Network Working Group   A. Lindem, Ed.
Internet-Draft   Y. Qu
Intended status: Standards Track   Cisco Systems
Expires: April 30, 2017   D. Yeung
                                                             Arrcus, Inc
                                                                 I. Chen
                                                                Ericsson
                                                                J. Zhang
                                                        Juniper Networks
                                                                 Y. Yang
                                                  Individual Contributor
                                                        October 27, 2016

Routing Key Chain YANG Data Model
draft-ietf-rtgwg-yang-key-chain-10.txt

Abstract

This document describes the key chain YANG data model. A key chain is a list of elements each containing a key, send lifetime, accept lifetime, and algorithm (authentication or encryption). By properly overlapping the send and accept lifetimes of multiple key chain elements, keys and algorithms may be gracefully updated. By representing them in a YANG data model, key distribution can be automated. Key chains are commonly used for routing protocol authentication and other applications. In some applications, the protocols do not use the key chain element key directly, but rather a key derivation function is used to derive a short-lived key from the key chain element key (e.g., the Master Keys used in the TCP Authentication Option.

Status of This Memo

Table of Contents

1. Introduction

This document describes the key chain YANG data model. A key chain is a list of elements each containing a key, send lifetime, accept lifetime, and algorithm (authentication or encryption). By properly overlapping the send and accept lifetimes of multiple key chain elements, keys and algorithms may be gracefully updated. By representing them in a YANG data model, key distribution can be automated. Key chains are commonly used for routing protocol authentication and other applications. In some applications, the protocols do not use the key chain element key directly, but rather a

key derivation function is used to derive a short-lived key from the key chain element key (e.g., the Master Keys used in [TCP-AO]).

1.1. Requirements Notation

The key words "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-KEYWORDS].

1.2. Tree Diagrams

A simplified graphical representation of the complete data tree is presented in Section 3.3. The following tree notation is used.

- o Brackets "[" and "]" enclose list keys.

- o Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.

- o Abbreviations before data node names: "rw" means configuration (read-write), "ro" state data (read-only), "-x" RPC operations, and "-n" notifications.

- o Symbols after data node names: "?" means an optional node, "!" a container with presence, and "*" denotes a "list" or "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. Problem Statement

This document describes a YANG [YANG] data model for key chains. Key chains have been implemented and deployed by a large percentage of network equipment vendors. Providing a standard YANG model will facilitate automated key distribution and non-disruptive key rollover. This will aid in tightening the security of the core routing infrastructure as recommended in [IAB-REPORT].

A key chain is a list containing one or more elements containing a Key ID, key, send/accept lifetimes, and the associated authentication or encryption algorithm. A key chain can be used by any service or application requiring authentication or encryption. In essence, the key-chain is a reusable key policy that can be referenced where ever it is required. The key-chain construct has been implemented by most networking vendors and deployed in many networks.

The module name was change from ietf-key-chain to ietf-routing-key-chain to avoid disambiguate it from the ietf-system-keychain module defined in [NETCONF-SERVER-CONF]. However, due to popular demand, the module name has been restored to simply ietf-key-chain.

A conceptual representation of a crypto key table is described in [CRYPTO-KEYTABLE]. The crypto key table also includes keys as well as their corresponding lifetimes and algorithms. Additionally, the key table includes key selection criteria and envisions a deployment model where the details of the applications or services requiring authentication or encryption permeate into the key database. The YANG key-chain model described herein doesn't include key selection criteria or support this deployment model. At the same time, it does not preclude it. The draft [YANG-CRYPTO-KEYTABLE] describes augmentations to the key chain YANG model in support of key selection criteria.

2.1. Applicability

Other YANG modules may reference ietf-key-chain YANG module key-chain names for authentication and encryption applications. A YANG type has been provided to facilate reference to the key-chain name without having to specify the complete YANG XML Path Language (XPath) selector.

2.2. Graceful Key Rollover using Key Chains

Key chains may be used to gracefully update the key and/or algorithm used by an application for authentication or encryption. This MAY be accomplished by accepting all the keys that have a valid accept lifetime and sending the key with the most recent send lifetime. One scenario for facilitating key rollover is to:

1. Distribute a key chain with a new key to all the routers or other network devices in the domain of that key chain. The new key's accept lifetime should be such that it is accepted during the key rollover period. The send lifetime should be a time in the future when it can be assured that all the routers in the domain of that key are upgraded. This will have no immediate impact on the keys used for transmission.

