NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019
K. Watsen
Juniper Networks
H. Wang
Huawei
October 22, 2018

Common YANG Data Types for Cryptography
draft-ietf-netconf-crypto-types-02

Abstract

This document defines YANG identities, typedefs, the groupings useful for cryptographic applications.

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

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

o "2018-10-22" --> 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

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 25, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG 1.1 [RFC7950] module specifying identities, typedefs, and groupings useful for cryptography.

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 the groupings as represented, as tree diagrams have no means to represent identities or typedefs.

```
[Note: '\' line wrapping for formatting only]

module: ietf-crypto-types

  grouping asymmetric-key-pair-grouping
    +-- algorithm?             asymmetric-key-encryption-algorithm-r\
ef
    +-- public-key?            binary
    +-- private-key?           union
    +---x generate-hidden-key
    |  +---w input
    |     +---w algorithm    asymmetric-key-encryption-algorithm-ref
    +---x install-hidden-key
       +---w input
          +---w algorithm      asymmetric-key-encryption-algorithm-r\
ef
          +---w public-key?    binary
          +---w private-key?   binary
  grouping public-key-grouping
    +-- algorithm?    asymmetric-key-encryption-algorithm-ref
    +-- public-key?   binary
  grouping asymmetric-key-pair-with-certs-grouping
    +-- algorithm?
    |       asymmetric-key-encryption-algorithm-ref
    +-- public-key?                              binary
    +-- private-key?                             union
    +---x generate-hidden-key
    |  +---w input
```

```
       |     +---w algorithm    asymmetric-key-encryption-algorithm-ref
       +---x install-hidden-key
       |  +---w input
       |     +---w algorithm      asymmetric-key-encryption-algorithm-r\
   ef
       |     +---w public-key?    binary
       |     +---w private-key?   binary
       +-- certificates
       |  +-- certificate* [name]
       |     +-- name?                     string
       |     +-- cert?                     end-entity-cert-cms
       |     +---n certificate-expiration
       |        +-- expiration-date    yang:date-and-time
       +---x generate-certificate-signing-request
          +---w input
          |  +---w subject       binary
          |  +---w attributes?   binary
          +--ro output
             +--ro certificate-signing-request    binary
     grouping end-entity-cert-grouping
       +-- cert?                     end-entity-cert-cms
       +---n certificate-expiration
          +-- expiration-date    yang:date-and-time
     grouping trust-anchor-cert-grouping
       +-- cert?                     trust-anchor-cert-cms
       +---n certificate-expiration
          +-- expiration-date    yang:date-and-time
```

2.2. YANG Module

This module has normative references to [RFC2404], [RFC2986], [RFC3174], [RFC3565], [RFC3686], [RFC4106], [RFC4253], [RFC4279], [RFC4309], [RFC4493], [RFC4494], [RFC4543], [RFC4868], [RFC5280], [RFC5652], [RFC5656], [RFC5915], [RFC6187], [RFC6234], [RFC6239], [RFC6507], [RFC6991], [RFC7539], [RFC7919], [RFC8017], [RFC8032], [RFC8268], [RFC8332], [RFC8341], [RFC8422], [RFC8446], and [ITU.X690.2015].

