format?            identityref
  +-- (key-type)
     +--:(key)
     |  +-- key?             binary
     +--:(hidden-key)
```

```
        |  +-- hidden-key?      empty
        +--:(encrypted-key)
           +-- encrypted-key
              +-- (key-type)
              |  +--:(symmetric-key-ref)
              |  |  +-- symmetric-key-ref?    leafref
              |  |          {keystore-supported}?
              |  +--:(asymmetric-key-ref)
              |     +-- asymmetric-key-ref?   leafref
              |             {keystore-supported}?
              +-- value?                     binary
    grouping asymmetric-key-pair-grouping
      +-- algorithm                      iasa:asymmetric-algorithm-type
      +-- public-key-format              identityref
      +-- public-key                     binary
      +-- private-key-format?            identityref
      +-- (private-key-type)
         +--:(private-key)
         |  +-- private-key?             binary
         +--:(hidden-private-key)
         |  +-- hidden-private-key?      empty
         +--:(encrypted-private-key)
            +-- encrypted-private-key
               +-- (key-type)
               |  +--:(symmetric-key-ref)
               |  |  +-- symmetric-key-ref?    leafref
               |  |          {keystore-supported}?
               |  +--:(asymmetric-key-ref)
               |     +-- asymmetric-key-ref?   leafref
               |             {keystore-supported}?
               +-- value?                     binary
    grouping asymmetric-key-pair-with-cert-grouping
      +-- algorithm
      |       iasa:asymmetric-algorithm-type
      +-- public-key-format                        identityref
      +-- public-key                               binary
      +-- private-key-format?                      identityref
      +-- (private-key-type)
      |  +--:(private-key)
      |  |  +-- private-key?                       binary
```



```
        |  +--:(hidden-private-key)
```

```
      |  |  +-- hidden-private-key?               empty
      |  +--:(encrypted-private-key)
      |     +-- encrypted-private-key
      |        +-- (key-type)
      |        |  +--:(symmetric-key-ref)
      |        |  |  +-- symmetric-key-ref?    leafref
      |        |  |           {keystore-supported}?
      |        |  +--:(asymmetric-key-ref)
      |        |     +-- asymmetric-key-ref?   leafref
      |        |              {keystore-supported}?
      |        +-- value?                      binary
      +-- cert?                                   end-entity-cert-cms
      +---n certificate-expiration
      |  +-- expiration-date    yang:date-and-time
      +---x generate-certificate-signing-request
         +---w input
         |  +---w subject       binary
         |  +---w attributes?   binary
         +--ro output
            +--ro certificate-signing-request    binary
    grouping asymmetric-key-pair-with-certs-grouping
      +-- algorithm
      |       iasa:asymmetric-algorithm-type
      +-- public-key-format                        identityref
      +-- public-key                               binary
      +-- private-key-format?                      identityref
      +-- (private-key-type)
      |  +--:(private-key)
      |  |  +-- private-key?                       binary
      |  +--:(hidden-private-key)
      |  |  +-- hidden-private-key?               empty
      |  +--:(encrypted-private-key)
      |     +-- encrypted-private-key
      |        +-- (key-type)
      |        |  +--:(symmetric-key-ref)
      |        |  |  +-- symmetric-key-ref?    leafref
      |        |  |           {keystore-supported}?
      |        |  +--:(asymmetric-key-ref)
      |        |     +-- asymmetric-key-ref?   leafref
      |        |              {keystore-supported}?
      |        +-- value?                      binary
      +-- certificates
      |  +-- certificate* [name]
      |     +-- name?                     string
      |     +-- cert?                     end-entity-cert-cms
      |     +---n certificate-expiration
      |        +-- expiration-date    yang:date-and-time
```

```
       +---x generate-certificate-signing-request
          +---w input
          |  +---w subject       binary
          |  +---w attributes?   binary
          +--ro output
             +--ro certificate-signing-request    binary
     grouping asymmetric-key-certificate-ref-grouping
       +-- asymmetric-key?   ks:asymmetric-key-ref
       +-- certificate?      leafref
     grouping local-or-keystore-symmetric-key-grouping
       +-- (local-or-keystore)
          +--:(local) {local-definitions-supported}?
          |  +-- local-definition
          |     +-- algorithm              isa:symmetric-algorithm-type
          |     +-- key-format?            identityref
          |     +-- (key-type)
          |        +--:(key)
          |        |  +-- key?             binary
          |        +--:(hidden-key)
          |        |  +-- hidden-key?      empty
          |        +--:(encrypted-key)
          |           +-- encrypted-key
          |              +-- (key-type)
          |              |  +--:(symmetric-key-ref)
          |              |  |  +-- symmetric-key-ref?    leafref
          |              |  |           {keystore-supported}?
          |              |  +--:(asymmetric-key-ref)
          |              |     +-- asymmetric-key-ref?   leafref
          |              |              {keystore-supported}?
          |              +-- value?                     binary
          +--:(keystore) {keystore-supported}?
             +-- keystore-reference?   ks:symmetric-key-ref
     grouping local-or-keystore-asymmetric-key-grouping
       +-- (local-or-keystore)
          +--:(local) {local-definitions-supported}?
          |  +-- local-definition
          |     +-- algorithm
          |     |       iasa:asymmetric-algorithm-type
          |     +-- public-key-format              identityref
          |     +-- public-key                     binary
          |     +-- private-key-format?            identityref
          |     +-- (private-key-type)
          |        +--:(private-key)
          |        |  +-- private-key?             binary
          |        +--:(hidden-private-key)
          |        |  +-- hidden-private-key?      empty
```

