NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2020
K. Watsen
Watsen Networks
H. Wang
Huawei
March 8, 2020

Common YANG Data Types for Cryptography
draft-ietf-netconf-crypto-types-14

Abstract

This document primarily defines a YANG module for identities, typedefs, groupings, and RPCs useful to cryptographic applications. This draft additionally defines a new IANA registry for cryptographic primitives, modifies existing SSH and TLS registries, and defines a process enabling IANA to automatically generate three new YANG modules from the new cryptographic primitives registry.

Editorial Note (To be removed by RFC Editor)

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

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

o "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:

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

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

o Appendix B. Change Log

Status of This Memo

Table of Contents

1. Introduction

This document primarily defines a YANG 1.1 [RFC7950] module for identities, typedefs, groupings, and RPCs useful to cryptographic applications.

This draft additionally defines a new IANA registry called "Cryptographic Primitives", and defines a process enabling IANA to automatically generate three new YANG modules ("iana-hash-algs", "iana-symmetric-key-algs", and "iana-asymmetric-key-algs") from the new cryptographic primitives registry.

Lastly, the draft also modifies existing SSH and TLS registries, adding a new column called "Primitives" to specific sub-registries identifying which primitives are used by that registration.

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

  rpcs:
    +---x generate-asymmetric-key {asymmetric-key-generation}?
    |  +---w input
    |  |  +---w algorithm    iasa:asymmetric-algorithm-type
    |  +--ro output
    |     +--ro algorithm
    |     |       iasa:asymmetric-algorithm-type
    |     +--ro public-key-format           identityref
    |     +--ro public-key                  binary
    |     +--ro private-key-format?         identityref
    |     +--ro (private-key-type)
    |        +--:(private-key)
    |        |  +--ro private-key?          binary
    |        +--:(hidden-private-key)
    |           +--ro hidden-private-key?   empty
    +---x generate-symmetric-key {symmetric-key-generation}?
       +---w input
       |  +---w algorithm    isa:symmetric-algorithm-type
       +--ro output
          +--ro algorithm           isa:symmetric-algorithm-type
          +--ro key-format?         identityref
          +--ro (key-type)
             +--:(key)
             |  +--ro key?          binary
             +--:(hidden-key)
                +--ro hidden-key?   empty

  grouping symmetric-key-grouping
    +-- algorithm           isa:symmetric-algorithm-type
    +-- key-format?         identityref
    +-- (key-type)
       +--:(key)
```