2. Assure that all the network devices have been updated with the updated key chain and that their system times are roughly synchronized. The system times of devices within an administrative domain are commonly synchronized (e.g., using Network Time Protocol (NTP) [NTP-PROTO]). This also may be automated.

3. When the send lifetime of the new key becomes valid, the network devices within the domain of key chain will start sending the new key.

4. At some point in the future, a new key chain with the old key removed may be distributed to the network devices within the domain of the key chain. However, this may be deferred until the next key rollover. If this is done, the key chain will always include two keys; either the current and future key (during key rollovers) or the current and previous keys (between key rollovers).

3. Design of the Key Chain Model

The ietf-key-chain module contains a list of one or more keys indexed by a Key ID. For some applications (e.g., OSPFv3 [OSPFV3-AUTH]), the Key-Id is used to identify the key chain entry to be used. In addition to the Key-ID, each key chain entry includes a key-string and a cryptographic algorithm. Optionally, the key chain entries include send/accept lifetimes. If the send/accept lifetime is unspecified, the key is always considered valid.

Note that asymmetric keys, i.e., a different key value used for transmission versus acceptance, may be supported with multiple key chain elements where the accept-lifetime or send-lifetime is not valid (e.g., has an end-time equal to the start-time).

Due to the differences in key chain implementations across various vendors, some of the data elements are optional. Additionally, the key-chain is made a grouping so that an implementation could support scoping other than at the global level. Finally, the crypto-algorithm-types grouping is provided for reuse when configuring legacy authentication and encryption not using key-chains.

A key-chain is identified by a unique name within the scope of the network device. The "key-chain-ref" typedef SHOULD be used by other YANG modules when they need to reference a configured key-chain.

3.1. Key Chain Operational State

The key chain operational state is maintained in a separate tree. The key string itself is omitted from the operational state to minimize visibility similar to what was done with keys in SNMP MIBs. The timestamp of the last key-chain modification is also maintained in the operational state. Additionally, the operational state includes an indication of whether or not a key chain entry is valid for sending or acceptance.

3.2. Key Chain Model Features

Features are used to handle differences between vendor implementations. For example, not all vendors support configuration an acceptance tolerance or configuration of key strings in hexadecimal. They are also used to support of security requirements (e.g., TCP-AO Algorithms [TCP-AO-ALGORITHMS]) not implemented by vendors or only a single vendor.

