NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

YANG Data Types and Groupings for Cryptography
draft-ietf-netconf-crypto-types-34

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:

* 2024-03-16 --> the publication date of this draft

The "Relation to other RFCs" section Section 1.1 contains the text "one or more YANG modules" and, later, "modules". This text is sourced from a file in a context where it is unknown how many modules a draft defines. The text is not wrong as is, but it may be improved by stating more directly how many modules are defined.

The "Relation to other RFCs" section Section 1.1 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

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

* Appendix A. Change Log

Status of This Memo

This Internet-Draft will expire on 17 September 2024.

Copyright Notice

Table of Contents

1. Introduction

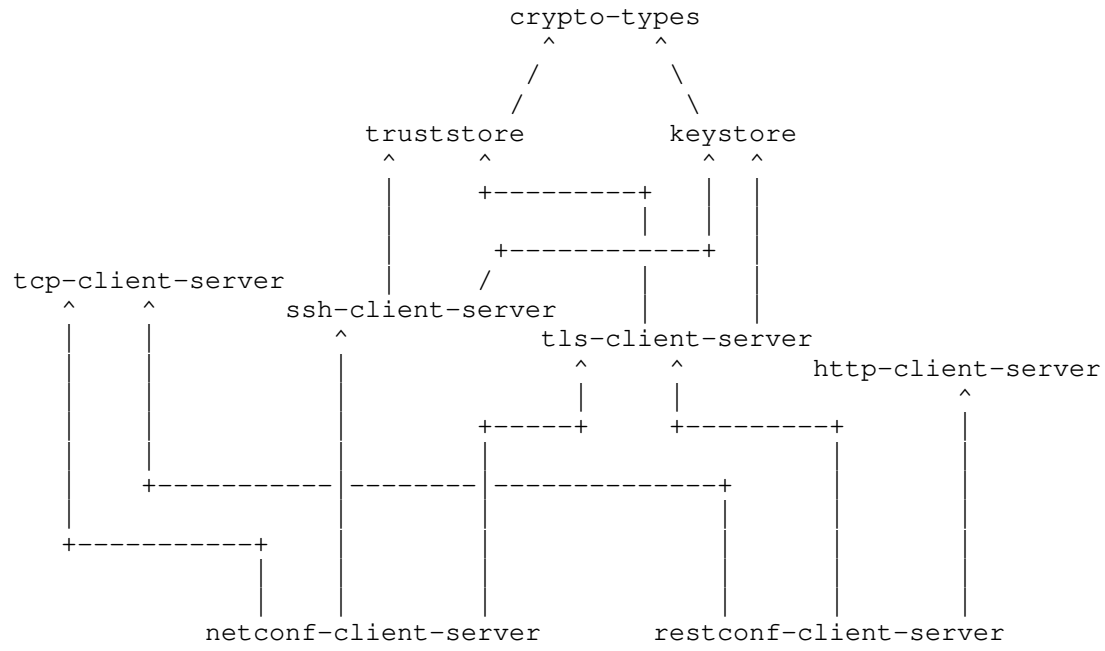
This document presents a YANG 1.1 [RFC7950] module defining identities, typedefs, and groupings useful to cryptographic applications.

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, ultimately, support the configuration of both the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The dependency relationship between the primary YANG groupings defined in the various RFCs is presented in the below diagram. In some cases, a draft may define secondary groupings that introduce dependencies not illustrated in the diagram. The labels in the diagram are a shorthand name for the defining RFC. The citation reference for shorthand name is provided below the diagram.

Please note that the arrows in the diagram point from referencer to referenced. For example, the "crypto-types" RFC does not have any dependencies, whilst the "keystore" RFC depends on the "crypto-types" RFC.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label in Diagram to RFC Mapping

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

1.4. Conventions

Various examples in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (per Section 9.8 of [RFC7950]). This placeholder value is used because real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-crypto-types" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-crypto-types". A high-level overview of the module is provided in Section 2.1. Examples illustrating the module's use are provided in Examples (Section 2.2). The YANG module itself is defined in Section 2.3.

2.1. Data Model Overview

This section provides an overview of the "ietf-crypto-types" module in terms of its features, identities, typedefs, and groupings.

2.1.1. Features

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

```
   Features:
     +-- one-symmetric-key-format
     +-- one-asymmetric-key-format
     +-- symmetrically-encrypted-value-format
     +-- asymmetrically-encrypted-value-format
     +-- cms-enveloped-data-format
     +-- cms-encrypted-data-format
     +-- p10-csr-format
     +-- csr-generation
     +-- certificate-expiration-notification
     +-- cleartext-passwords
     +-- encrypted-passwords
     +-- cleartext-symmetric-keys
     +-- hidden-symmetric-keys
     +-- encrypted-symmetric-keys
     +-- cleartext-private-keys
     +-- hidden-private-keys
     +-- encrypted-private-keys
```

The diagram above uses syntax that is similar to but not the same as that in [RFC8340].

2.1.2. Identities

The following diagram illustrates the hierarchical 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}?
  +-- symmetric-key-format
  |  +-- octet-string-key-format
  |  +-- one-symmetric-key-format
  |             {one-symmetric-key-format}?
  +-- encrypted-value-format
  |  +-- symmetrically-encrypted-value-format
  |  |  |       {symmetrically-encrypted-value-format}?
  |  |  +-- cms-encrypted-data-format
  |  |             {cms-encrypted-data-format}?
  |  +-- asymmetrically-encrypted-value-format
  |     |       {asymmetrically-encrypted-value-format}?
  |     +-- cms-enveloped-data-format
  |             {cms-enveloped-data-format}?
  +-- csr-format
        +-- p10-csr-format {p10-csr-format?}
```

The diagram above uses syntax that is similar to but not the same as that in [RFC8340].

Comments:

* The diagram shows that there are five base identities. The first three identities are used to indicate the format for the key data, while the fourth identity is used to indicate the format for encrypted values. The fifth identity is used to indicate the format for a certificate signing request. The base identities are "abstract", in the object oriented programming sense, in that they only define a "class" of formats, rather than a specific format.

* The various terminal identities define specific 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 used to specify uncommon formats are enabled by "feature" statements, allowing 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:

```
Typedefs:
  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
```

The diagram above uses syntax that is similar to but not the same as that in [RFC8340].

Comments:

* All 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.X680.2021] structure using DER [ITU.X690.2021].

* 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 "ietf-crypto-types" module defines the following "grouping" statements:

* encrypted-value-grouping
* password-grouping

* symmetric-key-grouping
* public-key-grouping
* private-key-grouping
* asymmetric-key-pair-grouping
* certificate-expiration-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-value-grouping" Grouping

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

```
  grouping encrypted-value-grouping:
    +-- encrypted-by
    +-- encrypted-value-format    identityref
    +-- 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 value, encrypted by the key referenced by the "encrypted-by" node, and encoded in the format appropriate for the kind of key it was encrypted by.