```
          |  +-- key?          binary
          +--:(hidden-key)
             +-- hidden-key?   empty
     grouping public-key-grouping
       +-- algorithm            iasa:asymmetric-algorithm-type
       +-- public-key-format    identityref
       +-- public-key           binary
     grouping asymmetric-key-pair-grouping
       +-- algorithm                   iasa:asymmetric-algorithm-type
       +-- public-key-format           identityref
       +-- public-key                  binary
       +-- private-key-format?         identityref
       +-- (private-key-type)
          +--:(private-key)
          |  +-- private-key?          binary
          +--:(hidden-private-key)
             +-- hidden-private-key?   empty
     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 asymmetric-key-pair-with-cert-grouping
       +-- algorithm
       |       iasa:asymmetric-algorithm-type
       +-- public-key-format                       identityref
       +-- public-key                              binary
       +-- private-key-format?                     identityref
       +-- (private-key-type)
       |  +--:(private-key)
       |  |  +-- private-key?                      binary
       |  +--:(hidden-private-key)
       |     +-- hidden-private-key?               empty
       +-- cert?                                   end-entity-cert-cms
       +---n certificate-expiration
       |  +-- expiration-date    yang:date-and-time
       +---x generate-certificate-signing-request
```

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

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@2020-03-08.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";
     }

     //import iana-hash-algs {
     //  prefix iha;
     //  reference
     //    "RFC AAAA: Common YANG Data Types for Cryptography";
     //}

     import iana-symmetric-algs {
       prefix isa;
       reference
         "RFC AAAA: Common YANG Data Types for Cryptography";
     }

     import iana-asymmetric-algs {
       prefix iasa;
       reference
         "RFC AAAA: Common YANG Data Types for Cryptography";
     }

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

     contact
       "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
        WG List:  <mailto:netconf@ietf.org>
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Wang Haiguang <wang.haiguang.shieldlab@huawei.com>";

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

        Copyright (c) 2019 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

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

        This version of this YANG module is part of RFC 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 2020-03-08 {
    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 symmetric-key-generation {
    description
```

```
         "Indicates that the server implements the 'generate-
          symmetric-key' RPC.";
     }

     feature asymmetric-key-generation {
       description
         "Indicates that the server implements the 'generate-
          asymmetric-key' RPC.";
     }


     /*************************************************/
     /*   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 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 algorithm {
         type isa:symmetric-algorithm-type;
```

```
         mandatory true;
         description
           "The algorithm to be used when generating the key.";
         reference
           "RFC AAAA: Common YANG Data Types for Cryptography";
       }
       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 algorithm {
         nacm:default-deny-write;
         type iasa:asymmetric-algorithm-type;
         mandatory true;
         description
           "Identifies the key's algorithm.";
         reference
           "RFC AAAA: Common YANG Data Types for Cryptography";
       }
```

```
       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 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;
         action generate-certificate-signing-request {
           nacm:default-deny-all;
           description
             "Generates a certificate signing request structure for
              the associated asymmetric key using the passed subject
              and attribute values.  The specified assertions need
              to be appropriate for the certificate's use.  For
              example, an entity certificate for a TLS server
              SHOULD have values that enable clients to satisfy
              RFC 6125 processing.";
           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 binary;
             mandatory true;
             description
               "A CertificationRequest structure as specified by
                RFC 2986, Section 4.2 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).";
          }
        }
      } // generate-certificate-signing-request
    } // 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;
        }
      } // certificates
      action generate-certificate-signing-request {
        nacm:default-deny-all;
        description
          "Generates a certificate signing request structure for
           the associated asymmetric key using the passed subject
           and attribute values.  The specified assertions need
           to be appropriate for the certificate's use.  For
           example, an entity certificate for a TLS server
           SHOULD have values that enable clients to satisfy
           RFC 6125 processing.";
```

```
         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 binary;
             mandatory true;
             description
               "A CertificationRequest structure as specified by
                RFC 2986, Section 4.2 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).";
           }
         }
       } // generate-certificate-signing-request
     } // asymmetric-key-pair-with-certs-grouping


     /*********************************/
     /*   Protocol-Accessible Nodes   */
     /*********************************/

     rpc generate-asymmetric-key {
       if-feature "asymmetric-key-generation";
       description
         "Requests the device to generate an asymmetric key using
          the specified key algorithm.";
       input {
         leaf algorithm {
           type iasa:asymmetric-algorithm-type;
           mandatory true;
           description
             "The algorithm to be used when generating the key.";
           reference
             "RFC AAAA: Common YANG Data Types for Cryptography";
         }
       }
       output {
         uses ct:asymmetric-key-pair-grouping;
       }
     } // end generate-asymmetric-key


     rpc generate-symmetric-key {
       if-feature "symmetric-key-generation";
       description
         "Requests the device to generate an symmetric key using
          the specified key algorithm.";
       input {
         leaf algorithm {
```

```
          type isa:symmetric-algorithm-type;
          mandatory true;
          description
            "The algorithm to be used when generating the key.";
          reference
            "RFC AAAA: Common YANG Data Types for Cryptography";
        }
      }
      output {
        uses ct:symmetric-key-grouping;
      }
    } // end generate-symmetric-key

  }

  <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 XXXX: 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 "1001-01-01" {
       description
        "Initial version";
       reference
        "RFC ????: Usage Example for RFC XXXX";
     }

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

```
      <key-format>ct:octet-string-key-format</key-format>
      <key>base64encodedvalue==</key>
    </symmetric-key>
    <symmetric-key>
      <name>ex-hidden-symmetric-key</name>
      <algorithm>aes-256-cbc</algorithm>
      <hidden-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-asymmetric-key</name>
      <algorithm>rsa2048</algorithm>
      <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-hidden-asymmetric-key</name>
      <algorithm>rsa2048</algorithm>
      <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-keys>
```

2.3.2. The "generate-symmetric-key" RPC

The following example illustrates the "generate-symmetric-key" RPC with the NETCONF protocol.

REQUEST

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

RESPONSE

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

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

2.3.3. The "generate-asymmetric-key" RPC

The following example illustrates the "generate-asymmetric-key" RPC with the NETCONF protocol.

REQUEST

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

RESPONSE

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

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

2.3.4. 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>
```

2.3.5. 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.

generate-symmetric-key: FIXME

generate-asymmetric-key: FIXME

4. IANA Considerations

4.1. Create the "Cryptographic Primitives" Registry

This section defines a new registry called "Cryptographic Primitives", following the guidelines described in Section 4 of [RFC5226].

This registery enumerates various primitive algorithms that are used by various cryptographic ciphers and protocols.

The following note shall be at the top of the registry:

> This registry enumerates cryptographic primitives that are or have been used by various cryptographic ciphers and protocols.

4.1.1. Introduction

The Cryptographic Primitives registry is composed of a number of sub-registries, one for each kind of primitive algorithm.

Each sub-registry has the same number of fields and update policy.

The fields for each sub-registry are:

o Name

  * The name of the algorithm (required).

  * The name must be the common "enumerated" value for the algorithm.

   * The name must be unique within the sub-registry.

   * The name must be a single word composed of one or more ASCII characters.

   * Each character must be either an uppercase or lowercase letter, a digit, a hyphen, or an underscore.

   * While not bounded, the name is expected to be relatively short; unlikely ever exceeding a couple dozen characters.

- Description

   * An arbitrary description of the algorithm (optional).

