Abstract

This document presents a YANG 1.1 (RFC 7950) module defining identities, typedefs, and groupings useful to cryptographic applications.

Editorial Note (To be removed by RFC Editor)

This draft contains 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 this draft

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

* "2020-07-10" --> 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-DraftYang Data Types and Groupings for Cryptography

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 11 January 2021.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction .......................... 3
   1.1. Relation to other RFCs ............... 3
   1.2. Specification Language ............... 5
   1.3. Adherence to the NMDA .......... 5
2. The "ietf-crypto-types" Module .......... 5
   2.1. Data Model Overview ............... 5
   2.2. Example Usage .................. 16
   2.3. YANG Module .................. 23
3. Security Considerations ............... 40
   3.1. No Support for CRMF ............... 41
   3.2. No Support for Key Generation ...... 41
   3.3. Strength of Keys Configured ....... 41
   3.4. Deletion of Cleartext Key Values .... 41
   3.5. The "ietf-crypto-types" YANG Module .. 41
4. IANA Considerations ............... 43
   4.1. The "IETF XML" Registry .......... 43
   4.2. The "YANG Module Names" Registry ... 43
5. References .................. 43
   5.1. Normative References .............. 43
   5.2. Informative References .......... 45
1. Introduction

This document presents a YANG 1.1 [RFC7950] module defining identities, typedefs, and groupings useful to cryptographic applications.

1.1. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to define configuration modules for clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The modules have been defined in a modular fashion to enable their use by other efforts, some of which are known to be in progress at the time of this writing, with many more expected to be defined in time.

The relationship between the various RFCs in the collection is...
presented in the below diagram. The labels in the diagram represent the primary purpose provided by each RFC. Links the each RFC are provided below the diagram.
Table 1: Label to RFC Mapping

<table>
<thead>
<tr>
<th>keystore</th>
<th>[I-D.ietf-netconf-keystore]</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp-client-server</td>
<td>[I-D.ietf-netconf-tcp-client-server]</td>
</tr>
<tr>
<td>ssh-client-server</td>
<td>[I-D.ietf-netconf-ssh-client-server]</td>
</tr>
<tr>
<td>tls-client-server</td>
<td>[I-D.ietf-netconf-tls-client-server]</td>
</tr>
<tr>
<td>netconf-client-server</td>
<td>[I-D.ietf-netconf-netconf-client-server]</td>
</tr>
<tr>
<td>restconf-client-server</td>
<td>[I-D.ietf-netconf-restconf-client-server]</td>
</tr>
</tbody>
</table>

1.2. Specification 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.

1.3. Adherence to the NMDA

This document in compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. It does not define any protocol accessible nodes that are "config false".

2. The "ietf-crypto-types" Module

This section defines a YANG 1.1 [RFC7950] module that defines data types (typedefs and identities) and groupings supporting downstream models needing cryptographic primitives.

2.1. Data Model Overview
2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-crypt-types" module:

Features:
- one-symmetric-key-format
- one-asymmetric-key-format
- encrypted-one-symmetric-key-format
- encrypted-one-asymmetric-key-format
- certificate-signing-request-generation

2.1.2. Identities

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-crypto-types" module:

Identities:
- public-key-format
  - subject-public-key-info-format
  - ssh-public-key-format
- private-key-format
  - rsa-private-key-format
  - ec-private-key-format
  - one-asymmetric-key-format
    - {one-asymmetric-key-format}?
    - encrypted-one-asymmetric-key-format
      - {encrypted-one-asymmetric-key-format}?
- symmetric-key-format
  - octet-string-key-format
  - one-symmetric-key-format
    - {one-symmetric-key-format}?
The diagram shows that there are three base identities. These identities are used by this module to define the format that key data is encoded in. The base identities are "abstract", in the object orientied programming sense, in that they only define a "class" of formats, rather than a specific format.

The various derived identities define specific key encoding formats. The derived identities defined in this document are sufficient for the effort described in Section 1.1 but, by nature of them being identities, additional derived identities MAY be defined by future efforts.

Identities use to specify uncommon formats are enabled by "feature" statements, enabling applications to support them when needed.

2.1.3. Typedefs

The following diagram illustrates the relationship amongst the "typedef" statements defined in the "ietf-crypto-types" module:

```yaml
+-- binary
    +-- csr-info
    +-- csr
    +-- x509
    |   +-- trust-anchor-cert-x509
    |   +-- end-entity-cert-x509
    +-- crl
```
--- ocsp-request
--- ocsp-response
--- cms
  --- data-content-cms
  --- signed-data-cms
  | --- trust-anchor-cert-cms
  | --- end-entity-cert-cms
  --- enveloped-data-cms
  --- digested-data-cms
  --- encrypted-data-cms
  --- authenticated-data-cms

Comments:

* All of the typedefs defined in the "ietf-crypto-types" module extend the "binary" type defined in [RFC7950].

* Additionally, all the typedefs define a type for encoding an ASN.1 [ITU.X680.2015] structure using DER [ITU.X690.2015].

* The "trust-anchor-*" and "end-entity-*" typedefs are syntactically identical to their base typedefs and only distinguish themselves by the expected nature of their content. These typedefs are defined to facilitate common modeling needs.

2.1.4. Groupings

The following diagram lists all the "grouping" statements defined in the "ietf-crypto-types" module:

Groupings:
  --- encrypted-key-value-grouping
  --- symmetric-key-grouping
  --- public-key-grouping
  --- asymmetric-key-pair-grouping
  --- trust-anchor-cert-grouping
  --- end-entity-cert-grouping
  --- generate-csr-grouping
  --- asymmetric-key-pair-with-cert-grouping
  --- asymmetric-key-pair-with-certs-grouping

Each of these groupings are presented in the following subsections.
2.1.4.1. The "encrypted-key-value-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "encrypted-key-value-grouping" grouping:

```plaintext
grouping encrypted-key-value-grouping
  +-- encrypted-by
  +-- encrypted-value  binary
```

Comments:
* The "encrypted-by" node is an empty container (difficult to see in the diagram) that a consuming module MUST augment key references into. The "ietf-crypto-types" module is unable to populate this container as the module only defines groupings. Section 2.2.1 presents an example illustrating a consuming module populating the "encrypted-by" container.
* The "encrypted-value" node is the key, encrypted by the other key referenced by the "encrypted-by" node, encoded in the format specified by the "format" identity Section 2.1.2 associated with the ancestor node using this grouping.