3.3. Key Chain Model Tree

```
+--rw key-chain
|  +--rw key-chain-list* [name]
|  |  +--rw name                  string
|  |  +--rw description?          string
|  |  +--rw accept-tolerance {accept-tolerance}?
|  |  |  +--rw duration?   uint32
|  |  +--rw key-chain-entries* [key-id]
|  |     +--rw key-id               uint64
|  |     +--rw lifetime
|  |     |  +--rw (lifetime)?
|  |     |     +--:(send-and-accept-lifetime)
|  |     |     |  +--rw send-accept-lifetime
|  |     |     |     +--rw (lifetime)?
|  |     |     |        +--:(always)
|  |     |     |        |  +--rw always?            empty
|  |     |     |        +--:(start-end-time)
|  |     |     |           +--rw start-date-time?
|  |     |     |           |    yang:date-and-time
|  |     |     |           +--rw (end-time)?
|  |     |     |              +--:(infinite)
|  |     |     |              |  +--rw no-end-time?       empty
|  |     |     |              +--:(duration)
|  |     |     |              |  +--rw duration?          uint32
|  |     |     |              +--:(end-date-time)
|  |     |     |                 +--rw end-date-time?
|  |     |     |                      yang:date-and-time
|  |     |     +--:(independent-send-accept-lifetime)
|  |     |             {independent-send-accept-lifetime}?
|  |     |        +--rw send-lifetime
|  |     |        |  +--rw (lifetime)?
|  |     |        |     +--:(always)
|  |     |        |     |  +--rw always?            empty
|  |     |        |     +--:(start-end-time)
|  |     |        |        +--rw start-date-time?
|  |     |        |        |    yang:date-and-time
|  |     |        |        +--rw (end-time)?
|  |     |        |           +--:(infinite)
```

```
   |  |     |        |           |  +--rw no-end-time?       empty
   |  |     |        |           +--:(duration)
   |  |     |        |           |  +--rw duration?          uint32
   |  |     |        |           +--:(end-date-time)
   |  |     |        |              +--rw end-date-time?
   |  |     |        |                   yang:date-and-time
   |  |     |        +--rw accept-lifetime
   |  |     |           +--rw (lifetime)?
   |  |     |              +--:(always)
   |  |     |              |  +--rw always?            empty
   |  |     |              +--:(start-end-time)
   |  |     |                 +--rw start-date-time?
   |  |     |                 |     yang:date-and-time
   |  |     |                 +--rw (end-time)?
   |  |     |                    +--:(infinite)
   |  |     |                    |  +--rw no-end-time?       empty
   |  |     |                    +--:(duration)
   |  |     |                    |  +--rw duration?          uint32
   |  |     |                    +--:(end-date-time)
   |  |     |                       +--rw end-date-time?
   |  |     |                            yang:date-and-time
   |  |     +--rw crypto-algorithm
   |  |     |  +--rw (algorithm)?
   |  |     |     +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
   |  |     |     |  +--rw hmac-sha1-12?            empty
   |  |     |     +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
   |  |     |     |  +--rw aes-cmac-prf-128?        empty
   |  |     |     +--:(md5)
   |  |     |     |  +--rw md5?                     empty
   |  |     |     +--:(sha-1)
   |  |     |     |  +--rw sha-1?                   empty
   |  |     |     +--:(hmac-sha-1)
   |  |     |     |  +--rw hmac-sha-1?              empty
   |  |     |     +--:(hmac-sha-256)
   |  |     |     |  +--rw hmac-sha-256?            empty
   |  |     |     +--:(hmac-sha-384)
   |  |     |     |  +--rw hmac-sha-384?            empty
   |  |     |     +--:(hmac-sha-512)
   |  |     |     |  +--rw hmac-sha-512?            empty
   |  |     |     +--:(clear-text) {clear-text}?
   |  |     |     |  +--rw clear-text?              empty
   |  |     |     +--:(replay-protection-only) {replay-protection-only}?
   |  |     |        +--rw replay-protection-only?   empty
   |  |     +--rw key-string
   |  |        +--rw (key-string-style)?
   |  |           +--:(keystring)
   |  |           |  +--rw keystring?            string
   |  |           +--:(hexadecimal) {hex-key-string}?
```

```
   |  |              +--rw hexadecimal-string?   yang:hex-string
   |  +--rw aes-key-wrap {aes-key-wrap}?