   * The discription may be used to provide a human-facing name and/or alternate names for the algorithm.

   * The description, when present, is expected to be no more than a few sentences.

   * The description is to be in English, but may contain UTF-8 characters as may be needed in some cases.

- Status

   * An enumerated value stating the current status of the algorithm (optional).

   * The value, when present, must be "Recommended", "Deprecated" or "Obsolete".

   * An algorithm having no "status" specified (i.e., not marked as "Recommended") does not necessarily mean that it is flawed; rather, it indicates that the item either has not been through the IETF consensus process, has limited applicability, or is intended only for specific use cases.

   * When requesting a registration for an algorithm having no status, the request should use an empty string value (i.e., "Status: ") to clearly indicate no status, as opposed to the value having been forgotten.

- References

   * One or more normative references for the algorithm (required).

* Each reference must declare its "type" as as either "text" or "rfc" and, if "rfc", must also declare an "data" value containing the RFC's number in the form "rfcxxxx" (or "rfcxxxxx"). In either case, the xref's content must contain a suitable textual citation, e.g., containing both a tracking number (e.g., RFC 2119) as well the document's title (e.g., Key words for use in RFCs to Indicate Requirement Levels). Rendering software (e.g., stylesheets) may choose to present the reference in any suitable manner.

* There must be at least one reference to a document that defines the algorithm.

* There must be a reference to the document that originated the algorithm's registration.

* The document that defines the algorithm and the document that defines originated the registration may be the same.

* While not bounded, the total number of references is unlikely to ever exceed a few.

The update policy is either "RFC Required" or "IETF Review", and maybe also "IESG Approval". In any case, it is always requires an "Expert Review" (a.k.a. "Designated Expert).

Whenever a sub-registry is updated, IANA must automatically update and re-published the corresponding YANG module, as described in IANA-maintained YANG Modules (Section 4.2).

4.1.2. The "Symmetric Key Algorithms" Sub-Registry

The "Symmetric Key Algorithms" sub-registry enumerates symmetric key algorithms used by cryptographic ciphers and protocols.

The format of this registry is described in the Introduction (Section 4.1.1) section above.

Following is the initial assignment for this sub-registry:

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

Record:
  Name: des
  Description: The Data Encryption Algorithm
  Status:
  Reference (type="text"): National Institute of Standards and Techn\
ology. FIPS Pub 46: Data Encryption Standard.  15 January 1977.

Record:
  Name: 3des
  Description: The Data Encryption Algorithm
  Status:
  Reference (type="rfc" data="1851"): RFC 3961: The ESP Triple DES T\
ransform

Record:
  Name: aes
  Description: The AES algorithm.
  Status:
  Reference (type="text"): National Institute of Standards.  FIPS Pu\
b 197: Advanced Encryption Standard (AES).  26 November 2001.
```

4.1.3. The "Asymmetric Key Algorithms" Sub-Registry

The "Asymmetric Key Algorithms" sub-registry enumerates asymmetric key algorithms used by cryptographic ciphers and protocols.

The format of this registry is described in the Introduction (Section 4.1.1) section above.

Following is the initial assignment for this sub-registry:

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

Record:
  Name: rsa
  Description: The RSA algorithm
  Status:
  Reference (type="rfc" data="rfc8017"): RFC 8017: PKCS #1: RSA Cryp\
tography Specifications Version 2.2

Record:
  Name: secp192r1
  Description: The asymmetric algorithm using a NIST P192 Curve
  Status:
  Reference (type="rfc" data="rfc6090"): RFC 6090: Fundamental Ellip\
```

```
tic Curve Cryptography Algorithms
  Reference (type="rfc" data="rfc5480"): RFC 5480: Elliptic Curve Cr\
yptography Subject Public Key Information

Record:
  Name: secp224r1
  Description: The asymmetric algorithm using a NIST P224 Curve
  Status:
  Reference (type="rfc" data="rfc6090"): RFC 6090: Fundamental Ellip\
tic Curve Cryptography Algorithms
  Reference (type="rfc" data="rfc5480"): RFC 5480: Elliptic Curve Cr\
yptography Subject Public Key Information

Record:
  Name: secp256r1
  Description: The asymmetric algorithm using a NIST P256 Curve
  Status:
  Reference (type="rfc" data="rfc6090"): RFC 6090: Fundamental Ellip\
tic Curve Cryptography Algorithms
  Reference (type="rfc" data="rfc5480"): RFC 5480: Elliptic Curve Cr\
yptography Subject Public Key Information

Record:
  Name: secp384r1
  Description: The asymmetric algorithm using a NIST P384 Curve
  Status:
  Reference (type="rfc" data="rfc6090"): RFC 6090: Fundamental Ellip\
tic Curve Cryptography Algorithms
  Reference (type="rfc" data="rfc5480"): RFC 5480: Elliptic Curve Cr\
yptography Subject Public Key Information

Record:
  Name: secp521r1
  Description: The asymmetric algorithm using a NIST P521 Curve
  Status:
  Reference (type="rfc" data="rfc6090"): RFC 6090: Fundamental Ellip\
tic Curve Cryptography Algorithms
  Reference (type="rfc" data="rfc5480"): RFC 5480: Elliptic Curve Cr\
yptography Subject Public Key Information

Record:
  Name: x25519
  Description: The asymmetric algorithm using a x.25519 Curve
  Status:
  Reference (type="rfc" data="rfc7748"): RFC 7748: Elliptic Curves f\
or Security

Record:
```

```
     Name: x448
     Description: The asymmetric algorithm using a x.448 Curve
     Status:
     Reference (type="rfc" data="rfc7748"): RFC 7748: Elliptic Curves f\
   or Security
```