This module has an informational reference to [RFC6125].

```
<CODE BEGINS> file "ietf-crypto-types@2018-10-22.yang"
module ietf-crypto-types {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-crypto-types";
  prefix "ct";

  import ietf-yang-types {
```

```
       prefix yang;
       reference
         "RFC 6991: Common YANG Data Types";
     }

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

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

     contact
      "WG Web:   <http://datatracker.ietf.org/wg/netconf/>
       WG List:  <mailto:netconf@ietf.org>

       Author:   Kent Watsen
                 <mailto:kwatsen@juniper.net>

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

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

       Copyright (c) 2018 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
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

     revision "2018-10-22" {
       description
        "Initial version";
       reference
        "RFC XXXX: Common YANG Data Types for Cryptography";
     }
```

```
    /*************************************/
    /*   Identities for Hash Algorithms   */
    /*************************************/

    identity hash-algorithm {
      description
        "A base identity for hash algorithm verification.";
    }

    identity sha-224 {
      base "hash-algorithm";
      description "The SHA-224 algorithm.";
      reference "RFC 6234: US Secure Hash Algorithms.";
    }

    identity sha-256 {
      base "hash-algorithm";
      description "The SHA-256 algorithm.";
      reference "RFC 6234: US Secure Hash Algorithms.";
    }

    identity sha-384 {
      base "hash-algorithm";
      description "The SHA-384 algorithm.";
      reference "RFC 6234: US Secure Hash Algorithms.";
    }

    identity sha-512 {
      base "hash-algorithm";
      description "The SHA-512 algorithm.";
      reference "RFC 6234: US Secure Hash Algorithms.";
    }

    /*********************************************************/
    /*  Identities for Asymmetric Key Encyption Algorithms  */
    /*********************************************************/

    identity asymmetric-key-encryption-algorithm {
      description
        "Base identity from which all asymmetric key
         encryption Algorithm.";
    }

    identity rsa1024 {
      base asymmetric-key-encryption-algorithm;
      description
        "The RSA algorithm using a 1024-bit key.";
      reference
```

```
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2.";
    }

    identity rsa2048 {
      base asymmetric-key-encryption-algorithm;
      description
        "The RSA algorithm using a 2048-bit key.";
      reference
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2.";
    }

    identity rsa3072 {
      base asymmetric-key-encryption-algorithm;
      description
        "The RSA algorithm using a 3072-bit key.";
      reference
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2.";
    }

    identity rsa4096 {
      base asymmetric-key-encryption-algorithm;
      description
        "The RSA algorithm using a 4096-bit key.";
      reference
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2.";
    }

    identity rsa7680 {
      base asymmetric-key-encryption-algorithm;
      description
        "The RSA algorithm using a 7680-bit key.";
      reference
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2.";
    }

    identity rsa15360 {
      base asymmetric-key-encryption-algorithm;
         description
        "The RSA algorithm using a 15360-bit key.";
      reference
        "RFC 8017:
           PKCS #1: RSA Cryptography Specifications Version 2.2.";
    }
```

```
   /*************************************/
   /*   Identities for MAC Algorithms   */
   /*************************************/

   identity mac-algorithm {
     description
       "A base identity for mac generation.";
   }

   identity hmac-sha1 {
     base "mac-algorithm";
     description "Generating MAC using SHA1 hash function";
     reference "RFC 3174: US Secure Hash Algorithm 1 (SHA1)";
   }

   identity hmac-sha1-96 {
     base "mac-algorithm";
     description "Generating MAC using SHA1 hash function";
     reference "RFC 2404: The Use of HMAC-SHA-1-96 within ESP and AH";
   }

   identity hmac-sha2-224 {
     base "mac-algorithm";
     description
       "Generating MAC using SHA2 hash function";
     reference
       "RFC 6234:
          US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)";
   }

   identity hmac-sha2-256 {
     base "mac-algorithm";
     description
       "Generating MAC using SHA2 hash function";
     reference
       "RFC 6234:
          US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)";
   }

   identity hmac-sha2-256-128 {
     base "mac-algorithm";
     description
       "Generating a 256 bits MAC using SHA2 hash function and truncate
        it to 128 bits";
     reference
       "RFC 4868:
          Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with
          IPsec";
```

```
    }

    identity hmac-sha2-384 {
      base "mac-algorithm";
      description
        "Generating MAC using SHA2 hash function";
      reference
        "RFC 6234:
           US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)";
    }

    identity hmac-sha2-384-192 {
      base "mac-algorithm";
      description
        "Generating a 384 bits MAC using SHA2 hash function and truncate
         it to 192 bits";
      reference
        "RFC 4868:
           Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with
           IPsec";
    }

    identity hmac-sha2-512 {
      base "mac-algorithm";
      description "Generating MAC using SHA2 hash function";
      reference
        "RFC 6234:
           US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)";
    }

    identity hmac-sha2-512-256 {
      base "mac-algorithm";
      description
        "Generating a 512 bits MAC using SHA2 hash function and
         truncating it to 256 bits";
      reference
        "RFC 4868:
           Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with
           IPsec";
    }

    identity aes-128-gmac {
      base "mac-algorithm";
      description
        "Generating MAC using the Advanced Encryption Standard (AES)
         Galois Message Authentication Code (GMAC) as a mechanism to
         provide data origin authentication";
      reference
```

```
        "RFC 4543:
           The Use of Galois Message Authentication Code (GMAC) in
           IPsec ESP and AH";
    }

    identity aes-192-gmac {
      base "mac-algorithm";
      description
        "Generating MAC using the Advanced Encryption Standard (AES)
         Galois Message Authentication Code (GMAC) as a mechanism to
