NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 6, 2018

K. Watsen
Juniper Networks

March 5, 2018

Common YANG Data Types for Cryptography
draft-kwatsen-netconf-crypto-types-00

Abstract

This document defines a YANG identities, typedefs, and groupings useful for when working with ASN.1 structures, algorithms, and private keys.

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:

- "2018-03-05" --> 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 September 6, 2018.

Table of Contents

1. Introduction 3
1.1. Requirements Language 3
1.2. Tree Diagram Notation 3
2. Tree Diagram 3
3. Examples 4
3.1. Private Key and Associated Certificate Configuration . . 4
3.2. Certificate Signing Request Action 5
3.3. Generate Private Key Action 6
3.4. Certificate Expiration Notification 6
4. YANG Module 7
5. Security Considerations 16
6. IANA Considerations 16
6.1. The IETF XML Registry 17
6.2. The YANG Module Names Registry 17
7. Acknowledgements 17
8. References 17
8.1. Normative References 17
8.2. Informative References 19
Appendix A. Example YANG Module 20
Appendix B. Change Log 21
Author's Address 21

1. Introduction

This document defines a YANG identities, typedefs, and groupings useful for when working with ASN.1 structures, algorithms, and private keys.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.2. Tree Diagram Notation

Tree diagrams used in this document follow the notation defined in [I-D.ietf-netmod-yang-tree-diagrams].

2. Tree Diagram

The following tree diagram provides an overview of the groupings, actions, and notifications defined in the ietf-crypto-types YANG module. The YANG module defines many identities and typedefs that are not represented by this tree diagram.

```
module: ietf-crypto-types

  notifications:
    +---n certificate-expiration
       +--ro certificate        instance-identifier
       +--ro expiration-date    yang:date-and-time

  grouping certificates-grouping
    +---- certificates
    |  +---- certificate* [name]
    |     +---- name?    string
    |     +---- value?   binary
    +---x generate-certificate-signing-request
       +---w input
       |  +---w subject       binary
       |  +---w attributes?   binary
       +--ro output
          +--ro certificate-signing-request    binary
  grouping private-key-grouping
    +---- algorithm?              identityref
    +---- private-key?            union
    +---- public-key?             binary
    +---x generate-private-key
       +---w input
          +---w algorithm    identityref
```

3. Examples

These examples illustrate the use of the groupings, actions, and notifications defined in the ietf-crypto-types YANG module. The YANG module defines many identities and typedefs that are not represented that are not represented by these examples.

3.1. Private Key and Associated Certificate Configuration

The following example illustrates a configured private key along with an associated certificate. This example uses the "ex-crypto-types-usage" module defined in Appendix A.

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

  <key xmlns="http://example.com/ns/example-crypto-types-usage">
    <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>
    <certificates>
      <certificate>
        <name>domain certificate</name>
        <value>base64encodedvalue==</value>
      </certificate>
    </certificates>
   </key>
```

3.2. Certificate Signing Request Action

The following example illustrates the "generate-certificate-signing-request" action in use with the NETCONF protocol. This example uses the "ex-crypto-types-usage" module defined in Appendix A.

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

3.3. Generate Private Key Action

The following example illustrates the "generate-private-key" action in use with the NETCONF protocol. This example uses the "ex-crypto-types-usage" module defined in Appendix A.

```
REQUEST
-------
[ note: '\' line wrapping for formatting only]

<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0\
">
  <action xmlns="urn:ietf:params:xml:ns:yang:1">
    <key xmlns="http://example.com/ns/example-crypto-types-usage">
      <generate-private-key>
        <algorithm xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto\
-types">ct:secp521r1</algorithm>
      </generate-private-key>
    </key>
  </action>
</rpc>

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

3.4. Certificate Expiration Notification

The following example illustrates the "certificate-expiration" notification in use with the NETCONF protocol. This example uses the "ex-crypto-types-usage" module defined in Appendix A.

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

   <notification
     xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
     <eventTime>2016-07-08T00:01:00Z</eventTime>
     <certificate-expiration
       xmlns="urn:ietf:params:xml:ns:yang:ietf-crypto-types">
       <certificate xmlns:ex="http://example.com/ns/example-crypto-type\
   s-usage">
         /ex:key/ex:certificates/ex:certificate[ex:name='domain certifi\
   cate']
       </certificate>
       <expiration-date>2016-08-08T14:18:53-05:00</expiration-date>
     </certificate-expiration>
   </notification>
```

4. YANG Module

This YANG module imports modules defined in [RFC6536] and [RFC6991]. This module uses data types defined in [RFC2315], [RFC2986], [RFC3447], [RFC4253], [RFC5280], [RFC5915], and [ITU.X690.1994]. This module uses algorithms defined in [RFC3447] and [RFC5480].

