Internet Draft
<draft-chen-rtgwg-key-table-yang-01.txt>
Intended Status: Standards Track
Expires in 6 months

I. Chen
Ericsson

November 2, 2015

YANG Data Model for RFC 7210 Key Table
<draft-chen-rtgwg-key-table-yang-00.txt>

Status of this Memo

Distribution of this memo is unlimited.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire in 6 months.

Copyright Notice

Abstract

RFC 7210 defines a key table that consists of cryptographic keys that stores information for many different types of routing protocols to ensure message security. This document defines a YANG data model that represents the key table defined in RFC 7210, with the information necessary for YANG and NETCONF to provide routing protocol authentication.

Table of Contents

1. Introduction

[RFC7210] defines a standards track key table that is used to store information for routing protocol authentication, including key values, cryptographic algorithms, timing attributes, and their relationships to allow different routing protocols to perform key selection, authentication, and smooth key rollover that ultimate provides routing protocol message security. This document defines a YANG [RFC6020] data model that corresponds to [RFC7210] and enables the use of NETCONF [RFC6241] and YANG to manage routing protocol authentication data.

An earlier version of the key table YANG model [I-D.chen-rtg-key-table-yang] augments from the key-chain YANG model [I-D.acee-rtg-yang-key-chain]. However, because the key-chain YANG model organizes keys in groups, which is different from the key definitions in [RFC7210], this document proposes a new key-table YANG model that is

structurally more consistent with the key database defined in [RFC7210].

1.1. Terminology

The keywords "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].

1.2. Tree Diagram

A simplified graphical representation of the data model is presented in Section 2.

The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Curly braces "{" and "}" contain names of optional features that make the corresponding node conditional.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" state data (read-only).
- Symbols after data node names: "?" means an optional node and "*" denotes a "list" or "leaf-list".
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Design of the Data Model

This data model is based on the key table defined in [RFC7210]. Because [RFC7210] defines a table that allows protocols to customize certain fields, the data model comes in two parts, the base model and

the customizable part. The base model defines the attributes in [RFC7210] that are immutable across all protocols, such as admin-key-name and send-lifetime-start. The customizable part defines the attributes that are different across protocols, such as the local-key-name, peer-key-name, and peers field. These customizable attributes are defined as YANG grouping statements that can be used and incorporated into protocol-specific key table modules.

2.1. Base Model

[RFC7210] defines a single table in which the rows represent the individual key entries of the key table. As such, the base model consists of one single top-level container named "key-table", in which a YANG list named "security-association-entry" is defined. The entries in the "security-association-entry" YANG list represents the rows that correspond to the cryptographic keys in [RFC7210] key table.

The base model further defines each "security-association-entry" to

consist of attributes that are common across all protocols:

- admin-key-name
- key
- interfaces
- send-lifetime-start
- send-lifetime-end
- accept-lifetime-start
- accept-lifetime-end

Additionally, the base model also defines two attributes, "peers" and "protocol-specific-info", as placeholders left for each protocol to define, as prescribed by [RFC7210].

```
module: ietf-key-table
   +--rw key-table
```

```
      +--rw security-association-entry* [admin-key-name]
         +--rw admin-key-name            string
         +--rw peers
         +--rw interfaces
         |  +--rw (interface-options)
         |     +--:(all-interfaces)
         |     |  +--rw all?          empty
         |     +--:(interface-list)
         |        +--rw interface*    if:interface-ref
         +--rw protocol                  identityref
         +--rw protocol-specific-info
         +--rw key                       yang:hex-string
         +--rw send-lifetime-start       lifetime-type
         +--rw send-lifetime-end         lifetime-type
         +--rw accept-lifetime-start     lifetime-type
         +--rw accept-lifetime-end       lifetime-type
```

2.2. Protocol Customization

Besides the attributes defined in the base model, for each row in the key table, [RFC7210] also defines the following attributes for which the format and range of valid values are protocol-specific.