```
          |        +--:(encrypted-private-key)
          |           +-- encrypted-private-key
```



```
          |              +-- (key-type)
          |              |  +--:(symmetric-key-ref)
          |              |  |  +-- symmetric-key-ref?    leafref
          |              |  |           {keystore-supported}?
          |              |  +--:(asymmetric-key-ref)
          |              |     +-- asymmetric-key-ref?   leafref
          |              |              {keystore-supported}?
          |              +-- value?                      binary
          |           +--:(keystore) {keystore-supported}?
          |              +-- keystore-reference?   ks:asymmetric-key-ref
          |        grouping local-or-keystore-asymmetric-key-with-certs-grouping
          |          +-- (local-or-keystore)
          |             +--:(local) {local-definitions-supported}?
          |             |  +-- local-definition
          |             |     +-- algorithm
          |             |     |       iasa:asymmetric-algorithm-type
          |             |     +-- public-key-format                        identityref
          |             |     +-- public-key                               binary
          |             |     +-- private-key-format?                      identityref
          |             |     +-- (private-key-type)
          |             |     |  +--:(private-key)
          |             |     |  |  +-- private-key?                       binary
          |             |     |  +--:(hidden-private-key)
          |             |     |  |  +-- hidden-private-key?                empty
          |             |     |  +--:(encrypted-private-key)
          |             |     |     +-- encrypted-private-key
          |             |     |        +-- (key-type)
          |             |     |        |  +--:(symmetric-key-ref)
          |             |     |        |  |  +-- symmetric-key-ref?    leafref
          |             |     |        |  |           {keystore-supported}?
          |             |     |        |  +--:(asymmetric-key-ref)
          |             |     |        |     +-- asymmetric-key-ref?   leafref
          |             |     |        |              {keystore-supported}?
          |             |     |        +-- value?                       binary
          |             |     +-- certificates
          |             |     |  +-- certificate* [name]
          |             |     |     +-- name?                      string
          |             |     |     +-- cert?                      end-entity-cert-cms
          |             |     |     +---n certificate-expiration
```

```
|     |        +-- expiration-date    yang:date-and-time
|     +---x generate-certificate-signing-request
|        +---w input
|        |  +---w subject       binary
|        |  +---w attributes?   binary
|        +--ro output
|           +--ro certificate-signing-request    binary
+--:(keystore) {keystore-supported}?
   +-- keystore-reference?   ks:asymmetric-key-ref
```



```
grouping local-or-keystore-end-entity-cert-with-key-grouping
  +-- (local-or-keystore)
     +--:(local) {local-definitions-supported}?
     |  +-- local-definition
     |     +-- algorithm
     |     |       iasa:asymmetric-algorithm-type
     |     +-- public-key-format                          identityref
     |     +-- public-key                                 binary
     |     +-- private-key-format?                        identityref
     |     +-- (private-key-type)
     |     |  +--:(private-key)
     |     |  |  +-- private-key?                         binary
     |     |  +--:(hidden-private-key)
     |     |  |  +-- hidden-private-key?                  empty
     |     |  +--:(encrypted-private-key)
     |     |     +-- encrypted-private-key
     |     |        +-- (key-type)
     |     |           |  +--:(symmetric-key-ref)
     |     |           |  |  +-- symmetric-key-ref?    leafref
     |     |           |  |           {keystore-supported}?
     |     |           |  +--:(asymmetric-key-ref)
     |     |           |     +-- asymmetric-key-ref?   leafref
     |     |           |              {keystore-supported}?
     |     |           +-- value?                       binary
     |     +-- cert?
     |     |       end-entity-cert-cms
     |     +---n certificate-expiration
     |     |  +-- expiration-date    yang:date-and-time
     |     +---x generate-certificate-signing-request
     |        +---w input
     |        |  +---w subject       binary
     |        |  +---w attributes?   binary
```

```
      |        +--ro output
      |           +--ro certificate-signing-request    binary
      +--:(keystore) {keystore-supported}?
         +-- keystore-reference
            +-- asymmetric-key?   ks:asymmetric-key-ref
            +-- certificate?      leafref
  grouping keystore-grouping
    +-- asymmetric-keys
    |  +-- asymmetric-key* [name]
    |     +-- name?                                   string
    |     +-- algorithm
    |     |       iasa:asymmetric-algorithm-type
    |     +-- public-key-format                       identityref
    |     +-- public-key                              binary
    |     +-- private-key-format?                     identityref
    |     +-- (private-key-type)
```



```
      |     |  +--:(private-key)
      |     |  |  +-- private-key?                    binary
      |     |  +--:(hidden-private-key)
      |     |  |  +-- hidden-private-key?             empty
      |     |  +--:(encrypted-private-key)
      |     |     +-- encrypted-private-key
      |     |        +-- (key-type)
      |     |        |  +--:(symmetric-key-ref)
      |     |        |  |  +-- symmetric-key-ref?    leafref
      |     |        |  |           {keystore-supported}?
      |     |        |  +--:(asymmetric-key-ref)
      |     |        |     +-- asymmetric-key-ref?   leafref
      |     |        |              {keystore-supported}?
      |     |        +-- value?                      binary
      |     +-- certificates
      |     |  +-- certificate* [name]
      |     |     +-- name?                    string
      |     |     +-- cert?                    end-entity-cert-cms
      |     |     +---n certificate-expiration
      |     |        +-- expiration-date    yang:date-and-time
      |     +---x generate-certificate-signing-request
      |        +---w input
      |        |  +---w subject       binary
      |        |  +---w attributes?   binary
      |        +--ro output
```

