

Workgroup: NETCONF Working Group

Internet-Draft: draft-ietf-netconf-sztp-csr-14

Updates: [8572](#) (if approved)

Published: 2 March 2022

Intended Status: Standards Track

Expires: 3 September 2022

Authors: K. Watsen                      R. Housley                      S. Turner  
          Watsen Networks      Vigil Security, LLC      sn3rd

**Conveying a Certificate Signing Request (CSR) in a Secure Zero Touch  
Provisioning (SZTP) Bootstrapping Request**

**Abstract**

This draft extends the input to the "get-bootstrapping-data" RPC defined in RFC 8572 to include an optional certificate signing request (CSR), enabling a bootstrapping device to additionally obtain an identity certificate (e.g., an LDevID from IEEE 802.1AR) as part of the "onboarding information" response provided in the RPC-reply.

**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 numerical RFC value for this draft

\*AAAA --> the assigned RFC value for I-D.ietf-netconf-crypto-types

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

\*2022-03-02 --> the publication date of this draft

This document contains references to other drafts in progress, both in the Normative References section, as well as in body text throughout. Please update the following references to reflect their final RFC assignments:

\*I-D.ietf-netconf-crypto-types

\*I-D.ietf-netconf-keystore

## 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 3 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. [Introduction](#)
  - 1.1. [Overview](#)
  - 1.2. [Terminology](#)
  - 1.3. [Requirements Language](#)
  - 1.4. [Conventions](#)
2. [The "ietf-sztp-csr" Module](#)
  - 2.1. [Data Model Overview](#)
  - 2.2. [Example Usage](#)
  - 2.3. [YANG Module](#)
3. [The "ietf-ztp-types" Module](#)
  - 3.1. [Data Model Overview](#)
  - 3.2. [YANG Module](#)

- 4. [Security Considerations](#)
  - 4.1. [SZTP-Client Considerations](#)
    - 4.1.1. [Ensuring the Integrity of Asymmetric Private Keys](#)
    - 4.1.2. [Reuse of a Manufacturer-generated Private Key](#)
    - 4.1.3. [Replay Attack Protection](#)
    - 4.1.4. [Connecting to an Untrusted Bootstrap Server](#)
    - 4.1.5. [Selecting the Best Origin Authentication Mechanism](#)
    - 4.1.6. [Clearing the Private Key and Associated Certificate](#)
  - 4.2. [SZTP-Server Considerations](#)
    - 4.2.1. [Verifying Proof of Possession](#)
    - 4.2.2. [Verifying Proof of Origin](#)
    - 4.2.3. [Supporting SZTP-Clients that don't trust the SZTP-Server](#)
  - 4.3. [Security Considerations for the "ietf-sztp-csr" YANG Module](#)
  - 4.4. [Security Considerations for the "ietf-ztp-types" YANG Module](#)
- 5. [IANA Considerations](#)
  - 5.1. [The "IETF XML" Registry](#)
  - 5.2. [The "YANG Module Names" Registry](#)
- 6. [References](#)
  - 6.1. [Normative References](#)
  - 6.2. [Informative References](#)
- [Acknowledgements](#)
- [Contributors](#)
- [Authors' Addresses](#)

## 1. Introduction

### 1.1. Overview

This draft extends the input to the "get-bootstrapping-data" RPC defined in [[RFC8572](#)] to include an optional certificate signing request (CSR) [[RFC2986](#)], enabling a bootstrapping device to additionally obtain an identity certificate (e.g., an LDevID [[Std-802.1AR-2018](#)]) as part of the "onboarding information" response provided in the RPC-reply.

The ability to provision an identity certificate that is purpose-built for a production environment during the bootstrapping process removes reliance on the manufacturer CA, and it also enables the bootstrapped device to join the production environment with an appropriate identity and other attributes in its identity certificate (e.g., an LDevID).

Two YANG [[RFC7950](#)] modules are defined. The "ietf-ztp-types" module defines three YANG groupings for the various messages defined in this document. The "ietf-sztp-csr" module augments two groupings into the "get-bootstrapping-data" RPC and defines a YANG Data Structure [[RFC8791](#)] around the third grouping.

## 1.2. Terminology

This document uses the following terms from [\[RFC8572\]](#):

- \*Bootstrap Server
- \*Bootstrapping Data
- \*Conveyed Information
- \*Device
- \*Manufacturer
- \*Onboarding Information
- \*Signed Data

This document defines the following new terms:

**SZTP-client** The term "SZTP-client" refers to a "device" that is using a "bootstrap server" as a source of "bootstrapping data".

**SZTP-server** The term "SZTP-server" is an alternative term for "bootstrap server" that is symmetric with the "SZTP-client" term.

## 1.3. 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.

## 1.4. Conventions

Various examples used in this document use a placeholder value for binary data that has been base64 encoded (e.g., "BASE64VALUE="). This placeholder value is used as real base64 encoded structures are often many lines long and hence distracting to the example being presented.

## 2. The "ietf-sztp-csr" Module

The "ietf-sztp-csr" module is a YANG 1.1 [\[RFC7950\]](#) module that augments the "ietf-sztp-bootstrap-server" module defined in [\[RFC8572\]](#) and defines a YANG "structure" that is to be conveyed in the "error-info" node defined in [Section 7.1](#) of [\[RFC8040\]](#).

### 2.1. Data Model Overview

The following tree diagram [\[RFC8340\]](#) illustrates the "ietf-sztp-csr" module.

```
module: ietf-sztp-csr
```

```
augment /sztp-svr:get-bootstrapping-data/sztp-svr:input:
```

```
+---w (msg-type)?
  +--:(csr-support)
    | +---w csr-support
    |   +---w key-generation!
    |   | +---w supported-algorithms
    |   |   +---w algorithm-identifier*   binary
    |   +---w csr-generation
    |     +---w supported-formats
    |       +---w format-identifier*   identityref
  +--:(csr)
    +---w (csr-type)
    +--:(p10-csr)
    | +---w p10-csr?   ct:csr
    +--:(cmc-csr)
    | +---w cmc-csr?   binary
    +--:(cmp-csr)
    +---w cmp-csr?   binary
```

```
structure csr-request:
```

```
+-- key-generation!
| +-- selected-algorithm
|   +-- algorithm-identifier   binary
+-- csr-generation
| +-- selected-format
|   +-- format-identifier   identityref
+-- cert-req-info?   ct:csr-info
```

The augmentation defines two kinds of parameters that an SZTP-client can send to an SZTP-server. The YANG structure defines one collection of parameters that an SZTP-server can send to an SZTP-client.