```
<CODE BEGINS> file "ietf-crypto-types@2018-03-05.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 6536: Network Configuration Protocol (NETCONF) Access
       Control Model";
  }

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

  contact
```

```
   "WG Web:   <http://tools.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 cryptography 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 "2018-03-05" {
    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.

       This identity is used in the ietf-zerotouch-information
       module (draft-ietf-netconf-zerotouch)";
  }

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

```
  /*****************************************************************/
  /*   Identities for Key Algorithms (used by Certificates)   */
  /*****************************************************************/

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

       This identity is used in the 'private-key-grouping' grouping
       and the 'generate-private-key' action below.";
  }

  identity rsa1024 {
    base key-algorithm;
    description
      "The RSA algorithm using a 1024-bit key.";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
                RSA Cryptography Specifications Version 2.1.";
  }

  identity rsa2048 {
    base key-algorithm;
    description
      "The RSA algorithm using a 2048-bit key.";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
                RSA Cryptography Specifications Version 2.1.";
  }

  identity rsa3072 {
    base key-algorithm;
    description
      "The RSA algorithm using a 3072-bit key.";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
                RSA Cryptography Specifications Version 2.1.";
  }

  identity rsa4096 {
    base key-algorithm;
    description
      "The RSA algorithm using a 4096-bit key.";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
                RSA Cryptography Specifications Version 2.1.";
  }
```

```
  identity rsa7680 {
    base key-algorithm;
    description
      "The RSA algorithm using a 7680-bit key.";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
                RSA Cryptography Specifications Version 2.1.";
  }

  identity rsa15360 {
    base key-algorithm;
    description
      "The RSA algorithm using a 15360-bit key.";
    reference
      "RFC3447: Public-Key Cryptography Standards (PKCS) #1:
                RSA Cryptography Specifications Version 2.1.";
  }

  identity secp192r1 {
    base key-algorithm;
    description
      "The secp192r1 algorithm.";
    reference
      "RFC5480:
         Elliptic Curve Cryptography Subject Public Key Information.";
  }

  identity secp256r1 {
    base key-algorithm;
    description
      "The secp256r1 algorithm.";
    reference
      "RFC5480:
         Elliptic Curve Cryptography Subject Public Key Information.";
  }

  identity secp384r1 {
    base key-algorithm;
    description
      "The secp384r1 algorithm.";
    reference
      "RFC5480:
         Elliptic Curve Cryptography Subject Public Key Information.";
  }

  identity secp521r1 {
    base key-algorithm;
    description
```

```
      "The secp521r1 algorithm.";
    reference
      "RFC5480:
         Elliptic Curve Cryptography Subject Public Key Information.";
  }


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

  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 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 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).";
  }



  /*****************************************************************/
  /*   Groupings for a Private Key and its Associated Certificates  */
```

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

  grouping private-key-grouping {
    description
      "A private/public key pair, and an action to request the
       system to generate a private key.

       This grouping is currently used by the YANG modules
       ietf-ssh-client, ietf-ssh-server, ietf-tls-client,
       and ietf-tls-server, where it populates the SSH/TLS
       peer object's private key parameters.";
    leaf algorithm {
      type identityref {
        base "key-algorithm";
      }
      description
        "Identifies the key's algorithm.  More specifically, this
         leaf specifies how the 'private-key' and 'public-key'
         binary leafs are encoded.";
    }
    leaf private-key {
      nacm:default-deny-all;
      type union {
        type binary;
        type enumeration {
          enum "hardware-protected" {
            description
             "The private key is inaccessible due to being
              protected by a cryptographic hardware module
              (e.g., a TPM).";
          }
        }
      }
      must "../algorithm";
      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
         [RFC3447], and an Elliptic Curve Cryptography (ECC) key
         is represented as ECPrivateKey as defined in [RFC5915]";
      reference
        "RFC 3447: Public-Key Cryptography Standards (PKCS) #1:
                   RSA Cryptography Specifications Version 2.1.
         RFC 5915: Elliptic Curve Private Key Structure.";
    }
    leaf public-key {
      type binary;
```