```
|           +--ro certificate-signing-request    binary
+-- symmetric-keys
   +-- symmetric-key* [name]
      +-- name?                   string
      +-- algorithm               isa:symmetric-algorithm-type
      +-- key-format?             identityref
      +-- (key-type)
         +--:(key)
         |  +-- key?              binary
         +--:(hidden-key)
         |  +-- hidden-key?       empty
         +--:(encrypted-key)
            +-- encrypted-key
               +-- (key-type)
               |  +--:(symmetric-key-ref)
               |  |  +-- symmetric-key-ref?    leafref
               |  |           {keystore-supported}?
               |  +--:(asymmetric-key-ref)
               |     +-- asymmetric-key-ref?   leafref
               |              {keystore-supported}?
               +-- value?                     binary
```


3.2. Example Usage

3.2.1. A Keystore Instance

The following example illustrates keys in <intended>. Please see Section 4 for an example illustrating built-in values in <operational>.

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ===========

<keystore xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <symmetric-keys>

    <symmetric-key>
      <name>cleartext-symmetric-key</name>
      <algorithm>aes-256-cbc</algorithm>
      <key-format>ct:octet-string-key-format</key-format>
```

```
        <key>base64encodedvalue==</key>
      </symmetric-key>

      <symmetric-key>
        <name>hidden-symmetric-key</name>
        <algorithm>aes-256-cbc</algorithm>
        <hidden-key/>
      </symmetric-key>

      <symmetric-key>
        <name>encrypted-symmetric-key</name> <!-- operator's key -->
        <algorithm>aes-256-cbc</algorithm>
        <key-format>ct:encrypted-one-symmetric-key-format</key-format>
        <encrypted-key>
          <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-key-re\
f>
          <value>base64encodedvalue==</value>
        </encrypted-key>
      </symmetric-key>

    </symmetric-keys>
    <asymmetric-keys>

      <asymmetric-key>
        <name>ssh-rsa-key</name>
        <algorithm>rsa2048</algorithm>
        <public-key-format>ct:ssh-public-key-format</public-key-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
```



```
        <private-key>base64encodedvalue==</private-key>
      </asymmetric-key>

      <asymmetric-key>
        <name>ssh-rsa-key-with-cert</name>
        <algorithm>rsa2048</algorithm>
        <public-key-format>ct:ssh-public-key-format</public-key-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
        <private-key>base64encodedvalue==</private-key>
```

```
        <certificates>
          <certificate>
            <name>ex-rsa-cert2</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>

      <asymmetric-key>
        <name>raw-private-key</name>
        <algorithm>rsa2048</algorithm>
        <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
        <private-key>base64encodedvalue==</private-key>
      </asymmetric-key>

      <asymmetric-key>
        <name>rsa-asymmetric-key</name>
        <algorithm>rsa2048</algorithm>
        <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
        <private-key>base64encodedvalue==</private-key>
        <certificates>
          <certificate>
            <name>ex-rsa-cert</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>

      <asymmetric-key>
```



```
        <name>ec-asymmetric-key</name>
        <algorithm>secp256r1</algorithm>
        <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
```

```
      <public-key>base64encodedvalue==</public-key>
      <private-key-format>ct:ec-private-key-format</private-key-form\
at>
      <private-key>base64encodedvalue==</private-key>
      <certificates>
        <certificate>
          <name>ex-ec-cert</name>
          <cert>base64encodedvalue==</cert>
        </certificate>
      </certificates>
    </asymmetric-key>

    <asymmetric-key>
      <name>hidden-asymmetric-key</name>
      <algorithm>rsa2048</algorithm>
      <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
      <public-key>base64encodedvalue==</public-key>
      <hidden-private-key/> <!-- e.g., TPM protected -->
      <certificates>
        <certificate>
          <name>builtin-idevid-cert</name>
        </certificate>
        <certificate>
          <name>my-ldevid-cert</name>
          <cert>base64encodedvalue==</cert>
        </certificate>
      </certificates>
    </asymmetric-key>

    <asymmetric-key>
      <name>encrypted-asymmetric-key</name>
      <algorithm>secp256r1</algorithm>
      <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
      <public-key>base64encodedvalue==</public-key>
      <private-key-format>ct:encrypted-one-asymmetric-key-format</pr\
ivate-key-format>
      <encrypted-private-key>
        <symmetric-key-ref>encrypted-symmetric-key</symmetric-key-re\
f>
        <value>base64encodedvalue==</value>
      </encrypted-private-key>
    </asymmetric-key>
```

```
    </asymmetric-keys>
  </keystore>
```

3.2.2. The "generate-symmetric-key" RPC

The following example illustrates how the "generate-symmetric-key" RPC defined in "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types] has been augmented to encrypt the generated symmetric key.

This example references the key defined in the keystore example in Section 3.2.1.