         provide data origin authentication";
      reference
        "RFC 4543:
           The Use of Galois Message Authentication Code (GMAC) in
           IPsec ESP and AH";

    }

    identity aes-256-gmac {
      base "mac-algorithm";
      description
        "Generating MAC using the Advanced Encryption Standard (AES)
         Galois Message Authentication Code (GMAC) as a mechanism to
         provide data origin authentication";
      reference
        "RFC 4543:
           The Use of Galois Message Authentication Code (GMAC) in
           IPsec ESP and AH";
    }

    identity aes-cmac-96 {
      base "mac-algorithm";
      description
        "Generating MAC using Advanced Encryption Standard (AES)
         Cipher-based Message Authentication Code (CMAC)";
      reference
        "RFC 4494: The AES-CMAC-96 Algorithm and its Use with IPsec";
    }

    identity aes-cmac-128 {
      base "mac-algorithm";
      description
        "Generating MAC using Advanced Encryption Standard (AES)
         Cipher-based Message Authentication Code (CMAC)";
      reference
        "RFC 4493: The AES-CMAC Algorithm";
    }
```

```
    identity mac-aes-128-ccm {
      base "mac-algorithm";
      description
        "Generating MAC using Advanced Encryption Standard (AES) in
         CCM (Counter with CBC-MAC) mode (AES CCM)";
      reference
        "RFC 4309:
           Using Advanced Encryption Standard (AES) CCM Mode with
           IPsec Encapsulating Security Payload (ESP)";
    }

    identity mac-aes-192-ccm {
      base "mac-algorithm";
      description
        "Generating MAC using Advanced Encryption Standard (AES) in
         CCM (Counter with CBC-MAC) mode (AES CCM)";
      reference
        "RFC 4309:
           Using Advanced Encryption Standard (AES) CCM Mode with
           IPsec Encapsulating Security Payload (ESP)";
    }

    identity mac-aes-256-ccm {
      base "mac-algorithm";
      description
        "Generating MAC using Advanced Encryption Standard (AES) in
         CCM (Counter with CBC-MAC) mode (AES CCM)";
      reference
        "RFC 4309:
           Using Advanced Encryption Standard (AES) CCM Mode with
           IPsec Encapsulating Security Payload (ESP)";
    }

    identity mac-aes-128-gcm {
      base "mac-algorithm";
      description
        "Generating MAC when using Advanced Encryption Standard (AES)
         GCM mode for encryption";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
    }

    identity mac-aes-192-gcm {
      base "mac-algorithm";
      description
        "Generating MAC when using Advanced Encryption Standard (AES)
```

```
         GCM mode for encryption";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
    }

    identity mac-aes-256-gcm {
      base "mac-algorithm";
      description
        "Generating MAC when using Advanced Encryption Standard (AES)
         GCM mode for encryption";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
    }

    identity mac-chacha20-poly1305 {
      base "mac-algorithm";
      description
        "Generating MAC using poly1305 algorithm";
      reference
        "RFC 7539: ChaCha20 and Poly1305 for IETF Protocols";
    }


    /*********************************************************/
    /*   Identities for Symmetric Key Encryption Algorithms*/
    /*********************************************************/

    identity symmetric-key-encryption-algorithm {
      description
        "A base identity for encryption algorithm.";
    }

    identity aes-128-cbc {
      base "symmetric-key-encryption-algorithm";
      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)";
    }

    identity aes-192-cbc {
```

```
      base "symmetric-key-encryption-algorithm";
      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)";
    }

    identity aes-256-cbc {
      base "symmetric-key-encryption-algorithm";
      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)";
    }

    identity aes-128-ctr {
      base "symmetric-key-encryption-algorithm";
      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)";
    }

    identity aes-192-ctr {
      base "symmetric-key-encryption-algorithm";
      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)";
    }

    identity aes-256-ctr {
      base "symmetric-key-encryption-algorithm";
      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)";
    }

    identity enc-aes-128-ccm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in CCM mode with a key
         length of 128 bits";
      reference
        "RFC 4309:
           Using Advanced Encryption Standard (AES) CCM Mode with IPsec
           Encapsulating Security Payload (ESP)";
    }

    identity enc-aes-192-ccm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in CCM mode with a key
         length of 192 bits";
      reference
        "RFC 4309:
           Using Advanced Encryption Standard (AES) CCM Mode with IPsec
           Encapsulating Security Payload (ESP)";
    }

    identity enc-aes-256-ccm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in CCM mode with a key
         length of 256 bits";
      reference
        "RFC 4309:
           Using Advanced Encryption Standard (AES) CCM Mode with IPsec
           Encapsulating Security Payload (ESP)";
    }

    identity enc-aes-128-gcm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in GCM mode with a key
         length of 128 bits";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
```

```
    }

    identity enc-aes-192-gcm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in GCM mode with a key
         length of 192 bits";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
    }

    identity enc-aes-256-gcm {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with AES algorithm in GCM mode with a key
         length of 256 bits";
      reference
        "RFC 4106:
           The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating
           Security Payload (ESP)";
    }

    identity enc-chacha20-poly1305 {
      base "symmetric-key-encryption-algorithm";
      description
        "Encrypt message with chacha20 algorithm and generate MAC with
         POLY1305";
      reference
        "RFC 7539: ChaCha20 and Poly1305 for IETF Protocols";
    }

    /*****************************************/
    /*   Identities for signature algorithm   */
    /*****************************************/

    identity signature-algorithm {
      description
        "A base identity for asymmetric key encryption algorithm.";
    }

    identity dsa-sha1 {
      base "signature-algorithm";
      description
        "The signature algorithm using DSA algorithm with SHA1 hash
         algorithm";
      reference
```