2.1.4.2. The "symmetric-key-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "symmetric-key-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```plaintext
grouping symmetric-key-grouping
  +-- key-format?  identityref
  +-- (key-type)
      ++-- (cleartext-key)
          |  ++-- cleartext-key?  binary
          ++-- (hidden-key)
          |  ++-- hidden-key?  empty
          ++-- (encrypted-key)
              ++-- encrypted-key
                  +----u encrypted-key-value-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:
grouping symmetric-key-grouping
  +- key-format?            identityref
  +- (key-type)
    +-:(cleartext-key)
      |  +- cleartext-key?   binary
    +-:(hidden-key)
      |  +- hidden-key?      empty
    +-:(encrypted-key)
      +- encrypted-key
        +- encrypted-by
        +- encrypted-value  binary

Comments:

* For the referenced grouping statement(s):

  - The "encrypted-key-value-grouping" grouping is discussed in Section 2.1.4.1.

* The "key-format" node is an identity-reference to the "symmetric-key-format" abstract base identity discussed in Section 2.1.2, enabling the symmetric key to be encoded using the format defined by any of the derived identities.

* The "choice" statement enables the private key data to be plain-text, encrypted, or hidden, as follows:

  - The "key" node can encode any plain-text key value.
  - The "hidden-key" node is of type "empty" as the real value cannot be presented via the management interface.
  - The "encrypted-key" node's structure is discussed in Section 2.1.4.1.

2.1.4.3. The "public-key-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "public-key-grouping" grouping:

grouping public-key-grouping
  +- public-key-format    identityref
  +- public-key          binary

Comments:

* The "public-key-format" node is an identity-reference to the "public-key-format" abstract base identity discussed in
Section 2.1.2, enabling the public key to be encoded using the format defined by any of the derived identities.

The "public-key" node is the public key data in the selected format. No "choice" statement is used to hide or encrypt the public key data because it is unnecessary to do so for public keys.

2.1.4.4. The "asymmetric-key-pair-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "asymmetric-key-pair-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
  grouping asymmetric-key-pair-grouping
    +---u public-key-grouping
    ++--- private-key-format? identityref
    ++--- (private-key-type)
       |     +---:(cleartext-private-key)
       |     +--- cleartext-private-key? binary
       |     +---:(hidden-private-key)
       |     +--- hidden-private-key? empty
       +---:(encrypted-private-key)
          +--- encrypted-private-key
          +---u encrypted-key-value-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
  grouping asymmetric-key-pair-grouping
    ++--- public-key-format identityref
    ++--- public-key binary
    ++--- private-key-format? identityref
    ++--- (private-key-type)
       |     +---:(cleartext-private-key)
       |     +--- cleartext-private-key? binary
       |     +---:(hidden-private-key)
       |     +--- hidden-private-key? empty
       +---:(encrypted-private-key)
          +--- encrypted-private-key
          +--- encrypted-by
          +--- encrypted-value binary
```
Comments:

* For the referenced grouping statement(s):

  - The "public-key-grouping" grouping is discussed in Section 2.1.4.3.
  - The "encrypted-key-value-grouping" grouping is discussed in Section 2.1.4.1.

2.1.4.5. The "certificate-expiration-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "certificate-expiration-grouping" grouping:

grouping certificate-expiration-grouping
  +---n certificate-expiration
  +-- expiration-date yang:date-and-time

Comments:

* This grouping's only purpose is to define the "certificate-expiration" notification statement, used by the groupings defined in Section 2.1.4.6 and Section 2.1.4.7.

* The "certificate-expiration" notification enables servers to notify clients when certificates are nearing expiration.
* The "expiration-date" node indicates when the designated certificate will (or did) expire.

* Identification of the certificate that is expiring is built into the notification itself. For an example, please see Section 2.2.3.

2.1.4.6. The "trust-anchor-cert-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "trust-anchor-cert-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
grouping trust-anchor-cert-grouping
   +-- cert-data?                         trust-anchor-cert-cms
   +---u certificate-expiration-grouping

Watsen                   Expires 11 January 2021               [Page 11]
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
grouping trust-anchor-cert-grouping
   +-- cert-data?                trust-anchor-cert-cms
   +--n certificate-expiration
       +-- expiration-date yang:date-and-time

Comments:

* For the referenced grouping statement(s):
  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.5.

* The "cert-data" node contains a chain of one or more certificates encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

2.1.4.7. The "end-entity-cert-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "end-entity-cert-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
grouping trust-anchor-cert-grouping
   +-- cert-data?                trust-anchor-cert-cms
   +--n certificate-expiration
       +-- expiration-date yang:date-and-time
```
The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
grouping end-entity-cert-grouping
  +-- cert-data?                         end-entity-cert-cms
    +---u certificate-expiration-grouping

Comments:
* For the referenced grouping statement(s):
  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.5.
* The "cert-data" node contains a chain of one or more certificates encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

2.1.4.8. The "generate-csr-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "generate-csr-grouping" grouping:

```
grouping generate-csr-grouping
  +---x generate-certificate-signing-request
    {certificate-signing-request-generation}? 
      +----w input
        |       +----w csr-info    ct:csr-info
        |      +--ro output
        |          +--ro certificate-signing-request ct:csr

Comments:
* This grouping's only purpose is to define the "generate-certificate-signing-request" action statement, used by the groupings defined in Section 2.1.4.9 and Section 2.1.4.10.
2.1.4.9. The "asymmetric-key-pair-with-cert-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "asymmetric-key-pair-with-cert-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):

```
grouping asymmetric-key-pair-with-cert-grouping
  +---u asymmetric-key-pair-grouping
  +---u end-entity-cert-grouping
  +---u generate-csr-grouping
```

The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
grouping asymmetric-key-pair-with-cert-grouping
  +-- public-key-format                       identityref
  +-- public-key                              binary
  +-- private-key-format?                     identityref
    |  +--:(cleartext-private-key)            binary
    |  |  +-- cleartext-private-key?           binary
    |  +--:(hidden-private-key)               empty
    |     +-- hidden-private-key?             empty
    +--:(encrypted-private-key)
```
This grouping defines an asymmetric key with at most one associated certificate, a commonly needed combination in protocol models.

For the referenced grouping statement(s):

- The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.4.
- The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.7.
- The "generate-csr-grouping" grouping is discussed in Section 2.1.4.8.