  - If the value is encrypted by a symmetric key, then the encrypted value is encoded using the format associated with the "symmetrically-encrypted-value-format" identity.

  - If the value is encrypted by an asymmetric key, then the encrypted value is encoded using the format associated with the "asymmetrically-encrypted-value-format" identity.

  See Section 2.1.2 for information about the "format" identities.

2.1.4.2. The "password-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "password-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping password-grouping:
    +-- (password-type)
       +--:(cleartext-password) {cleartext-passwords}?
       |  +-- cleartext-password?   string
       +--:(encrypted-password) {encrypted-passwords}?
          +-- encrypted-password
             +---u encrypted-value-grouping
```

Comments:

* The "password-grouping" enables configuration of credentials needed to authenticate to a remote system. The 'ianach:crypt-hash' typedef from [RFC7317] should be used instead when needing to configure a password to authencate a local account.

* For the referenced grouping statement(s):

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

* The "choice" statement enables the password data to be cleartext or encrypted, as follows:

  - The "cleartext-password" node can encode any cleartext value.
  - The "encrypted-password" node's structure is discussed in Section 2.1.4.1.

2.1.4.3. The "symmetric-key-grouping" Grouping

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

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

Comments:

* For the referenced grouping statement(s):

  - The "encrypted-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 any of the formats defined by the derived identities.

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

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

2.1.4.4. The "public-key-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "public-key-grouping" grouping. This tree diagram does not expand any internally used grouping statement(s):

```
  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 any of the formats defined by 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.

2.1.4.5. The "private-key-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "private-key-grouping" grouping. This tree diagram does not expand the internally used grouping statement(s):

```
  grouping private-key-grouping:
    +-- private-key-format?            identityref
    +-- (private-key-type)
       +--:(cleartext-private-key) {cleartext-private-keys}?
       |  +-- cleartext-private-key?   binary
       +--:(hidden-private-key) {hidden-private-keys}?
       |  +-- hidden-private-key?      empty
       +--:(encrypted-private-key) {encrypted-private-keys}?
          +-- encrypted-private-key
             +---u encrypted-value-grouping
```

Comments:

* For the referenced grouping statement(s):

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

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

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

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

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

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

```
  grouping asymmetric-key-pair-grouping:
    +---u public-key-grouping
    +---u private-key-grouping
```

Comments:

* For the referenced grouping statement(s):

  - The "public-key-grouping" grouping is discussed in Section 2.1.4.4.
  - The "private-key-grouping" grouping is discussed in Section 2.1.4.5.

2.1.4.7. The "certificate-expiration-grouping" Grouping

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

```
  grouping certificate-expiration-grouping:
    +---n certificate-expiration
            {certificate-expiration-notification}?
       +-- 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 2.1.4.8 and Section 2.1.4.9.

* 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.

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

This section presents a tree diagram [RFC8340] illustrating the "trust-anchor-cert-grouping" grouping. This 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
```

Comments:

* For the referenced grouping statement(s):

  - The "certificate-expiration-grouping" grouping is discussed in Section 2.1.4.7.

* The "cert-data" node contains a chain of one or more certificates containing at most one self-signed certificates (the "root" certificate), encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

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

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

```
  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 2.1.4.7.

* The "cert-data" node contains a chain of one or more certificates containing at most one certificate that is neither self-signed nor having Basic constraint "CA true", encoded using a "signed-data-cms" typedef discussed in Section 2.1.3.

2.1.4.10. The "generate-csr-grouping" Grouping

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

```
  grouping generate-csr-grouping:
    +---x generate-csr {csr-generation}?
       +---w input
       |  +---w csr-format    identityref
       |  +---w csr-info      csr-info
       +--ro output
          +--ro (csr-type)
             +--:(p10-csr)
                +--ro p10-csr?   p10-csr
```

Comments:

* This grouping's only purpose is to define the "generate-certificate-signing-request" action statement, used by the groupings defined in Section 2.1.4.11 and Section 2.1.4.12.

* This action takes as input a "csr-info" type and returns a "csr" type, both of which are discussed in Section 2.1.3.

* For an example, please see Section 2.2.2.

2.1.4.11. The "asymmetric-key-pair-with-cert-grouping" Grouping

This section presents a tree diagram [RFC8340] illustrating the "asymmetric-key-pair-with-cert-grouping" grouping. This 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
```

Comments:

* 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 2.1.4.6.
  - The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.9.
  - The "generate-csr-grouping" grouping is discussed in Section 2.1.4.10.

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

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

```
  grouping asymmetric-key-pair-with-certs-grouping:
    +---u asymmetric-key-pair-grouping
    +-- certificates
    |  +-- certificate* [name]
    |     +-- name?                        string
    |     +---u end-entity-cert-grouping
    +---u generate-csr-grouping
```

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 2.1.4.6.
  - The "end-entity-cert-grouping" grouping is discussed in Section 2.1.4.9.
  - The "generate-csr-grouping" grouping is discussed in Section 2.1.4.10.

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", "asymmetric-key-pair-with-certs-grouping", and "password-grouping" Groupings

The following non-normative module is constructed in order to illustrate the use of the "symmetric-key-grouping" (Section 2.1.4.3), the "asymmetric-key-pair-with-certs-grouping" (Section 2.1.4.12), and the "password-grouping" (Section 2.1.4.2) grouping statements.

