NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 21, 2020

K. Watsen
Watsen Networks

May 20, 2020

Common YANG Data Types for Cryptography
draft-ietf-netconf-crypto-types-15

Abstract

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

Editorial Note (To be removed by RFC Editor)

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

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

- "AAAA" --> the assigned RFC value for this draft

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

- "2020-05-20" --> the publication date of this draft

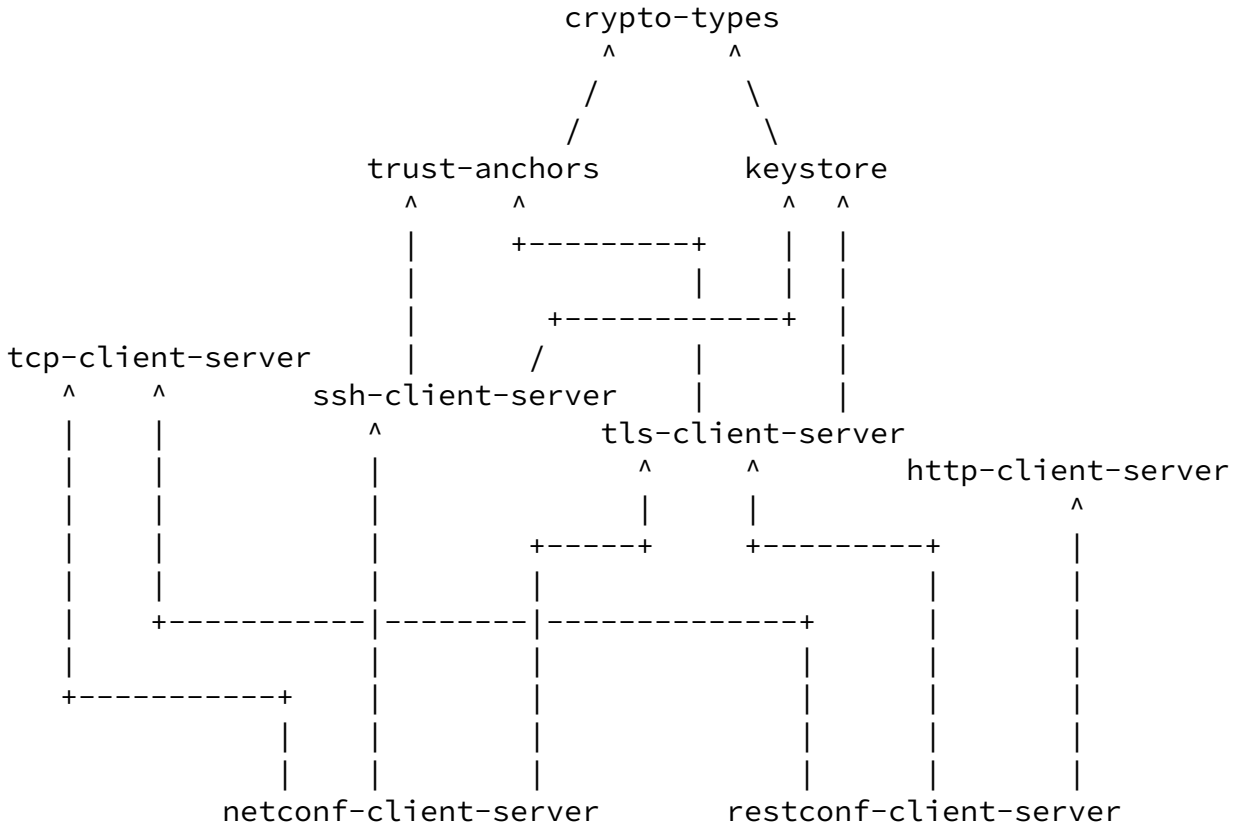
The following Appendix section is to be removed prior to publication:

- Appendix B. Change Log

Note to Reviewers (To be removed by RFC Editor)

This document presents a YANG module or modules that is/are part of a collection of drafts that work together to produce the ultimate goal of the NETCONF WG: to define configuration modules for NETCONF client and servers, and RESTCONF client and servers.

The relationship between the various drafts in the collection is presented in the below diagram.