In the order of their intended use:

\*The "csr-support" node is used by the SZTP-client to signal to the SZTP-server that it supports the ability to generate CSRs. This parameter conveys if the SZTP-client is able to generate a new asymmetric key and, if so, which key algorithms it supports, as well as conveys what kinds of CSR structures the SZTP-client is able to generate.

\*The "csr-request" structure is used by the SZTP-server to request the SZTP-client to generate a CSR. This structure is used to select the key algorithm the SZTP-client should use to generate a new asymmetric key, if supported, the kind of CSR structure the

SZTP-client should generate and, optionally, the content for the CSR itself.

\*The various "csr" nodes are used by the SZTP-client to communicate a CSR to the SZTP-server.

No data model is defined enabling an SZTP-server to communicate the signed certificate to the SZTP-client. How to do this is discussed in [Section 2.2](#).

To further illustrate how the augmentation and structure defined by the "ietf-sztp-csr" module are used, below are two additional tree diagrams showing these nodes placed where they are used.

The following tree diagram [[RFC8340](#)] illustrates SZTP's "get-bootstrapping-data" RPC with the augmentation in place.

===== NOTE: '\' line wrapping per RFC 8792 =====

module: ietf-sztp-bootstrap-server

```
rpcs:
  +---x get-bootstrapping-data
    +---w input
      | +---w signed-data-preferred?      empty
      | +---w hw-model?                   string
      | +---w os-name?                    string
      | +---w os-version?                 string
      | +---w nonce?                     binary
      | +---w (sztp-csr:msg-type)?
      |   +---:(sztp-csr:csr-support)
      |     | +---w sztp-csr:csr-support
      |     |   +---w sztp-csr:key-generation!
      |     |     | +---w sztp-csr:supported-algorithms
      |     |     |   +---w sztp-csr:algorithm-identifier*  bina\
ry
      |     |       +---w sztp-csr:csr-generation
      |     |       +---w sztp-csr:supported-formats
      |     |       +---w sztp-csr:format-identifier*  identit\
yref
      |   +---:(sztp-csr:csr)
      |     +---w (sztp-csr:csr-type)
      |       +---:(sztp-csr:p10-csr)
      |         | +---w sztp-csr:p10-csr?  ct:csr
      |         +---:(sztp-csr:cmc-csr)
      |           | +---w sztp-csr:cmc-csr?  binary
      |           +---:(sztp-csr:cmp-csr)
      |             +---w sztp-csr:cmp-csr?  binary
      +---ro output
        +---ro reporting-level?  enumeration {onboarding-server}?
        +---ro conveyed-information  cms
        +---ro owner-certificate?    cms
        +---ro ownership-voucher?    cms
```

The following tree diagram [[RFC8340](#)] illustrates RESTCONF's "errors" RPC-reply message with the "csr-request" structure in place.

```

module: ietf-restconf
+--ro errors
  +--ro error* []
    +--ro error-type      enumeration
    +--ro error-tag       string
    +--ro error-app-tag?  string
    +--ro error-path?     instance-identifier
    +--ro error-message?  string
    +--ro error-info
      +--ro sztp-csr:csr-request
        +--ro sztp-csr:key-generation!
          | +--ro sztp-csr:selected-algorithm
          |   +--ro sztp-csr:algorithm-identifier  binary
        +--ro sztp-csr:csr-generation
          | +--ro sztp-csr:selected-format
          |   +--ro sztp-csr:format-identifier  identityref
        +--ro sztp-csr:cert-req-info?  ct:csr-info

```

## 2.2. Example Usage

The examples below are encoded using JSON, but they could equally well be encoded using XML, as is supported by SZTP.

An SZTP-client implementing this specification would signal to the bootstrap server its willingness to generate a CSR by including the "csr-support" node in its "get-bootstrapping-data" RPC. In the example below, the SZTP-client additionally indicates that it is able to generate keys and provides a list of key algorithms it supports, as well as provide a list of certificate formats it supports.

REQUEST



===== NOTE: '\\' line wrapping per RFC 8792 =====

POST /restconf/operations/ietf-sztp-bootstrap-server:get-bootstrappi\ng-data HTTP/1.1  
HOST: example.com  
Content-Type: application/yang-data+json

```
{
  "ietf-sztp-bootstrap-server:input" : {
    "hw-model": "model-x",
    "os-name": "vendor-os",
    "os-version": "17.3R2.1",
    "nonce": "extralongbase64encodedvalue=",
    "ietf-sztp-csr:csr-support": {
      "key-generation": {
        "supported-algorithms": {
          "algorithm-identifier": [
            "BASE64VALUE1",
            "BASE64VALUE2",
            "BASE64VALUE3"
          ]
        }
      },
      "csr-generation": {
        "supported-formats": {
          "format-identifier": [
            "ietf-ztp-types:p10-csr",
            "ietf-ztp-types:cmc-csr",
            "ietf-ztp-types:cmp-csr"
          ]
        }
      }
    }
  }
}
```

Assuming the SZTP-server wishes to prompt the SZTP-client to provide a CSR, then it would respond with an HTTP 400 Bad Request error code. In the example below, the SZTP-server specifies that it wishes the SZTP-client to generate a key using a specific algorithm and generate a PKCS#10-based CSR containing specific content.