Notably, this example module and associated configuration data illustrates that a hidden private key (ex-hidden-asymmetric-key) has been used to encrypt a symmetric key (ex-encrypted-one-symmetric-based-symmetric-key) that has been used to encrypt another private key (ex-encrypted-rsa-based-asymmetric-key). Additionally, the symmetric key is also used to encrypt a password (ex-encrypted-password).

2.2.1.1. Example Module

```
module ex-crypto-types-usage {
  yang-version 1.1;
  namespace "https://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 example module illustrates the 'symmetric-key-grouping'
     and 'asymmetric-key-grouping' groupings defined in the
    'ietf-crypto-types' module defined in RFC AAAA.";

  revision 2024-03-16 {
    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-symmetric-key/"
                  + "encrypted-symmetric-key/encrypted-by" {
              description
                "Augments in a choice statement enabling the
                 encrypting key to be any other symmetric or
                 asymmetric key.";
              uses encrypted-by-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-grouping;
            }
          }
          description
            "An asymmetric key pair with associated certificates.";
        }
      }
      container passwords {
        description
          "A container of passwords.";
        list password {
          key "name";
          leaf name {
            type string;
            description
              "An arbitrary name for this password.";
          }
```

```
        uses ct:password-grouping {
          augment "password-type/encrypted-password/"
                + "encrypted-password/encrypted-by" {
            description
              "Augments in a choice statement enabling the
               encrypting key to be any symmetric or
               asymmetric key.";
            uses encrypted-by-grouping;
          }
        }
        description
          "A password.";
      }
    }

    grouping encrypted-by-grouping {
      description
        "A grouping that defines a choice enabling references
         to other keys.";
      choice encrypted-by {
        mandatory true;
        description
          "A choice amongst other symmetric or asymmetric keys.";
        case symmetric-key-ref {
          leaf symmetric-key-ref {
            type leafref {
              path "/ectu:symmetric-keys/ectu:symmetric-key/"
                 + "ectu:name";
            }
            description
              "Identifies the symmetric key that encrypts this key.";
          }
        }
        case asymmetric-key-ref {
          leaf asymmetric-key-ref {
            type leafref {
              path "/ectu:asymmetric-keys/ectu:asymmetric-key/"
                 + "ectu:name";
            }
            description
              "Identifies the asymmetric key that encrypts this key.";
          }
        }
      }
    }
  }
```

2.2.1.2. Tree Diagram for the Example Module

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-symmetric-key)
  |        |  +--rw cleartext-symmetric-key?   binary
  |        |          {cleartext-symmetric-keys}?
  |        +--:(hidden-symmetric-key) {hidden-symmetric-keys}?
  |        |  +--rw hidden-symmetric-key?      empty
  |        +--:(encrypted-symmetric-key) {encrypted-symmetric-keys}?
  |           +--rw encrypted-symmetric-key
  |              +--rw encrypted-by
  |              |  +--rw (encrypted-by)
  |              |     +--:(symmetric-key-ref)
  |              |     |  +--rw symmetric-key-ref?    leafref
  |              |     +--:(asymmetric-key-ref)
  |              |        +--rw asymmetric-key-ref?   leafref
  |              +--rw encrypted-value-format    identityref
  |              +--rw encrypted-value           binary
  +--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) {cleartext-private-keys}?
  |     |  |  +--rw cleartext-private-key?   binary
  |     |  +--:(hidden-private-key) {hidden-private-keys}?
  |     |  |  +--rw hidden-private-key?      empty
  |     |  +--:(encrypted-private-key) {encrypted-private-keys}?
  |     |     +--rw encrypted-private-key
  |     |        +--rw encrypted-by
  |     |        |  +--rw (encrypted-by)
  |     |        |     +--:(symmetric-key-ref)
  |     |        |     |  +--rw symmetric-key-ref?    leafref
  |     |        |     +--:(asymmetric-key-ref)
  |     |        |        +--rw asymmetric-key-ref?   leafref
  |     |        +--rw encrypted-value-format    identityref
  |     |        +--rw encrypted-value           binary
  |     +--rw certificates
  |     |  +--rw certificate* [name]
```

```
      |      |     +--rw name                       string
      |      |     +--rw cert-data                  end-entity-cert-cms
      |      |     +---n certificate-expiration
      |      |             {certificate-expiration-notification}?
      |      |        +-- expiration-date    yang:date-and-time
      |      +---x generate-csr {csr-generation}?
      |         +---w input
      |         |  +---w csr-format    identityref
      |         |  +---w csr-info      csr-info
      |         +--ro output
      |            +--ro (csr-type)
      |               +--:(p10-csr)
      |                  +--ro p10-csr?   p10-csr
      +--rw passwords
         +--rw password* [name]
            +--rw name                          string
            +--rw (password-type)
               +--:(cleartext-password) {cleartext-passwords}?
               |  +--rw cleartext-password?   string
               +--:(encrypted-password) {encrypted-passwords}?
                  +--rw encrypted-password
                     +--rw encrypted-by
                     |  +--rw (encrypted-by)
                     |     +--:(symmetric-key-ref)
                     |     |  +--rw symmetric-key-ref?    leafref
                     |     +--:(asymmetric-key-ref)
                     |        +--rw asymmetric-key-ref?   leafref
                     +--rw encrypted-value-format    identityref
                     +--rw encrypted-value           binary
```

2.2.1.3. Usage Example for the Example Module

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

```
================ NOTE: '\' line wrapping per RFC 8792 ================

<symmetric-keys
  xmlns="https://example.com/ns/example-crypto-types-usage"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <symmetric-key>
    <name>ex-hidden-symmetric-key</name>
    <hidden-symmetric-key/>
  </symmetric-key>
  <symmetric-key>
    <name>ex-octet-string-based-symmetric-key</name>
    <key-format>ct:octet-string-key-format</key-format>
    <cleartext-symmetric-key>BASE64VALUE=</cleartext-symmetric-key>
```

