

System Keychain Model
draft-ietf-netconf-system-keychain-00

Abstract

This document defines a YANG data module for a system-level keychain mechanism, that might be used to hold onto private keys and certificates that are trusted by the system advertising support for this module.

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.

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:

- o [draft-ietf-netconf-restconf](#)
- o [draft-ietf-netconf-call-home](#)
- o [draft-ietf-rtgwg-yang-key-chain](#)

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

- o "VVVV" --> the assigned RFC value for this draft
- o "XXXX" --> the assigned RFC value for [draft-ietf-netconf-restconf](#)
- o "YYYY" --> the assigned RFC value for [draft-ietf-netconf-call-home](#)

Artwork in this document contains placeholder values for ports pending IANA assignment from "[draft-ietf-netconf-call-home](#)". Please apply the following replacements:

- o "7777" --> the assigned port value for "netconf-ch-ssh"
- o "8888" --> the assigned port value for "netconf-ch-tls"
- o "9999" --> the assigned port value for "restconf-ch-tls"

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

- o "2016-07-08" --> the publication date of this draft

The following two Appendix sections are to be removed prior to publication:

- o [Appendix A](#). Change Log
- o [Appendix B](#). Open Issues

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 <http://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 January 9, 2017.

Copyright Notice

Copyright (c) 2016 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 (<http://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 Simplified BSD License text as described in [Section 4](#).e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Terminology	4
1.2.	Tree Diagrams	4
2.	The System Keychain Model	4
2.1.	Overview	5
2.2.	Example Usage	6
2.3.	YANG Module	17
3.	Design Considerations	28
4.	Security Considerations	29
5.	IANA Considerations	30
5.1.	The IETF XML Registry	30
5.2.	The YANG Module Names Registry	30
6.	Acknowledgements	30
7.	References	30
7.1.	Normative References	30
7.2.	Informative References	31
Appendix A.	Change Log	33
A.1.	server-model-09 to 00	33
Appendix B.	Open Issues	33
	Authors' Addresses	33

[1.](#) Introduction

This document defines a YANG [[RFC6020](#)] data module for a system-level keychain mechanism, which can be used to hold onto private keys and certificates that are trusted by the system advertising support for this module.

This module provides a centralized location for security sensitive data, so that the data can be then referenced by other modules. There are two types of data that are maintained by this module:

- o Private keys, and any associated public certificates.
- o Sets of trusted certificates.

This document extends special consideration for systems that have Trusted Protection Modules (TPMs). These systems are unique in that the TPM must be directed to generate new private keys (it is not possible to load a private key into a TPM) and it is not possible to backup/restore the TPM's private keys as configuration.

It is not required that a system has an operating system level keychain utility to implement this module.

1.1. Terminology

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

1.2. Tree Diagrams

A simplified graphical representation of the data models is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature names, and indicate that the named feature must be present for the subtree to be present.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2. The System Keychain Model

The system keychain module defined in this section provides a configurable object having the following characteristics:

- o A semi-configurable list of private keys, each with one or more associated certificates. Private keys MUST be either preinstalled (e.g., a key associated to an IDevID [[Std-802.1AR-2009](#)] certificate), be generated by request, or be loaded by request. Each private key is MAY have associated certificates, either preinstalled or configured after creation.
- o A configurable list of lists of trust anchor certificates. This enables the server to have use-case specific trust anchors. For instance, one list of trust anchors might be used to authenticate management connections (e.g., client certificate-based

authentication for NETCONF or RESTCONF connections), and a different list of trust anchors might be used for when connecting to a specific Internet-based service (e.g., a zero touch bootstrap server).

- o An RPC to generate a certificate signing request for an existing private key, a passed subject, and an optional attributes. The signed certificate returned from an external certificate authority (CA) can be later set using a standard configuration change request (e.g., <edit-config>).
- o An RPC to request the server to generate a new private key using the specified algorithm and key length.
- o An RPC to request the server to load a new private key.

2.1. Overview

The system keychain module has the following tree diagram. Please see [Section 1.2](#) for information on how to interpret this diagram.

```

module: ietf-system-keychain
  +--rw keychain
    +--rw private-keys
      | +--rw private-key* [name]
      | | +--rw name string
      | | +--ro algorithm? identityref
      | | +--ro key-length? uint32
      | | +--ro public-key binary
      | | +--rw certificate-chains
      | | | +--rw certificate-chain* [name]
      | | | | +--rw name string
      | | | | +--rw certificate* binary
      | | +---x generate-certificate-signing-request
      | | | +---w input
      | | | | +---w subject binary
      | | | | +---w attributes? binary
      | | | +--ro output
      | | | +--ro certificate-signing-request binary
      | +---x generate-private-key
      | | +---w input
      | | | +---w name string
      | | | +---w algorithm identityref
      | | | +---w key-length? uint32
      | +---x load-private-key
      | | +---w input
      | | | +---w name string
      | | | +---w private-key binary
  
```



```

+--rw trusted-certificates* [name]
| +--rw name                string
| +--rw description?       string
| +--rw trusted-certificate* [name]
|   +--rw name              string
|   +--rw certificate?     binary
+--rw trusted-ssh-host-keys* [name]
| +--rw name                string
| +--rw description?       string
| +--rw trusted-host-key* [name]
|   +--rw name              string
|   +--rw host-key         binary
+--rw user-auth-credentials
  +--rw user-auth-credential* [username]
  +--rw username            string
  +--rw auth-method* [priority]
  +--rw priority            uint8
  +--rw (auth-type)?
  +--:(certificate)
  | +--rw certificate*      -> /keychain/private-
keys/private-key/certificate-chains/certificate-chain/name
  +--:(public-key)
  | +--rw public-key*      -> /keychain/private-
keys/private-key/name
  +--:(ciphertext-password)
  | +--rw ciphertext-password? string
  +--:(cleartext-password)
  +--rw cleartext-password? string

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

```

2.2. Example Usage

The following example illustrates the "generate-private-key" action in use with the RESTCONF protocol and JSON encoding.

REQUEST

['\`' line wrapping added for formatting only]

POST https://example.com/restconf/data/ietf-system-keychain:keychain/
private-keys/generate-private-key HTTP/1.1
HOST: example.com
Content-Type: application/yang.operation+json

```
{  
  "ietf-system-keychain:input" : {  
    "name" : "ex-key-sect571r1",  
    "algorithm" : "sect571r1"  
  }  
}
```

RESPONSE

HTTP/1.1 204 No Content
Date: Mon, 31 Oct 2015 11:01:00 GMT
Server: example-server

The following example illustrates the "load-private-key" action in use with the RESTCONF protocol and JSON encoding.