```
      must "../algorithm";
      must "../private-key";
      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
         [RFC3447], and an Elliptic Curve Cryptography (ECC) key
         is represented using the 'publicKey' described in
         [RFC5915]";
      reference
        "RFC 3447: Public-Key Cryptography Standards (PKCS) #1:
                   RSA Cryptography Specifications Version 2.1.
         RFC 5915: Elliptic Curve Private Key Structure.";
    }
    action generate-private-key {
      description
        "Requests the device to generate a private key using the
         specified key algorithm.  This action is primarily to
         support cryptographic processors that must generate
         the private key themselves.  The resulting key is
         considered operational state and hence only present
         in <operational>.";
      input {
        leaf algorithm {
          type identityref {
            base "key-algorithm";
          }
          mandatory true;
          description
            "The algorithm to be used when generating the key.";
        }
      }
    } // end generate-private-key
  }



  grouping certificates-grouping {
    description
      "A container of certificates, and an action to generate
       a certificate signing request.

       This grouping is currently used by the YANG modules
       ietf-ssh-client, ietf-ssh-server, ietf-tls-client,
       and ietf-tls-server, where it populates the SSH/TLS
       peer object's value for a certificates associated
       with the private key.";
```

```
    container certificates {
      description
        "Certificates associated with this key.  More than one
         certificate supports, for instance, a TPM-protected
         key that has both IDevID and LDevID certificates
         associated.";
      list certificate {
        key name;
        description
          "A certificate for this private key.";
        leaf name {
          type string;
          description
            "An arbitrary name for the certificate.";
        }
        leaf value {
          type binary;
          description
           "A PKCS #7 SignedData structure, as specified by
            Section 9.1 in RFC 2315, containing just certificates
            (no content, signatures, or CRLs), encoded using ASN.1
            distinguished encoding rules (DER), as specified in
            ITU-T X.690.

            This structure contains the certificate itself as well
            as any intermediate certificates leading up to a trust
            anchor certificate.  The trust anchor certificate MAY
            be included as well.";
          reference
            "RFC 2315:
               PKCS #7: Cryptographic Message Syntax Version 1.5.
             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).";
        }
      }
    }
    action generate-certificate-signing-request {
      description
        "Generates a certificate signing request structure for
         the associated private 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 from 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 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).";
        }
      }
      output {
        leaf certificate-signing-request {
          type binary;
          mandatory true;
          description
            "A CertificationRequest 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).";

        }
      }
    }
  }


  notification certificate-expiration {
    description
      "A notification indicating that a 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.";
    leaf certificate {
      type instance-identifier;
      mandatory true;
      description
        "Identifies which certificate is expiring or is expired.";
    }
    leaf expiration-date {
      type yang:date-and-time;
      mandatory true;
      description
        "Identifies the expiration date on the certificate.";
    }
  }

}
<CODE ENDS>
```

5. Security Considerations

TBD

6. IANA Considerations

6.1. The IETF XML Registry

This document registers one URI in 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.
```

6.2. 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
```

7. Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name):

8. References

8.1. Normative References

[ITU.X690.1994]
International Telecommunications 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, 1994.

[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>.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, DOI 10.17487/RFC2315, March 1998, <https://www.rfc-editor.org/info/rfc2315>.

[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>.

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

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

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

[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>.

[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>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <https://www.rfc-editor.org/info/rfc6536>.

[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>.

8.2. Informative References

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-06 (work in progress), February 2018.

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

[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>.

Appendix A. Example YANG Module

The following example YANG module has been constructed to illustrate the groupings defined in the "ietf-crypto-types" module. This YANG module is uses as a basis for the protocol examples in Section 3.

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

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

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

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

  contact
   "WG Web:   <http://tools.ietf.org/wg/netconf/>
    WG List:  <mailto:netconf@ietf.org>
    Author:   Kent Watsen <mailto:kwatsen@juniper.net>";

  description
   "This module illustrates using groupings defined in the YANG
    module ietf-crypto-types.";

  revision "YYYY-MM-DD" {
    description
     "Initial version";
  }

  container key {
    uses ct:private-key-grouping;
    uses ct:certificates-grouping;
    description
      "A container of certificates, and an action to generate
       a certificate signing request.";
  }
}
```

Appendix B. Change Log

TBD

Author's Address

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net