RESPONSE

HTTP/1.1 400 Bad Request  
Date: Sat, 31 Oct 2021 17:02:40 GMT  
Server: example-server  
Content-Type: application/yang-data+json

```
{
  "ietf-restconf:errors" : {
    "error" : [
      {
        "error-type": "application",
        "error-tag": "missing-attribute",
        "error-message": "Missing input parameter",
        "error-info": {
          "ietf-sztp-csr:csr-request": {
            "key-generation": {
              "selected-algorithm": {
                "algorithm-identifier": "BASE64VALUE="
              }
            },
            "csr-generation": {
              "selected-format": {
                "format-identifier": "ietf-ztp-types:p10-csr"
              }
            },
            "cert-req-info": "BASE64VALUE="
          }
        }
      }
    ]
  }
}
```

Upon being prompted to provide a CSR, the SZTP-client would POST another "get-bootstrapping-data" request, but this time including one of the "csr" nodes to convey its CSR to the SZTP-server:

REQUEST

===== NOTE: '\\' line wrapping per RFC 8792 =====

```
POST /restconf/operations/ietf-sztp-bootstrap-server:get-bootstrappi\
ng-data HTTP/1.1
HOST: example.com
Content-Type: application/yang-data+json
```

```
{
  "ietf-sztp-bootstrap-server:input" : {
    "hw-model": "model-x",
    "os-name": "vendor-os",
    "os-version": "17.3R2.1",
    "nonce": "extralongbase64encodedvalue=",
    "ietf-sztp-csr:p10-csr": "BASE64VALUE="
  }
}
```

At this point, it is expected that the SZTP-server, perhaps in conjunction with other systems, such as a backend CA or RA, will validate the CSR's origin and proof-of-possession and, assuming the CSR is approved, issue a signed certificate for the bootstrapping device.

The SZTP-server responds with "onboarding-information" (encoded inside the "conveyed-information" node, shown below) containing a signed identity certificate for the CSR provided by the SZTP-client:

RESPONSE

```
HTTP/1.1 200 OK
Date: Sat, 31 Oct 2021 17:02:40 GMT
Server: example-server
Content-Type: application/yang-data+json
```

```
{
  "ietf-sztp-bootstrap-server:output" : {
    "reporting-level": "verbose",
    "conveyed-information": "BASE64VALUE="
  }
}
```

How the signed certificate is conveyed inside the onboarding information is outside the scope of this document. Some implementations may choose to convey it inside a script (e.g., SZTP's "pre-configuration-script"), while other implementations may choose to convey it inside the SZTP "configuration" node. SZTP onboarding information is described in [Section 2.2](#) of [\[RFC8572\]](#).

Below are two examples of conveying the signed certificate inside the "configuration" node. Both examples assume that the SZTP-client

understands the "ietf-keystore" module defined in [[I-D.ietf-netconf-keystore](#)].

This first example illustrates the case where the signed certificate is for the same asymmetric key used by the SZTP-client's manufacturer-generated identity certificate (e.g., an IDevID, from [[Std-802.1AR-2018](#)]). As such, the configuration needs to associate the newly signed certificate with the existing asymmetric key:

===== NOTE: '\\' line wrapping per RFC 8792 =====

```
{
  "ietf-keystore:keystore": {
    "asymmetric-keys": {
      "asymmetric-key": [
        {
          "name": "Manufacturer-Generated Hidden Key",
          "public-key-format": "ietf-crypto-types:subject-public-key\
-info-format",
          "public-key": "BASE64VALUE=",
          "hidden-private-key": [null],
          "certificates": {
            "certificate": [
              {
                "name": "Manufacturer-Generated IDevID Cert",
                "cert-data": "BASE64VALUE="
              },
              {
                "name": "Newly-Generated LDevID Cert",
                "cert-data": "BASE64VALUE="
              }
            ]
          }
        }
      ]
    }
  }
}
```

This second example illustrates the case where the signed certificate is for a newly generated asymmetric key. As such, the configuration needs to associate the newly signed certificate with the newly generated asymmetric key:

===== NOTE: '\' line wrapping per RFC 8792 =====

```
{
  "ietf-keystore:keystore": {
    "asymmetric-keys": {
      "asymmetric-key": [
        {
          "name": "Manufacturer-Generated Hidden Key",
          "public-key-format": "ietf-crypto-types:subject-public-key\
-info-format",
          "public-key": "BASE64VALUE=",
          "hidden-private-key": [null],
          "certificates": {
            "certificate": [
              {
                "name": "Manufacturer-Generated IDevID Cert",
                "cert-data": "BASE64VALUE="
              }
            ]
          }
        },
        {
          "name": "Newly-Generated Hidden Key",
          "public-key-format": "ietf-crypto-types:subject-public-key\
-info-format",
          "public-key": "BASE64VALUE=",
          "hidden-private-key": [null],
          "certificates": {
            "certificate": [
              {
                "name": "Newly-Generated LDevID Cert",
                "cert-data": "BASE64VALUE="
              }
            ]
          }
        }
      ]
    }
  }
}
```

In addition to configuring the signed certificate, it is often necessary to also configure the Issuer's signing certificate so that the device (i.e., STZP-client) can authenticate certificates presented by peer devices signed by the same issuer as its own. While outside the scope of this document, one way to do this would be to use the "ietf-truststore" module defined in [[I-D.ietf-netconf-trust-anchors](#)].

### 2.3. YANG Module

This module augments an RPC defined in [[RFC8572](#)]. The module uses a data types and groupings defined in [[RFC8572](#)], [[RFC8791](#)], and [[I-D.ietf-netconf-crypto-types](#)]. The module also has an informative reference to [[Std-802.1AR-2018](#)].

```
<CODE BEGINS> file "ietf-sztp-csr@2022-03-02.yang"
```

```

module ietf-sztp-csr {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-sztp-csr";
  prefix sztp-csr;

  import ietf-sztp-bootstrap-server {
    prefix sztp-svr;
    reference
      "RFC 8572: Secure Zero Touch Provisioning (SZTP)";
  }

  import ietf-yang-structure-ext {
    prefix sx;
    reference
      "RFC 8791: YANG Data Structure Extensions";
  }

  import ietf-ztp-types {
    prefix zt;
    reference
      "RFC XXXX: Conveying a Certificate Signing Request (CSR)
      in a Secure Zero Touch Provisioning (SZTP)
      Bootstrapping Request";
  }

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

  contact
    "WG Web:  https://datatracker.ietf.org/wg/netconf
    WG List:  NETCONF WG list <mailto:netconf@ietf.org>
    Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
              Russ Housley <mailto:housley@vigilsec.com>
              Sean Turner <mailto:sean@sn3rd.com>";

```

#### description

"This module augments the 'get-bootstrapping-data' RPC, defined in the 'ietf-sztp-bootstrap-server' module from SZTP (RFC 8572), enabling the SZTP-client to obtain a signed identity certificate (e.g., an LDevID from IEEE 802.1AR) as part of the SZTP onboarding information response.