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

    identity rsa-pkcs1-sha1 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PKCS1-v1_5 with the SHA1
        hash algorithm.";
      reference
        "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
    }

    identity rsa-pkcs1-sha256 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PKCS1-v1_5 with the
         SHA256 hash algorithm.";
      reference
        "RFC 8332:
           Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell
           (SSH) Protocol
         RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity rsa-pkcs1-sha384 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PKCS1-v1_5 with the
         SHA384 hash algorithm.";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity rsa-pkcs1-sha512 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PKCS1-v1_5 with the
         SHA512 hash algorithm.";
      reference
        "RFC 8332:
           Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell
           (SSH) Protocol
         RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }
```

```
    identity rsa-pss-rsae-sha256 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PSS with mask generation
         function 1 and SHA256 hash algorithm. If the public key is
         carried in an X.509 certificate, it MUST use the rsaEncryption
         OID";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity rsa-pss-rsae-sha384 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PSS with mask generation
         function 1 and SHA384 hash algorithm. If the public key is
         carried in an X.509 certificate, it MUST use the rsaEncryption
         OID";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity rsa-pss-rsae-sha512 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PSS with mask generation
         function 1 and SHA512 hash algorithm. If the public key is
         carried in an X.509 certificate, it MUST use the rsaEncryption
         OID";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity rsa-pss-pss-sha256 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PSS with mask generation
         function 1 and SHA256 hash algorithm. If the public key is
         carried in an X.509 certificate, it MUST use the RSASSA-PSS
         OID";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }
```

```
    identity rsa-pss-pss-sha384 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PSS with mask generation
         function 1 and SHA256 hash algorithm. If the public key is
         carried in an X.509 certificate, it MUST use the RSASSA-PSS
         OID";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity rsa-pss-pss-sha512 {
      base "signature-algorithm";
      description
        "The signature algorithm using RSASSA-PSS with mask generation
         function 1 and SHA256 hash algorithm. If the public key is
         carried in an X.509 certificate, it MUST use the RSASSA-PSS
         OID";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity ecdsa-secp256r1-sha256 {
      base "signature-algorithm";
      description
        "The signature algorithm using ECDSA wtih curve name secp256r1
         and SHA256 hash algorithm.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer
         RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity ecdsa-secp384r1-sha384 {
      base "signature-algorithm";
      description
        "The signature algorithm using ECDSA wtih curve name secp384r1
         and SHA384 hash algorithm.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer
         RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }
```

```
    identity ecdsa-secp521r1-sha512 {
      base "signature-algorithm";
      description
        "The signature algorithm using ECDSA wtih curve name secp521r1
         and SHA512 hash algorithm.";
      reference
        "RFC 5656: Elliptic Curve Algorithm Integration in the
           Secure Shell Transport Layer
         RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity x509v3-rsa-pkcs1-sha1 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-ssh-rsa key format and
         RSASSA-PKCS1-v1_5 with the SHA1 hash algorithm.";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-rsa2048-pkcs1-sha256 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-rsa2048-sha256
         key format and RSASSA-PKCS1-v1_5 with the SHA-256
         hash algorithm.";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-secp256r1-sha256 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-ecdsa-sha2-secp256r1 key
         format and ECDSA algorithm with the SHA-256 hash algorithm.";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-secp384r1-sha384 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-ecdsa-sha2-secp384r1 key
         format and ECDSA algorithm with the SHA-384 hash algorithm.";
```

```
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity x509v3-ecdsa-secp521r1-sha512 {
      base "signature-algorithm";
      description
        "The signature algorithm using x509v3-ecdsa-sha2-secp521r1 key
         format and ECDSA algorithm with the SHA-512 hash algorithm.";
      reference
        "RFC 6187:
           X.509v3 Certificates for Secure Shell Authentication";
    }

    identity ed25519 {
      base "signature-algorithm";
      description
        "The signature algorithm using EdDSA as defined in RFC 8032 or
         its successors.";
      reference
        "RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA)";
    }

    identity ed448 {
      base "signature-algorithm";
      description
        "The signature algorithm using EdDSA as defined in RFC 8032 or
         its successors.";
      reference
        "RFC 8032: Edwards-Curve Digital Signature Algorithm (EdDSA)";
    }

    identity eccsi {
      base "signature-algorithm";
      description
        "The signature algorithm using ECCSI signature as defined in
         RFC 6507.";
      reference
        "RFC 6507:
           Elliptic Curve-Based Certificateless Signatures for
           Identity-based Encryption (ECCSI)";
    }

    /**********************************************/
    /*   Identities for key exchange algorithms   */
    /**********************************************/
```

```
  identity key-exchange-algorithm {
    description
      "A base identity for Diffe-Hellman based key exchange
       algorithm.";
  }

  identity psk-only {
    base "key-exchange-algorithm";
    description
      "Using Pre-shared key for authentication and key exhange";
    reference
      "RFC 4279:
         Pre-Shared Key Ciphersuites for Transport Layer Security
        (TLS)";
  }

  identity dhe-ffdhe2048 {
    base "key-exchange-algorithm";
    description
      "Ephemeral Diffie Hellman key exhange with 2048 bit
       finite field";
    reference
      "RFC 7919:
         Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
         for Transport Layer Security (TLS)";
  }

  identity dhe-ffdhe3072 {
    base "key-exchange-algorithm";
    description
      "Ephemeral Diffie Hellman key exhange with 3072 bit finite
       field";
    reference
      "RFC 7919:
         Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
         for Transport Layer Security (TLS)";
  }

  identity dhe-ffdhe4096 {
    base "key-exchange-algorithm";
    description
      "Ephemeral Diffie Hellman key exhange with 4096 bit
       finite field";
    reference
      "RFC 7919:
         Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
         for Transport Layer Security (TLS)";
  }
```