4.1.4. The "Hash Algorithms" Sub-Registry

The "Hash Algorithms" sub-registry enumerates hashing algorithms used by cryptographic ciphers and protocols.

The format of this registry is described in the Introduction (Section 4.1.1) section above.

Following is the initial assignment for this sub-registry:

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

   Record:
     Name: sha1
     Description: The SHA1 algorithm
     Status: Obsolete
     Reference (type="rfc" data="rfc3174"): RFC 3174: US Secure Hash Al\
   gorithms 1 (SHA1)

   Record:
     Name: sha-224
     Description: The SHA-224 algorithm
     Status:
     Reference (type="rfc" data="rfc6234"): RFC 6234: US Secure Hash Al\
   gorithms

   Record:
     Name: sha-256
     Description: The SHA-256 algorithm
     Status:
     Reference (type="rfc" data="rfc6234"): RFC 6234: US Secure Hash Al\
   gorithms

   Record:
     Name: sha-384
     Description: The SHA-384 algorithm
     Status:
     Reference (type="rfc" data="rfc6234"): RFC 6234: US Secure Hash Al\
   gorithms

   Record:
```

```
     Name: sha-512
     Description: The SHA-512 algorithm
     Status:
     Reference (type="rfc" data="rfc6234"): RFC 6234: US Secure Hash Al\
   gorithms

   Record:
     Name: shake-128
     Description: The SHA3 algorithm with 128-bits output
     Status:
     Reference (type="text"): National Institute of Standards and Techn\
   ology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output \
   Functions, FIPS PUB 202, DOI 10.6028/NIST.FIPS.202, August 2015

   Record:
     Name: shake-224
     Description: The SHA3 algorithm with 224-bits output
     Status:
     Reference (type="text"): National Institute of Standards and Techn\
   ology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output \
   Functions, FIPS PUB 202, DOI 10.6028/NIST.FIPS.202, August 2015

   Record:
     Name: shake-256
     Description: The SHA3 algorithm with 256-bits output
     Status:
     Reference (type="text"): National Institute of Standards and Techn\
   ology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output \
   Functions, FIPS PUB 202, DOI 10.6028/NIST.FIPS.202, August 2015

   Record:
     Name: shake-384
     Description: The SHA3 algorithm with 384-bits output
     Status:
     Reference (type="text"): National Institute of Standards and Techn\
   ology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output \
   Functions, FIPS PUB 202, DOI 10.6028/NIST.FIPS.202, August 2015