o local key name

o peer key name

o key derivation function

o cryptographic algorithm

o direction

After a routing protocol augments the base model to incorporate the attributes above, the result is a key table with all the necessary attributes for routing protocol authentication, as shown in the key table below with RSVP customization.

```
module: ietf-key-table
   +--rw key-table
      +--rw security-association-entry* [admin-key-name]
```

```
            +--rw admin-key-name            string
            +--rw interfaces
            |  +--rw (interface-options)
            |     +--:(all-interfaces)
            |     |  +--rw all?         empty
            |     +--:(interface-list)
            |        +--rw interface*   if:interface-ref
            +--rw protocol                  identityref
            +--rw protocol-specific-info
            +--rw key                       yang:hex-string
            +--rw send-lifetime-start       lifetime-type
            +--rw send-lifetime-end         lifetime-type
            +--rw accept-lifetime-start     lifetime-type
            +--rw accept-lifetime-end       lifetime-type
            +--rw peers
   augment
   /keytable:key-table/keytable:security-association-entry +
   /keytable:peers:
            |  +--rw rsvp-security-association?   leafref
   augment
   /keytable:key-table/keytable:security-association-entry:
            +--rw kdf?         key-derivation-function-type
            +--rw alg-id?      cryptographic-algorithm-type
            +--rw direction?   enumeration

   module: example-rsvp-key-table
      +--rw example-rsvp-key-table
         +--rw rsvp-security-association-entry* [name]
            +--rw name                    string
            +--rw rsvp-local-key-name?    uint64
            +--rw rsvp-peer-key-name?     uint64
```


3. Key-chain vs. Key-table Comparison

The key-chain YANG model also proposes a YANG model for key management. This section compares the key-chain model and the key-table model proposed in this document.

3.1. Organization

The key-chain YANG model groups several keys into a single key chain. It is the key chain that is referenced and applied by routing protocols. Consequently, the key-chain YANG model defines a hierarchical database, which consists of a top-level key-chain database, and each key-chain database consists of the actual keys that are used by a protocol. A routing protocol that requires encryption or authentication must reference a key-chain instead of the individual keys.

In contrast, [RFC7210] defines a single flat database of keys and their attributes. [RFC7210] does not require that an implementation of key management explicitly group a set of keys into a separate entity that routing protocols reference and use. Consequently, the key-table base model defined in this document presents a flat view of the key database.

For routing protocol customization, the key-table base model can also be adapted to hierarchical key management as described in the key-chain YANG model. Section 5 provides an example of adapting a hierarchical key management model into the key-table model.

3.2. Missing Attributes

The key-chain YANG model defines only a subset of key attributes defined in [RFC7210]. Consequently, the key-chain YANG model can only support specific types of deployments requiring authentication. A list of key attributes defined in [RFC7210] and in this document, but not defined in [key-chain], are as follows:

- Peer key name
- Interfaces
- Protocol
- Key derivation function
- Direction

In addition to the list above, [key-chain] also does not define

attributes that [RFC7210] defined but left the detailed definitions to individual routing protocols. Similar to [RFC7210], this document defines YANG constructs for these attributes and intends for routing protocols to provided the details. The attributes in question are as follows:

- Peers
- ProtocolSpecificInfo

4. YANG Module

```
<CODE BEGINS> file "ietf-key-table@2015-08-28.yang"

module ietf-key-table {
  namespace "http://www.example.com/ietf-key-table";
  prefix "keytable";

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

  import ietf-routing {
    prefix "rt";
  }

  import ietf-interfaces {
    prefix "if";
  }

  organization
    "Ericsson";

  contact
    "I. Chen - ing-wher.chen@ericsson.com";

  description
    "A key table YANG data model based on RFC 7210";

  revision 2015-08-28 {
    description
      "Revision 3.  " +
      "Making RFC 7210 a global generic table.";
      reference "RFC 7210";
  }
```

```
    revision 2015-06-29 {
      description
        "Revision 2.";
      reference "RFC 7210";
    }

    /* Identities */

    identity key-derivation-function {
      description
        "Base identity from which key derivation function " +
        "identities are derived";
    }

    identity kdf-none {
      base key-derivation-function;
      description
        "This identity represents a cryptographic key that is" +
        "used directly, without a key derivation function.";
    }

    identity kdf-aes-128-cmac {
      base key-derivation-function;
      description
        "This identity represents the key derivation function that " +
        "uses AES-CMAC using 128-bit keys (RFC 4493).";
    }

    identity kdf-hmac-sha-1 {
      base key-derivation-function;
      description
        "This identity represents the key derivation function that " +
        "uses HMAC using the SHA-1 hash (RFC 2104).";
    }

    identity cryptographic-algorithm {
      description
        "Base identity from which cryptographic algorithms " +
        "are derived";
    }

    identity algid-aes-128-cmac {
      base cryptographic-algorithm;
      description
        "This identity represents the cryptographic algorithm " +
        "AES-CMAC using 128-bit keys (RFC 4493).";
```

```
}
```



```
identity algid-aes-128-cmac-92 {
  base cryptographic-algorithm;
  description
    "This identity represents the cryptographic algorithm " +
    "AES-128-CMAC truncated to 96 bits (RFC 5926).";
}

identity algid-hmac-sha-1-96 {
  base cryptographic-algorithm;
  description
    "This identity represents the cryptographic algorithm " +
    "HMAC SHA-1 truncated to 96 bits (RFC 2104).";
}

identity all-routing-protocols {
  base rt:routing-protocol;
  description
    "All routing protocols";
}

/* Typedefs */
typedef routing-protocol-type {
  type identityref {
    base rt:routing-protocol;
  }
  description
    "This type identifies the routing protocol";
}

typedef key-derivation-function-type {
  type identityref {
    base key-derivation-function;
  }
  description
    "This type identifies the key derivation function";
}

typedef cryptographic-algorithm-type {
  type identityref {
```

```
        base cryptographic-algorithm;
      }
      description
        "This type identifies the cryptographic algorithm";
    }

    typedef lifetime-type {
      type string {
        pattern '{4}{2}{2}{2}{2}Z';
```



```
      }
      description
        "This type identifies a time in the format YYYYMMDDHHSSZ, " +
        "where the first four digits specify the year, " +
        "the next two digits specify the month, " +
        "the next two digits specify the day, " +
        "the next two digits specify the hour, " +
        "the next two digits specify the second, " +
        "ending with the letter 'Z' as a clear indication " +
        "that the time is in Coordinated Universal Time (UTC).";
    }

    /* Groupings */

    grouping key-properties-grp {
      description
        "A grouping that to specify the properties of a key.  " +
        "This is defined as a grouping so that different " +
        "routing protocols can further refine the values of " +
        "each individual key properties.";
      leaf kdf {
        type key-derivation-function-type;
        description
          "Specify the key derivation function to be used " +
          "with this key.";
      }
      leaf alg-id {
        type cryptographic-algorithm-type;
        description
          "Specify the cryptographic algorithm to be used" +
          "with this key.";
      }
```

```
        leaf direction {
          type enumeration {
            enum "in" {
              description
                "The key is used for inbound traffic.";
            }
            enum "out" {
              description
                "The key is used for outbound traffic.";
            }
            enum "both" {
              description
                "The key is used for both inbound and outbound " +
                "traffic.";
            }
            enum "disabled" {
```



```
              description
                "The key is disabled and cannot be used for " +
                "either inbound or outbound traffic.";
            }
          }
          description
            "The value of the direction must be one of 'in', 'out', " +
            "'both', and 'disabled'.  The actual allowed value " +
            "is left for routing protocols to define.";
        }
      }

      /* The key-table model */

      container key-table {
        description
          "The key table of all managed cryptographic keys " +
          "of a device.";
        list security-association-entry {
          key "admin-key-name";
          description
            "A key table entry that specifies a key " +
            "and its attributes.";
          leaf admin-key-name {
            type string;
```

```
        description
          "A human-readable string that identifies the key.";
      }
      container peers {
        description
          "Specify the peer systems that also have this key " +
          "in their database.  The format of this field is " +
          "left to protocols to define.";
      }
      container interfaces  {
        description
          "Specify the interfaces to which they key may be applied.";
        choice interface-options {
          mandatory true;
          description
            "The option to apply this key to all interfaces or " +
            "to a pre-defined list of interfaces.";
          case all-interfaces {
            leaf all {
              type empty;
              description
                "This key applies to all interfaces.";
            }
```



```
          }
          case interface-list {
            leaf-list interface {
              type if:interface-ref;
              description
                "This key applies to the identified interfaces.";
            }
          }
        }
      }
      leaf protocol {
        type identityref {
          base rt:routing-protocol;
        }
        mandatory true;
        description
          "Specify a single routing protocol where this key " +
          "may be used to provide cryptographic protection.";
```

```
        }
        container protocol-specific-info {
          description
            "This field contains protocol-specified information " +
            "that maybe useful for a protocol to apply the key " +
            "correctly.  This field is left for each protocol " +
            "to define.";
        }
        leaf key {
          type yang:hex-string;
          mandatory true;
          description
            "The key";
        }
        leaf send-lifetime-start {
          type lifetime-type;
          mandatory true;
          description
            "Specify the earliest date and time at which this key " +
            "should be considered for use when sending traffic.";
        }
        leaf send-lifetime-end {
          type lifetime-type;
          mandatory true;
          description
            "Specify the latest date and time at which this key " +
            "should be considered for use when sending traffic.";
        }
        leaf accept-lifetime-start {
          type lifetime-type;
```



```
          mandatory true;
          description
            "Specify the earliest date and time at which this key " +
            "should be considered for processing received traffic.";
        }
        leaf accept-lifetime-end {
          type lifetime-type;
          mandatory true;
          description
            "Specify the latest date and time at which this key " +
            "should be considered for processing received traffic.";
```

```
        }
      }
    } }
```