```
    </symmetric-key>
    <symmetric-key>
      <name>ex-one-symmetric-based-symmetric-key</name>
      <key-format>ct:one-symmetric-key-format</key-format>
      <cleartext-symmetric-key>BASE64VALUE=</cleartext-symmetric-key>
    </symmetric-key>
    <symmetric-key>
      <name>ex-encrypted-one-symmetric-based-symmetric-key</name>
      <key-format>ct:one-symmetric-key-format</key-format>
      <encrypted-symmetric-key>
        <encrypted-by>
          <asymmetric-key-ref>ex-hidden-asymmetric-key</asymmetric-key\
-ref>
        </encrypted-by>
        <encrypted-value-format>ct:cms-enveloped-data-format</encrypte\
d-value-format>
        <encrypted-value>BASE64VALUE=</encrypted-value>
      </encrypted-symmetric-key>
    </symmetric-key>
  </symmetric-keys>

  <asymmetric-keys
    xmlns="https://example.com/ns/example-crypto-types-usage"
    xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <asymmetric-key>
      <name>ex-hidden-asymmetric-key</name>
      <public-key-format>ct:subject-public-key-info-format</public-key\
-format>
      <public-key>BASE64VALUE=</public-key>
      <hidden-private-key/>
      <certificates>
        <certificate>
          <name>ex-hidden-asymmetric-key-cert</name>
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-rsa-based-asymmetric-key</name>
      <public-key-format>ct:subject-public-key-info-format</public-key\
-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>ct:rsa-private-key-format</private-key-forma\
t>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
      <certificates>
        <certificate>
          <name>ex-cert</name>
```

```
          <cert-data>BASE64VALUE=</cert-data>
        </certificate>
      </certificates>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-one-asymmetric-based-asymmetric-key</name>
      <public-key-format>ct:subject-public-key-info-format</public-key\
-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>ct:one-asymmetric-key-format</private-key-fo\
rmat>
      <cleartext-private-key>BASE64VALUE=</cleartext-private-key>
    </asymmetric-key>
    <asymmetric-key>
      <name>ex-encrypted-rsa-based-asymmetric-key</name>
      <public-key-format>ct:subject-public-key-info-format</public-key\
-format>
      <public-key>BASE64VALUE=</public-key>
      <private-key-format>ct:rsa-private-key-format</private-key-forma\
t>
      <encrypted-private-key>
        <encrypted-by>
          <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetri\
c-key</symmetric-key-ref>
        </encrypted-by>
        <encrypted-value-format>ct:cms-encrypted-data-format</encrypte\
d-value-format>
        <encrypted-value>BASE64VALUE=</encrypted-value>
      </encrypted-private-key>
    </asymmetric-key>
  </asymmetric-keys>

  <passwords
    xmlns="https://example.com/ns/example-crypto-types-usage"
    xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
    <password>
      <name>ex-cleartext-password</name>
      <cleartext-password>super-secret</cleartext-password>
    </password>
    <password>
      <name>ex-encrypted-password</name>
      <encrypted-password>
        <encrypted-by>
          <symmetric-key-ref>ex-encrypted-one-symmetric-based-symmetri\
c-key</symmetric-key-ref>
        </encrypted-by>
        <encrypted-value-format>ct:cms-encrypted-data-format</encrypte\
d-value-format>
```

```
        <encrypted-value>BASE64VALUE=</encrypted-value>
      </encrypted-password>
    </password>
  </passwords>
```

2.2.2. The "generate-certificate-signing-request" Action

The following example illustrates the "generate-certificate-signing-request" action, discussed in Section 2.1.4.10, with the NETCONF protocol.

REQUEST

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <asymmetric-keys
      xmlns="https://example.com/ns/example-crypto-types-usage">
      <asymmetric-key>
        <name>ex-hidden-asymmetric-key</name>
        <generate-csr>
          <csr-format>ct:p10-csr-format</csr-format>
          <csr-info>BASE64VALUE=</csr-info>
        </generate-csr>
      </asymmetric-key>
    </asymmetric-keys>
  </action>
</rpc>
```

RESPONSE

```
============== NOTE: '\' line wrapping per RFC 8792 ================

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <p10-csr xmlns="https://example.com/ns/example-crypto-types-usage"\
>BASE64VALUE=</p10-csr>
</rpc-reply>
```

2.2.3. The "certificate-expiration" Notification

The following example illustrates the "certificate-expiration" notification, discussed in Section 2.1.4.7, with the NETCONF protocol.

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   <notification
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <eventTime>2018-05-25T00:01:00Z</eventTime>
     <asymmetric-keys xmlns="https://example.com/ns/example-crypto-type\
   s-usage">
       <asymmetric-key>
         <name>ex-hidden-asymmetric-key</name>
         <certificates>
           <certificate>
             <name>ex-hidden-asymmetric-key-cert</name>
             <certificate-expiration>
               <expiration-date>2018-08-05T14:18:53-05:00</expiration-d\
   ate>
             </certificate-expiration>
           </certificate>
         </certificates>
       </asymmetric-key>
     </asymmetric-keys>
   </notification>
```

2.3. YANG Module

This module has normative references to [RFC2119], [RFC2986], [RFC4253], [RFC5280], [RFC5652], [RFC5915], [RFC5958], [RFC6031], [RFC6960], [RFC6991], [RFC7093], [RFC8017], [RFC8174], [RFC8341], and [ITU.X690.2021].