   |     +--rw enable?   boolean
   +--ro key-chain-state
      +--ro key-chain-list* [name]
      |  +--ro name                 string
      |  +--ro description?         string
      |  +--ro accept-tolerance {accept-tolerance}?
      |  |  +--ro duration?   uint32
      |  +--ro key-chain-entries* [key-id]
      |     +--ro key-id                    uint64
      |     +--ro lifetime
      |     |  +--ro (lifetime)?
      |     |     +--:(send-and-accept-lifetime)
      |     |     |  +--ro send-accept-lifetime
      |     |     |     +--ro (lifetime)?
      |     |     |        +--:(always)
      |     |     |        |  +--ro always?            empty
      |     |     |        +--:(start-end-time)
      |     |     |           +--ro start-date-time?
      |     |     |           |    yang:date-and-time
      |     |     |           +--ro (end-time)?
      |     |     |              +--:(infinite)
      |     |     |              |  +--ro no-end-time?       empty
      |     |     |              +--:(duration)
      |     |     |              |  +--ro duration?          uint32
      |     |     |              +--:(end-date-time)
      |     |     |                 +--ro end-date-time?
      |     |     |                         yang:date-and-time
      |     |     +--:(independent-send-accept-lifetime)
      |     |              {independent-send-accept-lifetime}?
      |     |        +--ro send-lifetime
      |     |        |  +--ro (lifetime)?
      |     |        |     +--:(always)
      |     |        |     |  +--ro always?            empty
      |     |        |     +--:(start-end-time)
      |     |        |        +--ro start-date-time?
      |     |        |        |    yang:date-and-time
      |     |        |        +--ro (end-time)?
      |     |        |           +--:(infinite)
      |     |        |           |  +--ro no-end-time?       empty
      |     |        |           +--:(duration)
      |     |        |           |  +--ro duration?          uint32
      |     |        |           +--:(end-date-time)
      |     |        |              +--ro end-date-time?
      |     |        |                      yang:date-and-time
      |     |        +--ro accept-lifetime
      |     |           +--ro (lifetime)?
```

```
      |     |              +--:(always)
      |     |              |  +--ro always?            empty
      |     |              +--:(start-end-time)
      |     |                 +--ro start-date-time?
      |     |                 |    yang:date-and-time
      |     |                 +--ro (end-time)?
      |     |                    +--:(infinite)
      |     |                    |  +--ro no-end-time?       empty
      |     |                    +--:(duration)
      |     |                    |  +--ro duration?          uint32
      |     |                    +--:(end-date-time)
      |     |                       +--ro end-date-time?
      |     |                            yang:date-and-time
      |     +--ro crypto-algorithm
      |     |  +--ro (algorithm)?
      |     |     +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
      |     |     |  +--ro hmac-sha1-12?             empty
      |     |     +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
      |     |     |  +--ro aes-cmac-prf-128?         empty
      |     |     +--:(md5)
      |     |     |  +--ro md5?                      empty
      |     |     +--:(sha-1)
      |     |     |  +--ro sha-1?                    empty
      |     |     +--:(hmac-sha-1)
      |     |     |  +--ro hmac-sha-1?               empty
      |     |     +--:(hmac-sha-256)
      |     |     |  +--ro hmac-sha-256?             empty
      |     |     +--:(hmac-sha-384)
      |     |     |  +--ro hmac-sha-384?             empty
      |     |     +--:(hmac-sha-512)
      |     |     |  +--ro hmac-sha-512?             empty
      |     |     +--:(clear-text) {clear-text}?
      |     |     |  +--ro clear-text?               empty
      |     |     +--:(replay-protection-only) {replay-protection-only}?
      |     |        +--ro replay-protection-only?   empty
      |     +--ro send-lifetime-active?     boolean
      |     +--ro accept-lifetime-active?   boolean
      +--ro aes-key-wrap {aes-key-wrap}?
         +--ro enable?   boolean
```