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <generate-symmetric-key
    xmlns="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <algorithm>aes-256-cbc</algorithm>
    <encrypt-with
      xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
      <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-key-ref>
    </encrypt-with>
  </generate-symmetric-key>
</rpc>
```

Following is the RPC-reply.

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ===========

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:ks="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <!--<data> yanglint validation fails -->
    <ct:algorithm>aes-256-cbc</ct:algorithm>
    <ct:key-format>ct:encrypted-one-symmetric-key-format</ct:key-for\
mat>
    <ks:encrypted-key>
      <ks:asymmetric-key-ref>hidden-asymmetric-key</ks:asymmetric-ke\
y-ref>
      <ks:value>base64encodedvalue==</ks:value>
    </ks:encrypted-key>
  <!--</data> yanglint validation fails -->
</rpc-reply>
```

3.2.3. The "generate-asymmetric-key" RPC

The following example illustrates how the "generate-asymmetric-key" RPC defined in "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types] has been augmented to encrypt the generated asymmetric key.

This example references the key defined in the keystore example in Section 3.2.1.

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <generate-asymmetric-key
    xmlns="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <algorithm>secp256r1</algorithm>
    <encrypt-with
      xmlns="urn:ietf:params:xml:ns:yang:ietf-keystore">
      <symmetric-key-ref>encrypted-symmetric-key</symmetric-key-ref>
    </encrypt-with>
  </generate-asymmetric-key>
</rpc>
```

Following is the RPC-reply.

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ===========

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:ks="urn:ietf:params:xml:ns:yang:ietf-keystore"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <!--<data> yanglint validation fails -->
    <ct:algorithm>secp256r1</ct:algorithm>
    <ct:public-key-format>ct:subject-public-key-info-format</ct:publ\
ic-key-format>
    <ct:public-key>base64encodedvalue==</ct:public-key>
    <ct:private-key-format>ct:encrypted-one-asymmetric-key-format</c\
t:private-key-format>
    <ks:encrypted-private-key>
      <ks:symmetric-key-ref>encrypted-symmetric-key</ks:symmetric-ke\
y-ref>
```

```
        <ks:value>base64encodedvalue==</ks:value>
      </ks:encrypted-private-key>
    <!--</data> yanglint validation fails -->
  </rpc-reply>
```


3.2.4. Notable Keystore Groupings

The following non-normative module is used by subsequent examples to illustrate groupings defined in the ietf-crypto-types module.

```
module ex-keystore-usage {
  yang-version 1.1;

  namespace "http://example.com/ns/example-keystore-usage";
  prefix "eku";

  import ietf-keystore {
    prefix ks;
    reference
      "RFC XXXX: YANG Data Model for a 'Keystore' Mechanism";
  }

  organization
   "Example Corporation";

  contact
   "Author: YANG Designer <mailto:yang.designer@example.com>";

  description
   "This module illustrates the grouping in the keystore draft called
    'local-or-keystore-asymmetric-key-with-certs-grouping'.";

  revision "YYYY-MM-DD" {
    description
     "Initial version";
    reference
     "RFC XXXX: YANG Data Model for a 'Keystore' Mechanism";
  }
```

```
    container keystore-usage {
      description
        "An illustration of the various keystore groupings.";

      list just-a-key {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-asymmetric-key-grouping;
        description
          "An asymmetric key, with no certs, that may be configured
```



```
           locally or be a reference to an asymmetric key in the
           keystore.  The intent is to reference just the asymmetric
           key, not any certificates that may also be associated
           with the asymmetric key.";
      }

      list key-with-certs {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
        }
        uses ks:local-or-keystore-asymmetric-key-with-certs-grouping;
        description
          "An asymmetric key and its associated certs, that may be
           configured locally or be a reference to an asymmetric key
           (and its associated certs) in the keystore.";
      }

      list end-entity-cert-with-key {
        key name;
        leaf name {
          type string;
          description
            "An arbitrary name for this key.";
```

```
        }
        uses ks:local-or-keystore-end-entity-cert-with-key-grouping;
        description
          "An end-entity certificate, and its associated private key,
           that may be configured locally or be a reference to a
           specific certificate (and its associated private key) in
           the keystore.";
      }
    }

  }
```

The following example illustrates what two configured keys, one local and the other remote, might look like. This example consistent with other examples above (i.e., the referenced key is in an example above).

```
========== NOTE: '\' line wrapping per BCP XXX (RFC XXXX) ===========

<keystore-usage xmlns="http://example.com/ns/example-keystore-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
```



```
    <!-- ks:local-or-keystore-asymmetric-key-grouping -->

    <just-a-key>
      <name>a locally-defined key</name>
      <local-definition>
        <algorithm>rsa2048</algorithm>
        <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
        <private-key>base64encodedvalue==</private-key>
      </local-definition>
    </just-a-key>

    <just-a-key>
      <name>a keystore-defined key (and its associated certs)</name>
      <keystore-reference>rsa-asymmetric-key</keystore-reference>
    </just-a-key>
```

```
    <!-- ks:local-or-keystore-key-and-end-entity-cert-grouping -->

    <key-with-certs>
      <name>a locally-defined key with certs</name>
      <local-definition>
        <algorithm>rsa2048</algorithm>
        <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
        <private-key>base64encodedvalue==</private-key>
        <certificates>
          <certificate>
            <name>a locally-defined cert</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </local-definition>
    </key-with-certs>

    <key-with-certs>
      <name>a keystore-defined key (and its associated certs)</name>
      <keystore-reference>rsa-asymmetric-key</keystore-reference>
    </key-with-certs>

    <!-- ks:local-or-keystore-end-entity-cert-with-key-grouping -->
```



```
    <end-entity-cert-with-key>
      <name>a locally-defined end-entity cert with key</name>
      <local-definition>
        <algorithm>rsa2048</algorithm>
        <public-key-format>ct:subject-public-key-info-format</public-k\
ey-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>ct:rsa-private-key-format</private-key-for\
mat>
        <private-key>base64encodedvalue==</private-key>
        <cert>base64encodedvalue==</cert>
      </local-definition>
```

```
    </end-entity-cert-with-key>

    <end-entity-cert-with-key>
      <name>a keystore-defined certificate (and its associated key)</n\
ame>
      <keystore-reference>
        <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
        <certificate>ex-rsa-cert</certificate>
      </keystore-reference>
    </end-entity-cert-with-key>

  </keystore-usage>
```