Full draft names and link to drafts:

- draft-ietf-netconf-crypto-types (html [1])
- draft-ietf-netconf-trust-anchors (html [2])
- draft-ietf-netconf-keystore (html [3])
- draft-ietf-netconf-tcp-client-server (html [4])
- draft-ietf-netconf-ssh-client-server (html [5])
- draft-ietf-netconf-tls-client-server (html [6])
- draft-ietf-netconf-http-client-server (html [7])
- draft-ietf-netconf-netconf-client-server (html [8])
- draft-ietf-netconf-restconf-client-server (html [9])

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on November 21, 2020.

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.

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.

2. The Crypto Types Module

2.1. Tree Diagram

This section provides a tree diagram [RFC8340] for the "ietf-crypto-types" module. Only "grouping" statements are represented, as tree diagrams have no means to represent identities or typedefs.

```
module: ietf-crypto-types

  grouping symmetric-key-grouping
    +-- key-format?         identityref
    +-- (key-type)
       +--:(key)
       |  +-- key?          binary
```



```
       +--:(hidden-key)
          +-- hidden-key?   empty
  grouping public-key-grouping
    +-- public-key-format    identityref
    +-- public-key           binary
  grouping asymmetric-key-pair-grouping
    +-- public-key-format           identityref
    +-- public-key                  binary
    +-- private-key-format?         identityref
    +-- (private-key-type)
       +--:(private-key)
       |  +-- private-key?          binary
       +--:(hidden-private-key)
          +-- hidden-private-key?   empty
  grouping trust-anchor-cert-grouping
    +-- cert?                     trust-anchor-cert-cms
    +---n certificate-expiration
       +-- expiration-date    yang:date-and-time
  grouping trust-anchor-certs-grouping
    +-- cert*                     trust-anchor-cert-cms
    +---n certificate-expiration
       +-- expiration-date    yang:date-and-time
```

```
grouping end-entity-cert-grouping
  +-- cert?                     end-entity-cert-cms
  +---n certificate-expiration
     +-- expiration-date    yang:date-and-time
grouping end-entity-certs-grouping
  +-- cert*                     end-entity-cert-cms
  +---n certificate-expiration
     +-- expiration-date    yang:date-and-time
grouping generate-csr-grouping
  +---x generate-certificate-signing-request
          {certificate-signing-request-generation}?
     +---w input
     |  +---w subject       binary
     |  +---w attributes?   binary
     +--ro output
        +--ro certificate-signing-request    ct:csr
grouping asymmetric-key-pair-with-cert-grouping
  +-- public-key-format                        identityref
  +-- public-key                               binary
  +-- private-key-format?                      identityref
  +-- (private-key-type)
  |  +--:(private-key)
  |  |  +-- private-key?                       binary
  |  +--:(hidden-private-key)
  |     +-- hidden-private-key?                empty
  +-- cert?                                    end-entity-cert-cms
```



```
  +---n certificate-expiration
  |  +-- expiration-date    yang:date-and-time
  +---x generate-certificate-signing-request
          {certificate-signing-request-generation}?
     +---w input
     |  +---w subject       binary
     |  +---w attributes?   binary
     +--ro output
        +--ro certificate-signing-request    ct:csr
grouping asymmetric-key-pair-with-certs-grouping
  +-- public-key-format                        identityref
  +-- public-key                               binary
  +-- private-key-format?                      identityref
  +-- (private-key-type)
  |  +--:(private-key)
```

```
      |  |  +-- private-key?                        binary
      |  +--:(hidden-private-key)
      |     +-- hidden-private-key?                 empty
      +-- certificates
      |  +-- certificate* [name]
      |     +-- name?                       string
      |     +-- cert                        end-entity-cert-cms
      |     +---n certificate-expiration
      |        +-- expiration-date    yang:date-and-time
      +---x generate-certificate-signing-request
              {certificate-signing-request-generation}?
         +---w input
         |  +---w subject       binary
         |  +---w attributes?   binary
         +--ro output
            +--ro certificate-signing-request    ct:csr
```

2.2. YANG Module

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

<CODE BEGINS> file "ietf-crypto-types@2020-05-20.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:   <http://datatracker.ietf.org/wg/netconf/>
   WG List:  <mailto:netconf@ietf.org>
   Author:   Kent Watsen <mailto:kent+ietf@watsen.net>";

description
  "This module defines common YANG types for cryptographic
   applications.