4. Key Chain YANG Model

```
   <CODE BEGINS> file "ietf-key-chain@2016-10-27.yang"
   module ietf-key-chain {
       namespace "urn:ietf:params:xml:ns:yang:ietf-key-chain";
       // replace with IANA namespace when assigned
       prefix "key-chain";
```

```
      import ietf-yang-types {
          prefix "yang";
       }

       organization
          "IETF RTG (Routing) Working Group";
       contact
          "Acee Lindem - acee@cisco.com";

       description
          "This YANG module defines the generic configuration
        data for key-chain. It is intended that the module
        will be extended by vendors to define vendor-specific
        key-chain configuration parameters.

        Copyright (c) 2015 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Simplified BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).
        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

       revision 2016-10-27 {
           description
               "Restructure into separate config and state trees to
                match YANG structure.";
           reference
               "RFC XXXX: A YANG Data Model for key-chain";
       }
       revision 2016-08-17 {
           description
               "Add description and last-modified timestamp leaves.";
           reference
               "RFC XXXX: A YANG Data Model for key-chain";
       }
       revision 2016-07-01 {
           description
               "Rename module back to ietf-key-chain.
                Added replay-protection-only feature and algorithm.";
           reference
               "RFC XXXX: A YANG Data Model for key-chain";
       }
       revision 2016-03-15 {
```

```
         description
             "Rename module from ietf-key-chain to
        ietf-routing-key-chain.";
         reference
             "RFC XXXX: A YANG Data Model for Routing key-chain";
     }
     revision 2016-02-16 {
         description
             "Updated version. Added clear-text algorithm as a
        feature.";
         reference
             "RFC XXXX: A YANG Data Model for key-chain";
     }
     revision 2015-10-15 {
         description
             "Updated version, organization, and copyright.
              Added aes-cmac-prf-128 and aes-key-wrap features.";
         reference
             "RFC XXXX: A YANG Data Model for key-chain";
     }
     revision 2015-06-29 {
         description
             "Updated version. Added Operation State following
              draft-openconfig-netmod-opstate-00.";
         reference
             "RFC XXXX: A YANG Data Model for key-chain";
     }
     revision 2015-02-24 {
         description
             "Initial revision.";
         reference
             "RFC XXXX: A YANG Data Model for key-chain";
     }

     typedef key-chain-ref {
         type leafref {
             path "/key-chain:key-chain/key-chain:key-chain-list/"
                 + "key-chain:name";
         }
         description
             "This type is used by data models that need to reference
              configured key-chains.";
     }

     /* feature list */
     feature hex-key-string {
         description
             "Support hexadecimal key string.";
```

```
       }

       feature accept-tolerance {
           description
               "To specify the tolerance or acceptance limit.";
       }

       feature independent-send-accept-lifetime {
           description
               "Support for independent send and accept key lifetimes.";
       }

       feature crypto-hmac-sha-1-12 {
           description
               "Support for TCP HMAC-SHA-1 12 byte digest hack.";
       }

       feature clear-text {
           description
               "Support for clear-text algorithm. Usage is
                NOT RECOMMENDED.";
       }

       feature aes-cmac-prf-128 {
           description
               "Support for AES Cipher based Message Authentication
                Code Pseudo Random Function.";
       }

       feature aes-key-wrap {
           description
               "Support for Advanced Encryption Standard (AES)
                Key Wrap.";
       }

       feature replay-protection-only {
           description
               "Provide replay-protection without any authentication
                as required by protocols such as Bidirectional
                Forwarding Detection (BFD).";
       }

       /* groupings */
       grouping lifetime {
           description
               "Key lifetime specification.";
           choice lifetime {
               default always;
```

```
            description
                "Options for specifying key accept or send
                 lifetimes";
            case always {
                leaf always {
                    type empty;
                    description
                        "Indicates key lifetime is always valid.";
                }
            }
            case start-end-time {
                leaf start-date-time {
                    type yang:date-and-time;
                    description "Start time.";
                }
                choice end-time {
                    default infinite;
                    description
                        "End-time setting.";
                    case infinite {
                        leaf no-end-time {
                            type empty;
                            description
                                "Indicates key lifetime end-time in
                                 infinite.";
                        }
                    }
                    case duration {
                        leaf duration {
                            type uint32 {
                                range "1..2147483646";
                            }
                            units seconds;
                            description "Key lifetime duration,
                           in seconds";
                        }
                    }
                    case end-date-time {
                        leaf end-date-time {
                            type yang:date-and-time;
                            description "End time.";
                        }
                    }
                }
            }
        }
    }
```

```
       grouping crypto-algorithm-types {
           description "Cryptographic algorithm types.";
           choice algorithm {
               description
                   "Options for cryptographic algorithm specification.";
               case hmac-sha-1-12 {
                   if-feature crypto-hmac-sha-1-12;
                   leaf hmac-sha1-12 {
                       type empty;
                       description "The HMAC-SHA1-12 algorithm.";
                   }
               }
               case aes-cmac-prf-128 {
                   if-feature aes-cmac-prf-128;
                   leaf aes-cmac-prf-128 {
                       type empty;
                       description "The AES-CMAC-PRF-128 algorithm -
                                    required by RFC 5926 for TCP-AO key
                                    derivation functions.";
                   }
               }
               case md5 {
                   leaf md5 {
                       type empty;
                       description "The MD5 algorithm.";
                   }
               }
               case sha-1 {
                   leaf sha-1 {
                       type empty;
                       description "The SHA-1 algorithm.";
                   }
               }
               case hmac-sha-1 {
                   leaf hmac-sha-1 {
                       type empty;
                       description
                           "HMAC-SHA-1 authentication algorithm.";