   Record:
     Name: shake-512
     Description: The SHA3 algorithm with 512-bits output
     Status:
     Reference (type="text"): National Institute of Standards and Techn\
   ology, SHA-3 Standard: Permutation-Based Hash and Extendable-Output \
   Functions, FIPS PUB 202, DOI 10.6028/NIST.FIPS.202, August 2015
```

4.2. IANA-maintained YANG Modules

FIXME: this section needs elaboration!

Any time one of the "Primitive" registries defined in Section 4.1 is modified, IANA must:

Run the TBD script defined in TBD to generate the corresponding YANG module.

Publish the corresponding YANG module using the TBD process.

Sample resulting YANG modules are provided in Appendix A.

4.3. Update the "Secure Shell (SSH) Protocol Parameters" Registry

This section updates the "Secure Shell (SSH) Protocol Parameters" registry located at https://www.iana.org/assignments/ssh-parameters/ssh-parameters.xhtml, following the guidelines specified in Section 5.2 in [RFC5226].

The Secure Shell (SSH) Protocol Parameters registry is composed of a number of sub-registries. The update described in this section modifies only a subset of the sub-registries, as described in the subsections contained herein.

The modification includes both adding a new column to the sub-registry and initialing the new column's values for existing registrations.

The process to add the new column is the same for each subregistry and hence described only once here below.

How to initialize the new column's values for existing registrations is specific to each subregistry and hence specified in the subsections.

4.3.1. Common Update to Specified Sub-Registries

Add a new column called "Primitives" placed at the left-most position in the table.

This column must contain one or more primitive algorithms used by the given registration.

Each primitive algorithm must be listed in the "Cryptographic Primitives" registry defined in Section 4.1.

While unbounded, the number of primitive algorithms listed is never expected to be more than a few.

4.3.2. The "Public Key Algorithm Names" Sub-Registry

Public Key Algorithm Name	Primitives
ssh-dss	dss, sha1
ssh-rsa	rsa, sha1
rsa-sha2-256	rsa, sha-256
rsa-sha2-512	rsa,
spki-sign-rsa	rsa
spki-sign-dss	dss
pgp-sign-rsa	rsa
pgp-sign-dss	dss
null	N/A
ecdsa-sha2-*	
x509v3-ssh-dss	dss
x509v3-ssh-rsa	rsa
x509v3-rsa2048-sha256	rsa
x509v3-ecdsa-sha2-*	
ssh-ed25519	x25519
ssh-ed448	x448

4.4. Update the "Transport Layer Security (TLS) Parameters" Registry

This section updates the "Update the "Transport Layer Security (TLS) Parameters" registry located at https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml, following the guidelines specified in Section 5.2 in [RFC5226].

The Update the "Transport Layer Security (TLS) Parameters registry is composed of a number of sub-registries. The update described in this section modifies only a subset of the sub-registries, as described in the subsections contained herein.

The modification includes both adding a new column to the sub-registry and initialing the new column's values for existing registrations.

The process to add the new column is the same for each subregistry and hence described only once here below.

How to initialize the new column's values for existing registrations is specific to each subregistry and hence specified in the subsections.

4.4.1. Common Update to Specified Sub-Registries

Add a new column called "Primitives" placed at the left-most position in the table.

This column must contain one or more primitive algorithms used by the given registration.

Each primitive algorithm must be listed in the "Cryptographic Primitives" registry defined in Section 4.1.

While unbounded, the number of primitive algorithms listed is never expected to be more than a few.

4.4.2. The "TLS Supported Groups" Sub-Registry

Any unspecified row should have the Primitive value "N/A".

Value	Description	Primitives
1	sect163k1	FIXME
2	sect163r1	FIXME?
3	sect163r2	FIXME?
4	sect193r1	FIXME?
5	sect193r2	FIXME?
6	sect233k1	FIXME?
7	sect233r1	FIXME?
8	sect239k1	FIXME?
9	sect283k1	FIXME?
10	sect283r1	FIXME?
11	sect409k1	FIXME?
12	sect409r1	FIXME?
13	sect571k1	FIXME?
14	sect571r1	FIXME?
15	secp160k1	FIXME?
16	secp160r1	FIXME?
17	secp160r2	FIXME?
18	secp192k1	FIXME?
19	secp192r1	secp192r1
20	secp224k1	FIXME?
21	secp224r1	secp224r1
22	secp256k1	FIXME?
23	secp256r1	secp256r1
24	secp384r1	secp384r1
25	secp521r1	secp521r1
26	brainpoolP256r1	FIXME?
27	brainpoolP384r1	FIXME?
28	brainpoolP512r1	FIXME?
29	x25519	x25519
30	x448	x448
31	brainpoolP256r1tls13	FIXME?
32	brainpoolP384r1tls13	FIXME?
33	brainpoolP512r1tls13	FIXME?
256	ffdhe2048	FIXME?
257	ffdhe3072	FIXME?
258	ffdhe4096	FIXME?
259	ffdhe6144	FIXME?
260	ffdhe8192	FIXME?

4.4.3. The "TLS SignatureAlgorithm" Sub-Registry

Any unspecified row should have the Primitive value "N/A".

Value	Description	Primitives
0	anonymous	FIXME?
1	rsa	rsa
2	dsa	dsa
3	ecdsa	FIXME?
7	ed25519	x25519
8	ed448	x448

4.4.4. The "TLS SignatureScheme" Sub-Registry

Any unspecified row should have the Primitive value "N/A".

Value	Description	Primitives
0x0201	rsa_pkcs1_sha1	rsa
0x0203	ecdsa_sha1	dsa
0x0401	rsa_pkcs1_sha256	rsa
0x0403	ecdsa_secp256r1_sha256	secp256r1
0x0501	rsa_pkcs1_sha384	rsa
0x0503	ecdsa_secp384r1_sha384	secp384r1
0x0601	rsa_pkcs1_sha512	rsa
0x0603	ecdsa_secp521r1_sha512	secp521r1
0x0804	rsa_pss_rsae_sha256	rsa
0x0805	rsa_pss_rsae_sha384	rsa
0x0806	rsa_pss_rsae_sha512	rsa
0x0807	ed25519	x25519
0x0808	ed448	x448
0x0809	rsa_pss_pss_sha256	rsa
0x080A	rsa_pss_pss_sha384	rsa
0x080B	rsa_pss_pss_sha512	rsa
0x081A	ecdsa_brainpoolP256r1tls13_sha256	dsa
0x081B	ecdsa_brainpoolP384r1tls13_sha384	dsa
0x081C	ecdsa_brainpoolP512r1tls13_sha512	dsa

4.5. Update the "IETF XML" Registry

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

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

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

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

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

4.6. Update the "YANG Module Names" Registry

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

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

      name: iana-symmetric-algs
      namespace: urn:ietf:params:xml:ns:yang:iana-symmetric-algs
      prefix: isa
      reference: RFC XXXX

      name: iana-asymmetric-algs
      namespace: urn:ietf:params:xml:ns:yang:iana-asymmetric-algs
      prefix: iasa
      reference: RFC XXXX

      name: iana-hash-algs
      namespace: urn:ietf:params:xml:ns:yang:iana-hash-algs
      prefix: iha
      reference: RFC XXXX

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>.

Appendix A. Sample IANA Modules

This non-normative section presents the YANG modules produced by running the TBD script presented in Section 4.2 over the registries defined in Section 4.1.

A.1. The Symmetric Algorithms Module