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

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

   This version of this YANG module is part of RFC AAAA
   (https://www.rfc-editor.org/info/rfcAAAA); see the RFC
   itself for full legal notices.

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

revision 2020-05-20 {
  description
    "Initial version";
  reference
    "RFC AAAA: Common YANG Data Types for Cryptography";
```



```
}
```

```
/****************/
/*   Features   */
/****************/

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

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

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

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

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


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

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

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

```
  }

  identity symmetric-key-format {
    description "Base key-format identity for symmetric 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 3447).";
    reference
      "RFC 3447: PKCS #1: RSA Cryptography
                 Specifications Version 2.2";
  }

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

  identity one-asymmetric-key-format {
    if-feature "one-asymmetric-key-format";
    base "private-key-format";
    description
      "Indicates that the private key value is a CMS
       OneAsymmetricKey structure, as defined in RFC 5958,
       encoded using ASN.1 distinguished encoding rules
       (DER), as specified in ITU-T X.690.";
    reference
      "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).";
  }
```

```
identity encrypted-one-asymmetric-key-format {
```



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


  /****************************************************/
  /*   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).";
    }


    /****************************************************/
    /*   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.";
    }

    identity one-symmetric-key-format {
      if-feature "one-symmetric-key-format";
      base "symmetric-key-format";
      description
        "Indicates that the private key value is a CMS
         OneSymmetricKey structure, as defined in RFC 6031,
         encoded using ASN.1 distinguished encoding rules
         (DER), as specified in ITU-T X.690.";
      reference
        "RFC 6031: Cryptographic Message Syntax (CMS)
                   Symmetric Key Package Content Type
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
```

```
        Specification of Basic Encoding Rules (BER),
        Canonical Encoding Rules (CER) and Distinguished
        Encoding Rules (DER).";
    }

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



```
         RFC 6031: Cryptographic Message Syntax (CMS)
                   Symmetric Key Package Content Type
         ITU-T X.690:
           Information technology - ASN.1 encoding rules:
           Specification of Basic Encoding Rules (BER),
           Canonical Encoding Rules (CER) and Distinguished
           Encoding Rules (DER).";
    }


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

    typedef csr {
      type binary;
      description
        "A CertificationRequest structure, as specified in RFC 2986,
         encoded using ASN.1 distinguished encoding rules (DER), as
         specified in ITU-T X.690.";
      reference
        "RFC 2986:
           PKCS #10: Certification Request Syntax Specification
           Version 1.7
         ITU-T X.690:
```

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

    /*************************************************/
    /*   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).";
    }

    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).";
    }


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

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

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

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

     This CMS structure MAY (as applicable where this type is
     used) also contain suitably fresh (as defined by local
```

```
          policy) revocation objects with which the device can
          verify the revocation status of the certificates.

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

      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 that is commonly used to disseminate X.509
           certificates and revocation objects (RFC 5280).";
        reference
          "RFC 5280:
            Internet X.509 Public Key Infrastructure Certificate
            and Certificate Revocation List (CRL) Profile.";
      }


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

      grouping symmetric-key-grouping {
        description
          "A symmetric key and algorithm.";
        leaf key-format {
```