2.1.4.10. The "asymmetric-key-pair-with-certs-grouping" Grouping

This section presents two tree diagrams [RFC8340] illustrating the "asymmetric-key-pair-with-certs-grouping" grouping. The first tree diagram does not expand the internally used grouping statement(s):
The following tree diagram expands the internally used grouping statement(s), enabling the grouping's full structure to be seen:

```
grouping asymmetric-key-pair-with-certs-grouping
  +-- public-key-format identityref
  +-- public-key binary
  +-- private-key-format? identityref
  +-- (private-key-type)
    |  +--:(cleartext-private-key)
    |    |  +-- cleartext-private-key? binary
    |    +--:(hidden-private-key)
    |    |  +-- hidden-private-key? empty
    |    +--:(encrypted-private-key)
    |    +-- encrypted-private-key
    |    |  +-- encrypted-by
    |    |     +-- encrypted-value binary
  +-- certificates
    |  +-- certificate* [name]
    |    +-- name? string
    |    +-- cert-data end-entity-cert-cms
    |     +-- certificate-expiration
    |       +-- expiration-date yang:date-and-time
    +---x generate-certificate-signing-request
      {certificate-signing-request-generation}?
        +--w input
        |  +--w csr-info ct:csr-info
        +--ro output
        +--ro certificate-signing-request ct:csr
```

Comments:

* This grouping defines an asymmetric key with one or more associated certificates, a commonly needed combination in configuration models.

* For the referenced grouping statement(s):

  - The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.4.
- The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.7.
- The "generate-csr-grouping" grouping is discussed in Section 2.1.4.8.

2.1.5. Protocol-accessible Nodes

The "ietf-crypto-types" module does not contain any protocol-accessible nodes, but the module needs to be "implemented", as described in Section 5.6.5 of [RFC7950], in order for the identities in Section 2.1.2 to be defined.

2.2. Example Usage

2.2.1. The "symmetric-key-grouping" and "asymmetric-key-pair-with-certs-grouping" Grouping

The following non-normative module is constructed in order to illustrate the use of the "symmetric-key-grouping" (Section 2.1.4.2) and the "asymmetric-key-pair-with-certs-grouping" (Section 2.1.4.10) grouping statements:

module ex-crypto-types-usage {
  yang-version 1.1;

  namespace "http://example.com/ns/example-crypto-types-usage";
  prefix "ectu";

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

  organization "Example Corporation";
  contact "YANG Designer <mailto:yang.designer@example.com>";

  description
    "This module illustrates the 'symmetric-key-grouping'
    and 'asymmetric-key-pair-with-certs-grouping' groupings defined in
    the 'ietf-crypto-types' module defined in RFC AAAA."

  revision "2020-07-10" {
    description
      "Initial version";
    reference
      "RFC AAAA: Common YANG Data Types for Cryptography";
container symmetric-keys {
    description "A container of symmetric keys."
    list symmetric-key {
        key name;
        description "A symmetric key"
        leaf name {
            type string;
            description "An arbitrary name for this key."
        }
    }
}
uses ct:symmetric-key-grouping {
    augment "key-type/encrypted-key/encrypted-key/" + "encrypted-by" {
        description "Augments in a choice statement enabling the encrypting key to be any other symmetric or asymmetric key."
        uses encrypted-by-choice-grouping;
    }
}
}

container asymmetric-keys {
    description "A container of asymmetric keys."
    list asymmetric-key {
        key name;
        leaf name {
            type string;
            description "An arbitrary name for this key."
        }
    }
}
uses ct:asymmetric-key-pair-with-certs-grouping {
    augment "private-key-type/encrypted-private-key/" + "encrypted-private-key/encrypted-by" {
        description "Augments in a choice statement enabling the
encrypting key to be any other symmetric or asymmetric key."
uses encrypted-by-choice-grouping;
}
}
description
"An asymmetric key pair with associated certificates."
;
}
The tree diagram [RFC8340] for this example module follows:

module: ex-crypto-types-usage
  +--rw symmetric-keys
    |  +--rw symmetric-key* [name]
    |     +--rw name                   string
    |     +--rw key-format?            identityref
    |     +--rw (key-type)
    |        +--:(cleartext-key)
    |           +--rw cleartext-key?   binary
    |        +--:(hidden-key)
    |           +--rw hidden-key?     empty
    |        +--:(encrypted-key)
    |           +--rw encrypted-key

  +--rw asymmetric-keys
    +--rw asymmetric-key* [name]
      +--rw name                                    string
      +--rw public-key-format                       identityref
      +--rw public-key                              binary
      +--rw private-key-format?                     identityref
      +--rw (private-key-type)
      |  +--:(cleartext-private-key)
      |         +--rw cleartext-private-key?          binary
      |         +--:(hidden-private-key)
      |           +--rw hidden-private-key?           empty
      |         +--:(encrypted-private-key)
      |           +--rw encrypted-private-key
      |             +--rw encrypted-by
      |                +--rw (encrypted-by-choice)
      |                       +--:(symmetric-key-ref)
      |                           +--rw symmetric-key-ref? leafref
      |                           +--:(asymmetric-key-ref)
      |                                     +--rw asymmetric-key-ref? leafref
      +--rw encrypted-value                       binary

grouping encrypted-by-choice-grouping
  (encrypted-by-choice)
    (symmetric-key-ref)
      symmetric-key-ref?
      /symmetric-keys/symmetric-key/name
    (asymmetric-key-ref)
      asymmetric-key-ref?
      /asymmetric-keys/asymmetric-key/name

Finally, the following example illustrates various symmetric and asymmetric keys as they might appear in configuration:

================= NOTE: '\ line wrapping per RFC 8792 =================
<asymmetric-key>
  <name>ex-one-asymmetric-based-symmetric-key</name>
  <public-key-format>
    ct:subject-public-key-info-format
  </public-key-format>
  <public-key>base64encodedvalue==</public-key>
  <private-key-format>
    ct:one-asymmetric-key-format
  </private-key-format>
  <cleartext-private-key>base64encodedvalue==</cleartext-private-key>
</asymmetric-key>

<asymmetric-key>
  <name>ex-encrypted-one-asymmetric-based-symmetric-key</name>
  <public-key-format>
    ct:subject-public-key-info-format
  </public-key-format>
  <public-key>base64encodedvalue==</public-key>
  <private-key-format>
    ct:encrypted-one-asymmetric-key-format
  </private-key-format>
  <encrypted-private-key>
    <encrypted-by>
      <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetric-key</symmetric-key-ref>
    </encrypted-by>
    <encrypted-value>base64encodedvalue==</encrypted-value>
  </encrypted-private-key>
</asymmetric-key>