3.3. YANG Module

This YANG module has normative references to [RFC8341] and [I-D.ietf-netconf-crypto-types], and an informative reference to [RFC8342].

<CODE BEGINS> file "ietf-keystore@2020-03-08.yang"

```
module ietf-keystore {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-keystore";
  prefix ks;

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

  import ietf-crypto-types {
    prefix ct;
    reference
      "RFC AAAA: Common YANG Data Types for Cryptography";
```



```
  }

  organization
    "IETF NETCONF (Network Configuration) Working Group";
```

```
  contact
    "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
     WG List:  <mailto:netconf@ietf.org>
     Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

  description
    "This module defines a keystore to centralize management
     of security credentials.

     Copyright (c) 2019 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
     (https://www.rfc-editor.org/info/rfcXXXX); 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 2020-03-08 {
    description
      "Initial version";
    reference
      "RFC XXXX: A YANG Data Model for a Keystore";
  }

  /****************/
  /*   Features   */
  /****************/

  feature keystore-supported {
    description
```

```
    "The 'keystore-supported' feature indicates that the server
     supports the keystore.";
}

feature local-definitions-supported {
  description
    "The 'local-definitions-supported' feature indicates that the
     server supports locally-defined keys.";
}

/****************/
/*   Typedefs   */
/****************/

typedef symmetric-key-ref {
  type leafref {
    path "/ks:keystore/ks:symmetric-keys/ks:symmetric-key"
       + "/ks:name";
  }
  description
    "This typedef enables modules to easily define a reference
     to a symmetric key stored in the keystore.";
}

typedef asymmetric-key-ref {
  type leafref {
    path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key"
       + "/ks:name";
  }
  description
    "This typedef enables modules to easily define a reference
     to an asymmetric key stored in the keystore.";
}

/*****************/
/*   Groupings   */
/*****************/


grouping key-reference-type-grouping {
  description
    "A reusable grouping for a choice for the type of key
     referenced in the keystore.";
  choice key-type {
    mandatory true;
    description
      "A choice between a reference to a symmetric or asymmetric
       key in the keystore.";
```

```
        leaf symmetric-key-ref {
          if-feature "keystore-supported";
          type leafref {
            path "/ks:keystore/ks:symmetric-keys/ks:symmetric-key/"
                 + "ks:name";
          }
          description
            "Identifies a symmetric key used to encrypt this key.";
        }
        leaf asymmetric-key-ref {
          if-feature "keystore-supported";
          type leafref {
            path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key/"
                 + "ks:name";
          }
          description
            "Identifies an asymmetric key used to encrypt this key.";
        }
      }
    }

    grouping encrypted-value-grouping {
      description
        "A reusable grouping for a value that has been encrypted by
         a symmetric or asymmetric key in the keystore.";
      uses "key-reference-type-grouping";
      leaf value {
        type binary;
        description
          "The private key, encrypted using the specified symmetric
           or asymmetric key.";
      }
    }

    grouping symmetric-key-grouping {
      description
        "This grouping is identical to the one in ietf-crypto-types
         except that it adds a case statement enabling the key
         value to be encrypted by a symmetric or an asymmetric
         key known to the keystore.";
      uses ct:symmetric-key-grouping {
        augment "key-type" {
```

```
          description
            "Augments a new 'case' statement into the 'choice'
             statement defined by the ietf-crypto-types module.";
          container encrypted-key {
            must "../key-format";
            description
```



```
              "A container for the encrypted symmetric key value.";
            uses encrypted-value-grouping;
          }
        }
      }
    }

    grouping asymmetric-key-pair-grouping {
      description
        "This grouping is identical to the one in ietf-crypto-types
         except that it adds a case statement enabling the key
         value to be encrypted by a symmetric or an asymmetric
         key known to the keystore.";
      uses ct:asymmetric-key-pair-grouping {
        augment "private-key-type" {
          description
            "Augments a new 'case' statement into the 'choice'
             statement defined by the ietf-crypto-types module.";
          container encrypted-private-key {
            must "../private-key-format";
            description
              "A container for the encrypted asymmetric private
               key value.";
            uses encrypted-value-grouping;
          }
        }
      }
    }

    grouping asymmetric-key-pair-with-cert-grouping {
      description
        "This grouping is identical to the one in ietf-crypto-types
         except that it adds a case statement enabling the key
         value to be encrypted by a symmetric or an asymmetric
         key known to the keystore.";
```

```
      uses ct:asymmetric-key-pair-with-cert-grouping {
        augment "private-key-type" {
          description
            "Augments a new 'case' statement into the 'choice'
             statement defined by the ietf-crypto-types module.";
          container encrypted-private-key {
            must "../private-key-format";
            description
              "A container for the encrypted asymmetric private
               key value.";
            uses encrypted-value-grouping;
          }
        }
```



```
      }
    }

    grouping asymmetric-key-pair-with-certs-grouping {
      description
        "This grouping is identical to the one in ietf-crypto-types
         except that it adds a case statement enabling the key
         value to be encrypted by a symmetric or an asymmetric
         key known to the keystore.";
      uses ct:asymmetric-key-pair-with-certs-grouping {
        augment "private-key-type" {
          description
            "Augments a new 'case' statement into the 'choice'
             statement defined by the ietf-crypto-types module.";
          container encrypted-private-key {
            must "../private-key-format";
            description
              "A container for the encrypted asymmetric private
               key value.";
            uses encrypted-value-grouping;
          }
        }
      }
    }

    grouping asymmetric-key-certificate-ref-grouping {
      leaf asymmetric-key {
        type ks:asymmetric-key-ref;
```

```
        must '../certificate';
        description
          "A reference to an asymmetric key in the keystore.";
      }
      leaf certificate {
        type leafref {
          path "/ks:keystore/ks:asymmetric-keys/ks:asymmetric-key[ks:"
               + "name = current()/../asymmetric-key]/ks:certificates"
               + "/ks:certificate/ks:name";
        }
        must '../asymmetric-key';
        description
          "A reference to a specific certificate of the
           asymmetric key in the keystore.";
      }
      description
        "This grouping defines a reference to a specific certificate
         associated with an asymmetric key stored in the keystore.";
    }
```



```
    // local-or-keystore-* groupings

    grouping local-or-keystore-symmetric-key-grouping {
      description