```
    identity dhe-ffdhe6144 {
      base "key-exchange-algorithm";
      description
        "Ephemeral Diffie Hellman key exhange with 6144 bit
         finite field";
      reference
        "RFC 7919:
           Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
           for Transport Layer Security (TLS)";
    }

    identity dhe-ffdhe8192 {
      base "key-exchange-algorithm";
      description
        "Ephemeral Diffie Hellman key exhange with 8192 bit
         finite field";
      reference
        "RFC 7919:
           Negotiated Finite Field Diffie-Hellman Ephemeral Parameters
           for Transport Layer Security (TLS)";
    }

    identity psk-dhe-ffdhe2048 {
      base "key-exchange-algorithm";
      description
        "Key exchange using pre-shared key with Diffie-Hellman key
         generation mechansim, where the DH group is FFDHE2048";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity psk-dhe-ffdhe3072 {
      base "key-exchange-algorithm";
      description
        "Key exchange using pre-shared key with Diffie-Hellman key
         generation mechansim, where the DH group is FFDHE3072";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity psk-dhe-ffdhe4096 {
      base "key-exchange-algorithm";
      description
        "Key exchange using pre-shared key with Diffie-Hellman key
         generation mechansim, where the DH group is FFDHE4096";
      reference
```

```
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity psk-dhe-ffdhe6144 {
      base "key-exchange-algorithm";
      description
        "Key exchange using pre-shared key with Diffie-Hellman key
         generation mechansim, where the DH group is FFDHE6144";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity psk-dhe-ffdhe8192 {
      base "key-exchange-algorithm";
      description
        "Key exchange using pre-shared key with Diffie-Hellman key
         generation mechansim, where the DH group is FFDHE8192";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity ecdhe-secp256r1 {
      base "key-exchange-algorithm";
      description
        "Ephemeral Diffie Hellman key exhange with elliptic group
         over curve secp256r1";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity ecdhe-secp384r1 {
      base "key-exchange-algorithm";
      description
        "Ephemeral Diffie Hellman key exhange with elliptic group
         over curve secp384r1";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity ecdhe-secp521r1 {
      base "key-exchange-algorithm";
```

```
      description
        "Ephemeral Diffie Hellman key exhange with elliptic group
         over curve secp521r1";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity ecdhe-x25519 {
      base "key-exchange-algorithm";
      description
        "Ephemeral Diffie Hellman key exhange with elliptic group
         over curve x25519";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity ecdhe-x448 {
      base "key-exchange-algorithm";
      description
        "Ephemeral Diffie Hellman key exhange with elliptic group
         over curve x448";
      reference
        "RFC 8422:
           Elliptic Curve Cryptography (ECC) Cipher Suites for
           Transport Layer Security (TLS) Versions 1.2 and Earlier";
    }

    identity psk-ecdhe-secp256r1 {
      base "key-exchange-algorithm";
      description
        "Key exchange using pre-shared key with elliptic group-based
         Ephemeral Diffie Hellman key exhange over curve secp256r1";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    identity psk-ecdhe-secp384r1 {
      base "key-exchange-algorithm";
      description
        "Key exchange using pre-shared key with elliptic group-based
         Ephemeral Diffie Hellman key exhange over curve secp384r1";
      reference
        "RFC 8446:
```

```
           The Transport Layer Security (TLS) Protocol Version 1.3";
     }

     identity psk-ecdhe-secp521r1 {
       base "key-exchange-algorithm";
       description
         "Key exchange using pre-shared key with elliptic group-based
          Ephemeral Diffie Hellman key exhange over curve secp521r1";
       reference
         "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
     }

     identity psk-ecdhe-x25519 {
       base "key-exchange-algorithm";
       description
         "Key exchange using pre-shared key with elliptic group-based
          Ephemeral Diffie Hellman key exhange over curve x25519";
       reference
         "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
     }

     identity psk-ecdhe-x448 {
       base "key-exchange-algorithm";
       description
         "Key exchange using pre-shared key with elliptic group-based
          Ephemeral Diffie Hellman key exhange over curve x448";
       reference
         "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
     }

     identity diffie-hellman-group14-sha1 {
       base "key-exchange-algorithm";
       description
         "Using DH group14 and SHA1 for key exchange";
       reference
         "RFC 4253: The Secure Shell (SSH) Transport Layer Protocol";
     }

    identity diffie-hellman-group14-sha256 {
       base "key-exchange-algorithm";
       description
         "Using DH group14 and SHA256 for key exchange";
       reference
         "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
```

```
           Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

    identity diffie-hellman-group15-sha512 {
      base "key-exchange-algorithm";
      description
        "Using DH group15 and SHA512 for key exchange";
      reference
        "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
           Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

    identity diffie-hellman-group16-sha512 {
      base "key-exchange-algorithm";
      description
        "Using DH group16 and SHA512 for key exchange";
      reference
        "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
           Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

    identity diffie-hellman-group17-sha512 {
      base "key-exchange-algorithm";
      description
        "Using DH group17 and SHA512 for key exchange";
      reference
        "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
           Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

    identity diffie-hellman-group18-sha512 {
      base "key-exchange-algorithm";
      description
        "Using DH group18 and SHA512 for key exchange";
      reference
        "RFC 8268:
           More Modular Exponentiation (MODP) Diffie-Hellman (DH)
           Key Exchange (KEX) Groups for Secure Shell (SSH)";
    }

    identity ecdh-sha2-secp256r1 {
      base "key-exchange-algorithm";
      description
        "Elliptic curve-based Diffie Hellman key exhange over curve
         secp256r1 and using SHA2 for MAC generation";
```