Copyright (c) 2022 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 Revised BSD License set forth in Section 4.c of the IETF Trust's

Legal Provisions Relating to IETF Documents  
(<https://trustee.ietf.org/license-info>).

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

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

```
revision 2022-03-02 {
  description
    "Initial version";
  reference
    "RFC XXXX: Conveying a Certificate Signing Request (CSR)
      in a Secure Zero Touch Provisioning (SZTP)
      Bootstrapping Request";
}

// Protocol-accessible nodes

augment "/sztp-svr:get-bootstrapping-data/sztp-svr:input" {
  description
    "This augmentation adds the 'csr-support' and 'csr' nodes to
    the SZTP (RFC 8572) 'get-bootstrapping-data' request message,
    enabling the SZTP-client to obtain an identity certificate
    (e.g., an LDevID from IEEE 802.1AR) as part of the onboarding
    information response provided by the SZTP-server.

    The 'csr-support' node enables the SZTP-client to indicate
    that it supports generating certificate signing requests
    (CSRs), and to provide details around the CSRs it is able
    to generate.

    The 'csr' node enables the SZTP-client to relay a CSR to
    the SZTP-server.";
  reference
    "IEEE 802.1AR: IEEE Standard for Local and metropolitan
      area networks - Secure Device Identity
    RFC 8572: Secure Zero Touch Provisioning (SZTP)";
  choice msg-type {
    description
      "Messages are mutually exclusive.";
    case csr-support {
      description
```



```

    "Indicates how the SZTP-client supports generating CSRs.

    If present and a SZTP-server wishes to request the
    SZTP-client generate a CSR, the SZTP-server MUST
    respond with HTTP code 400 Bad Request with an
    'ietf-restconf:errors' message having the 'error-tag'
    value 'missing-attribute' and the 'error-info' node
    containing the 'csr-request' structure described
    in this module.";
    uses zt:csr-support-grouping;
}
case csr {
    description
        "Provides the CSR generated by the SZTP-client.

        When present, the SZTP-server SHOULD respond with
        an SZTP onboarding information message containing
        a signed certificate for the conveyed CSR. The
        SZTP-server MAY alternatively respond with another
        HTTP error containing another 'csr-request', in
        which case the SZTP-client MUST delete any key
        generated for the previously generated CSR.";
        uses zt:csr-grouping;
    }
}
}

sx:structure csr-request {
    description
        "A YANG data structure, per RFC 8791, that specifies
        details for the CSR that the ZTP-client is to generate.";
    reference
        "RFC 8791: YANG Data Structure Extensions";
    uses zt:csr-request-grouping;
}
}

```

<CODE ENDS>

### 3. The "ietf-ztp-types" Module

This section defines a YANG 1.1 [\[RFC7950\]](#) module that defines three YANG groupings, one each for messages sent between a ZTP-client and ZTP-server. This module is defined independently of the "ietf-sztp-csr" module so that it's groupings may be used by bootstrapping protocols other than SZTP [\[RFC8572\]](#).

#### 3.1. Data Model Overview

The following tree diagram [\[RFC8340\]](#) illustrates the three groupings defined in the "ietf-ztp-types" module.

module: ietf-ztp-types

```
grouping csr-support-grouping
  +-- csr-support
    +-- key-generation!
      | +-- supported-algorithms
      |   +-- algorithm-identifier*   binary
    +-- csr-generation
      +-- supported-formats
        +-- format-identifier*   identityref
grouping csr-request-grouping
  +-- key-generation!
    | +-- selected-algorithm
    |   +-- algorithm-identifier   binary
  +-- csr-generation
    | +-- selected-format
    |   +-- format-identifier   identityref
  +-- cert-req-info?   ct:csr-info
grouping csr-grouping
  +-- (csr-type)
    +--:(p10-csr)
      | +-- p10-csr?   ct:csr
    +--:(cmc-csr)
      | +-- cmc-csr?   binary
    +--:(cmp-csr)
      +-- cmp-csr?   binary
```

#### 3.2. YANG Module

This module uses a data types and groupings [\[RFC8791\]](#) and [\[I-D.ietf-netconf-crypto-types\]](#). The module has additional normative references to [\[RFC2986\]](#), [\[RFC4210\]](#), [\[RFC5272\]](#), and [\[ITU.X690.2015\]](#), and an informative reference to [\[Std-802.1AR-2018\]](#).

<CODE BEGINS> file "ietf-ztp-types@2022-03-02.yang"

```

module ietf-ztp-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ztp-types";
  prefix zt;

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

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

  contact
    "WG Web:  https://datatracker.ietf.org/wg/netconf
    WG List:  NETCONF WG list <mailto:netconf@ietf.org>
    Authors:  Kent Watsen <mailto:kent+ietf@watsen.net>
              Russ Housley <mailto:housley@vigilsec.com>
              Sean Turner <mailto:sean@sn3rd.com>";

  description
    "This module defines three groupings that enable
    bootstrapping devices to 1) indicate if and how they
    support generating CSRs, 2) obtain a request to
    generate a CSR, and 3) communicate the requested CSR.

    Copyright (c) 2022 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 Revised
    BSD License set forth in Section 4.c of the IETF Trust's
    Legal Provisions Relating to IETF Documents
    (https://trustee.ietf.org/license-info).

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

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

  revision 2022-03-02 {
    description

```

```

    "Initial version";
reference
    "RFC XXXX: Conveying a Certificate Signing Request (CSR)
        in a Secure Zero Touch Provisioning (SZTP)
        Bootstrapping Request";
}

identity certificate-request-format {
description
    "A base identity for the request formats supported
        by the ZTP-client.

        Additional derived identities MAY be defined by
        future efforts.";
}

identity p10-csr {
base certificate-request-format;
description
    "Indicates that the ZTP-client supports generating
        requests using the 'CertificationRequest' structure
        defined in RFC 2986.";
reference
    "RFC 2986: PKCS #10: Certification Request Syntax
        Specification Version 1.7";
}

identity cmp-csr {
base certificate-request-format;
description
    "Indicates that the ZTP-client supports generating
        requests using a profiled version of the PKIMessage
        that MUST contain a PKIHeader followed by a PKIBody
        containing only the ir, cr, kur, or p10cr structure
        defined in RFC 4210.";
reference
    "RFC 4210: Internet X.509 Public Key Infrastructure
        Certificate Management Protocol (CMP)";
}

identity cmc-csr {
base certificate-request-format;
description
    "Indicates that the ZTP-client supports generating
        requests using a profiled version of the 'Full
        PKI Request' structure defined in RFC 5272.";
reference
    "RFC 5272: Certificate Management over CMS (CMC)";
}