<asymmetric-key>
  <name>ex-encrypted-one-asymmetric-based-symmetric-key</name>
  <public-key-format>
    ct:subject-public-key-info-format
  </public-key-format>
  <public-key>base64encodedvalue==</public-key>
  <private-key-format>
    ct:encrypted-one-asymmetric-key-format
  </private-key-format>
  <encrypted-private-key>
    <encrypted-by>
      <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetric-key</symmetric-key-ref>
    </encrypted-by>
    <encrypted-value>base64encodedvalue==</encrypted-value>
  </encrypted-private-key>
</asymmetric-key>

<asymmetric-key>
  <name>ex-encrypted-one-asymmetric-based-symmetric-key</name>
  <public-key-format>
    ct:subject-public-key-info-format
  </public-key-format>
  <public-key>base64encodedvalue==</public-key>
  <private-key-format>
    ct:encrypted-one-asymmetric-key-format
  </private-key-format>
  <encrypted-private-key>
    <encrypted-by>
      <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetric-key</symmetric-key-ref>
    </encrypted-by>
    <encrypted-value>base64encodedvalue==</encrypted-value>
  </encrypted-private-key>
</asymmetric-key>
2.2.2. The "generate-certificate-signing-request" Action

The following example illustrates the "generate-certificate-signing-request" action, discussed in Section 2.1.4.8, with the NETCONF protocol.

REQUEST

```xml
<rpc message-id="101"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="urn:ietf:params:xml:ns:yang:1">
   <asymmetric-keys
       xmlns="http://example.com/ns/example-crypto-types-usage">
     <asymmetric-key>
       <name>ex-key-sect571r1</name>
       <generate-certificate-signing-request>
         <csr-info>base64encodedvalue==</csr-info>
       </generate-certificate-signing-request>
     </asymmetric-key>
   </asymmetric-keys>
 </action>
</rpc>
```

RESPONSE

```xml
<rpc-reply message-id="101"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <certificate-signing-request
     xmlns="http://example.com/ns/example-crypto-types-usage">
   base64encodedvalue==
 </certificate-signing-request>
</rpc-reply>
```

2.2.3. The "certificate-expiration" Notification

The following example illustrates the "certificate-expiration" notification, discussed in Section 2.1.4.5, with the NETCONF protocol.
<notification>
  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2018-05-25T00:01:00Z</eventTime>
  <asymmetric-keys xmlns="http://example.com/ns/example-crypto-types-usage">
    <asymmetric-key>
      <name>ex-hidden-asymmetric-key</name>
      <certificates>
        <certificate>
          <name>ex-hidden-asymmetric-key</name>
          <certificate-expiration>
            <expiration-date>2018-08-05T14:18:53-05:00</expiration-date>
          </certificate-expiration>
        </certificate>
      </certificates>
    </asymmetric-key>
  </asymmetric-keys>
</notification>

2.3. YANG Module

This module has normative references to [RFC2119], [RFC2986], [RFC3447], [RFC4253], [RFC5280], [RFC5652], [RFC5915], [RFC5958], [RFC6031], [RFC6125], [RFC6991], [RFC8174], [RFC8341], and [ITU.X690.2015].

<CODE BEGINS> file "ietf-crypto-types@2020-07-10.yang"

module ietf-crypto-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-crypto-types";
  prefix ct;

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

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

</CODE BEGINS>
Internet-Draft YANG Data Types and Groupings for Cryptography

July 2020

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 common YANG types for cryptographic applications.

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 AAAA (https://www.rfc-editor.org/info/rfcAAAA); 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 2020-07-10 {

description

"Initial version"

reference

"RFC AAAA: YANG Data Types and Groupings for Cryptography"

}

/********************/
feature one-symmetric-key-format {
    description "Indicates that the server supports the
    'one-symmetric-key-format' identity.";
}

feature one-asymmetric-key-format {
    description "Indicates that the server supports the
    'one-asymmetric-key-format' identity.";
}

feature encrypted-one-symmetric-key-format {
    description "Indicates that the server supports the
    'encrypted-one-symmetric-key-format' identity.";
}

feature encrypted-one-asymmetric-key-format {
    description "Indicates that the server supports the
    'encrypted-one-asymmetric-key-format' identity.";
}

feature certificate-signing-request-generation {
    description "Indicates that the server implements the
    'generate-certificate-signing-request' action.";
}

identity symmetric-key-format {
    description "Base key-format identity for symmetric keys.";
}
identity public-key-format {
  description "Base key-format identity for public keys.";
}

identity private-key-format {
  description "Base key-format identity for private keys.";
}

identity rsa-private-key-format {
  base "private-key-format";
  description "Indicates that the private key value is encoded as an RSAPrivateKey (from RFC 3447).";
  reference "RFC 3447: PKCS #1: RSA Cryptography Specifications Version 2.2";
}

identity ec-private-key-format {
  base "private-key-format";
  description "Indicates that the private key value is encoded as an ECPrivateKey (from RFC 5915)";
  reference "RFC 5915: Elliptic Curve Private Key Structure";
}

identity one-asymmetric-key-format {
  if-feature "one-asymmetric-key-format";
  base "private-key-format";
  description "Indicates that the private key value is a CMS OneAsymmetricKey structure, as defined in RFC 5958, encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690.";
  reference
}
identity encrypted-one-asymmetric-key-format {
  if-feature "encrypted-one-asymmetric-key-format";
  base "private-key-format";
  description
    "Indicates that the private key value is a CMS EnvelopedData structure, per Section 8 in RFC 5652, containing a OneAsymmetricKey structure, as defined in RFC 5958, encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690.";
  reference
    "RFC 5652: Cryptographic Message Syntax (CMS)"
    RFC 5958: Asymmetric Key Packages
}

identity ssh-public-key-format {
  base "public-key-format";
  description
    "Indicates that the public key value is an SSH public key, as specified by RFC 4253, Section 6.6, i.e.:

    string certificate or public key format identifier
    byte[n] key/certificate data.";
identity subject-public-key-info-format {
    base "public-key-format";
    description
        "Indicates that the public key value is a SubjectPublicKeyInfo structure, as described in RFC 5280 encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690.";
    reference
        "RFC 5280:
         Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
      ITU-T X.690:
      Information technology - ASN.1 encoding rules:
      Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).";
}

identity octet-string-key-format {
    base "symmetric-key-format";
    description
        "Indicates that the key is encoded as a raw octet string. The length of the octet string MUST be appropriate for the associated algorithm's block size.";
}

identity one-symmetric-key-format {
    if-feature "one-symmetric-key-format";
    base "symmetric-key-format";
    description
        "Indicates that the private key value is a CMS OneSymmetricKey structure, as defined in RFC 6031,";
encoded using ASN.1 distinguished encoding rules (DER), as specified in ITU-T X.690.";

reference

"RFC 6031": Cryptographic Message Syntax (CMS)