```
<CODE BEGINS> file "iana-symmetric-algs@2020-03-08.yang"

module iana-symmetric-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-symmetric-algs";
  prefix isa;

  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>
     Author:   Wang Haiguang <wang.haiguang.shieldlab@huawei.com>";

  description
    "This module defines a typedef for symmetric algorithms, and
     a container for a list of symmetric algorithms supported by
     the server.

     Copyright (c) 2019 IETF Trust and the persons identified
     as authors of the code. All rights reserved.
     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Simplified
     BSD License set forth in Section 4.c of the IETF Trust's
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

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

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

```
     revision 2020-03-08 {
       description
         "Initial version";
       reference
         "RFC XXXX: Common YANG Data Types for Cryptography";
     }

     // Typedefs

     typedef symmetric-algorithm-type {
       type enumeration {
         enum aes-128-cbc {
           value 1;
           description
             "Encrypt message with AES algorithm in CBC mode with
              a key length of 128 bits.";
           reference
             "RFC 3565: Use of the Advanced Encryption Standard (AES)
              Encryption Algorithm in Cryptographic Message Syntax
              (CMS)";
         }
         enum aes-192-cbc {
           value 2;
           description
             "Encrypt message with AES algorithm in CBC mode with
              a key length of 192 bits";
           reference
             "RFC 3565: Use of the Advanced Encryption Standard (AES)
              Encryption Algorithm in Cryptographic Message Syntax
              (CMS)";
         }
         enum aes-256-cbc {
           value 3;
           description
             "Encrypt message with AES algorithm in CBC mode with
              a key length of 256 bits";
           reference
             "RFC 3565: Use of the Advanced Encryption Standard (AES)
              Encryption Algorithm in Cryptographic Message Syntax
              (CMS)";
         }
         enum aes-128-ctr {
           value 4;
           description
             "Encrypt message with AES algorithm in CTR mode with
              a key length of 128 bits";
           reference
             "RFC 3686:
```

```
              Using Advanced Encryption Standard (AES) Counter
              Mode with IPsec Encapsulating Security Payload
              (ESP)";
        }
        enum aes-192-ctr {
          value 5;
          description
            "Encrypt message with AES algorithm in CTR mode with
             a key length of 192 bits";
          reference
            "RFC 3686:
               Using Advanced Encryption Standard (AES) Counter
               Mode with IPsec Encapsulating Security Payload
               (ESP)";
        }
        enum aes-256-ctr {
          value 6;
          description
            "Encrypt message with AES algorithm in CTR mode with
             a key length of 256 bits";
          reference
            "RFC 3686:
               Using Advanced Encryption Standard (AES) Counter
               Mode with IPsec Encapsulating Security Payload
               (ESP)";
        }
        enum des3-cbc-sha1-kd {
          value 7;
          description
            "Encrypt message with 3DES algorithm in CBC mode
             with sha1 function for key derivation";
          reference
            "RFC 3961:
               Encryption and Checksum Specifications for
               Kerberos 5";
        }
        enum rc4-hmac {
          value 8;
          description
            "Encrypt message with rc4 algorithm";
          reference
            "RFC 4757:
               The RC4-HMAC Kerberos Encryption Types Used by
               Microsoft Windows";
        }
        enum rc4-hmac-exp {
          value 9;
          description
```

```
             "Encrypt message with rc4 algorithm that is exportable";
           reference
             "RFC 4757:
                The RC4-HMAC Kerberos Encryption Types Used by
                Microsoft Windows";
         }
       }
       description
         "A typedef enumerating various symmetric key algorithms.";
     }

     // Protocol-accessible Nodes

     container supported-symmetric-algorithms {
       config false;
       description
         "A container for a list of supported symmetric algorithms.
          How algorithms come to be supported is outside the scope
          of this module.";
       list supported-symmetric-algorithm {
         key algorithm;
         description
           "A lists of symmetric algorithms supported by the server.";
         leaf algorithm {
           type symmetric-algorithm-type;
           description
             "An symmetric algorithms supported by the server.";
         }
       }
     }

   }

   <CODE ENDS>
```