```

```

// Protocol-accessible nodes

grouping csr-support-grouping {
  description
    "A grouping enabling use by other efforts.";
  container csr-support {
    description
      "Enables a ZTP-client to indicate that it supports
      generating certificate signing requests (CSRs) and
      provides details about the CSRs it is able to
      generate.";
    container key-generation {
      presence
        "Indicates that the ZTP-client is capable of
        generating a new asymmetric key pair.

        If this node is not present, the ZTP-server MAY
        request a CSR using the asymmetric key associated
        with the device's existing identity certificate
        (e.g., an IDevID from IEEE 802.1AR).";
      description
        "Specifies details for the ZTP-client's ability to
        generate a new asymmetric key pair.";
      container supported-algorithms {
        description
          "A list of public key algorithms supported by the
          ZTP-client for generating a new asymmetric key.";
        leaf-list algorithm-identifier {
          type binary;
          min-elements 1;
          description
            "An AlgorithmIdentifier, as defined in RFC 2986,
            encoded using ASN.1 distinguished encoding rules
            (DER), as specified in ITU-T X.690.";
          reference
            "RFC 2986: PKCS #10: Certification Request Syntax
            Specification Version 1.7
            ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
        }
      }
    }
  }
  container csr-generation {
    description
      "Specifies details for the ZTP-client's ability to

```

```

        generate a certificate signing requests.";
    container supported-formats {
        description
            "A list of certificate request formats supported
            by the ZTP-client for generating a new key.";
        leaf-list format-identifier {
            type identityref {
                base zt:certificate-request-format;
            }
            min-elements 1;
            description
                "A certificate request format supported by the
                ZTP-client.";
        }
    }
}

}

}

}

grouping csr-request-grouping {
    description
        "A grouping enabling use by other efforts.";
    container key-generation {
        presence
            "Provided by a ZTP-server to indicate that it wishes
            the ZTP-client to generate a new asymmetric key.

            This statement is present so the mandatory descendant
            nodes do not imply that this node must be configured.";
        description
            "The key generation parameters selected by the ZTP-server.

            This leaf MUST only appear if the ZTP-client's
            'csr-support' included the 'key-generation' node.";
        container selected-algorithm {
            description
                "The key algorithm selected by the ZTP-server. The
                algorithm MUST be one of the algorithms specified by
                the 'supported-algorithms' node in the ZTP-client's
                message containing the 'csr-support' structure.";
            leaf algorithm-identifier {
                type binary;
                mandatory true;
                description
                    "An AlgorithmIdentifier, as defined in RFC 2986,
                    encoded using ASN.1 distinguished encoding rules
                    (DER), as specified in ITU-T X.690.";
                reference
                    "RFC 2986: PKCS #10: Certification Request Syntax

```

Specification Version 1.7

ITU-T X.690:

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

```
    }
  }
}
container csr-generation {
  description
    "Specifies details for the CSR that the ZTP-client
    is to generate.";
  container selected-format {
    description
      "The CSR format selected by the ZTP-server. The
      format MUST be one of the formats specified by
      the 'supported-formats' node in the ZTP-client's
      request message.";
    leaf format-identifier {
      type identityref {
        base zt:certificate-request-format;
      }
      mandatory true;
      description
        "A certificate request format to be used by the
        ZTP-client.";
    }
  }
}
leaf cert-req-info {
  type ct:csr-info;
  description
    "A CertificationRequestInfo structure, as defined in
    RFC 2986, and modeled via a 'typedef' statement by
    RFC AAAA.

    Enables the ZTP-server to provide a fully-populated
    CertificationRequestInfo structure that the ZTP-client
    only needs to sign in order to generate the complete
    'CertificationRequest' structure to send to ZTP-server
    in its next 'get-bootstrapping-data' request message.

    When provided, the ZTP-client MUST use this structure
    to generate its CSR; failure to do so will result in a
    400 Bad Request response containing another 'csr-request'
    structure.

    When not provided, the ZTP-client SHOULD generate a CSR
```

using the same structure defined in its existing identity certificate (e.g., an IDevID from IEEE 802.1AR).

If the 'AlgorithmIdentifier' field contained inside the certificate 'SubjectPublicKeyInfo' field does not match the algorithm identified by the 'selected-algorithm' node, then the client MUST reject the certificate and raise an error.";

```
reference
  "RFC 2986:
    PKCS #10: Certification Request Syntax Specification
  RFC AAAA:
    YANG Data Types and Groupings for Cryptography";
}
}

grouping csr-grouping {
  description
    "Enables a ZTP-client to convey a certificate signing
    request, using the encoding format selected by a
    ZTP-server's 'csr-request' response to the ZTP-client's
    previously sent request containing the 'csr-support'
    node.";
  choice csr-type {
    mandatory true;
    description
      "A choice amongst certificate signing request formats.

      Additional formats MAY be augmented into this 'choice'
      statement by future efforts.";
    case p10-csr {
      leaf p10-csr {
        type ct:csr;
        description
          "A CertificationRequest structure, per RFC 2986.
          Encoding details are defined in the 'ct:csr'
          typedef defined in RFC AAAA.

          A raw P10 does not support origin authentication in
          the CSR structure. External origin authentication
          may be provided via the ZTP-client's authentication
          to the ZTP-server at the transport layer (e.g., TLS).";
        reference
          "RFC 2986: PKCS #10: Certification Request Syntax
            Specification
          RFC AAAA: YANG Data Types and Groupings for
            Cryptography";
      }
    }
  }
}
```



```

}
case cmc-csr {
  leaf cmc-csr {
    type binary;
    description
      "A profiled version of the 'Full PKI Request'
      message defined in RFC 5272, encoded using ASN.1
      distinguished encoding rules (DER), as specified
      in ITU-T X.690.