```
      reference
        "RFC 6239: Suite B Cryptographic Suites for Secure Shell (SSH)";
    }

    identity ecdh-sha2-secp384r1 {
      base "key-exchange-algorithm";
      description
        "Elliptic curve-based Diffie Hellman key exhange over curve
         secp384r1 and using SHA2 for MAC generation";
      reference
        "RFC 6239: Suite B Cryptographic Suites for Secure Shell (SSH)";
    }

    /********************************************************/
    /*   Typedefs for identityrefs to above base identites   */
    /********************************************************/

    typedef hash-algorithm-ref {
      type identityref {
        base "hash-algorithm";
      }
      description
        "This typedef enables importing modules to easily define an
         identityref to the 'hash-algorithm' base identity.";
    }

    typedef signature-algorithm-ref {
      type identityref {
        base "signature-algorithm";
      }
      description
        "This typedef enables importing modules to easily define an
         identityref to the 'signature-algorithm' base identity.";
    }

    typedef mac-algorithm-ref {
      type identityref {
        base "mac-algorithm";
      }
      description
        "This typedef enables importing modules to easily define an
         identityref to the 'mac-algorithm' base identity.";
    }

    typedef symmetric-key-encryption-algorithm-ref {
      type identityref {
        base "symmetric-key-encryption-algorithm";
      }
```

```
      description
        "This typedef enables importing modules to easily define an
         identityref to the 'symmetric-key-encryption-algorithm'
         base identity.";
    }

    typedef asymmetric-key-encryption-algorithm-ref {
      type identityref {
        base "asymmetric-key-encryption-algorithm";
      }
      description
        "This typedef enables importing modules to easily define an
         identityref to the 'asymmetric-key-encryption-algorithm'
         base identity.";
    }

    typedef key-exchange-algorithm-ref {
      type identityref {
        base "key-exchange-algorithm";
      }
      description
        "This typedef enables importing modules to easily define an
         identityref to the 'key-exchange-algorithm' base identity.";
    }

    /***************************************************/
    /*   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 structures related to RFC 4253   */
```

```
    /*****************************************************/

    typedef ssh-host-key {
      type binary;
      description
        "The binary public key data for this SSH 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";
    }

    /***********************************************************/
    /*   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 public-key-grouping {
  description
    "A public key.";
  leaf algorithm {
    type asymmetric-key-encryption-algorithm-ref;
    description
      "Identifies the key's algorithm.  More specifically,
       this leaf specifies how the 'public-key' binary leaf
       is encoded.";
    reference
      "RFC CCCC: Common YANG Data Types for Cryptography";
  }
  leaf public-key {
    type binary;
    description
      "A binary that contains the value of the public key.  The
       interpretation of the content is defined by the key
       algorithm.  For example, a DSA key is an integer, an RSA
       key is represented as RSAPublicKey as defined in
       RFC 8017, and an Elliptic Curve Cryptography (ECC) key
       is represented using the 'publicKey' described in
       RFC 5915.";
    reference
      "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
                 RSA Cryptography Specifications Version 2.2.
       RFC 5915: Elliptic Curve Private Key Structure.";
  }
} // end public-key-grouping


grouping asymmetric-key-pair-grouping {
  description
    "A private/public key pair.";
  uses public-key-grouping;
  leaf private-key {
    nacm:default-deny-all;
    type union {
      type binary;
      type enumeration {
        enum "permanently-hidden" {
          description
           "The private key is inaccessible due to being
            protected by the system (e.g., a cryptographic
            hardware module).  It is not possible to
```

```
                configure a permanently hidden key, as a real
                private key value must be set.  Permanently
                hidden keys cannot be archived or backed up.";
            }
          }
        }
        description
          "A binary that contains the value of the private key.  The
           interpretation of the content is defined by the key
           algorithm.  For example, a DSA key is an integer, an RSA
           key is represented as RSAPrivateKey as defined in
           RFC 8017, and an Elliptic Curve Cryptography (ECC) key
           is represented as ECPrivateKey as defined in RFC 5915.";
        reference
          "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
                     RSA Cryptography Specifications Version 2.2.
           RFC 5915: Elliptic Curve Private Key Structure.";
      } // end private-key

      action generate-hidden-key {
        description
          "Requests the device to generate a hidden key using the
           specified asymmetric key algorithm.  This action is
           used to request the system to generate a key that
           is 'permanently-hidden', perhaps protected by a
           cryptographic hardware module.  The resulting
           asymmetric key values are considered operational
           state and hence present only in <operational>.";
        input {
          leaf algorithm {
            type asymmetric-key-encryption-algorithm-ref;
            mandatory true;
            description
              "The algorithm to be used when generating the
               asymmetric key.";
            reference
              "RFC CCCC: Common YANG Data Types for Cryptography";
          }
        }
      } // end generate-hidden-key

      action install-hidden-key {
        description
          "Requests the device to load the specified values into
           a hidden key.  The resulting asymmetric key values are
           considered operational state and hence present only in
           <operational>.";
        input {
```