```
        nacm:default-deny-write;
```



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

    grouping public-key-grouping {
      description
        "A public key and its associated algorithm.";
      leaf public-key-format {
        nacm:default-deny-write;
        type identityref {
          base public-key-format;
        }
        mandatory true;
        description "Identifies the key's format.";
      }
      leaf public-key {
```

```
      nacm:default-deny-write;
      type binary;
      mandatory true;
      description
        "The binary value of the public key.  The interpretation
         of the value is defined by 'public-key-format' field.";
    }
  }
```



```
  grouping asymmetric-key-pair-grouping {
    description
      "A private key and its associated public key and algorithm.";
    uses public-key-grouping;
    leaf private-key-format {
      nacm:default-deny-write;
      type identityref {
        base private-key-format;
      }
      description "Identifies the key's format.";
    }
    choice private-key-type {
      mandatory true;
      description
        "Choice between key types.";
      leaf 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.";
      }
      leaf hidden-private-key {
        nacm:default-deny-write;
        type empty;
        must "not(../private-key-format)";
        description
          "A permanently hidden key.  How such keys are created
           is outside the scope of this module.";
      }
    }
  }
```

```
  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 {
      nacm:default-deny-write;
      type trust-anchor-cert-cms;
      description
        "The binary certificate data for this certificate.";
      reference
        "RFC YYYY: Common YANG Data Types for Cryptography";
    }
    notification certificate-expiration {
      description
```



```
        "A notification indicating that the configured certificate
         is either about to expire or has already expired.  When to
         send notifications is an implementation specific decision,
         but it is RECOMMENDED that a notification be sent once a
         month for 3 months, then once a week for four weeks, and
         then once a day thereafter until the issue is resolved.";
      leaf expiration-date {
        type yang:date-and-time;
        mandatory true;
        description
          "Identifies the expiration date on the certificate.";
      }
    }
  }

  grouping trust-anchor-certs-grouping {
    description
      "A list of trust anchor certificates, and a notification
       for when one is about to (or already has) expire.";
    leaf-list cert {
      nacm:default-deny-write;
      type trust-anchor-cert-cms;
      description
        "The binary certificate data for this certificate.";
      reference
        "RFC YYYY: Common YANG Data Types for Cryptography";
```

```
    }
    notification certificate-expiration {
      description
        "A notification indicating that the configured certificate
         is either about to expire or has already expired.  When to
         send notifications is an implementation specific decision,
         but it is RECOMMENDED that a notification be sent once a
         month for 3 months, then once a week for four weeks, and
         then once a day thereafter until the issue is resolved.";
      leaf expiration-date {
        type yang:date-and-time;
        mandatory true;
        description
          "Identifies the expiration date on the certificate.";
      }
    }
  }

  grouping 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 {
      nacm:default-deny-write;
      type end-entity-cert-cms;
      description
        "The binary certificate data for this certificate.";
      reference
        "RFC YYYY: Common YANG Data Types for Cryptography";
    }
    notification certificate-expiration {
      description
        "A notification indicating that the configured certificate
         is either about to expire or has already expired.  When to
         send notifications is an implementation specific decision,
         but it is RECOMMENDED that a notification be sent once a
         month for 3 months, then once a week for four weeks, and
         then once a day thereafter until the issue is resolved.";
      leaf expiration-date {
```

```
        type yang:date-and-time;
        mandatory true;
        description
          "Identifies the expiration date on the certificate.";
      }
    }
  }

  grouping end-entity-certs-grouping {
    description
      "A list of end entity certificates, and a notification for
       when one is about to (or already has) expire.";
    leaf-list cert {
      nacm:default-deny-write;
      type end-entity-cert-cms;
      description
        "The binary certificate data for this certificate.";
      reference
        "RFC YYYY: Common YANG Data Types for Cryptography";
    }
    notification certificate-expiration {
      description
        "A notification indicating that the configured certificate
         is either about to expire or has already expired.  When to
         send notifications is an implementation specific decision,
         but it is RECOMMENDED that a notification be sent once a
         month for 3 months, then once a week for four weeks, and
         then once a day thereafter until the issue is resolved.";
      leaf expiration-date {
```



```
        type yang:date-and-time;
        mandatory true;
        description
          "Identifies the expiration date on the certificate.";
      }
    }
  }

  grouping generate-csr-grouping {
    description
      "Defines the 'generate-certificate-signing-request' action.";
    action generate-certificate-signing-request {
```

```
        if-feature certificate-signing-request-generation;
        nacm:default-deny-all;
        description
          "Generates a certificate signing request structure for
           the associated asymmetric key using the passed subject
           and attribute values.

           This action statement is only available when the
           associated 'public-key-format' node's value is
           'subject-public-key-info-format'.";
        reference
          "RFC 6125:
            Representation and Verification of Domain-Based
            Application Service Identity within Internet Public Key
            Infrastructure Using X.509 (PKIX) Certificates in the
            Context of Transport Layer Security (TLS)";
        input {
          leaf subject {
            type binary;
            mandatory true;
            description
              "The 'subject' field per the CertificationRequestInfo
                structure as specified by RFC 2986, Section 4.1
                encoded using the 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).";
          }
          leaf attributes {
            type binary;
```



```
            description
              "The 'attributes' field from the structure
               CertificationRequestInfo as specified by RFC 2986,
               Section 4.1 encoded using the 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).";
          }
        }
        output {
          leaf certificate-signing-request {
            type ct:csr;
            mandatory true;
            description
              "The requested certificate signing request structure.";
          }
        }
      }
    } // generate-csr-grouping

    grouping asymmetric-key-pair-with-cert-grouping {
      description
        "A private/public key pair and an associated certificate.
         Implementations SHOULD assert that certificates contain
         the matching public key.";
      uses asymmetric-key-pair-grouping;
      uses end-entity-cert-grouping;
      uses generate-csr-grouping;
    } // asymmetric-key-pair-with-cert-grouping

    grouping asymmetric-key-pair-with-certs-grouping {
      description
        "A private/public key pair and associated certificates.
         Implementations SHOULD assert that certificates contain
         the matching public key.";
      uses asymmetric-key-pair-grouping;
      container certificates {
        nacm:default-deny-write;
        description
          "Certificates associated with this asymmetric key.
           More than one certificate supports, for instance,
           a TPM-protected asymmetric key that has both IDevID
```

```
           and LDevID certificates associated.";
        list certificate {
          key "name";
          description
            "A certificate for this asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the certificate.  If the name
               matches the name of a certificate that exists
               independently in <operational> (i.e., an IDevID),
               then the 'cert' node MUST NOT be configured.";
          }
          uses end-entity-cert-grouping {
            refine cert {
              mandatory true;
            }
          }
        }
      }
      uses generate-csr-grouping;
    } // asymmetric-key-pair-with-certs-grouping

  }

  <CODE ENDS>
```