        "A grouping that expands to allow the symmetric key to be
         either stored locally, within the using data model, or be
         a reference to an symmetric key stored in the keystore.";
      choice local-or-keystore {
        mandatory true;
        case local {
          if-feature "local-definitions-supported";
          container local-definition {
            description
              "Container to hold the local key definition.";
            uses symmetric-key-grouping;
          }
        }
        case keystore {
          if-feature "keystore-supported";
          leaf keystore-reference {
            type ks:symmetric-key-ref;
```

```
          description
            "A reference to an symmetric key that exists in
             the keystore.";
        }
      }
      description
        "A choice between an inlined definition and a definition
         that exists in the keystore.";
    }
  }

  grouping local-or-keystore-asymmetric-key-grouping {
    description
      "A grouping that expands to allow the asymmetric key to be
       either stored locally, within the using data model, or be
       a reference to an asymmetric key stored in the keystore.";
    choice local-or-keystore {
      mandatory true;
      case local {
        if-feature "local-definitions-supported";
        container local-definition {
          description
            "Container to hold the local key definition.";
          uses asymmetric-key-pair-grouping;
        }
      }
      case keystore {
```



```
        if-feature "keystore-supported";
        leaf keystore-reference {
          type ks:asymmetric-key-ref;
          description
            "A reference to an asymmetric key that exists in
             the keystore.  The intent is to reference just the
             asymmetric key without any regard for any certificates
             that may be associated with it.";
        }
      }
      description
        "A choice between an inlined definition and a definition
         that exists in the keystore.";
    }
```

```
    }

    grouping local-or-keystore-asymmetric-key-with-certs-grouping {
      description
        "A grouping that expands to allow an asymmetric key and its
         associated certificates to be either stored locally, within
         the using data model, or be a reference to an asymmetric key
         (and its associated certificates) stored in the keystore.";
      choice local-or-keystore {
        mandatory true;
        case local {
          if-feature "local-definitions-supported";
          container local-definition {
            description
              "Container to hold the local key definition.";
            uses asymmetric-key-pair-with-certs-grouping;
          }
        }
        case keystore {
          if-feature "keystore-supported";
          leaf keystore-reference {
            type ks:asymmetric-key-ref;
            description
              "A reference to an asymmetric-key (and all of its
               associated certificates) in the keystore.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
      }
    }

    grouping local-or-keystore-end-entity-cert-with-key-grouping {
```



```
      description
        "A grouping that expands to allow an end-entity certificate
         (and its associated private key) to be either stored locally,
         within the using data model, or be a reference to a specific
         certificate in the keystore.";
      choice local-or-keystore {
        mandatory true;
```

```
        case local {
          if-feature "local-definitions-supported";
          container local-definition {
            description
              "Container to hold the local key definition.";
            uses asymmetric-key-pair-with-cert-grouping;
          }
        }
        case keystore {
          if-feature "keystore-supported";
          container keystore-reference {
            uses asymmetric-key-certificate-ref-grouping;
            description
              "A reference to a specific certificate (and its
               associated private key) in the keystore.";
          }
        }
        description
          "A choice between an inlined definition and a definition
           that exists in the keystore.";
      }
    }

    grouping keystore-grouping {
      description
        "Grouping definition enables use in other contexts.  If ever
         done, implementations SHOULD augment new 'case' statements
         into local-or-keystore 'choice' statements to supply leafrefs
         to the new location.";
      container asymmetric-keys {
        description
          "A list of asymmetric keys.";
        list asymmetric-key {
          key "name";
          description
            "An asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the asymmetric key.";
          }
```

```
      uses ks:asymmetric-key-pair-with-certs-grouping;
    }
  }
  container symmetric-keys {
    description
      "A list of symmetric keys.";
    list symmetric-key {
      key "name";
      description
        "A symmetric key.";
      leaf name {
        type string;
        description
          "An arbitrary name for the symmetric key.";
      }
      uses ks:symmetric-key-grouping;
    }
  }
} // grouping keystore-grouping


/****************/
/*   Augments   */
/****************/

augment "/ct:generate-symmetric-key/ct:input" {
  description
    "This augmentation adds an input parameter to the
     'generate-symmetric-key' RPC to specify another key
     that is to be used to encypt the generated key
     before it is returned in the RPC-reply.";
  container encrypt-with {
    presence
      "Indicates that the key should be encrypted using
       the specified symmetric or asymmetric key.  If not
       specified, then the private key is not encrypted
       when returned.";
    description
      "A container for the 'key-type' choice.";
    uses key-reference-type-grouping;
  }
}

augment "/ct:generate-symmetric-key/ct:output/ct:key-type" {
  description
    "This augmentation extends the 'generate-symmetric-key'
     RPC-reply to encode an encrypted key.";
  container encrypted-key {
```

```
        must "../ct:key-format";
        description
          "A container for the encrypted symmetric key value.";
        uses encrypted-value-grouping;
      }
    }

    augment "/ct:generate-asymmetric-key/ct:input" {
      description
        "This augmentation adds an input parameter to the
         'generate-asymmetric-key' RPC to specify another key
         that is to be used to encypt the generated key
         before it is returned in the RPC-reply.";
      container encrypt-with {
        presence
          "Indicates that the key should be encrypted using
           the specified symmetric or asymmetric key.  If not
           specified, then the private key is not encrypted
           when returned.";
        description
          "A container for the 'key-type' choice.";
        uses key-reference-type-grouping;
      }
    }

    augment
      "/ct:generate-asymmetric-key/ct:output/ct:private-key-type" {
      description
        "This augmentation extends the 'generate-asymmetric-key'
         RPC-reply to encode an encrypted key.";
      container encrypted-private-key {
        must "../ct:private-key-format";
        description
          "A container for the encrypted asymmetric private
           key value.";
        uses encrypted-value-grouping;
      }
    }


    /*********************************/
    /*   Protocol accessible nodes   */
    /*********************************/
```

```
    container keystore {
      nacm:default-deny-write;
      description
        "The keystore contains a list of keys.";
```



```
      uses keystore-grouping;
    }

  }
```