A.2. The Asymmetric Algorithms Module

```
   <CODE BEGINS> file "iana-asymmetric-algs@2020-03-08.yang"

   module iana-asymmetric-algs {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:iana-asymmetric-algs";
     prefix iasa;

     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>
        Author:   Wang Haiguang <wang.haiguang.shieldlab@huawei.com>";

     description
       "This module defines a typedef for asymmetric algorithms, and
        a container for a list of asymmetric algorithms supported by
        the server.

        Copyright (c) 2019 IETF Trust and the persons identified
        as authors of the code. All rights reserved.
        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Simplified
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

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

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

     revision 2020-03-08 {
       description
         "Initial version";
       reference
         "RFC XXXX: Common YANG Data Types for Cryptography";
     }

     // Typedefs

     typedef asymmetric-algorithm-type {
       type enumeration {
         enum rsa1024 {
           value 1;
           description
             "The RSA algorithm using a 1024-bit key.";
           reference
             "RFC 8017: PKCS #1: RSA Cryptography
              Specifications Version 2.2.";
```

```
         }
         enum rsa2048 {
           value 2;
           description
             "The RSA algorithm using a 2048-bit key.";
           reference
             "RFC 8017:
              PKCS #1: RSA Cryptography Specifications Version 2.2.";
         }
         enum rsa3072 {
           value 3;
           description
             "The RSA algorithm using a 3072-bit key.";
           reference
             "RFC 8017:
              PKCS #1: RSA Cryptography Specifications Version 2.2.";
         }
         enum rsa4096 {
           value 4;
           description
             "The RSA algorithm using a 4096-bit key.";
           reference
             "RFC 8017:
              PKCS #1: RSA Cryptography Specifications Version 2.2.";
         }
         enum rsa7680 {
           value 5;
           description
             "The RSA algorithm using a 7680-bit key.";
           reference
             "RFC 8017:
              PKCS #1: RSA Cryptography Specifications Version 2.2.";
         }
         enum rsa15360 {
           value 6;
           description
             "The RSA algorithm using a 15360-bit key.";
           reference
             "RFC 8017:
              PKCS #1: RSA Cryptography Specifications Version 2.2.";
         }
         enum secp192r1 {
           value 7;
           description
             "The asymmetric algorithm using a NIST P192 Curve.";
           reference
             "RFC 6090:
                Fundamental Elliptic Curve Cryptography Algorithms.
```

```
          RFC 5480:
             Elliptic Curve Cryptography Subject Public Key
             Information.";
      }
      enum secp224r1 {
        value 8;
        description
          "The asymmetric algorithm using a NIST P224 Curve.";
        reference
          "RFC 6090:
             Fundamental Elliptic Curve Cryptography Algorithms.
           RFC 5480:
             Elliptic Curve Cryptography Subject Public Key
             Information.";
      }
      enum secp256r1 {
        value 9;
        description
          "The asymmetric algorithm using a NIST P256 Curve.";
        reference
          "RFC 6090:
             Fundamental Elliptic Curve Cryptography Algorithms.
           RFC 5480:
             Elliptic Curve Cryptography Subject Public Key
             Information.";
      }
      enum secp384r1 {
        value 10;
        description
          "The asymmetric algorithm using a NIST P384 Curve.";
        reference
          "RFC 6090:
             Fundamental Elliptic Curve Cryptography Algorithms.
           RFC 5480:
             Elliptic Curve Cryptography Subject Public Key
             Information.";
      }
      enum secp521r1 {
        value 11;
        description
          "The asymmetric algorithm using a NIST P521 Curve.";
        reference
          "RFC 6090:
             Fundamental Elliptic Curve Cryptography Algorithms.
           RFC 5480:
             Elliptic Curve Cryptography Subject Public Key
             Information.";
      }
```

```
        enum x25519 {
          value 12;
          description
            "The asymmetric algorithm using a x.25519 Curve.";
          reference
            "RFC 7748:
               Elliptic Curves for Security.";
        }
        enum x448 {
          value 13;
          description
            "The asymmetric algorithm using a x.448 Curve.";
          reference
            "RFC 7748:
               Elliptic Curves for Security.";
        }
      }
      description
        "A typedef enumerating various asymmetric key algorithms.";
    }

    // Protocol-accessible Nodes

    container supported-asymmetric-algorithms {
      config false;
      description
        "A container for a list of supported asymmetric algorithms.
         How algorithms come to be supported is outside the scope
         of this module.";
      list supported-asymmetric-algorithm {
        key algorithm;
        description
          "A lists of asymmetric algorithms supported by the server.";
        leaf algorithm {
          type asymmetric-algorithm-type;
          description
            "An asymmetric algorithms supported by the server.";
        }
      }
    }

  }

  <CODE ENDS>
```