```

For asymmetric key-based origin authentication of a CSR based on the initial device identity certificate's private key for the associated identity certificate's public key, the PKIData contains one reqSequence element and no cmsSequence or otherMsgSequence elements. The reqSequence is the TaggedRequest and it is the tcr CHOICE branch. The tcr is the TaggedCertificationRequest and it is the bodyPartId and the certificateRequest elements. The certificateRequest is signed with the initial device identity certificate's private key. The initial device identity certificate and optionally its certificate chain is included in the SignedData certificates that encapsulates the PKIData.

For asymmetric key-based origin authentication based on the initial device identity certificate's private key that signs the encapsulated CSR signed by the local device identity certificate's private key, the PKIData contains one cmsSequence element and no reqSequence or otherMsgSequence elements. The cmsSequence is the TaggedContentInfo and it includes a bodyPartID element and a contentInfo. The contentInfo is a SignedData encapsulating a PKIData with one reqSequence element and no cmsSequence or otherMsgSequence elements. The reqSequence is the TaggedRequest and it is the tcr CHOICE. The tcr is the TaggedCertificationRequest and it is the bodyPartId and the certificateRequest elements. PKIData contains one cmsSequence element and no controlSequence, reqSequence, or otherMsgSequence elements. The certificateRequest is signed with the local device identity certificate's private key. The initial device identity certificate and optionally its certificate chain is included in the SignedData certificates that encapsulates the PKIData.

For shared secret-based origin authentication of a CSR signed by the local device identity certificate's private key, the PKIData contains one cmsSequence

element and no reqSequence or otherMsgSequence elements. The cmsSequence is the TaggedContentInfo and it includes a bodyPartID element and a contentInfo. The contentInfo is an AuthenticatedData encapsulating a PKIData with one reqSequence element and no cmsSequences or otherMsgSequence elements. The reqSequence is the TaggedRequest and it is the tcr CHOICE. The tcr is the TaggedCertificationRequest and it is the bodyPartId and the certificateRequest elements. The certificateRequest is signed with the local device identity certificate's private key. The initial device identity certificate and optionally its certificate chain is included in the SignedData certificates that encapsulates the PKIData.";

reference

"RFC 5272: Certificate Management over CMS (CMC)

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).";

}

}

case cmp-csr {

leaf cmp-csr {

type binary;

description

"A PKIMessage structure, as defined in RFC 4210,  
encoded using ASN.1 distinguished encoding rules  
(DER), as specified in ITU-T X.690.

For asymmetric key-based origin authentication of a CSR based on the initial device identity certificate's private key for the associated initial device identity certificate's public key, PKIMessages contains one PKIMessage with the header and body elements, no protection element, and SHOULD contain the extraCerts element. The header element contains the pvno, sender, and recipient elements. The pvno contains cmp2000, and the sender contains the subject of the initial device identity certificate. The body element contains an ir, cr, kur, or p10cr CHOICE of type CertificationRequest. It is signed with the initial device identity certificate's private key. The extraCerts element contains the initial device identity certificate, optionally followed by its certificate chain excluding the trust anchor.

For asymmetric key-based origin authentication based

on the initial device identity certificate's private key that signs the encapsulated CSR signed by the local device identity certificate's private key, PKIMessages contains one PKIMessage with the header, body, and protection elements, and SHOULD contain the extraCerts element. The header element contains the pvno, sender, recipient, protectionAlg, and optionally senderKID elements. The pvno contains cmp2000, the sender contains the subject of the initial device identity certificate, the protectionAlg contains the AlgorithmIdentifier of the used signature algorithm, and the senderKID contains the subject key identifier of the initial device identity certificate. The body element contains an ir, cr, kur, or p10cr CHOICE of type CertificationRequest. It is signed with the local device identity certificate's private key. The protection element contains the digital signature generated with the initial device identity certificate's private key. The extraCerts element contains the initial device identity certificate, optionally followed by its certificate chain excluding the trust anchor.

For shared secret-based origin authentication of a CSR signed by the local device identity certificate's private key, PKIMessages contains one PKIMessage with the header, body, and protection element, and no extraCerts element. The header element contains the pvno, sender, recipient, protectionAlg, and senderKID elements. The pvno contains cmp2000, the protectionAlg contains the AlgorithmIdentifier of the used MAC algorithm, and the senderKID contains a reference the recipient can use to identify the shared secret. The body element contains an ir, cr, kur, or p10cr CHOICE of type CertificationRequest. It is signed with the local device identity certificate's private key. The protection element contains the MAC value generated with the shared secret.";

reference

"RFC 4210:

Internet X.509 Public Key Infrastructure  
Certificate Management Protocol (CMP)

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).";

}

}

}  
}  
}

<CODE ENDS>

## 4. Security Considerations

This document builds on top of the solution presented in [\[RFC8572\]](#) and therefore all the Security Considerations discussed in RFC 8572 apply here as well.

For the various CSR formats, when using PKCS#10, the security considerations in [\[RFC2986\]](#) apply, when using CMP, the security considerations in [\[RFC4210\]](#) apply and, when using CMC, the security considerations in [\[RFC5272\]](#) apply.

For the various authentication mechanisms, when using TLS-level authentication, the security considerations in [\[RFC8446\]](#) apply and, when using HTTP-level authentication, the security considerations in [\[RFC7235\]](#) apply.

### 4.1. SZTP-Client Considerations

#### 4.1.1. Ensuring the Integrity of Asymmetric Private Keys

The private key the SZTP-client uses for the dynamically-generated identity certificate MUST be protected from inadvertent disclosure in order to prevent identity fraud.

The security of this private key is essential in order to ensure the associated identity certificate can be used to authenticate the device it is issued to.

It is RECOMMENDED that devices are manufactured with an HSM (hardware security module), such as a TPM (trusted platform module), to generate and contain the private key within the security perimeter of the HSM. In such cases, the private key, and its associated certificates, MAY have long validity periods.

In cases where the SZTP-client does not possess an HSM, or is unable to use an HSM to protect the private key, it is RECOMMENDED to periodically reset the private key (and associated identity certificates) in order to minimize the lifetime of unprotected private keys. For instance, an NMS controller/orchestrator application could periodically prompt the SZTP-client to generate a new private key and provide a certificate signing request (CSR) or, alternatively, push both the key and an identity certificate to the SZTP-client using, e.g., a PKCS #12 message [\[RFC7292\]](#). In another example, the SZTP-client could be configured to periodically reset the configuration to its factory default, thus causing removal of the private key and associated identity certificates and re-execution of the SZTP protocol.