2.3. Examples

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

The following example module illustrates the use of both the "symmetric-key-grouping" and the "asymmetric-key-pair-with-certs-grouping" groupings defined in the "ietf-crypto-types" module.

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

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

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

  organization
```

```
      "Example Corporation";

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

    description
     "This module illustrates the grouping
      defined in the crypto-types draft called
      'asymmetric-key-pair-with-certs-grouping'.";

    revision "2020-05-20" {
      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;
      }
    }
    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;
        description
          "An asymmetric key pair with associated certificates.";
      }
    }
  }
```


Given the above example usage module, the following example illustrates some configured keys.

```
<symmetric-keys
  xmlns="http://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-key/>
  </symmetric-key>
  <symmetric-key>
    <name>ex-octet-string-based-symmetric-key</name>
    <key-format>ct:octet-string-key-format</key-format>
    <key>base64encodedvalue==</key>
  </symmetric-key>
  <symmetric-key>
    <name>ex-one-symmetric-based-symmetric-key</name>
    <key-format>ct:one-symmetric-key-format</key-format>
    <key>base64encodedvalue==</key>
  </symmetric-key>
  <symmetric-key>
    <name>ex-encrypted-one-symmetric-based-symmetric-key</name>
    <key-format>ct:encrypted-one-symmetric-key-format</key-format>
    <key>base64encodedvalue==</key>
  </symmetric-key>
</symmetric-keys>

<asymmetric-keys
  xmlns="http://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>base64encodedvalue==</public-key>
        <hidden-private-key/>
        <certificates>
          <certificate>
            <name>ex-hidden-key-cert</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>
      <asymmetric-key>
        <name>ex-subject-public-info-based-asymmetric-key</name>
        <public-key-format>
          ct:subject-public-key-info-format
```



```
        </public-key-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>
          ct:rsa-private-key-format
        </private-key-format>
        <private-key>base64encodedvalue==</private-key>
        <certificates>
          <certificate>
            <name>ex-cert</name>
            <cert>base64encodedvalue==</cert>
          </certificate>
        </certificates>
      </asymmetric-key>
      <asymmetric-key>
        <name>ex-one-asymmetric-based-symmetric-key</name>
        <public-key-format>
          ct:subject-public-key-info-format
        </public-key-format>
        <public-key>base64encodedvalue==</public-key>
        <private-key-format>
          ct:one-asymmetric-key-format
        </private-key-format>
        <private-key>base64encodedvalue==</private-key>
      </asymmetric-key>
      <asymmetric-key>
        <name>ex-encrypted-one-asymmetric-based-symmetric-key</name>
        <public-key-format>
          ct:subject-public-key-info-format
```