```
   <CODE BEGINS> file "ietf-crypto-types@2024-03-16.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";
     }
```

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

  contact
    "WG Web:   https://datatracker.ietf.org/wg/netconf
     WG List:  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) 2024 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 Revised
     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 2024-03-16 {
    description
      "Initial version";
    reference
      "RFC AAAA: YANG Data Types and Groupings for Cryptography";
  }

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

  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 symmetrically-encrypted-value-format {
    description
      "Indicates that the server supports the
       'symmetrically-encrypted-value-format' identity.";
  }

  feature asymmetrically-encrypted-value-format {
    description
      "Indicates that the server supports the
       'asymmetrically-encrypted-value-format' identity.";
  }

  feature cms-enveloped-data-format {
    description
      "Indicates that the server supports the
       'cms-enveloped-data-format' identity.";
  }

  feature cms-encrypted-data-format {
    description
      "Indicates that the server supports the
       'cms-encrypted-data-format' identity.";
  }

  feature p10-csr-format {
    description
      "Indicates that the server implements support
       for generating P10-based CSRs, as defined
       in RFC 2986.";
    reference
      "RFC 2986: PKCS #10: Certification Request Syntax
                 Specification Version 1.7";
  }

  feature csr-generation {
    description
      "Indicates that the server implements the
       'generate-csr' action.";
  }
```

```
    feature certificate-expiration-notification {
      description
        "Indicates that the server implements the
         'certificate-expiration' notification.";
    }

    feature cleartext-passwords {
      description
        "Indicates that the server supports cleartext
         passwords.";
    }

    feature encrypted-passwords {
      description
        "Indicates that the server supports password
         encryption.";
    }

    feature cleartext-symmetric-keys {
      description
        "Indicates that the server supports cleartext
         symmetric keys.";
    }

    feature hidden-symmetric-keys {
      description
        "Indicates that the server supports hidden keys.";
    }

    feature encrypted-symmetric-keys {
      description
        "Indicates that the server supports encryption
         of symmetric keys.";
    }

    feature cleartext-private-keys {
      description
        "Indicates that the server supports cleartext
         private keys.";
    }

    feature hidden-private-keys {
      description
        "Indicates that the server supports hidden keys.";
    }

    feature encrypted-private-keys {
      description
```

```
        "Indicates that the server supports encryption
         of private keys.";
    }

    /************************************************/
    /*   Base Identities for Key Format Structures   */
    /************************************************/

    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.";
    }

    /*****************************************************/
    /*   Identities for Private Key Format Structures   */
    /*****************************************************/

    identity rsa-private-key-format {
      base private-key-format;
      description
        "Indicates that the private key value is encoded as
         an RSAPrivateKey (from RFC 8017), encoded using ASN.1
         distinguished encoding rules (DER), as specified in
         ITU-T X.690.";
      reference
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2
         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) 02/2021.";
    }

    identity ec-private-key-format {
      base private-key-format;
      description
        "Indicates that the private key value is encoded as
```

```
          an ECPrivateKey (from RFC 5915), encoded using ASN.1
          distinguished encoding rules (DER), as specified in
          ITU-T X.690.";
        reference
          "RFC 5915:
            Elliptic Curve Private Key Structure
          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) 02/2021.";
      }

      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
          "RFC 5958: Asymmetric Key Packages
           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) 02/2021.";
      }

      /*************************************************/
      /*   Identities for Public Key Format Structures   */
      /*************************************************/

      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.";
        reference
          "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
      }

      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) 02/2021.";
    }

    /*****************************************************/
    /*   Identities for Symmetric Key Format Structures   */
    /*****************************************************/

    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.

          The identity of the associated algorithm is outside the
          scope of this specification.  This is also true when
          the octet string has been encrypted.";
    }

    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) 02/2021.";
    }

    /************************************************/
    /*   Identities for Encrypted Value Structures   */
    /************************************************/

    identity encrypted-value-format {
      description
        "Base format identity for encrypted values.";
    }

    identity symmetrically-encrypted-value-format {
      if-feature "symmetrically-encrypted-value-format";
      base encrypted-value-format;
      description
        "Base format identity for symmetrically encrypted
         values.";
    }

    identity asymmetrically-encrypted-value-format {
      if-feature "asymmetrically-encrypted-value-format";
      base encrypted-value-format;
      description
        "Base format identity for asymmetrically encrypted
         values.";
    }

    identity cms-encrypted-data-format {
      if-feature "cms-encrypted-data-format";
      base symmetrically-encrypted-value-format;
      description
        "Indicates that the encrypted value conforms to
         the 'encrypted-data-cms' type with the constraint
         that the 'unprotectedAttrs' value is not set.";
      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) 02/2021.";
    }

    identity cms-enveloped-data-format {
      if-feature "cms-enveloped-data-format";
      base asymmetrically-encrypted-value-format;
      description
```

```
        "Indicates that the encrypted value conforms to the
         'enveloped-data-cms' type with the following constraints:

         The EnvelopedData structure MUST have exactly one
         'RecipientInfo'.

         If the asymmetric key supports public key cryptography
         (e.g., RSA), then the 'RecipientInfo' must be a
         'KeyTransRecipientInfo' with the 'RecipientIdentifier'
         using a 'subjectKeyIdentifier' with the value set using
         'method 1' in RFC 7093 over the recipient's public key.

         Otherwise, if the asymmetric key supports key agreement
         (e.g., ECC), then the 'RecipientInfo' must be a
         'KeyAgreeRecipientInfo'.  The 'OriginatorIdentifierOrKey'
         value must use the 'OriginatorPublicKey' alternative.
         The 'UserKeyingMaterial' value must not be present.
         There must be exactly one 'RecipientEncryptedKeys' value
         having the 'KeyAgreeRecipientIdentifier' set to 'rKeyId'
         with the value set using 'method 1' in RFC 7093 over the
         recipient's public key.";
      reference
        "RFC 5652: Cryptographic Message Syntax (CMS)
         RFC 7093:
           Additional Methods for Generating Key
           Identifiers Values
         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) 02/2021.";
    }

    /*****************************************************/
    /*   Identities for Certificate Signing Request Formats  */
    /*****************************************************/

    identity csr-format {
      description
        "A base identity for the certificate signing request
         formats.  Additional derived identities MAY be defined
         by future efforts.";
    }

    identity p10-csr-format {
      if-feature "p10-csr-format";
      base csr-format;
      description
```

```
        "Indicates the 'CertificationRequest' structure
         defined in RFC 2986.";
      reference
        "RFC 2986: PKCS #10: Certification Request Syntax
                   Specification Version 1.7";
    }


    /***************************************************/
    /*   Typedefs for ASN.1 structures from RFC 2986   */
    /***************************************************/

    typedef csr-info {
      type binary;
      description
        "A CertificationRequestInfo structure, as defined 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) 02/2021.";
    }

    typedef p10-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) 02/2021.";
    }

    /***************************************************/
    /*   Typedefs for ASN.1 structures from RFC 5280   */
    /***************************************************/
```

```
     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:
            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) 02/2021.";
     }

     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) 02/2021.";
     }

     /*************************************************/
     /*   Typedefs for ASN.1 structures from RFC 6960   */
     /*************************************************/

     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) 02/2021.";
      }

      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) 02/2021.";
      }

      /*********************************************/
      /*   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) 02/2021.";
      }

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

      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)";
    }

    /*****************************************************/
    /*   Typedefs for ASN.1 structures related to RFC 5280   */
    /*****************************************************/

    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.";
    }

    /*****************************************************/
    /*   Typedefs for ASN.1 structures related to RFC 5652   */
    /*****************************************************/