#### 4.1.2. Reuse of a Manufacturer-generated Private Key

It is RECOMMENDED that a new private key is generated for each CSR described in this document.

Implementations must randomly generate nonces and private keys. The use of inadequate pseudo-random number generators (PRNGs) to generate cryptographic keys can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the keys, searching the resulting small set of possibilities, rather than brute force searching the whole key space. As an example of predictable random numbers see CVE-2008-0166 [[CVE-2008-0166](#)], and some consequences of low-entropy random numbers are discussed in Mining Your Ps and Qs [[MiningPsQs](#)]. The generation of quality random numbers is difficult. [[ISO.20543-2019](#)], [[NIST.SP.800-90Ar1](#)], BSI AIS 31 [[AIS31](#)], BCP 106 [[RFC4086](#)], and others offer valuable guidance in this area.

This private key SHOULD be protected as well as the built-in private key associated with the SZTP-client's initial device identity certificate (e.g., the IDevID, from [[Std-802.1AR-2018](#)]).

In cases where it is not possible to generate a new private key that is protected as well as the built-in private key, it is RECOMMENDED to reuse the built-in private key rather than generate a new private key that is not as well protected.

#### 4.1.3. Replay Attack Protection

This RFC enables an SZTP-client to announce an ability to generate a new key to use for its CSR.

When the SZTP-server responds with a request for the SZTP-client to generate a new key, it is essential that the SZTP-client actually generates a new key.

Generating a new key each time enables the random bytes used to create the key to also serve the dual-purpose of acting like a "nonce" used in other mechanisms to detect replay attacks.

When a fresh public/private key pair is generated for the request, confirmation to the SZTP-client that the response has not been replayed is enabled by the SZTP-client's fresh public key appearing in the signed certificate provided by the SZTP-server.

When a public/private key pair associated with the manufacturer-generated identity certificate (e.g., IDevID) is used for the request, there may not be confirmation to the SZTP-client that the response has not been replayed; however, the worst case result is a lost certificate that is associated to the private key known only to

the SZTP-client. Protection of the private-key information is vital to public-key cryptography. Disclosure of the private-key material to another entity can lead to masquerades.

#### **4.1.4. Connecting to an Untrusted Bootstrap Server**

[[RFC8572](#)] allows SZTP-clients to connect to untrusted SZTP-servers, by blindly authenticating the SZTP-server's TLS end-entity certificate.

As is discussed in [Section 9.5](#) of [[RFC8572](#)], in such cases the SZTP-client MUST assert that the bootstrapping data returned is signed, if the SZTP-client is to trust it.

However, the HTTP error message used in this document cannot be signed data, as described in RFC 8572.

Therefore, the solution presented in this document cannot be used when the SZTP-client connects to an untrusted SZTP-server.

Consistent with the recommendation presented in [Section 9.6](#) of [[RFC8572](#)], SZTP-clients SHOULD NOT pass the "csr-support" input parameter to an untrusted SZTP-server. SZTP-clients SHOULD pass instead the "signed-data-preferred" input parameter, as discussed in [Appendix B](#) of [[RFC8572](#)].

#### **4.1.5. Selecting the Best Origin Authentication Mechanism**

The origin of the CSR must be verified before a certificate is issued.

When generating a new key, it is important that the SZTP-client be able to provide additional proof that it was the entity that generated the key.

The CMP and CMC certificate request formats defined in this document support origin authentication. A raw PKCS#10 CSR does not support origin authentication.

The CMP and CMC request formats support origin authentication using both PKI and shared secret.

Typically, only one possible origin authentication mechanism can possibly be used but, in the case that the SZTP-client authenticates itself using both TLS-level (e.g., IDevID) and HTTP-level credentials (e.g., Basic), as is allowed by [Section 5.3](#) of [[RFC8572](#)], then the SZTP-client may need to choose between the two options.

In the case that the SZTP-client must choose between an asymmetric key option versus a shared secret for origin authentication, it is RECOMMENDED that the SZTP-client choose using the asymmetric key.

#### **4.1.6. Clearing the Private Key and Associated Certificate**

Unlike a manufacturer-generated identity certificate (e.g., IDevID), the deployment-generated identity certificate (e.g., LDevID) and the associated private key (assuming a new private key was generated for the purpose), are considered user data and SHOULD be cleared whenever the SZTP-client is reset to its factory default state, such as by the "factory-reset" RPC defined in [[RFC8808](#)].

### **4.2. SZTP-Server Considerations**

#### **4.2.1. Verifying Proof of Possession**

Regardless if using a new asymmetric key or the bootstrapping device's manufacturer-generated key (e.g., the IDevID key), the public key is placed in the CSR and the CSR is signed by that private key. Proof-of-possession of the private key is verified by ensuring the signature over the CSR using the public key placed in the CSR.

#### **4.2.2. Verifying Proof of Origin**

When the bootstrapping device's manufacturer-generated private key (e.g., the IDevID key) is reused for the CSR, proof-of-origin is verified by validating the IDevID-issuer cert and ensuring that the CSR uses the same key pair.

When the bootstrapping device's manufacturer-generated private key (e.g., an IDevID key from IEEE 802.1AR) is reused for the CSR, proof-of-origin is verified by validating the IDevID certification path and ensuring that the CSR uses the same key pair.

When a fresh asymmetric key is used with the CMP or CMC formats, the authentication is part of the protocols, which could employ either the manufacturer-generated private key or a shared secret. In addition, CMP and CMC support processing by a RA before the request is passed to the CA, which allows for more robust handling of errors.

#### **4.2.3. Supporting SZTP-Clients that don't trust the SZTP-Server**

[[RFC8572](#)] allows SZTP-clients to connect to untrusted SZTP-servers, by blindly authenticating the SZTP-server's TLS end-entity certificate.



As is recommended in [Section 4.1.4](#) in this document, in such cases, SZTP-clients SHOULD pass the "signed-data-preferred" input parameter.