```
      </public-key-format>
      <public-key>base64encodedvalue==</public-key>
      <private-key-format>
        ct:encrypted-one-asymmetric-key-format
      </private-key-format>
      <private-key>base64encodedvalue==</private-key>
    </asymmetric-key>
  </asymmetric-keys>
```

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

The following example illustrates the "generate-certificate-signing-request" action with the NETCONF protocol.

REQUEST

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

RESPONSE

```
<rpc-reply message-id="101"
```

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

2.3.3. The "certificate-expiration" Notification

The following example illustrates the "certificate-expiration" notification with the NETCONF protocol.


```
   <notification
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <eventTime>2018-05-25T00:01:00Z</eventTime>
     <keys xmlns="http://example.com/ns/example-crypto-types-usage">
       <key>
         <name>locally-defined key</name>
         <certificates>
           <certificate>
             <name>my-cert</name>
             <certificate-expiration>
               <expiration-date>
                 2018-08-05T14:18:53-05:00
               </expiration-date>
             </certificate-expiration>
```

```
          </certificate>
        </certificates>
      </key>
    </keys>
  </notification>
```

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 is 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. Access to Data Nodes

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

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

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

There are a number of data nodes defined by the grouping statements that are writable/creatable/deletable (i.e., config true, which is

the default). Some of these data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

*: All of the data nodes defined by all the groupings are

considered sensitive to write operations. For instance, 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 by all the groupings.

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

/private-key: The "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 to it here.

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:

*: All of the "action" statements defined by groupings SHOULD only be executed by authorized users. For this reason, the NACM extension "default-deny-all" has been applied to all of them. 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".

generate-certificate-signing-request: 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. Update 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 NETCONF WG of the IETF.
XML: N/A, the requested URI is an XML namespace.
```

4.2. Update the "YANG Module Names" Registry

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

```
name:         ietf-crypto-types
namespace:    urn:ietf:params:xml:ns:yang:ietf-crypto-types
prefix:       ct
reference:    RFC AAAA
```

5. References

5.1. Normative References

[ITU.X690.2015]
: 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, August 2015, <https://www.itu.int/rec/T-REC-X.690/>.

[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>.

[RFC3447]
: Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, DOI 10.17487/RFC3447, February 2003, <https://www.rfc-editor.org/info/rfc3447>.

[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>.

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

[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>.

[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

[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>.

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

[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>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

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

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF

Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[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>.

5.3. URIs

[1] https://tools.ietf.org/html/draft-ietf-netconf-crypto-types

[2] https://tools.ietf.org/html/draft-ietf-netconf-trust-anchors

[3] https://tools.ietf.org/html/draft-ietf-netconf-keystore

[4] https://tools.ietf.org/html/draft-ietf-netconf-tcp-client-server

[5] https://tools.ietf.org/html/draft-ietf-netconf-ssh-client-server

[6] https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server

[7] https://tools.ietf.org/html/draft-ietf-netconf-http-client-server

[8] https://tools.ietf.org/html/draft-ietf-netconf-netconf-client-server

[9] https://tools.ietf.org/html/draft-ietf-netconf-restconf-client-server

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

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

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

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

- Add in des3-cbc-sha1 for IPSec
- Add in sha1-des3-kd for IPSec
- Add in definit for rc4-hmac and rc4-hmac-exp. These two algorithms have been deprecated in RFC 8429. But some existing draft in i2nsf may still want to use them.
- Add x25519 and x448 curve for asymmetric algorithms
- Add signature algorithms ed25519, ed25519-cts, ed25519ph
- add signature algorithms ed448, ed448ph
- Add in rsa-sha2-256 and rsa-sha2-512 for SSH protocols (rfc8332)

A.12. 10 to 11

- Added a "key-format" identity.
- Added symmetric keys to the example in Section 2.3.

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.

Acknowledgements

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

Author's Address

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net