REQUEST

['\ ' line wrapping added for formatting only]

```
POST https://example.com/restconf/data/ietf-system-keychain:keychain/\
private-keys/load-private-key HTTP/1.1
HOST: example.com
Content-Type: application/yang.operation+xml
```

```
<input xmlns="urn:ietf:params:xml:ns:yang:ietf-system-keychain">
  <name>ex-key-sect571r1</name>
  <private-key>
    NGcEk3UE90cnNFVjRwTUNBd0VBQWFPQ0FSSXdnZ0VPck1CMEdBMVVkRGd\
    VEJiZ0JTWEdlbUEKMnhpRHV0TVkvVHFLNwd4cFJBZ1Z0YUU0cERZd05ER\
    V6QVJCZ05WQkFNVENrTlNUQ0JKYzNOMVpYS0NDUUNVRHBNS1l6UG8zREF\
    Z05WSFI4RV1qQmdNRjZnSXFbZ2hoNw9kSFJ3T2k4d1pYaGgKY1hCc1pTN\
    Qmd0VkJBwVRBbFZUTVJBd0RnWURWUVFLRXdkbAp1R0Z0Y0d4bE1RNHdEQ\
    MkF6a3hqUD1VQWtHR0dvS1U1eUc1SVR0Wm0vK3B0R2FieXVDMjBRd2kvZ\
    NQmd0VkhSTUJBZjhFckFqQUFNQTRHQTfVZER3RUIvd1FFQXdJSGdEQnBC\
    WmdsK2gyTTg3QmtGMjhWbW1CdFFVaWc30EgrRkYyRTFwdSt4ZVRJbVFFM\
    lLQ1l1sdWp0cjFTMnRLR05EMUC20VJpK2FWNGw2NTdZNCtadVJMZgpRYjk\
    zSFNwSDdwVXBCYnA4dmtNanFtZjJma3RqZHBxeFppUUtTbndWZTF2Zwot\
    25PZnpZNEhONApXY0pTaUpZK2xtYws3RTR0RUZXS9RdGp4NU1XZmdvN2\
    WpiMjB2WlhoaGJYQnNaUzVqY215aU9L=
  </private-key>
</input>
```

RESPONSE

```
HTTP/1.1 204 No Content
Date: Mon, 31 Oct 2015 11:01:00 GMT
Server: example-server
```

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">
    <keychain
      xmlns="urn:ietf:params:xml:ns:yang:ietf-system-keychain">
```



```

<private-keys>
  <private-key>
    <name>ex-key-sect571r1</name>
    <generate-certificate-signing-request>
      <subject>
        cztvaWRoc2RmZ2tqaHNkZmdramRzZnZzZGtmam5idnNvO2R
        manZvO3NkZmJpdmhzzGZpbHvidjtvvc2lkZmhidm11bHNlmo
        Z2aXNiZGZpYmhzzG87ZmJvO3NkZ25iO29pLmR6Zgo=
      </subject>
      <attributes>
        bwtakWRoc2RmZ2tqaHNkZmdramRzZnZzZGtmam5idnNvut4
        arnZvO3NkZmJpdmhzzGZpbHvidjtvvc2lkZmhidm11bHNkYm
        Z2aXNiZGZpYmhzzG87ZmJvO3NkZ25iO29pLmC6Rhp=
      </attributes>
    </generate-certificate-signing-request>
  </private-key>
</private-keys>
</keychain>
</action>
</rpc>

```

RESPONSE

```

<rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <certificate-signing-request
    xmlns="urn:ietf:params:xml:ns:yang:ietf-system-keychain">
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0LS0tCk1JSUNrekNDQWZ5Z
    0F3SUJBZ0lKQUpRT2t3bGpNK2pjTUEwR0NTcUdTSWIZRFFFQkJRvU
    FNRFF4Q3pBSklnTlYkQkFZVEFsVlRNUkF3RGdZRFZRUUtd2RszUd
    GdGNHeGxNUk13RVFZRFZRUURFd3BEVWt3Z1NYTnpkV1Z5TUI0WApE
    diR1V4RXpBUklnTlZCQU1UQ2t0OU1RDQkpcjM04xw1hJd2da0HdEUVl
    KS29aSwH2Y04KQVFFQkJRQURnWTBtBU1HSkFvR0JBtXVvZmFPNEV3
    EL1QWMrQ1RsTkNmc0d6cEw1Um5ydXZsOFRIcUJtdGZQY3N0Zk1KT1
    FaNzlnNlNWVldsMldzaHE1bUViCkJNNitGNzdjbTAvU25FcFE0TnV
    bXBDT2YkQWdNqkFBR2pnYXd3Z2Frd0hrWURWUjBPQkZRUZKY1o2W
    URiR0lPNDB4ajlPb3JtREdsRUNCVTFNR1FHQTFVZApJd1JkTUZ1QU
    ZKY1o2WURiR0lPNDB4ajlPb3JtREdsRUNCVTFvVGlrTmPBME1Rc3d
    mMKTUE0R0ExVWREd0VCL3dRRUF3SUNCREFTQmd0VkhSTUJBZjhFQ0
    RBR0FRSC9BZ0VBTUEwR0NTcUdTSWIZRFFFQgpCUVVBQTRHQkFMMmx
    rWmFGNWcyAGR6MVNhZnZPbnBneHA4eG00SHRhbStadHpLazFlS3Bx
    TXp4YXJCbFpDSHlLck1VbC9GVzRtV1RQS1VDeEtFTE40NEY2Zmk2d
    c4d0tSSElkYw1Wl0pGTmlQS0VXSTF4K1I1aDZmazcrQzQ1QXg1RWV
    SWHgZzjdVM2xZTgotLS0tLUVORCBDRVJUSUZJQ0FURSB0LS0tCg==
  </certificate-signing-request>
</rpc-reply>

```


The following example illustrates what a fully configured keychain object might look like. The private-key shown below is consistent with the generate-private-key and generate-certificate-signing-request examples above. This example also assumes that the resulting CA-signed certificate has been configured back onto the server. Lastly, this example shows that three lists of trusted certificates having been configured.