Symmetric Key Package Content Type

ITU-T X.690:
Information technology - ASN.1 encoding rules:
Specification of Basic Encoding Rules (BER),
Canonical Encoding Rules (CER) and Distinguished
Encoding Rules (DER).";

identity encrypted-one-symmetric-key-format {
  if-feature "encrypted-one-symmetric-key-format";
  base "symmetric-key-format";
  description
    "Indicates that the private key value is a CMS
EnvelopedData structure, per Section 8 in RFC 5652,
containing a OneSymmetricKey structure, as defined
in RFC 6031, encoded using ASN.1 distinguished
encoding rules (DER), as specified in ITU-T X.690.";

reference

"RFC 5652": Cryptographic Message Syntax (CMS)

"RFC 6031": Cryptographic Message Syntax (CMS)

Symmetric Key Package Content Type

ITU-T X.690:
Information technology - ASN.1 encoding rules:
Specification of Basic Encoding Rules (BER),
Canonical Encoding Rules (CER) and Distinguished
Encoding Rules (DER).";

}";
typedef csr {
  type binary;
  description
    "A CertificationRequest structure, as specified in
    RFC 2986, encoded using ASN.1 distinguished encoding
    rules (DER), as specified in ITU-T X.690."
  reference
    "RFC 2986: PKCS #10: Certification Request Syntax
    Specification Version 1.7
    ITU-T X.690:
    Information technology - ASN.1 encoding rules:
    Specification of Basic Encoding Rules (BER),
    Canonical Encoding Rules (CER) and Distinguished
    Encoding Rules (DER).";
}

typedef x509 {
  type binary;
  description
    "A Certificate structure, as specified in RFC 5280,
    encoded using ASN.1 distinguished encoding rules (DER),
    as specified in ITU-T X.690."
  reference
    "RFC 5280:
    
    */
    typedefs for ASN.1 structures from RFC 5280 */
    
    Internet-Draft/YANG Data Types and Groupings for Cryptography  July 2020
typedef crl {
  type binary;
  description
    "A CertificateList structure, as specified in RFC 5280,
    encoded using ASN.1 distinguished encoding rules (DER),
    as specified in ITU-T X.690."
  reference
    "RFC 5280:
    Internet X.509 Public Key Infrastructure Certificate
    and Certificate Revocation List (CRL) Profile
    ITU-T X.690:
    Information technology - ASN.1 encoding rules:
    Specification of Basic Encoding Rules (BER),
    Canonical Encoding Rules (CER) and Distinguished
    Encoding Rules (DER)."
};

typedef oscp-request {
  type binary;
  description
    "A OCSPRequest structure, as specified in RFC 6960,
    encoded using ASN.1 distinguished encoding rules
    (DER), as specified in ITU-T X.690."
  reference
    "RFC 6960:
    X.509 Internet Public Key Infrastructure Online
    Certificate Status Protocol - OCSP
    ITU-T X.690:
    Information technology - ASN.1 encoding rules:
    Specification of Basic Encoding Rules (BER),
    Canonical Encoding Rules (CER) and Distinguished
    Encoding Rules (DER)."
};
typedef oscp-response {
  type binary;
  description
    "A OCSPResponse structure, as specified in RFC 6960,
    encoded using ASN.1 distinguished encoding rules
    (DER), as specified in ITU-T X.690.";
  reference
    "RFC 6960:
    X.509 Internet Public Key Infrastructure Online
    Certificate Status Protocol - OCSP
    ITU-T X.690:
    Information technology - ASN.1 encoding rules:
    Specification of Basic Encoding Rules (BER),
    Canonical Encoding Rules (CER) and Distinguished
    Encoding Rules (DER).";
}

/**************************************************************/
/*   Typedefs for ASN.1 structures from 5652   */
/**************************************************************/

typedef cms {
  type binary;
  description
    "A ContentInfo structure, as specified in RFC 5652,
    encoded using ASN.1 distinguished encoding rules (DER),
    as specified in ITU-T X.690.";
  reference
    "RFC 5652:
    Cryptographic Message Syntax (CMS)
    ITU-T X.690:
    Information technology - ASN.1 encoding rules:
    Specification of Basic Encoding Rules (BER),
    Canonical Encoding Rules (CER) and Distinguished
    Encoding Rules (DER).";
}

typedef data-content-cms {
  type cms;
  description
    "A CMS structure whose top-most content type MUST be the
    data content type, as described by Section 4 in RFC 5652.";
typedef signed-data-cms {

type cms;

description
   "A CMS structure whose top-most content type MUST be the signed-data content type, as described by Section 5 in RFC 5652.";

reference
   "RFC 5652: Cryptographic Message Syntax (CMS)";
}

typedef enveloped-data-cms {

type cms;

description
   "A CMS structure whose top-most content type MUST be the enveloped-data content type, as described by Section 6 in RFC 5652.";

reference
   "RFC 5652: Cryptographic Message Syntax (CMS)";
}

typedef digested-data-cms {

type cms;

description
   "A CMS structure whose top-most content type MUST be the digested-data content type, as described by Section 7 in RFC 5652.";

reference
   "RFC 5652: Cryptographic Message Syntax (CMS)";
}

typedef encrypted-data-cms {

type cms;

description
   "A CMS structure whose top-most content type MUST be the encrypted-data content type, as described by Section 8 in RFC 5652.";

reference
   "RFC 5652: Cryptographic Message Syntax (CMS)";
}
typedef authenticated-data-cms {
    type cms;
    description
        "A CMS structure whose top-most content type MUST be the
         authenticated-data content type, as described by Section 9
         in RFC 5652.";
    reference
        "RFC 5652: Cryptographic Message Syntax (CMS)";
}

typedef trust-anchor-cert-x509 {
    type x509;
    description
        "A Certificate structure that MUST encode a self-signed
         root certificate."
}

typedef end-entity-cert-x509 {
    type x509;
    description
        "A Certificate structure that MUST encode a certificate
         that is neither self-signed nor having Basic constraint
         CA true."
}

typedef trust-anchor-cert-cms {
    type signed-data-cms;
    description
        "A CMS SignedData structure that MUST contain the chain of
         X.509 certificates needed to authenticate the certificate"
presented by a client or end-entity.

The CMS MUST contain only a single chain of certificates. The client or end-entity certificate MUST only authenticate to last intermediate CA certificate listed in the chain.

In all cases, the chain MUST include a self-signed root certificate. In the case where the root certificate is itself the issuer of the client or end-entity certificate, only one certificate is present.