    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 the 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 (RFC 5652, Section 5.2) 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.
           RFC 5652:
             Cryptographic Message Syntax (CMS)";
      }

      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.

           This CMS encodes the degenerate form of the SignedData
           structure (RFC 5652, Section 5.2) 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.
           RFC 5652:
             Cryptographic Message Syntax (CMS)";
      }

      /*****************/
      /*   Groupings   */
      /*****************/
```

```
  grouping encrypted-value-grouping {
    description
      "A reusable grouping for a value that has been encrypted by
       a referenced symmetric or asymmetric key.";
    container encrypted-by {
      nacm:default-deny-write;
      description
        "An empty container enabling a reference to the key that
         encrypted the value to be augmented in.  The referenced
         key MUST be a symmetric key or an asymmetric key.

         A symmetric key MUST be referenced via a leaf node called
         'symmetric-key-ref'.  An asymmetric key MUST be referenced
         via a leaf node called 'asymmetric-key-ref'.

         The leaf nodes MUST be direct descendants in the data tree,
         and MAY be direct descendants in the schema tree (e.g.,
         choice/case statements are allowed, but not a container).";
    }
    leaf encrypted-value-format {
      type identityref {
        base encrypted-value-format;
      }
      mandatory true;
      description
        "Identifies the format of the 'encrypted-value' leaf.

         If 'encrypted-by' points to a symmetric key, then a
         'symmetrically-encrypted-value-format' based identity
         MUST be set (e.g., cms-encrypted-data-format).

         If 'encrypted-by' points to an asymmetric key, then an
         'asymmetrically-encrypted-value-format' based identity
         MUST be set (e.g., cms-enveloped-data-format).";
    }
    leaf encrypted-value {
      nacm:default-deny-write;
      type binary;
      must '../encrypted-by';
      mandatory true;
      description
        "The value, encrypted using the referenced symmetric
         or asymmetric key.  The value MUST be encoded using
         the format associated with the 'encrypted-value-format'
         leaf.";
    }
  }
```

```
     grouping password-grouping {
       description
         "A password used for authenticating to a remote system.

          The 'ianach:crypt-hash' typedef from RFC 7317 should be
          used instead when needing a password to authencate a
          local account.";
       choice password-type {
         nacm:default-deny-write;
         mandatory true;
         description
           "Choice between password types.";
         case cleartext-password {
           if-feature "cleartext-passwords";
           leaf cleartext-password {
             nacm:default-deny-all;
             type string;
             description
               "The cleartext value of the password.";
           }
         }
         case encrypted-password {
           if-feature "encrypted-passwords";
           container encrypted-password {
             description
               "A container for the encrypted password value.";
             uses encrypted-value-grouping;
           }
         }
       }
     }

     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.  Implementations
            SHOULD ensure that the incoming symmetric key value is
            encoded in the specified format.

            For encrypted keys, the value is the decrypted key's
            format (i.e., the 'encrypted-value-format' conveys the
            encrypted key's format.";
```

```
      }
      choice key-type {
        nacm:default-deny-write;
        mandatory true;
        description
          "Choice between key types.";
        case cleartext-symmetric-key {
          leaf cleartext-symmetric-key {
            if-feature "cleartext-symmetric-keys";
            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-symmetric-key {
          if-feature "hidden-symmetric-keys";
          leaf hidden-symmetric-key {
            type empty;
            must 'not(../key-format)';
            description
              "A hidden key is not exportable, and not extractable,
               and therefore, it is of type 'empty' as its value is
               inaccessible via management interfaces. Though hidden
               to users, such keys are not hidden to the server and
               may be referenced by configuration to indicate which
               key a server should use for a cryptographic operation.
               How such keys are created is outside the scope of this
               module.";
          }
        }
        case encrypted-symmetric-key {
          if-feature "encrypted-symmetric-keys";
          container encrypted-symmetric-key {
            must '../key-format';
            description
              "A container for the encrypted symmetric key value.
               The interpretation of the 'encrypted-value' node
               is via the 'key-format' node";
            uses encrypted-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 public key's format. Implementations SHOULD
           ensure that the incoming public key value is encoded in the
           specified 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 private-key-grouping {
      description
        "A private key.";
      leaf private-key-format {
        nacm:default-deny-write;
        type identityref {
          base private-key-format;
        }
        description
          "Identifies the private key's format.  Implementations SHOULD
           ensure that the incoming private key value is encoded in the
           specified format.

           For encrypted keys, the value is the decrypted key's
           format (i.e., the 'encrypted-value-format' conveys the
           encrypted key's format.";
      }
      choice private-key-type {
        nacm:default-deny-write;
        mandatory true;
        description
          "Choice between key types.";
        case cleartext-private-key {
          if-feature "cleartext-private-keys";
          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 {
          if-feature "hidden-private-keys";
          leaf hidden-private-key {
            type empty;
            must 'not(../private-key-format)';
            description
              "A hidden key.  It is of type 'empty' as its value is
               inaccessible via management interfaces.  Though hidden
               to users, such keys are not hidden to the server and
               and may be referenced by configuration to indicate which
               key a server should use for a cryptographic operation.
               How such keys are created is outside the scope of this
               module.";
          }
        }
        case encrypted-private-key {
          if-feature "encrypted-private-keys";
          container encrypted-private-key {
            must '../private-key-format';
            description
              "A container for the encrypted asymmetric private key
               value.  The interpretation of the 'encrypted-value'
               node is via the 'private-key-format' node";
            uses encrypted-value-grouping;
          }
        }
      }
    }

    grouping asymmetric-key-pair-grouping {
      description
        "A private key and, optionally, its associated public key.
         Implementations MUST ensure that the two keys, when both
         are specified, are a matching pair.";
      uses public-key-grouping {
        refine public-key-format {
          mandatory false;
        }
        refine public-key {
          mandatory false;
```

```
        }
      }
      uses private-key-grouping;
    }

    grouping certificate-expiration-grouping {
      description
        "A notification for when a certificate is about to, or
         already has, expired.";
      notification certificate-expiration {
        if-feature "certificate-expiration-notification";
        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.

           If the certificate's Issuer maintains a Certificate
           Revocation List (CRL), the expiration notification MAY
           be sent if the CRL is about to expire.";
        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-all;
        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-all;
          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-csr' action.";
        action generate-csr {
          if-feature "csr-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'.";
          input {
            leaf csr-format {
              type identityref {
                base csr-format;
              }
              mandatory true;
              description
                "Specifies the format for the returned certificate.";
            }
            leaf csr-info {
              type 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 {
          choice csr-type {
            mandatory true;
            description
              "A choice amongst certificate signing request formats.
               Additional formats MAY be augmented into this 'choice'
               statement by future efforts.";
            case p10-csr {
              leaf p10-csr {
                type p10-csr;
                description
                  "A CertificationRequest, as defined in RFC 2986.";
              }
              description
                "A CertificationRequest, as defined in RFC 2986.";
              reference
                "RFC 2986:
                   PKCS #10: Certification Request Syntax Specification
                 RFC AAAA:
                   YANG Data Types and Groupings for Cryptography";
            }
          }
        }
      }
    } // generate-csr-grouping

    grouping asymmetric-key-pair-with-cert-grouping {
      description
        "A private/public key pair and an associated certificate.
         Implementations MUST assert that the certificate contains
         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 a list of associated
         certificates.  Implementations MUST 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.";
        list certificate {
          key "name";
          description
            "A certificate for this asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the certificate.";
          }
          uses end-entity-cert-grouping {
            refine "cert-data" {
              mandatory true;
            }
          }
        }
      }
      uses generate-csr-grouping;
    } // asymmetric-key-pair-with-certs-grouping

  }

  <CODE ENDS>
```