```
<keychain xmlns="urn:ietf:params:xml:ns:yang:ietf-system-keychain">

  <!-- private keys and associated certificates -->
  <private-keys>
    <private-key>
      <name>my-rsa-user-key</name>
      <algorithm>rsa</algorithm>
      <public-key>
        cztvaWRoc2RmZ2tqaHNkZmdramRzZnZzZGtmam5idnNvO2RmanZvO3NkZ
        mJpdmhzZGZpbHVidjtvvc2lkZmhidml1bHNkYmZ2aXNiZGZpYmhzZG87Zm
        JvO3NkZ25iO29pLmR6Zgo=
      </public-key>
      <certificate-chains>
        <certificate-chain>
          <name>my-rsa-chain</name>
          <certificate>
            ZKY1o2WURiR0lPNDB4ajlPb3JtREdsRUNCVTFvVGlRtmpBME1Rc3d
            diR1V4RXpBUKJnTlZCQU1UQ2tOU1RDQkpjM04xWlhJd2daOHdEUVl
            LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUNrekNDQWZ5Z
            KS29aSWh2Y04KQVFFQkJRQRnWTBBTUlHskFvR0JBTXVvZmFPNEV3
            0F3SUJBZ0lKQUpRT2t3bGpNK2pjTUEwR0NTcUdTSWIZRFFFQkJRUV
            FNRRFF4Q3pBSkJnTlYkQkFZVEFsVlRNUkF3RGdZRFZRUUtFd2RsZUd
            GdGNHeGxNUk13RVFZRFZRUURFd3BEVWt3Z1NYTnpkV1Z5TUI0WApE
            mMKTUE0R0ExVWREd0VCL3dRRUF3SUNCREFTQmd0VkhSTUJBZjhFQ0
            RBR0FRSC9BZ0VBTUEwR0NTcUdTSWIZRFFFQgpCUVVBQTRHkFMmX
            rWmFGNWcyaGR6MVNhZnZPbnBneHA4eG00SHRhbStadHpLazFlS3Bx
            TXp4YXJCbFpDSHlLCk1VbC9GVzRtV1RQS1VDeEtFTE40NEY2Zmk2d
            c4d0tSSElkYW1WL0pGTmlQS0VXSTF4K1I1aDZmazcrQzQ1QXg1RWV
            SWM2xZTgotLS0tLUVORCBDRVJUSUZJQ0FURSB0tLS0tCg==
          </certificate>
        </certificate-chain>
      </certificate-chains>
    </private-key>

    <private-key>
      <name>my-ec-user-key</name>
      <algorithm>secp256r1</algorithm>
      <public-key>
        mJpdmhzZGZpbHVidjtvvc2lkZmhidml1bHNkYmZ2aXNiZGZpYmhzZG87Zm
        cztvaWRoc2RmZ2tqaHNkZmdramRzZnZzZGtmam5idnNvO2RmanZvO3NkZ
```



```

    Jv03NkZ25i029pLmR6Zgo=
  </public-key>
  <certificate-chains>
    <certificate-chain>
      <name>my-ec-chain</name>
      <certificate>
        0F3SUJBZ0lKQUpRT2t3bGpNK2pjTUEwr0NTcUdTSWIzRFFFQkJRvU
        ZKY1o2WURiR0lPNDB4ajlPb3JtREdsRUNCVTFvVGlrTmPBME1Rc3d
        diR1V4RXpBUkJnTlZCQU1UQ2t0U1RDQkpjM04xWlhJd2da0HdEUVl
        LS0tLS1CRudJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUNrekNDQWZ5Z
        KS29aSWh2Y04KQVFFQkJRQURnWTBBTUlHskFvR0JBTXVvZmFPNEV3
        FNRRFF4Q3pBSkJnTlYkQkFZVEFsVlRNUkF3RGdZRFZRUUtFd2RsZUd
        GdGNHeGxNUk13RVFZRFZRUURFd3BEVwt3Z1NYTnpkV1Z5TUI0WApE
        mMKTUE0R0ExVwREd0VCL3dRRUF3SUNCREFTQmd0VkhSTUJBZjhFQ0
        RBR0FRSC9BZ0VBTUEwr0NTcUdTSWIzRFFFQgpCUVVBQTRHqkFMMmx
        rWmFGNWcyaGR6MVNhZnZPbnBneHA4eG00SHRhbStadHpLazFlS3Bx
        TXp4YXJCbFpDSH1Lck1VbC9GVzRtV1RQS1VDeEtFTE40NEY2Zmk2d
        c4d0tSSElkYw1WL0pGTmlQS0VXSTF4K1I1aDZmazcrQzQ1QXg1RWV
        SWM2xZTgotLS0tLUVORCBDRVJUSUZJQ0FURSB0tLS0tCg==
      </certificate>
    </certificate-chain>
  </certificate-chains>
</private-key>

<private-key>
  <name>tpm-protected-key</name>
  <algorithm>sect571r1</algorithm>
  <public-key>
    cztvaWRoc2RmZ2tqaHNkZmdramRzZnZzZGtmam5idnNv02RmanZv03NkZ
    mJpdmhzZGZpbHvidjtvC2lkZmhidml1bHNkYmZ2aXNiZGZpYmhZzG87Zm
    Jv03NkZ25i029pLmR6Zgo=
  </public-key>
  <certificate-chains>
    <certificate-chain>
      <name>default-idevid-chain</name>
      <certificate>
        diR1V4RXpBUkJnTlZCQU1UQ2t0U1RDQkpjM04xWlhJd2da0HdEUVl
        LS0tLS1CRudJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSUNrekNDQWZ5Z
        KS29aSWh2Y04KQVFFQkJRQURnWTBBTUlHskFvR0JBTXVvZmFPNEV3
        0F3SUJBZ0lKQUpRT2t3bGpNK2pjTUEwr0NTcUdTSWIzRFFFQkJRvU
        FNRRFF4Q3pBSkJnTlYkQkFZVEFsVlRNUkF3RGdZRFZRUUtFd2RsZUd
        GdGNHeGxNUk13RVFZRFZRUURFd3BEVwt3Z1NYTnpkV1Z5TUI0WApE
        ZKY1o2WURiR0lPNDB4ajlPb3JtREdsRUNCVTFvVGlrTmPBME1Rc3d
        mMKTUE0R0ExVwREd0VCL3dRRUF3SUNCREFTQmd0VkhSTUJBZjhFQ0
        RBR0FRSC9BZ0VBTUEwr0NTcUdTSWIzRFFFQgpCUVVBQTRHqkFMMmx
        rWmFGNWcyaGR6MVNhZnZPbnBneHA4eG00SHRhbStadHpLazFlS3Bx
        TXp4YXJCbFpDSH1Lck1VbC9GVzRtV1RQS1VDeEtFTE40NEY2Zmk2d
        c4d0tSSElkYw1WL0pGTmlQS0VXSTF4K1I1aDZmazcrQzQ1QXg1RWV

```