This CMS structure MAY (as applicable where this type is used) also contain suitably fresh (as defined by local policy) revocation objects with which the device can verify the revocation status of the certificates.

This CMS encodes the degenerate form of the SignedData structure that is commonly used to disseminate X.509 certificates and revocation objects (RFC 5280)."

typedef end-entity-cert-cms {
  type signed-data-cms;
  description
    "A CMS SignedData structure that MUST contain the end entity certificate itself, and MAY contain any number of intermediate certificates leading up to a trust anchor certificate. The trust anchor certificate MAY be included as well.

The CMS MUST contain a single end entity certificate. The CMS MUST NOT contain any spurious certificates.

This CMS structure MAY (as applicable where this type is used) also contain suitably fresh (as defined by local policy) revocation objects with which the device can verify the revocation status of the certificates.";

reference
  "RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.";
This CMS encodes the degenerate form of the SignedData structure that is commonly used to disseminate X.509 certificates and revocation objects (RFC 5280).

reference

"RFC 5280:
Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile."

/*****************************/
/* Groupings for keys and/or certificates */
/*****************************/

grouping encrypted-key-value-grouping {
  description
    "A reusable grouping for a value that has been encrypted by a symmetric or asymmetric key in the Keystore."
  container encrypted-by {
    nacm:default-deny-write;
    description
      "An empty container enabling references to other keys that encrypt these keys to be augmented in. The referenced key MAY be a symmetric or an asymmetric key.";
  }
}

leaf encrypted-value {
  nacm:default-deny-write;
  type binary;
  must ".../encrypted-by";
  mandatory true;
  description
    "The key data, encrypted using the referenced symmetric or asymmetric key. The format of the encrypted value is identified by the associated key format identity.";
}

grouping symmetric-key-grouping {
  description
    "A symmetric key.";
leaf key-format {
    nacm:default-deny-write;
    type identityref {
        base symmetric-key-format;
    }
    description "Identifies the symmetric key's format."
}

choice key-type {
    nacm:default-deny-write;
    mandatory true;
    description "Choice between key types."
    case cleartext-key {
        leaf cleartext-key {
            nacm:default-deny-all;
            type binary;
            must "../key-format";
            description "The binary value of the key. The interpretation of the value is defined by the 'key-format' field."
        }
    }
    case hidden-key {
        leaf hidden-key {
            type empty;
            must "not(../key-format)";
            description "A hidden key. How such keys are created is outside the scope of this module."
        }
    }
    case encrypted-key {
        container encrypted-key {
            must "../key-format";
            description "A container for the encrypted symmetric key value."
            uses encrypted-key-value-grouping;
        }
    }
}
grouping public-key-grouping {
    description "A public key.";
    leaf public-key-format {
        nacm:default-deny-write;
        type identityref {
            base public-key-format;
        }
        mandatory true;
        description "Identifies the key's format.";
    }
    leaf public-key {
        nacm:default-deny-write;
        type binary;
        mandatory true;
        description "The binary value of the public key. The interpretation of the value is defined by 'public-key-format' field.";
    }
}

grouping asymmetric-key-pair-grouping {
    description "A private key and its associated public key.";
    uses public-key-grouping;
    leaf private-key-format {
        nacm:default-deny-write;
        type identityref {
            base private-key-format;
        }
        description "Identifies the key's format.";
    }
    choice private-key-type {
        nacm:default-deny-write;
        mandatory true;
        description "Choice between key types.";
        case cleartext-private-key {
            leaf cleartext-private-key {
            nacm:default-deny-all;
        }
type binary;
must "./private-key-format";
description
    "The value of the binary key The key's value is interpreted by the 'private-key-format' field."
};

case hidden-private-key {
    leaf hidden-private-key {
        type empty;
must "not(.//private-key-format)";
description
    "A hidden key. How such keys are created is outside the scope of this module.";
    }
}

case encrypted-private-key {
    container encrypted-private-key {
        must ".//private-key-format";
description
    "A container for the encrypted asymmetric private key value.";
    uses encrypted-key-value-grouping;
    }
}
}


grouping certificate-expiration-grouping {
description
    "A notification for when a certificate is about to, or already has, expired.";
navigation certificate-expiration {
    description
    "A notification indicating that the configured certificate is either about to expire or has already expired. When to send notifications is an implementation specific decision, but it is RECOMMENDED that a notification be sent once a month for 3 months, then once a week for four weeks, and then once a day thereafter until the issue is resolved.";
    leaf expiration-date {
        type yang:date-and-time;
        mandatory true;
description
    "Identifies the expiration date on the certificate.";
    }
}
}
grouping trust-anchor-cert-grouping {
    description
        "A trust anchor certificate, and a notification for when
        it is about to (or already has) expire.";
    leaf cert-data {
        nacm:default-deny-write;
        type trust-anchor-cert-cms;
        description
            "The binary certificate data for this certificate.";
    }
    uses certificate-expiration-grouping;
}

grouping end-entity-cert-grouping {
    description
        "An end entity certificate, and a notification for when
        it is about to (or already has) expire. Implementations
        SHOULD assert that, where used, the end entity certificate
        contains the expected public key.";
    leaf cert-data {
        nacm:default-deny-write;
        type end-entity-cert-cms;
        description
            "The binary certificate data for this certificate.";
    }
    uses certificate-expiration-grouping;
}

grouping generate-csr-grouping {
    description
        "Defines the 'generate-certificate-signing-request' action.";
    action generate-certificate-signing-request {
        if-feature certificate-signing-request-generation;
        nacm:default-deny-all;
        description
            "Generates a certificate signing request structure for
            the associated asymmetric key using the passed subject
            and attribute values.

            This action statement is only available when the
            associated 'public-key-format' node's value is
            'subject-public-key-info-format'.";
        reference
            "RFC 6125":
}
input {
  leaf csr-info {
    type ct:csr-info;
    mandatory true;
    description
    "A CertificationRequestInfo structure, as defined in
    RFC 2986."

    Enables the client to provide a fully-populated
    CertificationRequestInfo structure that the server
    only needs to sign in order to generate the complete
    'CertificationRequest' structure to return in the
    'output'.