The reciprocal of this statement is that SZTP-servers, wanting to support SZTP-clients that don't trust them, SHOULD support the "signed-data-preferred" input parameter, as discussed in [Appendix B](#) of [\[RFC8572\]](#).

#### **4.3. Security Considerations for the "ietf-sztp-csr" YANG Module**

The recommended format for documenting the Security Considerations for YANG modules is described in [Section 3.7](#) of [\[RFC8407\]](#). However, this module only augments two input parameters into the "get-bootstrapping-data" RPC in [\[RFC8572\]](#), and therefore only needs to point to the relevant Security Considerations sections in that RFC.

\*Security considerations for the "get-bootstrapping-data" RPC are described in [Section 9.16](#) of [\[RFC8572\]](#).

\*Security considerations for the "input" parameters passed inside the "get-bootstrapping-data" RPC are described in [Section 9.6](#) of [\[RFC8572\]](#).

#### **4.4. Security Considerations for the "ietf-ztp-types" YANG Module**

The recommended format for documenting the Security Considerations for YANG modules is described in [Section 3.7](#) of [\[RFC8407\]](#). However, this module does not define any protocol-accessible nodes (it only defines "identity" and "grouping" statements) and therefore there are no Security considerations to report.

### **5. IANA Considerations**

#### **5.1. The "IETF XML" Registry**

This document registers two URIs in the "ns" subregistry of the IETF XML Registry [\[RFC3688\]](#) maintained at <https://www.iana.org/assignments/xml-registry/xml-registry.xhtml#ns>. Following the format in [\[RFC3688\]](#), the following registrations are requested:

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

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

## 5.2. The "YANG Module Names" Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020] maintained at <https://www.iana.org/assignments/yang-parameters/yang-parameters.xhtml>. Following the format defined in [RFC6020], the below registrations are requested:

name: ietf-sztp-csr  
namespace: urn:ietf:params:xml:ns:yang:ietf-sztp-csr  
prefix: sztp-csr  
reference: RFC XXXX

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

## 6. References

### 6.1. Normative References

#### [I-D.ietf-netconf-crypto-types]

Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-21, 14 September 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-21>>.

[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>>.

[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>>.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <<https://www.rfc-editor.org/info/rfc4210>>.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <<https://www.rfc-editor.org/info/rfc5272>>.
- [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>>.
- [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <<https://www.rfc-editor.org/info/rfc7235>>.
- [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>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [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>>.
- [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>>.
- [RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/

RFC8572, April 2019, <<https://www.rfc-editor.org/info/rfc8572>>.

[RFC8791] Bierman, A., Björklund, M., and K. Watsen, "YANG Data Structure Extensions", RFC 8791, DOI 10.17487/RFC8791, June 2020, <<https://www.rfc-editor.org/info/rfc8791>>.

## 6.2. Informative References

[AIS31] Bundesamt für Sicherheit in der Informationstechnik (BSI), Killmann, W., and W. Schindler, "A proposal for: Functionality classes for random number generators, version 2.0", 18 September 2011, <[https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS\\_31\\_Functionality\\_classes\\_for\\_random\\_number\\_generators\\_e.pdf](https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zertifizierung/Interpretationen/AIS_31_Functionality_classes_for_random_number_generators_e.pdf)>.

[CVE-2008-0166] National Institute of Science and Technology (NIST), "National Vulnerability Database - CVE-2008-0166", 13 May 2008, <<https://nvd.nist.gov/vuln/detail/CVE-2008-0166>>.

[I-D.ietf-netconf-keystore] Watsen, K., "A YANG Data Model for a Keystore", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-23, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-23>>.

[I-D.ietf-netconf-trust-anchors] Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-16, 14 December 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-16>>.

[ISO.20543-2019] International Organization for Standardization (ISO), "Information technology -- Security techniques -- Test and analysis methods for random bit generators within ISO/IEC 19790 and ISO/IEC 15408", ISO Draft Standard 20543-2019, October 2019.

[MiningPsQs] Security'12: Proceedings of the 21st USENIX conference on Security symposium, Heninger, N., Durumeric, Z., Wustrow, E., and J. A. Halderman, "Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices", August 2012, <<https://www.usenix.org/conference/>

[usenixsecurity12/technical-sessions/presentation/heninger](https://usenixsecurity12/technical-sessions/presentation/heninger)>.

- [NIST.SP.800-90Ar1] Barker, Elaine B. and John M. Kelsey, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", DOI 10.6028/NIST.SP.800-90Ar1, NIST NIST SP 800-90Ar1, June 2015, <<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90Ar1.pdf>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC7292] Moriarty, K., Ed., Nystrom, M., Parkinson, S., Rusch, A., and M. Scott, "PKCS #12: Personal Information Exchange Syntax v1.1", RFC 7292, DOI 10.17487/RFC7292, July 2014, <<https://www.rfc-editor.org/info/rfc7292>>.
- [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>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [RFC8808] Wu, Q., Lengyel, B., and Y. Niu, "A YANG Data Model for Factory Default Settings", RFC 8808, DOI 10.17487/RFC8808, August 2020, <<https://www.rfc-editor.org/info/rfc8808>>.
- [Std-802.1AR-2018] Group, W. -. H. L. L. P. W., "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", 14 June 2018, <[https://standards.ieee.org/standard/802\\_1AR-2018.html](https://standards.ieee.org/standard/802_1AR-2018.html)>.

## Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by first name): Benjamin Kaduk, David von Oheimb, Dan Romascanu, Eric Vyncke, Hendrik Brockhaus, Guy Fedorkow, Joe Clarke, Meral Shirazipour, Murray Kucherawy, Rich Salz, Rob Wilton, Roman Danyliw, Qin Wu, Yaron Sheffer, and Zaheduzzaman Sarkar.

## Contributors

Special thanks go to David von Oheimb and Hendrik Brockhaus for helping with the descriptions for the "cmc-csr" and "cmp-csr" nodes.

## Authors' Addresses

Kent Watsen  
Watsen Networks

Email: [kent+ietf@watsen.net](mailto:kent+ietf@watsen.net)

Russ Housley  
Vigil Security, LLC

Email: [housley@vigilsec.com](mailto:housley@vigilsec.com)

Sean Turner  
sn3rd

Email: [sean@sn3rd.com](mailto:sean@sn3rd.com)