```
          leaf algorithm {
            type asymmetric-key-encryption-algorithm-ref;
            mandatory true;
            description
              "The algorithm to be used when generating the
               asymmetric key.";
            reference
              "RFC CCCC: Common YANG Data Types for Cryptography";
          }
          leaf public-key {
            type binary;
            description
              "A binary that contains the value of the public key.
               The interpretation of the content is defined by the key
               algorithm.  For example, a DSA key is an integer, an
               RSA key is represented as RSAPublicKey as defined in
               RFC 8017, and an Elliptic Curve Cryptography (ECC) key
               is represented using the 'publicKey' described in
               RFC 5915.";
            reference
              "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
                         RSA Cryptography Specifications Version 2.2.
               RFC 5915: Elliptic Curve Private Key Structure.";
          }
          leaf private-key {
            type binary;
            description
              "A binary that contains the value of the private key.
               The interpretation of the content is defined by the key
               algorithm.  For example, a DSA key is an integer, an RSA
               key is represented as RSAPrivateKey as defined in
               RFC 8017, and an Elliptic Curve Cryptography (ECC) key
               is represented as ECPrivateKey as defined in RFC 5915.";
            reference
              "RFC 8017: Public-Key Cryptography Standards (PKCS) #1:
                         RSA Cryptography Specifications Version 2.2.
               RFC 5915: Elliptic Curve Private Key Structure.";
          }
        }
      } // end install-hidden-key
    } // end asymmetric-key-pair-grouping


    grouping trust-anchor-cert-grouping {
      description
       "A certificate, and a notification for when it might expire.";
      leaf cert {
        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.";
        }
      }
    } // end trust-anchor-cert-grouping


    grouping end-entity-cert-grouping {
      description
       "A certificate, and a notification for when it might expire.";
      leaf cert {
        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.";
        }
      }
```

```
    } // end end-entity-cert-grouping


    grouping asymmetric-key-pair-with-certs-grouping {
      description
        "A private/public key pair and associated certificates.";
      uses asymmetric-key-pair-grouping;
      container certificates {
        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;
        } // end certificate
      } // end certificates

      action generate-certificate-signing-request {
        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.";
        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).";

        }
```

```
        }
      } // end generate-certificate-signing-request
    } // end asymmetric-key-pair-with-certs-grouping


  }
  <CODE ENDS>
```

3. Security Considerations

In order to use YANG identities for algorithm identifiers, only the most commonly used RSA key lengths are supported for the RSA algorithm. Additional key lengths can be defined in another module or added into a future version of this document.

This document limits the number of elliptical curves supported. This was done to match industry trends and IETF best practice (e.g., matching work being done in TLS 1.3). If additional algorithms are needed, they can be defined by another module or added into a future version of this document.

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

generate-certificate-signing-request: 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.

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, placing a "choice" statement in both the input and output statements, along with an "if-feature" statement on the CRMF option, would enable a backwards compatible solution.

NACM:default-deny-all is set on asymmetric-key-pair-grouping's "private-key" node, as private keys should never be revealed without explicit permission.

4. IANA Considerations

4.1. The IETF XML Registry

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

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

4.2. The YANG Module Names Registry

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

```
name:         ietf-crypto-types
namespace:    urn:ietf:params:xml:ns:yang:ietf-crypto-types
prefix:       ct
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>.

[RFC2404] Madson, C. and R. Glenn, "The Use of HMAC-SHA-1-96 within ESP and AH", RFC 2404, DOI 10.17487/RFC2404, November 1998, <https://www.rfc-editor.org/info/rfc2404>.

[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>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <https://www.rfc-editor.org/info/rfc3174>.

[RFC3565] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, DOI 10.17487/RFC3565, July 2003, <https://www.rfc-editor.org/info/rfc3565>.

[RFC3686] Housley, R., "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)", RFC 3686, DOI 10.17487/RFC3686, January 2004, <https://www.rfc-editor.org/info/rfc3686>.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, DOI 10.17487/RFC4106, June 2005, <https://www.rfc-editor.org/info/rfc4106>.

[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>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <https://www.rfc-editor.org/info/rfc4279>.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, DOI 10.17487/RFC4309, December 2005, <https://www.rfc-editor.org/info/rfc4309>.

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006, <https://www.rfc-editor.org/info/rfc4493>.

[RFC4494] Song, JH., Poovendran, R., and J. Lee, "The AES-CMAC-96 Algorithm and Its Use with IPsec", RFC 4494, DOI 10.17487/RFC4494, June 2006, <https://www.rfc-editor.org/info/rfc4494>.