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 generically identify the key-algorithm to use. Hence, the solution presented in this document only supports keys to be configured via an external client.

Separate protocol-specific modules can present protocol-specific key-generating RPCs (e.g., the "generate-public-key" RPC in [I-D.ietf-netconf-ssh-client-server] and [I-D.ietf-netconf-tls-client-server]).

3.3. Unconstrained Public Key Usage

This module defines the "public-key-grouping" grouping, which enables the configuration of public keys without constraints on their usage, e.g., what operations the key is allowed to be used for (encryption, verification, both).

The "asymmetric-key-pair-grouping" grouping uses the aforementioned "public-key-grouping" grouping, and carries the same traits.

The "asymmetric-key-pair-with-cert-grouping" grouping uses the aforementioned "asymmetric-key-pair-grouping" grouping, whereby associated certificates MUST constrain the usage of the public key according to local policy.

3.4. Unconstrained Private Key Usage

This module defines the "asymmetric-key-pair-grouping" grouping, which enables the configuration of private keys without constraints on their usage, e.g., what operations the key is allowed to be used for (e.g., signature, decryption, both).

The "asymmetric-key-pair-with-cert-grouping" uses the aforementioned "asymmetric-key-pair-grouping" grouping, whereby configured certificates (e.g., identity certificates) may constrain the use of the public key according to local policy.

3.5. Cleartext Passwords and Keys

The module contained within this document enables, only when specific "feature" statements are enabled, for the cleartext value of passwords and keys to be stored in the configuration database. Storing cleartext values for passwords and keys is NOT RECOMMENDED.

3.6. Encrypting Passwords and Keys

The module contained within this document enables cleartext passwords and keys to be encrypted via another key, either symmetric or asymmetric. Both formats use a CMS structure (EncryptedData and EnvelopedData respectively), which allows any encryption algorithm to be used.

To securely encrypt a password or key with a symmetric key, a proper block cipher mode such as an AEAD or CBC MUST be used. This ensures that a random IV is part of the input, which guarantees that the output for encrypting the same password or key still produces a different unpredictable ciphertext. This avoids leaking that some encrypted keys or passwords are the same and makes it much harder to pre-generate rainbow tables to brute force attack weak passwords. The ECB block cipher mode MUST NOT be used.

3.7. 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 values are the "cleartext-symmetric-key" node defined in the "symmetric-key-grouping" grouping (Section 2.1.4.3) and the "cleartext-private-key" node defined in the "asymmetric-key-pair-grouping" grouping ("Section 2.1.4.6).

3.8. Considerations for the "ietf-crypto-types" YANG Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

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 Network 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 defines 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. The following subtrees and data nodes have particular sensitivity/vulnerability:

* The "cleartext-password" node:

  The "cleartext-password" node defined in the "password-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 "cleartext-symmetric-key" node:

  The "cleartext-symmetric-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 "cleartext-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.

* The "cert-data" node:

  The "cert-data" node, defined in both the "trust-anchor-cert-grouping" and "end-entity-cert-grouping" groupings, is additionally sensitive to read operations, as certificates may provide insight into which other resources/applications/servers this particular server communicates with, as well as potentially divulge personally identifying information (e.g., end-entity certificates). For this reason, the NACM extension "default-deny-all" has been applied.