```

    SWM2xZTgotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
  </certificate>
  <certificate>
    KS29aSWh2Y04KQVFFQkJRQURnWTBBTU1HskFvR0JBTXVvZmFPNEV3
    El1QWMrQ1RsTkNmc0d6cEw1Um5ydXZs0FRicUJTdGZQY3N0Zk1KT1
    FaNzlnN1NwVldsMldzaHE1bUViCkJNNitGNzdjbTAvU25FcFE0TnV
    bXBDT2YKQWdNQkFBR2pnYXd3Z2Frd0hRWURWUjBPQkJZRUZKY1o2W
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNrekNDQWZ5Z
    0F3SUJBZ0lKQUUpRT2t3bGpNK2pjTUEwR0NTcUdTSWIZRFFFQkJRVU
    FNRRF4Q3pBSkNjTlYkQkFZVEFsVlRNUkF3RGdZRFZRUUtFd2RsZUd
    GdGNHeGxNUk13RVFZRFZRUURFd3BEVWt3Z1NYTnpkV1Z5TUI0WApE
    diR1V4RXpBUkNjTlZCQU1UQ2t0U1RDQkpjM04xWlhJd2da0HdEUV1
    URiR0lPNDB4ajlPb3JtREdsRUNCVTFNR1FHQTFVZApJd1JkTUZ1QU
    RBR0FRSC9BZ0VBTUEwR0NTcUdTSWIZRFFFQgpCUVVBQTRHkFMmMx
    rWmFGNWcyaGR6MVNhZnZPbnBneHA4eG00SHRhbStadHpLazFlS3Bx
    c4d0tSSElkYw1WL0pGTmlQS0VXSTF4K1I1aDZmazcrQzQ1QXg1RWV
    SSUZJQ0FURS0tLS0tCg==
  </certificate>
</certificate-chain>
<certificate-chain>
  <name>my-ldevid-chain</name>
  <certificate>
    0F3SUJBZ0lKQUUpRT2t3bGpNK2pjTUEwR0NTcUdTSWIZRFFFQkJRVU
    FNRRF4Q3pBSkNjTlYkQkFZVEFsVlRNUkF3RGdZRFZRUUtFd2RsZUd
    GdGNHeGxNUk13RVFZRFZRUURFd3BEVWt3Z1NYTnpkV1Z5TUI0WApE
    diR1V4RXpBUkNjTlZCQU1UQ2t0U1RDQkpjM04xWlhJd2da0HdEUV1
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNrekNDQWZ5Z
    KS29aSWh2Y04KQVFFQkJRQURnWTBBTU1HskFvR0JBTXVvZmFPNEV3
    El1QWMrQ1RsTkNmc0d6cEw1Um5ydXZs0FRicUJTdGZQY3N0Zk1KT1
    FaNzlnN1NwVldsMldzaHE1bUViCkJNNitGNzdjbTAvU25FcFE0TnV
    ZKY1o2WURiR0lPNDB4ajlPb3JtREdsRUNCVTFvVGl1rTmPbME1Rc3d
    mMKTUE0R0ExVWREd0VCL3dRRUF3SUNCREFTQmdOVkhSTUJBZjhFQ0
    RBR0FRSC9BZ0VBTUEwR0NTcUdTSWIZRFFFQgpCUVVBQTRHkFMmMx
    rWmFGNWcyaGR6MVNhZnZPbnBneHA4eG00SHRhbStadHpLazFlS3Bx
    TXp4YXJCbFpDSHl1LCK1VbC9GVzRtV1RQS1VDeEtFTE40NEY2Zmk2d
    c4d0tSSElkYw1WL0pGTmlQS0VXSTF4K1I1aDZmazcrQzQ1QXg1RWV
    SWM2xZTgotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
  </certificate>
  <certificate>
    LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUNrekNDQWZ5Z
    0F3SUJBZ0lKQUUpRT2t3bGpNK2pjTUEwR0NTcUdTSWIZRFFFQkJRVU
    FNRRF4Q3pBSkNjTlYkQkFZVEFsVlRNUkF3RGdZRFZRUUtFd2RsZUd
    GdGNHeGxNUk13RVFZRFZRUURFd3BEVWt3Z1NYTnpkV1Z5TUI0WApE
    diR1V4RXpBUkNjTlZCQU1UQ2t0U1RDQkpjM04xWlhJd2da0HdEUV1
    KS29aSWh2Y04KQVFFQkJRQURnWTBBTU1HskFvR0JBTXVvZmFPNEV3
    El1QWMrQ1RsTkNmc0d6cEw1Um5ydXZs0FRicUJTdGZQY3N0Zk1KT1
    FaNzlnN1NwVldsMldzaHE1bUViCkJNNitGNzdjbTAvU25FcFE0TnV
    bXBDT2YKQWdNQkFBR2pnYXd3Z2Frd0hRWURWUjBPQkJZRUZKY1o2W

```



```

    URiR0lPNDB4ajlPb3JtREdsRUNCVTfNR1FHQTFVZApJd1JkTUZ1QU
    ZKY1o2WURiR0lPNDB4ajlPb3JtREdsRUNCVTfVvGlrTmpBME1Rc3d
    mMKTUE0R0ExVWREd0VCL3dRRUF3SUNCREFTQmd0VkhSTUJBZjhFQ0
    RBR0FRSC9BZ0VBTUEwR0NTcUdTSWIzRFFFQgpCUVVBQTRHqkFMMmx
    rWmFGNWcyaGR6MVNhZnZPbnBneHA4eG00SHRhStadHpLazFlS3Bx
    TXp4YXJCbFpDShLLcKlVbC9GVzRtV1RQS1VDeEtFTE40NEY2Zmk2d
    c4d0tSSElkYw1WL0pGTmlQS0VXSTF4K1I1aDZmazcrQzQ1QXg1RWV
    SWHgZzjdVM2xZTgotLS0tLUV0RCBDRVJUSUZJQ0FURS0tLS0tCg==
  </certificate>
</certificate-chain>
</certificate-chains>
</private-key>
</private-keys>