A.3. The Hash Algorithms Module

```
<CODE BEGINS> file "iana-hash-algs@2020-03-08.yang"

module iana-hash-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-hash-algs";
  prefix iha;

  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>
     Author:   Wang Haiguang <wang.haiguang.shieldlab@huawei.com>";

  description
    "This module defines a typedef for hash algorithms, and
     a container for a list of hash algorithms supported by
     the server.

     Copyright (c) 2019 IETF Trust and the persons identified
     as authors of the code. All rights reserved.
     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Simplified
     BSD License set forth in Section 4.c of the IETF Trust's
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).

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

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

  revision 2020-03-08 {
    description
      "Initial version";
    reference
      "RFC XXXX: Common YANG Data Types for Cryptography";
```

```
     }

     // Typedefs

     typedef hash-algorithm-type {
       type enumeration {
         enum sha1 {
           value 1;
           status obsolete;
           description
             "The SHA1 algorithm.";
           reference
             "RFC 3174: US Secure Hash Algorithms 1 (SHA1).";
         }
         enum sha-224 {
           value 2;
           description
             "The SHA-224 algorithm.";
           reference
             "RFC 6234: US Secure Hash Algorithms.";
         }
         enum sha-256 {
           value 3;
           description
             "The SHA-256 algorithm.";
           reference
             "RFC 6234: US Secure Hash Algorithms.";
         }
         enum sha-384 {
           value 4;
           description
             "The SHA-384 algorithm.";
           reference
             "RFC 6234: US Secure Hash Algorithms.";
         }
         enum sha-512 {
           value 5;
           description
             "The SHA-512 algorithm.";
           reference
             "RFC 6234: US Secure Hash Algorithms.";
         }
         enum shake-128 {
           value 6;
           description
             "The SHA3 algorithm with 128-bits output.";
           reference
             "National Institute of Standards and Technology,
```

```
              SHA-3 Standard: Permutation-Based Hash and
              Extendable-Output Functions, FIPS PUB 202, DOI
              10.6028/NIST.FIPS.202, August 2015.";
          }
          enum shake-224 {
            value 7;
            description
              "The SHA3 algorithm with 224-bits output.";
            reference
              "National Institute of Standards and Technology,
               SHA-3 Standard: Permutation-Based Hash and
               Extendable-Output Functions, FIPS PUB 202, DOI
               10.6028/NIST.FIPS.202, August 2015.";
          }
          enum shake-256 {
            value 8;
            description
              "The SHA3 algorithm with 256-bits output.";
            reference
              "National Institute of Standards and Technology,
               SHA-3 Standard: Permutation-Based Hash and
               Extendable-Output Functions, FIPS PUB 202, DOI
               10.6028/NIST.FIPS.202, August 2015.";
          }
          enum shake-384 {
            value 9;
            description
              "The SHA3 algorithm with 384-bits output.";
            reference
              "National Institute of Standards and Technology,
               SHA-3 Standard: Permutation-Based Hash and
               Extendable-Output Functions, FIPS PUB 202, DOI
               10.6028/NIST.FIPS.202, August 2015.";
          }
          enum shake-512 {
            value 10;
            description
              "The SHA3 algorithm with 384-bits output.";
            reference
              "National Institute of Standards and Technology,
               SHA-3 Standard: Permutation-Based Hash and
               Extendable-Output Functions, FIPS PUB 202, DOI
               10.6028/NIST.FIPS.202, August 2015.";
          }
        }
        description
          "A typedef enumerating various hash key algorithms.";
      }
```

```
     // Protocol-accessible Nodes

     container supported-hash-algorithms {
       config false;
       description
         "A container for a list of supported hash algorithms.
          How algorithms come to be supported is outside the
          scope of this module.";
       list supported-hash-algorithm {
         key algorithm;
         description
           "A lists of hash algorithms supported by the server.";
         leaf algorithm {
           type hash-algorithm-type;
           description
             "An hash algorithms supported by the server.";
         }
       }
     }

   }

   <CODE ENDS>
```

Appendix B. Change Log

B.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.

B.2. 00 to 01

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

B.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).

B.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.

B.5. 03 to 04

- ran YANG module through formatter.

B.6. 04 to 05

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

B.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.

B.8. 06 to 07

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

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

B.9. 07 to 08

o Removed the 'generate-key and 'hidden-key' features.

o Added grouping symmetric-key-grouping

o 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.

B.10. 08 to 09

o Converting algorithm from identities to enumerations.

B.11. 09 to 10

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

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

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

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

o Add in des3-cbc-sha1 for IPSec

o Add in sha1-des3-kd for IPSec

o 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.

o Add x25519 and x448 curve for asymmetric algorithms

o Add signature algorithms ed25519, ed25519-cts, ed25519ph

o add signature algorithms ed448, ed448ph

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

B.12. 10 to 11

- Added a "key-format" identity.

- Added symmetric keys to the example in Section 2.3.

B.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.

B.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.

B.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.

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, Balazs Kovacs, Juergen Schoenwaelder, Eric Voit, and Liang Xia.

Authors' Addresses

Kent Watsen
Watsen Networks

EMail: kent+ietf@watsen.net

Wang Haiguang
Huawei

EMail: wang.haiguang.shieldlab@huawei.com