<CODE ENDS>

4. Support for Built-in Keys

In some implementations, a device's hardware may define some built-in keys set during the manufacturing process, and/or the operating system the device runs may dynamically generate some "hidden" keys upon first boot. As an example, a built-in key may exist in conjunction with a secure device identity certificate (e.g., an IDevID certificate).

Built-in keys are expected to be set by a vendor-specific process. Any ability for operators to modify the built-in keys is outside the scope of this docuemnt.

As built-in keys are provided by the system (not configuration), they are present in <operational>. The following example illustrates built-in keys in <operational>.

(FIXME: add illustration with origin="system" here)

In order for the built-in keys to be referenced by configuration, they must first be copied into <intended> as the example in Section 3.2 illustrates for the built-in keys above. Note that this strategy is chosen, rather then setting "require-instance false" for the various leafrefs, as built-in keys are relatively few in number and hence not worth relaxing the validation for.

5. Support for RMAs

FIXME: https://mailarchive.ietf.org/arch/msg/netconf/

pn0LucnWx3Xz0rfBRqB31BVg-Bk

6. Security Considerations

The YANG module defined in this document is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The NETCONF access control model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

There are a number of data nodes defined in this YANG module 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:

/: The entire data tree defined by this module is sensitive to write operations. For instance, the addition or removal of keys, certificates, etc., can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for the entire data tree.

/keystore/asymmetric-keys/asymmetric-key/private-key: When writing this node, implementations MUST ensure that the strength of the key being configured is not greater than the strength of the underlying secure transport connection over which it is communicated. Implementations SHOULD fail the write-request if ever the strength of the private key is greater then the strength of the underlying transport, and alert the client that the strength of the key may have been compromised. Additionally, when deleting this node, implementations SHOULD automatically (without explicit request)

zeroize these keys in the most secure manner available, so as to prevent the remnants of their persisted storage locations from being analyzed in any meaningful way.

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:

/keystore/asymmetric-keys/asymmetric-key/private-key: This node is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. The best reason for returning this node is to support backup/restore type workflows. For this reason, the NACM extension "default-deny-all" has been set for this data node.