<!-- trusted netconf/restconf client certificates -->
<trusted-certificates>
  <name>explicitly-trusted-client-certs</name>
  <description>
    Specific client authentication certificates that are to be
    explicitly trusted NETCONF/RESTCONF clients. These are
    needed for client certificates not signed by our CA.
  </description>
  <trusted-certificate>
    <name>George Jetson</name>
    <certificate>
      Qmd0VkJBwVRBbFZUTVJBd0RnWURWUVFLRXdkbAp1R0Z0Y0d4bE1RNHdEQ
      MkF6a3hqUDlVQWtHR0dvs1U1eUc1SVR0Wm0vK3B0R2FieXVDMjBRd2kvZ
      25PZnpZNEhONApXY0pTaUpZK2xtYws3RTR0RUZXZS9RdGp4NU1XZmdvN2
      RV0JCU2t2MXI2SFNHeUFUVkpwSmYy0WtXbUU0NEo5akJrQmd0VkhTTUVY
      VEJiZ0JTWEdlbUEKMnhpRHV0TVkvVHFLNwd4cFJBZ1ZOYUU0cERZd05ER
      UxNQWtHQTFVRUJ0TUNwVnk14RURBT0JnTlZCQw9UQjJWNApZVzF3YkdVeE
      V6QVJCZ05WQkFNVENrTlNUQ0JKYzNOMVpYS0NDUUNVRHBNS1l6UG8zREF
      NQmd0VkhSTUJBZjhFckFqQUFNQTRHQTfVZER3RUIvd1FFQXdJSGdEQnBC
      Z05WSFI4RVlqQmdNRjZnSXFbZ2hoNW9kSFJ3T2k4dlpYaGgKYlhCc1pTN
      WpiMjB2WlhoaGJYQnNaUzVqY215aU9LUTJNRFF4Q3pBSkJnTlZCQVlUQW
      xWVE1SQXdEZ1lEVlFRSwpFd2RsZUdGdGNHeGxNUk13RVFZRFZRURFd3B
      EVWt3Z1NYTnpkV1Z5TUEwR0NTcUdTSWIzRFFFQkJRvUFBNEdCCKFFc3BK
      WmdsK2gyTTg3QmtGMjhWbW1CdFFVaWc30EgrRkYyRTFwdSt4ZVRJbVFFM
      TQzCjFZSjk0M1FQLzV5eGUKN2QxMkxCV0dxUjUrbE15N01YL21ka2M4a1
      zSFNwSDdwVXBCYnA4dmtNanFtZjJma3RqZHBxeFppUUtTbndWZTF2Zwot
      LS0tLUV0RCBDRVJUSUZJQ0FURS0tLS0tCg==
    </certificate>
  </trusted-certificate>
  <trusted-certificate>
    <name>Fred Flintstone</name>
    <certificate>
      V1EVlFRREv3Vm9ZWEJ3ZVRDQm56QU5CZ2txaGtpRz13MEJBUUVGQUFPQm
      pRQXdnWwtDCmdZRUE1RzRFSWZsS1p2bDlXTW44eUhyM2hObUFRaUHVUZV

```



```

rRUpPQy9hSFA3eGJXQW1ra054ZStUa2hrZnBsL3UKbVhsTjhsZUd10DhG
NGcEk3UE90cnNFVjRwTUNBd0VBQWFPQ0FSSXdnZ0VPck1CMEdBMVVKRGd
VEJiZ0JTWEdlbUEKMnhpRHVOTVkvVHFLNwd4cFJBZ1ZOYUU0cERZd05ER
V6QVJCZ05WQkFNVENrTlNUQ0JKYzNOMVpYS0NDUUNVRHBNSl16UG8zREF
NQmdOVkhSTUJBZjhFckFqQUFNQTRHQTFVZER3RUIvd1FFQXdJSgdeQnBC
Z05WSFI4RVlqQmdNRjZnSXFbZ2hoNW9kSFJ3T2k4d1pYaGgKYlhCc1pTN
WpiMjB2WlhoaGJYQnNaUzVqY215aU9LUTJNRFF4Q3pBSkJnTlZCQVlUQW
xWVE1SQXdEZ1lEVlFRSwpFd2RsZUdGdGNHeGxNUk13RVFZRFZRUURFd3B
EVWt3Z1NYTnpkV1Z5TUEwR0NTcUdTSWIZRFFFQkJRvUFBNEdCCkFFc3BK
WmdsK2gyTTg3QmtGMjhWbW1CdFFVaWc30EgrRkYyRTFwdSt4ZVRJbVFFM
lLQl1sdWpOcjFTMnRLR05EMUc20VJpK2FWNGw2NTdZNCtadVJMZgpRYjk
zSFNwSDdwVXBCYnA4dmtNanFtZjJma3RqZHBxeFppUUtTbndWZTF2Zwot
QwtUOCBDRVUUZJ0RUF==

```

```

</certificate>
</trusted-certificate>
</trusted-certificates>

<!-- trust anchors (CA certs) for netconf/restconf clients -->
<trusted-certificates>
  <name>deployment-specific-ca-certs</name>
  <description>
    Trust anchors used only to authenticate NETCONF/RESTCONF
    client connections. Since our security policy only allows
    authentication for clients having a certificate signed by
    our CA, we only configure its certificate below.
  </description>
  <trusted-certificate>
    <name>ca.example.com</name>
    <certificate>
      WmdsK2gyTTg3QmtGMjhWbW1CdFFVaWc30EgrRkYyRTFwdSt4ZVRJbVFFM
      lLQl1sdWpOcjFTMnRLR05EMUc20VJpK2FWNGw2NTdZNCtadVJMZgpRYjk
      zSFNwSDdwVXBCYnA4dmtNanFtZjJma3RqZHBxeFppUUtTbndWZTF2Zwot
      NGcEk3UE90cnNFVjRwTUNBd0VBQWFPQ0FSSXdnZ0VPck1CMEdBMVVKRGd
      VEJiZ0JTWEdlbUEKMnhpRHVOTVkvVHFLNwd4cFJBZ1ZOYUU0cERZd05ER
      V6QVJCZ05WQkFNVENrTlNUQ0JKYzNOMVpYS0NDUUNVRHBNSl16UG8zREF
      NQmdOVkhSTUJBZjhFckFqQUFNQTRHQTFVZER3RUIvd1FFQXdJSgdeQnBC
      Z05WSFI4RVlqQmdNRjZnSXFbZ2hoNW9kSFJ3T2k4d1pYaGgKYlhCc1pTN
      WpiMjB2WlhoaGJYQnNaUzVqY215aU9LUTJNRFF4Q3pBSkJnTlZCQVlUQW
      QmdOVkjbWVRBbFZUTVJBd0RnWURWUVFLRXdkbAp1R0Z0Y0d4bE1RNHdEQ
      MkF6a3hqUDlVQWtHR0dvs1U1eUc1SVR0Wm0vK3B0R2FieXVDMjBRd2kvZ
      25PZnpZNEhONApXY0pTaUpZK2xtYws3RTR0RUZXZS9RdGp4NUlXZmdvN2
      RJSUJQFRStS0Cg==
    </certificate>
  </trusted-certificate>
</trusted-certificates>

<!-- trust anchors for random HTTPS servers on Internet -->
<trusted-certificates>

```