[RFC4543] McGrew, D. and J. Viega, "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, DOI 10.17487/RFC4543, May 2006, <https://www.rfc-editor.org/info/rfc4543>.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <https://www.rfc-editor.org/info/rfc4868>.

[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>.

[RFC5656] Stebila, D. and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer", RFC 5656, DOI 10.17487/RFC5656, December 2009, <https://www.rfc-editor.org/info/rfc5656>.

[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>.

[RFC6187] Igoe, K. and D. Stebila, "X.509v3 Certificates for Secure Shell Authentication", RFC 6187, DOI 10.17487/RFC6187, March 2011, <https://www.rfc-editor.org/info/rfc6187>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC6239] Igoe, K., "Suite B Cryptographic Suites for Secure Shell (SSH)", RFC 6239, DOI 10.17487/RFC6239, May 2011, <https://www.rfc-editor.org/info/rfc6239>.

[RFC6507] Groves, M., "Elliptic Curve-Based Certificateless Signatures for Identity-Based Encryption (ECCSI)", RFC 6507, DOI 10.17487/RFC6507, February 2012, <https://www.rfc-editor.org/info/rfc6507>.

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

[RFC7539] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 7539, DOI 10.17487/RFC7539, May 2015, <https://www.rfc-editor.org/info/rfc7539>.

[RFC7919] Gillmor, D., "Negotiated Finite Field Diffie-Hellman Ephemeral Parameters for Transport Layer Security (TLS)", RFC 7919, DOI 10.17487/RFC7919, August 2016, <https://www.rfc-editor.org/info/rfc7919>.

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

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

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[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>.

[RFC8268] Baushke, M., "More Modular Exponentiation (MODP) Diffie-Hellman (DH) Key Exchange (KEX) Groups for Secure Shell (SSH)", RFC 8268, DOI 10.17487/RFC8268, December 2017, <https://www.rfc-editor.org/info/rfc8268>.

[RFC8332] Bider, D., "Use of RSA Keys with SHA-256 and SHA-512 in the Secure Shell (SSH) Protocol", RFC 8332, DOI 10.17487/RFC8332, March 2018, <https://www.rfc-editor.org/info/rfc8332>.

[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>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/info/rfc8422>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

5.2. Informative References

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

[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>.

[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>.

[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. Examples

A.1. The "asymmetric-key-pair-with-certs-grouping" Grouping

The following example module has been constructed to illustrate use of the "asymmetric-key-pair-with-certs-grouping" grouping defined in the "ietf-crypto-types" module.

Note that the "asymmetric-key-pair-with-certs-grouping" grouping uses both the "asymmetric-key-pair-grouping" and "end-entity-cert-grouping" groupings, and that the "asymmetric-key-pair-grouping" grouping uses the "public-key-grouping" grouping. Thus, a total of four of the five groupings defined in the "ietf-crypto-types" module are illustrated through the use of this one grouping. The only grouping not represented is the "trust-anchor-cert-grouping" grouping.

```
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 keys {
    description
      "A container of keys.";
    list 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.

```
<keys xmlns="http://example.com/ns/example-crypto-types-usage">
  <key>
    <name>ex-key</name>
    <algorithm
      xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
      ct:rsa2048
    </algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
    <certificates>
      <certificate>
        <name>ex-cert</name>
        <cert>base64encodedvalue==</cert>
      </certificate>
    </certificates>
  </key>
</keys>
```

A.2. The "generate-hidden-key" Action

The following example illustrates the "generate-hidden-key" action in use 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">
    <keys xmlns="http://example.com/ns/example-crypto-types-usage">
      <key>
        <name>empty-key</name>
        <generate-hidden-key>
          <algorithm
           xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
              ct:rsa2048
          </algorithm>
        </generate-hidden-key>
      </key>
    </keys>
  </action>
</rpc>

RESPONSE
--------
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

A.3. The "install-hidden-key" Action

The following example illustrates the "install-hidden-key" action in use 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">
    <keys xmlns="http://example.com/ns/example-crypto-types-usage">
      <key>
        <name>empty-key</name>
        <install-hidden-key>
          <algorithm
           xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
              ct:rsa2048
          </algorithm>
          <public-key>base64encodedvalue==</public-key>
          <private-key>base64encodedvalue==</private-key>
        </install-hidden-key>
      </key>
    </keys>
  </action>
</rpc>

RESPONSE
--------
<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

A.4. The "generate-certificate-signing-request" Action

The following example illustrates the "generate-certificate-signing-request" action in use 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">
    <keys xmlns="http://example.com/ns/example-crypto-types-usage">
      <key>
        <name>ex-key-sect571r1</name>
        <generate-certificate-signing-request>
          <subject>base64encodedvalue==</subject>
          <attributes>base64encodedvalue==</attributes>
        </generate-certificate-signing-request>
      </key>
    </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>
```

A.5. The "certificate-expiration" Notification

The following example illustrates the "certificate-expiration" notification in use 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>
```


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).

Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Martin Bjorklund, Balazs Kovacs, Eric Voit, and Liang Xia.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Wang Haiguang
Huawei

EMail: wang.haiguang.shieldlab@huawei.com