    The 'AlgorithmIdentifier' field contained inside
    the 'SubjectPublicKeyInfo' field MUST be one known
    to be supported by the device.";
    reference
    "RFC 2986:
    PKCS #10: Certification Request Syntax Specification
    RFC AAAA:
    YANG Data Types and Groupings for Cryptography";
  }
}

output {
  leaf certificate-signing-request {
    type ct:csr;
    mandatory true;
    description
    "A CertificationRequest structure, as defined in
    RFC 2986.";
    reference
    "RFC 2986:
    PKCS #10: Certification Request Syntax Specification
    RFC AAAA:
    YANG Data Types and Groupings for Cryptography";
  }
}
grouping asymmetric-key-pair-with-cert-grouping {
  description
  "A private/public key pair and an associated certificate. Implementations SHOULD assert that certificates contain the matching public key.";
}

uses asymmetric-key-pair-grouping;
uses end-entity-cert-grouping;
uses generate-csr-grouping;
} // asymmetric-key-pair-with-cert-grouping

grouping asymmetric-key-pair-with-certs-grouping {
  description
  "A private/public key pair and associated certificates. Implementations SHOULD assert that certificates contain the matching public key.";
  uses asymmetric-key-pair-grouping;
  container certificates {
    nacm:default-deny-write;
    description
      "Certificates associated with this asymmetric key. More than one certificate supports, for instance, a TPM-protected asymmetric key that has both IDevID and LDevID certificates associated.";
    list certificate {
      key "name";
      description
        "A certificate for this asymmetric key.";
      leaf name {
        type string;
        description
          "An arbitrary name for the certificate. If the name matches the name of a certificate that exists independently in <operational> (i.e., an IDevID), then the 'cert' node MUST NOT be configured.";
      }
    }
  }
  uses end-entity-cert-grouping {
    refine cert-data {

3. Security Considerations

3.1. No Support for CRMF

This document uses PKCS #10 [RFC2986] for the "generate-certificate-signing-request" action. The use of Certificate Request Message Format (CRMF) [RFC4211] was considered, but it was unclear if there was market demand for it. If it is desired to support CRMF in the future, a backwards compatible solution can be defined at that time.

3.2. No Support for Key Generation

Early revisions of this document included "rpc" statements for generating symmetric and asymmetric keys. These statements were removed due to an inability to obtain consensus for how to identify the key-algorithm to use. Thusly, the solution presented in this document only supports keys to be configured via an external client, which does not support Security best practice.

3.3. Strength of Keys Configured

When configuring key values, implementations SHOULD 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 than the strength of the underlying transport.
### 3.4. Deletion of Cleartext Key Values

This module defines storage for cleartext key values that SHOULD be zeroized when deleted, so as to prevent the remnants of their persisted storage locations from being analyzed in any meaningful way.

The cleartext key nodes are the "key" node defined in the "symmetric-key-grouping" grouping ([Section 2.1.4.2](#)) and the "private-key" node defined in the "asymmetric-key-pair-grouping" grouping (["Section 2.1.4.4](#)).

### 3.5. The "ietf-crypto-types" YANG Module

The YANG module in this document defines "grouping" statements that are 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.

Since the module in this document only define groupings, these considerations are primarily for the designers of other modules that use these groupings.

Some of the readable data nodes defined 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:

* The "key" node:

  The cleartext "key" node defined in the "symmetric-key-grouping" grouping is additionally sensitive to read operations
such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied to it.

* The "private-key" node:

The cleartext "private-key" node defined in the "asymmetric-key-pair-grouping" grouping is additionally sensitive to read operations such that, in normal use cases, it should never be returned to a client. For this reason, the NACM extension "default-deny-all" has been applied.

All of the writable data nodes defined by all the groupings defined in this module may be considered sensitive or vulnerable in some network environments. For instance, even the modification of a public key or a certificate can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been applied to all the data nodes defined in the module.

Some of the operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

* generate-certificate-signing-request:

This "action" statement SHOULD only be executed by authorized users. For this reason, the NACM extension "default-deny-all" has been applied. Note that NACM uses "default-deny-all" to protect "RPC" and "action" statements; it does not define, e.g., an extension called "default-deny-execute".

For this action, it is RECOMMENDED that implementations assert channel binding [RFC5056], so as to ensure that the application layer that sent the request is the same as the device authenticated when the secure transport layer was established.
4. IANA Considerations

4.1. 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:

- Registrant Contact: The NETCONF WG of the IETF.
- XML: N/A, the requested URI is an XML namespace.

4.2. The "YANG Module Names" Registry

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

- name: ietf-crypto-types
- prefix: ct
- reference: RFC AAAA

5. References

5.1. Normative References

[ITU.X680.2015]

[ITU.X690.2015]


5.2. Informative References

[I-D.ietf-netconf-crypto-types]

[I-D.ietf-netconf-http-client-server]

[I-D.ietf-netconf-keystore]

[I-D.ietf-netconf-netconf-client-server]

[I-D.ietf-netconf-restconf-client-server]

[I-D.ietf-netconf-ssh-client-server]
Watsen, K. and G. Wu, "YANG Groupings for SSH Clients and
SSH Servers", Work in Progress, Internet-Draft, draft-

Watsen                   Expires 11 January 2021               [Page 45]

Internet-Draft

YANG Data Types and Groupings for Cryptography    July 2020

ietf-netconf-ssh-client-server-19, 20 May 2020,

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers",
Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-06, 16 June 2020,

[I-D.ietf-netconf-tls-client-server]
Watsen, K. and G. Wu, "YANG Groupings for TLS Clients and TLS Servers",
Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-19, 20 May 2020,

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore",
Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-10, 20 May 2020,

RFC 2986, DOI 10.17487/RFC2986, November 2000,

BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004,

[RFC4211]  Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)",
RFC 4211, DOI 10.17487/RFC4211, September 2005,

[RFC5056]  Williams, N., "On the Use of Channel Bindings to Secure Channels",
RFC 5056, DOI 10.17487/RFC5056, November 2007,
Appendix A. Change Log

This section is to be removed before publishing as an RFC.
A.1. I-D to 00

* Removed groupings and notifications.
* Added typedefs for identityrefs.
* Added typedefs for other RFC 5280 structures.
* Added typedefs for other RFC 5652 structures.
* Added convenience typedefs for RFC 4253, RFC 5280, and RFC 5652.

A.2. 00 to 01

* Moved groupings from the draft-ietf-netconf-keystore here.

A.3. 01 to 02

* Removed unwanted "mandatory" and "must" statements.
* Added many new crypto algorithms (thanks Haiguang!)
* Clarified in asymmetric-key-pair-with-certs-grouping, in certificates/certificate/name/description, that if the name MUST NOT match the name of a certificate that exists independently in <operational>, enabling certs installed by the manufacturer (e.g., an IDevID).

A.4. 02 to 03

* renamed base identity 'asymmetric-key-encryption-algorithm' to 'asymmetric-key-algorithm'.
* added new 'asymmetric-key-algorithm' identities for secp192r1, secp224r1, secp256r1, secp384r1, and secp521r1.
* for all -cbc and -ctr identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-algorithm'.