```
<name>common-ca-certs</name>
<description>
  Trusted certificates to authenticate common HTTPS servers.
  These certificates are similar to those that might be
  shipped with a web browser.
</description>
<trusted-certificate>
  <name>ex-certificate-authority</name>
  <certificate>
    NGcEk3UE90cnNFVjRwTUNBd0VBQWFPQ0FSSXdnZ0VPck1CMEdBMVVKRGd
    VEJiZ0JTWEdlbUEKMnhpRHV0TVkvVHFLNwd4cFJBZ1ZOYUU0cERZd05ER
    V6QVJCZ05WQkFNVENrTlNUQ0JKYzNOMVpYS0NDUUNVRHBNS1l6UG8zREF
    Z05WSFI4RVlqQmdNRjZnSXFbZ2hoNW9kSFJ3T2k4d1pYaGgKYlhCc1pTN
    Qmd0VkJBwVRBbFZUTVJBd0RnWURWUVFLRXdkbAp1R0Z0Y0d4bE1RNHdEQ
    MkF6a3hqUDlVQWtHR0dvs1U1eUc1SVR0Wm0vK3B0R2FieXVDMjBRd2kvZ
    NQmd0VkhSTUJBZjhFckFqQUFNQTRHQTfVZER3RUIvd1FFQXdJSgdeQnBC
    WmdsK2gyTTg3QmtGMjhWbW1CdFFVaWc30EgrRkYyRTFwdSt4ZVRJbVFFM
    lLQl1sdWp0cjFTMnRLR05EMUc20VJpK2FWNGw2NTdZNCtadvJMZgpRYjk
    zSFNwSDdwVXBCYnA4dmtNanFtZjJma3RqZHBxeFppUUtTbndWZTF2Zwot
    25PZnpZNEhONApXY0pTaUpZK2xtYWs3RTR0RUZXZS9RdGp4NUlXZmdvN2
    WpiMjB2WlhoaGJYQnNaUzVqY215aU9L=
  </certificate>
</trusted-certificate>
</trusted-certificates>

<!-- trusted SSH host keys -->
<trusted-ssh-host-keys>
  <name>explicitly-trusted-ssh-host-keys</name>
  <description>
    Trusted SSH host keys used to authenticate SSH servers.
    These host keys would be analogous to those stored in
    a known_hosts file in OpenSSH.
  </description>
  <trusted-host-key>
    <name>corp-fw1</name>
    <host-key>
      VEJiZ0JTWEdlbUEKMnhpRHV0TVkvVHFLNwd4cFJBZ1ZOYUU0cERZd05ER
      NGcEk3UE90cnNFVjRwTUNBd0VBQWFPQ0FSSXdnZ0VPck1CMEdBMVVKRGd
      WpiMjB2WlhoaGJYQnNaUzVqY215aU9L=
    </host-key>
  </trusted-host-key>
</trusted-ssh-host-keys>

<!-- user credentials and associated authentication methods -->
<user-auth-credentials>
  <user-auth-credential>
    <username>admin</username>
    <auth-method>
```



```

    <priority>1</priority>
    <certificate-chain>my-ec-chain</certificate-chain>
    <certificate-chain>my-rsa-chain</certificate-chain>
  </auth-method>
  <auth-method>
    <priority>2</priority>
    <public-key>my-rsa-user-key</public-key>
  </auth-method>
</user-auth-credential>
<user-auth-credential>
  <username>tester</username>
  <auth-method>
    <priority>1</priority>
    <cleartext-password>testing123</cleartext-password>
  </auth-method>
</user-auth-credential>
<user-auth-credential>
  <username>ldevid</username>
  <auth-method>
    <priority>1</priority>
    <certificate-chain>my-ldevid-chain</certificate-chain>
  </auth-method>
</user-auth-credential>
</user-auth-credentials>

</keychain>

```

The following example illustrates a "certificate-expiration" notification in XML.

['\ ' line wrapping added 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-system-keychain">
    <certificate>
      /kc:keychain/kc:private-keys/kc:private-key/kc:certificate-chains\
      /kc:certificate-chain/kc:certificate[3]
    </certificate>
    <expiration-date>2016-08-08T14:18:53-05:00</expiration-date>
  </certificate-expiration>
</notification>

```


2.3. YANG Module

This YANG module makes extensive use of data types defined in [\[RFC5280\]](#) and [\[RFC5958\]](#).

```
<CODE BEGINS> file "ietf-system-keychain@2016-07-08.yang"

module ietf-system-keychain {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-system-keychain";
  prefix "kc";

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

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

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

    WG Chair: Mehmet Ersue
              <mailto:mehmet.ersue@nsn.com>

    WG Chair: Mahesh Jethanandani
              <mailto:mjethanandani@gmail.com>

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

  description
    "This module defines a keychain to centralize management of
    security credentials.

    Copyright (c) 2014 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 VVVV; see the RFC itself for full legal notices.";

```
revision "2016-07-08" {
  description
    "Initial version";
  reference
    "RFC VVVV: NETCONF Server and RESTCONF Server Configuration
      Models";
}

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

identity rsa {
  base key-algorithm;
  description
    "The RSA algorithm.";
  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.";
}

container keychain {
  description
    "A list of private-keys and their associated certificates, as
    well as lists of trusted certificates for client certificate
    authentication.  RPCs are provided to generate a new private
    key and to generate a certificate signing requests.";

  container private-keys {
    description
      "A list of private key maintained by the keychain.";
    list private-key {
      key name;
      description
        "A private key.";
      leaf name {
        type string;
        description
          "An arbitrary name for the private key.";
      }
      leaf algorithm {
        type identityref {
          base "key-algorithm";
        }
        config false;
        description
          "The algorithm used by the private key.";
      }
      leaf key-length {
        type uint32;
        config false;
        description
```



```
    "The key-length used by the private key.";
}
leaf public-key {
    type binary;
    config false;
    mandatory true;
    description
        "An OneAsymmetricKey 'publicKey' structure as specified
        by RFC 5958, Section 2 encoded using the ASN.1
        distinguished encoding rules (DER), as specified
        in ITU-T X.690.";
    reference
        "RFC 5958:
        Asymmetric Key Packages
        ITU-T X.690:
        Information technology - ASN.1 encoding rules:
        Specification of Basic Encoding Rules (BER),
        Canonical Encoding Rules (CER) and Distinguished
        Encoding Rules (DER).";
}
container certificate-chains {
    description
        "Certificate chains associated with this private key.
        More than one chain per key is enabled to support,
        for instance, a TPM-protected key that has associated
        both IDevID and LDevID certificates.";
    list certificate-chain {
        key name;
        description
            "A certificate chain for this public key.";
        leaf name {
            type string;
            description
                "An arbitrary name for the certificate chain. The
                name must be a unique across all private keys, not
                just within this private key.";
        }
    }
    leaf-list certificate {
        type binary;
        ordered-by user;
        description
            "An X.509 v3 certificate structure as specified by RFC
            5280, Section 4 encoded using the ASN.1 distinguished
            encoding rules (DER), as specified in ITU-T X.690.
            The list of certificates that run from the server
            certificate towards the trust anchor. The chain MAY
            include the trust anchor certificate itself.";
        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).";
}
}
}
action generate-certificate-signing-request {
  description
    "Generates a certificate signing request structure for
    the associated private key using the passed subject and
    attribute values. Please review both the Security
    Considerations and Design Considerations sections in
    RFC VVVV for more information regarding this action
    statement.";
  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).";
    }
}
}
}