7. IANA Considerations

7.1. The IETF XML Registry

This document registers one URI in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

```
URI: urn:ietf:params:xml:ns:yang:ietf-keystore
Registrant Contact: The NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

7.2. The YANG Module Names Registry

This document registers one YANG module in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the the following registration is requested:

```
name:         ietf-keystore
namespace:    urn:ietf:params:xml:ns:yang:ietf-keystore
prefix:       ks
reference:    RFC XXXX
```

8. References

8.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K. and H. Wang, "Common YANG Data Types for Cryptography", draft-ietf-netconf-crypto-types-13 (work in progress), November 2019.

[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>.

[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>.

[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>.

[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>.

8.2. Informative References

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

[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>.

[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>.

[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>.

[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>.

[Std-802.1AR-2009] Group, W. -. H. L. L. P. W., "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <http://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

Appendix A. Change Log

A.1. 00 to 01

- Replaced the 'certificate-chain' structures with PKCS#7 structures. (Issue #1)

- Added 'private-key' as a configurable data node, and removed the

'generate-private-key' and 'load-private-key' actions. (Issue #2)

- Moved 'user-auth-credentials' to the ietf-ssh-client module. (Issues #4 and #5)

A.2. 01 to 02

- Added back 'generate-private-key' action.
- Removed 'RESTRICTED' enum from the 'private-key' leaf type.
- Fixed up a few description statements.

A.3. 02 to 03

- Changed draft's title.
- Added missing references.
- Collapsed sections and levels.
- Added RFC 8174 to Requirements Language Section.
- Renamed 'trusted-certificates' to 'pinned-certificates'.
- Changed 'public-key' from config false to config true.
- Switched 'host-key' from OneAsymmetricKey to definition from RFC 4253.

A.4. 03 to 04

- Added typedefs around leafrefs to common keystore paths
- Now tree diagrams reference ietf-netmod-yang-tree-diagrams
- Removed Design Considerations section
- Moved key and certificate definitions from data tree to groupings

A.5. 04 to 05

- Removed trust anchors (now in their own draft)
- Added back global keystore structure
- Added groupings enabling keys to either be locally defined or a reference to the keystore.

A.6. 05 to 06

- Added feature "local-keys-supported"
- Added nacm:default-deny-all and nacm:default-deny-write
- Renamed generate-asymmetric-key to generate-hidden-key
- Added an install-hidden-key action
- Moved actions inside fo the "asymmetric-key" container
- Moved some groupings to draft-ietf-netconf-crypto-types

A.7. 06 to 07

- Removed a "require-instance false"
- Clarified some description statements
- Improved the keystore-usage examples

A.8. 07 to 08

- Added "local-definition" containers to avoid posibility of the action/notification statements being under a "case" statement.
- Updated copyright date, boilerplate template, affiliation, folding algorithm, and reformatted the YANG module.

A.9. 08 to 09

- Added a 'description' statement to the 'must' in the /keystore/asymmetric-key node explaining that the descendent values may exist in <operational> only, and that implementation MUST assert that the values are either configured or that they exist in <operational>.

- Copied above 'must' statement (and description) into the local-or-keystore-asymmetric-key-grouping, local-or-keystore-asymmetric-key-with-certs-grouping, and local-or-keystore-end-entity-cert-with-key-grouping statements.

A.10. 09 to 10

- Updated draft title to match new truststore draft title
- Moved everything under a top-level 'grouping' to enable use in other contexts.
- Renamed feature from 'local-keys-supported' to 'local-definitions-supported' (same name used in truststore)
- Removed the either-all-or-none 'must' expressions for the key's 3-tuple values (since the values are now 'mandatory true' in crypto-types)
- Example updated to reflect 'mandatory true' change in crypto-types draft

A.11. 10 to 11

- Replaced typedef asymmetric-key-certificate-ref with grouping asymmetric-key-certificate-ref-grouping.
- Added feature feature 'key-generation'.
- Cloned groupings symmetric-key-grouping, asymmetric-key-pair-grouping, asymmetric-key-pair-with-cert-grouping, and asymmetric-key-pair-with-certs-grouping from crypto-keys, augmenting into each new case statements for values that have been encrypted by other keys in the keystore. Refactored keystore model to use these groupings.
- Added new 'symmetric-keys' lists, as a sibling to the existing 'asymmetric-keys' list.
- Added RPCs (not actions) 'generate-symmetric-key' and 'generate-asymmetric-key' to *return* a (potentially encrypted) key.

A.12. 11 to 12

- Updated to reflect crypto-type's draft using enumerations over identities.

- Added examples for the 'generate-symmetric-key' and 'generate-asymmetric-key' RPCs.
- Updated the Introduction section.

A.13. 12 to 13

- Updated examples to incorporate new "key-format" identities.
- Made the two "generate-*-key" RPCs be "action" statements instead.

A.14. 13 to 14

- Updated YANG module and examples to incorporate the new iana-*-algorithm modules in the crypto-types draft..

A.15. 14 to 15

- Added new "Support for Built-in Trust Anchors" section.
- Added 'must' expressions asserting that the 'key-format' leaf whenever an encrypted key is specified.
- Added local-or-keystore-symmetric-key-grouping for PSK support.

A.16. 15 to 16

- Moved the generate key actions to ietf-crypt-types as RPCs, which are augmented by ietf-keystore to support encrypted keys. Examples updated accordingly.
- Added a SSH certificate-based key (RFC 6187) and a raw private key to the example instance document (partly so they could be referenced by examples in the SSH and TLS client/server drafts.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy

Bierman, Benoit Claise, Bert Wijnen, Balazs Kovacs, David Lamparter, Eric Voit, Ladislav Lhotka, Liang Xia, Juergen Schoenwaelder, Mahesh Jethanandani, Martin Bjorklund, Mehmet Ersue, Phil Shafer, Radek Krejci, Ramkumar Dhanapal, Reshad Rahman, Sean Turner, and Tom Petch.

Author's Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net