                   }
               }
               case hmac-sha-256 {
                   leaf hmac-sha-256 {
                       type empty;
                       description
                           "HMAC-SHA-256 authentication algorithm.";
                   }
               }
               case hmac-sha-384 {
```

```
                  leaf hmac-sha-384 {
                      type empty;
                      description
                          "HMAC-SHA-384 authentication algorithm.";
                  }
              }
              case hmac-sha-512 {
                  leaf hmac-sha-512 {
                      type empty;
                      description
                          "HMAC-SHA-512 authentication algorithm.";
                  }
              }
              case clear-text {
                  if-feature clear-text;
                  leaf clear-text {
                      type empty;
                      description "Clear text.";
                  }
              }
              case replay-protection-only {
                  if-feature replay-protection-only;
                  leaf replay-protection-only {
                      type empty;
                      description
                          "Provide replay-protection without any
                           authentication as required by protocols
                           such as Bidirectional Forwarding
                           Detection (BFD).";
                  }
              }
          }
      }

      grouping key-chain-common-entry {
          description "Key-chain entry data nodes common to
                       configuration and state.";
          container lifetime {
              description "Specify a key's lifetime.";
              choice lifetime {
                  description
                      "Options for specification of send and accept
            lifetimes.";
                  case send-and-accept-lifetime {
                      description
                          "Send and accept key have the same
                           lifetime.";
                      container send-accept-lifetime {
```

```
                            uses lifetime;
                            description
                                "Single lifetime specification for both
                                 send and accept lifetimes.";
                        }
                    }
                    case independent-send-accept-lifetime {
                        if-feature independent-send-accept-lifetime;
                        description
                            "Independent send and accept key lifetimes.";
                        container send-lifetime {
                            uses lifetime;
                            description
                                "Separate lifetime specification for send
                                 lifetime.";
                        }
                        container accept-lifetime {
                            uses lifetime;
                            description
                                "Separate lifetime specification for
                                 accept lifetime.";
                        }
                    }
                }
            }
            container crypto-algorithm {
                uses crypto-algorithm-types;
                description
                    "Cryptographic algorithm associated with key.";
            }
        }

        grouping key-chain-config-entry {
            description "Key-chain configuration entry.";
            uses key-chain-common-entry;
            container key-string {
                description "The key string.";
                choice key-string-style {
                    description
                        "Key string styles";
                    case keystring {
                        leaf keystring {
                            type string;
                            description
                                "Key string in ASCII format.";
                        }
                    }
                    case hexadecimal {
```

```
                if-feature hex-key-string;
                leaf hexadecimal-string {
                    type yang:hex-string;
                    description
                        "Key in hexadecimal string format.";
                }
            }
        }
    }
}

grouping key-chain-state-entry {
    description "Key-chain state entry.";
    uses key-chain-common-entry;
    leaf send-lifetime-active {
        type boolean;
        config false;
        description
            "Indicates if the send lifetime of the
             key-chain entry is currently active.";
    }
    leaf accept-lifetime-active {
        type boolean;
        config false;
        description
            "Indicates if the accept lifetime of the
             key-chain entry is currently active.";
    }
}

grouping key-chain-common {
    description
        "key-chain common grouping.";
    leaf name {
        type string;
        description "Name of the key-chain.";
    }
    leaf description {
        type string;
        description "A description of the key-chain";
    }
    container accept-tolerance {
        if-feature accept-tolerance;
        description
            "Tolerance for key lifetime acceptance (seconds).";
        leaf duration {
            type uint32;
            units seconds;
```

```
                default "0";
                description
                    "Tolerance range, in seconds.";
            }
        }
    }

    grouping key-chain-config {
        description
            "key-chain configuration grouping.";
        uses key-chain-common;
        list key-chain-entries {
            key "key-id";
            description "One key.";
            leaf key-id {
                type uint64;
                description "Key ID.";
            }
            uses key-chain-config-entry;
        }
    }

    grouping key-chain-state {
        description
            "key-chain state grouping.";
        uses key-chain-common;
        list key-chain-entries {
            key "key-id";
            description "One key.";
            leaf key-id {
                type uint64;
                description "Key ID.";
            }
            uses key-chain-state-entry;
        }
    }

    container key-chain {
        list key-chain-list {
            key "name";
            description
                "List of key-chains.";
            uses key-chain-config;
        }

        container aes-key-wrap {
            if-feature aes-key-wrap;
            description
```

```
                "AES Key Wrap password encryption.";
            leaf enable {
                type boolean;
                default false;
                description
                    "Enable AES Key Wrap encryption.";
            }
        }
        description "All configured key-chains
                     on the device.";
    }

    container key-chain-state {
        config false;
        list key-chain-list {
            key "name";
            description
                "List of key-chains and operational state.";
            uses key-chain-state;
        }
        container aes-key-wrap {
            if-feature aes-key-wrap;
            description
                "AES Key Wrap password encryption.";
            leaf enable {
                type boolean;
                description
                    "Indicates whether AES Key Wrap encryption
                     is enabled.";
            }
        }
        description "State for all configured key-chains
                     on the device.";
    }
}
<CODE ENDS>
```