action generate-private-key {
    description
        "Requests the device to generate a private key using the
        specified algorithm and key length.";
    input {
        leaf name {
            type string;
            mandatory true;
            description
                "The name this private-key should have when listed
                in /keychain/private-keys. As such, the passed
                value must not match any existing 'name' value.";
        }
        leaf algorithm {
            type identityref {
                base "key-algorithm";
            }
        }
    }
}
```



```
        mandatory true;
        description
            "The algorithm to be used when generating the key.";
    }
    leaf key-length {
        type uint32;
        description
            "For algorithms that need a key length specified
            when generating the key.";
    }
}
}

action load-private-key {
    description
        "Requests the device to load a private key";
    input {
        leaf name {
            type string;
            mandatory true;
            description
                "The name this private-key should have when listed
                in /keychain/private-keys. As such, the passed
                value must not match any existing 'name' value.";
        }
        leaf private-key {
            type binary;
            mandatory true;
            description
                "An OneAsymmetricKey structure as specified by RFC
                5958, Section 2 encoded using the ASN.1 distinguished
                encoding rules (DER), as specified in ITU-T X.690.
                Note that this is the raw private with no shrouding
                to protect it. The strength of this private key
                MUST NOT be greater than the strength of the secure
                connection over which it is communicated. Devices
                SHOULD fail this request if ever that happens.";
        }
        reference
            "RFC 5958:
            Asymmetric Key Packages
            ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
    }
}
}
```



```
}  
  
list trusted-certificates {  
  key name;  
  description  
    "A list of trusted certificates.  These certificates  
    can be used by a server to authenticate clients, or by clients  
    to authenticate servers.  The certificates may be endpoint  
    specific or for certificate authorities (to authenticate many  
    clients at once.  Each list of certificates SHOULD be specific  
    to a purpose, as the list as a whole may be referenced by other  
    modules.  For instance, a NETCONF server model might point to  
    a list of certificates to use when authenticating client  
    certificates.";  
  leaf name {  
    type string;  
    description  
      "An arbitrary name for this list of trusted certificates.";  
  }  
  leaf description {  
    type string;  
    description  
      "An arbitrary description for this list of trusted  
      certificates.";  
  }  
  list trusted-certificate {  
    key name;  
    description  
      "A trusted certificate for a specific use.  Note, this  
      'certificate' is a list in order to encode any  
      associated intermediate certificates.";  
    leaf name {  
      type string;  
      description  
        "An arbitrary name for this trusted certificate.  Must  
        be unique across all lists of trusted certificates  
        (not just this list) so that a leafref to it from  
        another module can resolve to unique values.";  
    }  
    leaf certificate { // rename to 'data'?  
      type binary;  
      description  
        "An X.509 v3 certificate structure as specified by RFC  
5280, Section 4 encoded using the 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).";
  }
}
}

list trusted-ssh-host-keys {
  key name;
  description
    "A list of trusted host-keys. These host-keys can be used
    by clients to authenticate SSH servers. The host-keys are
    endpoint specific. Each list of host-keys SHOULD be
    specific to a purpose, as the list as a whole may be
    referenced by other modules. For instance, a NETCONF
    client model might point to a list of host-keys to use
    when authenticating servers host-keys.";
  leaf name {
    type string;
    description
      "An arbitrary name for this list of trusted SSH host keys.";
  }
  leaf description {
    type string;
    description
      "An arbitrary description for this list of trusted SSH host
      keys.";
  }
}

list trusted-host-key {
  key name;
  description
    "A trusted host key.";
  leaf name {
    type string;
    description
      "An arbitrary name for this trusted host-key. Must be
      unique across all lists of trusted host-keys (not just
      this list) so that a leafref to it from another module
      can resolve to unique values.

```

Note that, for when the SSH client is able to listen for call-home connections as well, there is no reference identifier (e.g., hostname, IP address, etc.) that it can use to uniquely identify the server with. The call-home draft recommends SSH servers use X.509v3


```

        certificates (RFC6187) when calling home.";
    }
    leaf host-key { // rename to 'data'?
        type binary;
        mandatory true;
        description
            "An OneAsymmetricKey 'publicKey' structure as specified
            by RFC 5958, Section 2 encoded using the ASN.1
            distinguished encoding rules (DER), as specified
            in ITU-T X.690.";
        reference
            "RFC 5958:
            Asymmetric Key Packages
            ITU-T X.690:
            Information technology - ASN.1 encoding rules:
            Specification of Basic Encoding Rules (BER),
            Canonical Encoding Rules (CER) and Distinguished
            Encoding Rules (DER).";
    }
}
}
}

/*
Are the auth credentials truly limited to SSH?
Could they be used by an HTTP client to log into an HTTP server?
If truly just for SSH, maybe rename?
*/
    container user-auth-credentials {
        description
            "A list of user authentication credentials that can be used
            by an SSH client to log into an SSH server, using any of
            the supported authentication methods (e.g., password,
            public key, client certificate, etc.).";
        list user-auth-credential {
            key username;
            description
                "The authentication credentials for a specific user.";
            leaf username {
                type string;
                description
                    "The username of this user. This will be the username
                    used, for instance, to log into an SSH server.";
            }
        }
        list auth-method {
            key priority;
            description
                "A method of authenticating as this user.";
            leaf priority {

```



```
    type uint8;
    description
      "When multiple authentication methods in this list are
       supported by the server, the one with the lowest priority
       value will be the one that is used.";
  }
choice auth-type {
  description
    "The authentication type.";
  leaf-list certificate {
    type leafref {
      path "/keychain/private-keys/private-key/"
        + "certificate-chains/certificate-chain/name";
    }
    ordered-by user;
    description
      "A list of references to certificates that can be used for
       user authentication. When multiple certificates in this
       list supported by the server, the one that comes
       before the others in the leaf-list will be used.";
  }
  leaf-list public-key {
    type leafref {
      path "/keychain/private-keys/private-key/name";
    }
    ordered-by user;
    description
      "A list of references to public keys that can be used for
       user authentication. When multiple public keys in this
       list supported by the server, the one that comes
       before the others in the leaf-list will be used.";
  }
  leaf ciphertext-password {
    type string;
    description
      "An ciphertext password. The method of encipherment and
       how that method can be determined from this string is
       implementation-specific.";
  }
  leaf cleartext-password {
    type string;
    description
      "An cleartext password.";
  }
}
}
```



```
}  
  
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>

3. Design Considerations

This document uses PKCS #10 [[RFC2986](#)] for the "generate-certificate-signing-request" action. The use of Certificate Request Message Format (CRMF) [[RFC4211](#)] was considered, but it was unclear if there was market demand for it, and so support for CRMF has been left out of this specification. If it is desired to support CRMF in the future, placing a "choice" statement in both the input and output statements, along with an "if-feature" statement on the CRMF option, would enable a backwards compatible solution.