* for all -ccm and -gcm identities, renamed base identity 'symmetric-key-encryption-algorithm' to 'encryption-and-mac-algorithm' and renamed the identity to remove the "enc-" prefix.

* for all the 'signature-algorithm' based identities, renamed from 'rsa-*' to 'rsassa-*'.

* removed all of the "x509v3-" prefixed 'signature-algorithm' based identities.

* added 'key-exchange-algorithm' based identities for 'rsaes-oaep' and 'rsaes-pkcs1-v1_5'.

* renamed typedef 'symmetric-key-encryption-algorithm-ref' to 'symmetric-key-algorithm-ref'.

* renamed typedef 'asymmetric-key-encryption-algorithm-ref' to 'asymmetric-key-algorithm-ref'.

* added typedef 'encryption-and-mac-algorithm-ref'.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

A.5. 03 to 04

* ran YANG module through formatter.

A.6. 04 to 05

* fixed broken symlink causing reformatted YANG module to not show.

A.7. 05 to 06

* Added NACM annotations.

* Updated Security Considerations section.
* Added 'asymmetric-key-pair-with-cert-grouping' grouping.

* Removed text from 'permanently-hidden' enum regarding such keys not being backed up or restored.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

* Added an explanation to the 'public-key-grouping' and 'asymmetric-key-pair-grouping' statements as for why the nodes are not mandatory (e.g., because they may exist only in <operational>.

* Added 'must' expressions to the 'public-key-grouping' and 'asymmetric-key-pair-grouping' statements ensuring sibling nodes are either all exist or do not all exist.

* Added an explanation to the 'permanently-hidden' that the value cannot be configured directly by clients and servers MUST fail any attempt to do so.

* Added 'trust-anchor-certs-grouping' and 'end-entity-certs-grouping' (the plural form of existing groupings).

* Now states that keys created in <operational> by the *-hidden-key actions are bound to the lifetime of the parent 'config true' node, and that subsequent invocations of either action results in a failure.

* Added clarifications that implementations SHOULD assert that configured certificates contain the matching public key.

* Replaced the 'generate-hidden-key' and 'install-hidden-key' actions with special 'crypt-hash' -like input/output values.

* Removed the 'generate-key and 'hidden-key' features.
* Added grouping symmetric-key-grouping

* Modified 'asymmetric-key-pair-grouping' to have a 'choice' statement for the keystone module to augment into, as well as replacing the 'union' with leafs (having different NACM settings.

A.10. 08 to 09

* Converting algorithm from identities to enumerations.

A.11. 09 to 10

* All of the below changes are to the algorithm enumerations defined in ietf-crypto-types.

* Add in support for key exchange over x.25519 and x.448 based on RFC 8418.

* Add in SHAKE-128, SHAKE-224, SHAKE-256, SHAKE-384 and SHAKE 512

* Revise/add in enum of signature algorithm for x25519 and x448

* Add in des3-cbc-sha1 for IPSec

* Add in sha1-des3-kd for IPSec

* Add in definit for rc4-hmac and rc4-hmac-exp. These two algorithms have been deprecated in RFC 8429. But some existing draft in i2nsf may still want to use them.

* Add x25519 and x448 curve for asymmetric algorithms

* Add signature algorithms ed25519, ed25519-cts, ed25519ph

* add signature algorithms ed448, ed448ph

* Add in rsa-sha2-256 and rsa-sha2-512 for SSH protocols (rfc8332)

A.12. 10 to 11

* Added a "key-format" identity.
* Added symmetric keys to the example in Section 2.2.

A.13. 11 to 12

* Removed all non-essential (to NC/RC) algorithm types.
* Moved remaining algorithm types each into its own module.
* Added a 'config false' "algorithms-supported" list to each of the algorithm-type modules.

A.14. 12 to 13

* Added the four features: "[encrypted-]one-[a]symmetric-key-format", each protecting a 'key-format' identity of the same name.
* Added 'must' expressions asserting that the 'key-format' leaf exists whenever a non-hidden key is specified.
* Improved the 'description' statements and added 'reference' statements for the 'key-format' identities.
* Added a questionable forward reference to "encrypted-*" leafs in a couple 'when' expressions.
* Did NOT move "config false" alg-supported lists to SSH/TLS drafts.

A.15. 13 to 14

* Resolved the "FIXME: forward ref" issue by modulating 'must', 'when', and 'mandatory' expressions.
* Moved the 'generatesymmetric-key' and 'generate-asymmetric-key' actions from ietf-keystore to ietf-crypto-types, now as RPCs.
* Cleaned up various description statements and removed lingering FIXMEs.
* Converted the "iana-<alg-type>-algs" YANG modules to IANA registries with instructions for how to generate modules from the registries, whenever they may be updated.
A.16. 14 to 15

* Removed the IANA-maintained registries for symmetric, asymmetric, and hash algorithms.

* Removed the "generate-symmetric-key" and "generate-asymmetric-key" RPCs.

* Removed the "algorithm" node in the various symmetric and asymmetric key groupings.

* Added 'typedef csr' and 'feature certificate-signing-request-generation'.

* Refined a usage of "end-entity-cert-grouping" to make the "cert" node mandatory true.

* Added a "Note to Reviewers" note to first page.

A.17. 15 to 16

* Updated draft title (refer to "Groupings" too).

* Removed 'end-entity-certs-grouping' as it wasn't being used anywhere.

* Removed 'trust-anchor-certs-grouping' as it was no longer being used after modifying 'local-or-truststore-certs-grouping' to use lists (not leaf-lists).

* Renamed "cert" to "cert-data" in trust-anchor-cert-grouping.

* Added "csr-info" typedef, to complement the existing "csr" typedef.

* Added "ocsp-request" and "ocsp-response" typedefs, to complement the existing "crl" typedef.

* Added "encrypted" cases to both symmetric-key-grouping and asymmetric-key-pair-grouping (Moved from Keystore draft).

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Updated the Security Considerations section.

A.18. 16 to 17
* [Re]-added a "Strength of Keys Configured" Security Consideration
* Prefixed "cleartext-" in the "key" and "private-key" node names.

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Balazs Kovacs, Eric Voit, Juergen Schoenwaelder, Liang Xia, Martin Bjorklund, Nick Hancock, Rich Salz, Rob Wilton, Tom Petch, and Wang Haiguang.

Author's Address

Kent Watsen
Watsen Networks

Email: kent+ietf@watsen.net