5. Security Considerations

This document enables the automated distribution of industry standard key chains using the NETCONF [NETCONF] protocol. As such, the security considerations for the NETCONF protocol are applicable. Given that the key chains themselves are sensitive data, it is RECOMMENDED that the NETCONF communication channel be encrypted. One way to do accomplish this would be to invoke and run NETCONF over SSH as described in [NETCONF-SSH].

When configured, the key-strings can be encrypted using the AES Key Wrap algorithm [AES-KEY-WRAP]. The AES key-encryption key (KEK) is not included in the YANG model and must be set or derived independent of key-chain configuration.

The key strings are not included in the operational state. This is a practice carried over from SNMP MIB modules and is an area for further discussion.

The clear-text algorithm is included as a YANG feature. Usage is NOT RECOMMENDED except in cases where the application and device have no other alternative (e.g., a legacy network device that must authenticate packets at intervals of 10 milliseconds or less for many peers using Bidirectional Forwarding Detection [BFD]). Keys used with the clear-text algorithm are considered insecure and SHOULD NOT be reused with more secure algorithms.

6. IANA Considerations

This document registers a URI in the IETF XML registry [XML-REGISTRY]. Following the format in [XML-REGISTRY], the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-key-chain

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [YANG].

name: ietf-key-chain namespace: urn:ietf:params:xml:ns:yang:ietf-key-chain prefix: ietf-key-chain reference: RFC XXXX

7. References

7.1. Normative References

[NETCONF] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[NETCONF-SSH] Wasserman, M., "Using NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC-KEYWORDS]
Bradner, S., "Key words for use in RFC's to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[XML-REGISTRY]
Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[YANG] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

7.2. Informative References

[AES-KEY-WRAP]
Housley, R. and M. Dworkin, "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm", RFC 5649, August 2009.

[BFD] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, June 2010.

[CRYPTO-KEYTABLE]
Housley, R., Polk, T., Hartman, S., and D. Zhang, "Table of Cryptographic Keys", RFC 7210, April 2014.

[IAB-REPORT]
Andersson, L., Davies, E., and L. Zhang, "Report from the IAB workshop on Unwanted Traffic March 9-10, 2006", RFC 4948, August 2007.

[NETCONF-SERVER-CONF]
Watsen, K. and J. Schoenwaelder, "NETCONF Server and RESTCONF Server Configuration Models", draft-ietf-netconf-server-model-08.txt (work in progress), October 2015.

[NTP-PROTO]
Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[OSPFV3-AUTH]
Bhatia, M., Manral, V., and A. Lindem, "Supporting Authentication Trailer for OSPFv3", RFC 7166, March 2014.

[TCP-AO] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, June 2010.

[TCP-AO-ALGORITHMS]
Lebovitz, G. and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)", RFC 5926, June 2010.

[YANG-CRYPTO-KEYTABLE]
Chen, I., "YANG Data Model for RFC 7210 Key Table", draft-chen-rtg-key-table-yang-02.txt (work in progress), November 2015.

Appendix A. Acknowledgments

The RFC text was produced using Marshall Rose's xml2rfc tool.

Thanks to Brian Weis for fruitful discussions on security requirements.

Thanks to Ines Robles for Routing Directorate QA review comments.

Authors' Addresses

Acee Lindem (editor)
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

Email: acee@cisco.com

Yingzhen Qu
Cisco Systems
170 West Tasman Drive
San Jose, CA 95134
USA

Email: yiqu@cisco.com

Derek Yeung
Arrcus, Inc

Email: derek@arrcus.com

Ing-Wher Chen
Ericsson

Email: ichen@kuatrotech.com

Jeffrey Zhang
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: zzhang@juniper.net

Yi Yang
Individual Contributor

Email: yiyang1998@gmail.com