This document puts a limit on the number of elliptical curves supported by default. This was done to match industry trends in IETF best practice (e.g., matching work being done in TLS 1.3). If additional algorithms are needed, they MAY be augmented in by another module, or added directly in a future version of this document.

Both this document and Key Chain YANG Data Model [[draft-ietf-rtgwg-yang-key-chain](#)] define keychain YANG modules. The

authors looked at this and agree that they two modules server different purposes and hence not worth merging into one document. To underscore this further, this document renamed its module from "ietf-keychain" to "ietf-system-keychain" and that other document renamed its module from "ietf-key-chain" to "ietf-routing-key-chain".

For the trusted-certificates list, Trust Anchor Format [[RFC5914](#)] was evaluated and deemed inappropriate due to this document's need to also support pinning. That is, pinning a client-certificate to support NETCONF over TLS client authentication.

4. Security Considerations

This document defines a keychain mechanism that is entrusted with the safe keeping of private keys, and the safe keeping of trusted certificates. Nowhere in this API is there an ability to access (read out) a private key once it is known to the keychain. Further, associated public keys and attributes (e.g., algorithm name, key length, etc.) are read-only. That said, this document allows for the deletion of private keys and their certificates, as well the deletion of trusted certificates. Access control mechanisms (e.g., NACM [[RFC6536](#)]) MUST be in place so as to authorize such client actions. Further, whilst the data model allows for private keys and trusted certificates in general to be deleted, implementations should be well aware that some private keys (e.g., those in a TPM) and some trusted certificates, should never be deleted, regardless if the authorization mechanisms would generally allow for such actions.

For the "generate-certificate-signing-request" action, it is RECOMMENDED that devices implement assert channel binding [[RFC5056](#)], so as to ensure that the application layer that sent the request is the same as the device authenticated in the secure transport layer was established.

This document defines a data model that includes a list of private keys. These private keys MAY be deleted using standard NETCONF or RESTCONF operations (e.g., <edit-config>). Implementations SHOULD automatically (without explicit request) zeroize these keys in the most secure manner available, so as to prevent the remnants of their persisted storage locations from being analyzed in any meaningful way.

The keychain module define within this document defines the "load-private-key" action enabling a device to load a client-supplied private key. This is a private key with no shrouding to protect it. The strength of this private key MUST NOT be greater than the strength of the underlying secure transport connection over which it is communicated. Devices SHOULD fail this request if ever the

strength of the private key is greater than the strength of the underlying transport.

5. IANA Considerations

5.1. The IETF XML Registry

This document registers one URI in the IETF XML registry [[RFC2119](#)]. Following the format in [[RFC3688](#)], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-system-keychain
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 one YANG module in the YANG Module Names registry [[RFC6020](#)]. Following the format in [[RFC6020](#)], the the following registration is requested:

name: ietf-system-keychain
namespace: urn:ietf:params:xml:ns:yang:ietf-system-keychain
prefix: kc
reference: RFC VVVV

6. Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): Andy Bierman, Martin Bjorklund, Benoit Claise, Mehmet Ersue, David Lamparter, Alan Luchuk, Ladislav Lhotka, Radek Krejci, Tom Petch, Juergen Schoenwaelder; Phil Shafer, Sean Turner, and Bert Wijnen.

7. References

7.1. Normative References

[[draft-ietf-netconf-restconf](#)]

Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", [draft-ietf-netconf-restconf-04](#) (work in progress), 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://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, <<http://www.rfc-editor.org/info/rfc2986>>.
- [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, <<http://www.rfc-editor.org/info/rfc5280>>.
- [RFC5958] Turner, S., "Asymmetric Key Packages", [RFC 5958](#), DOI 10.17487/RFC5958, August 2010, <<http://www.rfc-editor.org/info/rfc5958>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.

7.2. Informative References

- [[draft-ietf-rtgwg-yang-key-chain](#)]
Lindem, A., Qu, Y., Yeung, D., Chen, I., Zhang, J., and Y. Yang, "Key Chain YANG Data Model", [draft-ietf-rtgwg-yang-key-chain](#) (work in progress), 2016, <<https://datatracker.ietf.org/html/draft-ietf-rtgwg-yang-key-chain>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), DOI 10.17487/RFC3688, January 2004, <<http://www.rfc-editor.org/info/rfc3688>>.
- [RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", [RFC 4211](#), DOI 10.17487/RFC4211, September 2005, <<http://www.rfc-editor.org/info/rfc4211>>.
- [RFC5056] Williams, N., "On the Use of Channel Bindings to Secure Channels", [RFC 5056](#), DOI 10.17487/RFC5056, November 2007, <<http://www.rfc-editor.org/info/rfc5056>>.

- [RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Format", [RFC 5914](#), DOI 10.17487/RFC5914, June 2010, <<http://www.rfc-editor.org/info/rfc5914>>.
- [RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", [RFC 6536](#), DOI 10.17487/RFC6536, March 2012, <<http://www.rfc-editor.org/info/rfc6536>>.
- [Std-802.1AR-2009]
IEEE SA-Standards Board, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", December 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.

[Appendix A](#). Change Log

[A.1](#). server-model-09 to 00

- o This draft was split out from [draft-ietf-netconf-server-model-09](#).
- o Removed key-usage parameter from generate-private-key action.
- o Now /private-keys/private-key/certificates/certificate/name must be globally unique (unique across all private keys).
- o Added top-level 'trusted-ssh-host-keys' and 'user-auth-credentials' to support SSH client modules.

[Appendix B](#). Open Issues

Please see: <https://github.com/netconf-wg/system-keychain/issues>.

Authors' Addresses

Kent Watsen
Juniper Networks

EMail: kwatsen@juniper.net

Gary Wu
Cisco Networks

EMail: garywu@cisco.com

