NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 24, 2019

K. Watsen
Juniper Networks

September 20, 2018

Common YANG Data Types for Cryptography
draft-ietf-netconf-crypto-types-01

Abstract

This document defines YANG identities and typedefs 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:

- "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:

- "2018-09-20" --> the publication date of this draft

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

- Appendix A. Change Log

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

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

This Internet-Draft will expire on March 24, 2019.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG 1.1 [RFC7950] module specifying identities and typedefs 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 diagrams [RFC8340] for the "ietf-crypto-types" module that, in addition to typedefs and identities, also presents groupings intended for external usage.

```
   module: ietf-crypto-types

     grouping asymmetric-key-pair-grouping
       +-- algorithm?              ct:key-algorithm-ref
       +-- public-key?             binary
       +-- private-key?            union
       +---x generate-hidden-key
       |  +---w input
       |     +---w algorithm    ct:key-algorithm-ref
       +---x install-hidden-key
          +---w input
             +---w algorithm      ct:key-algorithm-ref
             +---w public-key?    binary
             +---w private-key?   binary
     grouping public-key-grouping
       +-- algorithm?    ct:key-algorithm-ref
       +-- public-key?   binary
     grouping asymmetric-key-pair-with-certs-grouping
       +-- algorithm?                                ct:key-algorithm-ref
       +-- public-key?                               binary
       +-- private-key?                              union
       +---x generate-hidden-key
       |  +---w input
       |     +---w algorithm    ct:key-algorithm-ref
       +---x install-hidden-key
       |  +---w input
       |     +---w algorithm      ct:key-algorithm-ref
       |     +---w public-key?    binary
       |     +---w private-key?   binary
```

```
  +-- certificates
  |  +-- certificate* [name]
  |     +-- name?                      string
  |     +-- cert                       ct: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                       ct:end-entity-cert-cms
  +---n certificate-expiration
     +-- expiration-date?   yang:date-and-time
grouping trust-anchor-cert-grouping
  +-- cert    ct:trust-anchor-cert-cms
```


2.2. YANG Module

This module has normative references to [RFC4253], [RFC5280], [RFC5480], [RFC5652], [RFC6991], [RFC8341] and [ITU.X690.2015].

This module has informational references to [RFC2986], [RFC5915], [RFC6125], [RFC6234], and [RFC8017]

<CODE BEGINS> file "ietf-crypto-types@2018-09-20.yang"

```
module ietf-crypto-types {
  yang-version 1.1;

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

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

  import ietf-netconf-acm {
```

```
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

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

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

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

  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 "2018-09-20" {
    description
     "Initial version";
    reference
     "RFC XXXX: Common YANG Data Types for Cryptography";
  }


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

```
/*   Identities for Hashing Algorithms   */
/*****************************************/

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

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

/*************************************/
/*   Identities for Key Algorithms   */
/*************************************/

identity key-algorithm {
  description
    "Base identity from which all key-algorithms are derived.";
}

identity rsa1024 {
  base key-algorithm;
  description
    "The RSA algorithm using a 1024-bit key.";
  reference
    "RFC 8017:
       PKCS #1: RSA Cryptography Specifications Version 2.2.";
}

identity rsa2048 {
```



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

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

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

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

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

identity secp192r1 {
  base key-algorithm;
  description
    "The secp192r1 algorithm.";
```



```
  reference
```

```
      "RFC 5480: Elliptic Curve Cryptography Subject Public
                 Key Information.";
  }

  identity secp256r1 {
    base key-algorithm;
    description
      "The secp256r1 algorithm.";
    reference
      "RFC 5480: Elliptic Curve Cryptography Subject Public
                 Key Information.";
  }

  identity secp384r1 {
    base key-algorithm;
    description
      "The secp384r1 algorithm.";
    reference
      "RFC 5480: Elliptic Curve Cryptography Subject Public
                 Key Information.";
  }

  identity secp521r1 {
    base key-algorithm;
    description
      "The secp521r1 algorithm.";
    reference
      "RFC 5480: Elliptic Curve Cryptography Subject Public
                 Key Information.";
  }


  /*****************************************************/
  /*   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 key-algorithm-ref {
    type identityref {
```

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



```
      }
      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 the 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 ct:key-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 ct:key-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 ct:trust-anchor-cert-cms;
        mandatory true;
        description
         "The binary certificate data for this certificate.";
        reference
         "RFC YYYY: Common YANG Data Types for Cryptography";
      }
    } // end trust-anchor-cert-grouping


    grouping end-entity-cert-grouping {
      description
       "A certificate, and a notification for when it might expire.";
      leaf cert {
        type ct:end-entity-cert-cms;
        mandatory true;
        description
         "The binary certificate data for this certificate.";
        reference
```

```
      }
      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 {
          must "../../algorithm
                 and ../../public-key
                   and ../../private-key";
          key name;
          description
            "A certificate for this asymmetric key.";
          leaf name {
            type string;
            description
              "An arbitrary name for the certificate.";
```

```
          }
          uses end-entity-cert-grouping;
        } // 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 Syntaxi
                           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.

The YANG module defined in this document specifies only typedefs and identities, and hence there are no YANG-specific security considerations that need to be addressed.

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>.

[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>.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, DOI 10.17487/RFC5480, March 2009, <https://www.rfc-editor.org/info/rfc5480>.

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

[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>.

[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>.

[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>.

[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>.

[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>locally-defined key</name>
    <algorithm
      xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
      ct:secp521r1
    </algorithm>
    <private-key>base64encodedvalue==</private-key>
    <public-key>base64encodedvalue==</public-key>
  </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:secp521r1
          </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:secp521r1
          </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.

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.

Author's Address

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net