All 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:

```
URI: urn:ietf:params:xml:ns:yang:ietf-crypto-types
Registrant Contact: The IESG
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
namespace:    urn:ietf:params:xml:ns:yang:ietf-crypto-types
prefix:       ct
reference:    RFC AAAA
```

5. References

5.1. Normative References

[ITU.X680.2021]
International Telecommunication Union, "Information technology - Abstract Syntax Notation One (ASN.1):

Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.680-202102-I>.

[ITU.X690.2021] International Telecommunication Union, "Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.690-202102-I>.

[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>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <https://www.rfc-editor.org/info/rfc4253>.

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

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

[RFC5958] Turner, S., "Asymmetric Key Packages", RFC 5958, DOI 10.17487/RFC5958, August 2010, <https://www.rfc-editor.org/info/rfc5958>.

[RFC6031] Turner, S. and R. Housley, "Cryptographic Message Syntax (CMS) Symmetric Key Package Content Type", RFC 6031, DOI 10.17487/RFC6031, December 2010, <https://www.rfc-editor.org/info/rfc6031>.

[RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <https://www.rfc-editor.org/info/rfc6960>.

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

[RFC7093] Turner, S., Kent, S., and J. Manger, "Additional Methods for Generating Key Identifiers Values", RFC 7093, DOI 10.17487/RFC7093, December 2013, <https://www.rfc-editor.org/info/rfc7093>.

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

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

[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>.

[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>.

5.2. Informative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-19, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[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-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[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-27, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

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

[RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <https://www.rfc-editor.org/info/rfc4211>.

[RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", RFC 5056, DOI 10.17487/RFC5056, November 2007, <https://www.rfc-editor.org/info/rfc5056>.

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

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

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

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

[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>.

[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>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

Appendix A. Change Log

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.
* removed 'mac-algorithm' identities for mac-aes-128-ccm, mac-aes-192-ccm, mac-aes-256-ccm, mac-aes-128-gcm, mac-aes-192-gcm, mac-aes-256-gcm, and mac-chacha20-poly1305.

* 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.

A.8. 06 to 07

* 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.

A.9. 07 to 08

* 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 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 'inline-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.

A.19. 17 to 18

* Fixed issues found by the SecDir review of the "keystore" draft.

* Added "password-grouping", discussed during the IETF 108 session.

A.20. 18 to 19

* Added a "Unconstrained Public Key Usage" Security Consideration to address concern raised by SecDir of the 'truststore' draft.

* Added a "Unconstrained Private Key Usage" Security Consideration to address concern raised by SecDir of the 'truststore' draft.

* Changed the encryption strategy, after conferring with Russ Housley.

* Added a "password-grouping" example to the "crypto-types-usage" example.

* Added an "Encrypting Passwords" section to Security Consideration.

* Addressed other comments raised by YANG Doctor.

A.21. 19 to 20

* Nits found via YANG Doctors reviews.

* Aligned modules with ‘pyang -f‘ formatting.

A.22. 20 to 21

* Replaced "base64encodedvalue==" with "BASE64VALUE=".

* Accommodated SecDir review by Valery Smyslov.

A.23. 21 to 22

* fixup the ’WG Web’ and ’WG List’ lines in YANG module(s)

* fixup copyright (i.e., s/Simplified/Revised/) in YANG module(s)

* added ’hidden-keys’ feature.

A.24. 22 to 23

* Fixed an example to reference correct key.

* Fixed an example to not have line-returns around the encoding for a binary value.

A.25. 23 to 24

* Added mandatory leaf "csr-format" to action "generate-csr".

* s/certificate-signing-request/csr/g in the YANG module.

A.26. 24 to 25

* Updated per Shepherd reviews impacting the suite of drafts.

A.27. 25 to 26

* Updated per Shepherd reviews impacting the suite of drafts.

A.28. 26 to 27

* Updated per Tom Petch and AD reviews.

* Renamed numerous "feature" statements and some "grouping" statements (in YANG)

* Added "csr-format" and "p10-csr-format" identities to doc (they were already in YANG)

* Clarified that the 'rsa-private-key-format' and 'ec-private-key-format' formats must be encoded using DER

* Added 'if-feature cleartext-passwords' statement to 'case cleartext-password' in grouping 'password-grouping'.

* Added 'if-feature cleartext-keys' statement to 'case cleartext-key' in grouping 'symmetric-key-grouping'.

* Added 'if-feature cleartext-cleartext-private-keys' statement to 'case cleartext-private-key' in grouping 'asymmetric-key-grouping'.

* Updated Section titles.

* Clarified Security Considerations about the "generate-public-key" RPCs.

A.29. 27 to 28

* Mostly addresses AD review comments.

* Also addresses on-list comment regarding public-keys being "mandatory true."

* Added note to Editor to fix line foldings.

* Factored 'private-key-grouping' from 'asymmetric-key-pair-grouping'.

* Made public-key in 'asymmetric-key-pair-grouping' be "mandatory false".

* Renamed 'encrypted-by-choice-grouping' to 'encrypted-by-grouping'.

A.30. 28 to 29

* Addresses Gen-ART review by Dale Worley.

* Addresses review by Tom Petch.

A.31. 29 to 30

* Addresses 1st-round of IESG reviews.

A.32. 30 to 32

* Addresses issues found in OpsDir of the ssh-client-server draft.

* Removed "Strength of Keys Conveyed" section.

* Renamed Security Considerations section s/Template for/ Considerations for/

* Improved Security Consideration for 'cert-data' node.

A.33. 32 to 34

* Nothing changed. Only bumped for automation...

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Balázs Kovács, Carsten Bormann, Dale Worley, Eric Voit, Éric Vyncke, Francesca Palombini, Jürgen Schönwälder, Lars Eggert, Liang Xia, Martin Björklund, Mahesh Jethanandani, Murray Kucherawy, Nick Hancock, Orie Steele, Paul Wouters, Rich Salz, Rifaat Shekh-Yusef, Rob Wilton, Roman Danyliw, Russ Housley, Sandra Murphy, Tom Petch, Valery Smyslov, Wang Haiguang, Warren Kumari, and Zaheduzzaman Sarker.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net