<CODE ENDS>

5. Usage

A routing protocol that requires cryptographic key authentication should customize the key table base model such that the resulting YANG modules describe all the necessary attributes and also the valid values of these attributes.

An an example, the customized example-ospf-key-table YANG module below augments the key table base model to include "key-derivation-function", "cryptofgraphic-algorithm", and "direction" attributes, as well as specifying the valid valuesfor those attributes. The module below further uses the "peers" field in the base key table to reference keys that are organized into key chains as expected by the existing OSPF YANG module [I-D.ospf-yang].

```
module example-ospf-key-table {
  namespace "http://www.example.com/example-ospf-key-table";
  prefix "ospf-keytable";

  import ietf-key-table {
    prefix "keytable";
  }

  import ietf-routing {
    prefix "rt";
  }

  organization
    "Ericsson";

  contact
```



```
    "I. Chen - ing-wher.chen@ericsson.com";

  description
    "OSPF's customized key table";
```

```
  revision 2015-08-28 {
    description "Initial revision";
    reference "";
  }

  /* Identities */

  identity ospf-cryptographic-algorithm {
    base keytable:cryptographic-algorithm;
    description
      "Base identity from which OSPF cryptographic algorithm " +
      "identities are derived";
  }

  identity ospf-algid-md5 {
    base ospf-cryptographic-algorithm;
    description
      "This identity represents the cryptographic algorithm " +
      "MD5";
  }

  identity ospf-algid-hmac-md5 {
    base ospf-cryptographic-algorithm;
    description
      "This identity represents the cryptographic algorithm "  +
      "HMAC-MD5";
  }

  identity ospf-algid-sha-1 {
    base ospf-cryptographic-algorithm;
    description
      "This identity represents the cryptographic algorithm " +
      "SHA-1";
  }

  identity ospf-algid-hmac-sha-1 {
    base ospf-cryptographic-algorithm;
    description
      "This identity represents the cryptographic algorithm " +
      "HMAC-SHA-1";
  }

  identity ospf-algid-hmac-sha-1-12 {
```

```
    base ospf-cryptographic-algorithm;
    description
      "This identity represents the cryptographic algorithm " +
      "HMAC-SHA-1-12";
  }

  identity ospf-algid-hmac-sha-256 {
    base ospf-cryptographic-algorithm;
    description
      "This identity represents the cryptographic algorithm " +
      "HMAC-SHA-256";
  }

  identity ospf-algid-hmac-sha-384 {
    base ospf-cryptographic-algorithm;
    description
      "This identity represents the cryptographic algorithm " +
      "HMAC-SHA-384";
  }

  identity ospf-algid-hmac-sha-512 {
    base ospf-cryptographic-algorithm;
    description
      "This identity repesents the cryptographic algorithm " +
      "HMAC-SHA-512";
  }

  /* Typedef */

  typedef ospf-cryptographic-algorithm-type {
    type identityref {
      base ospf-cryptographic-algorithm;
    }
    description
      "This type identifies the cryptographic algorithm";
  }

  augment "/keytable:key-table/keytable:security-association-entry" {
    when "keytable:protocol == 'rt:ospfv2' or " +
         "keytable:protocol == 'rt:ospfv3'" {
      description
        "Applies only to OSPFv2 and OSPFv3.";
    }
    uses keytable:key-properties-grp {
      refine "kdf" {
        must ". == 'keytable:kdf-none'" {
          description
```

```
            "KDF is not used.";
```



```
          }
        }
        refine "alg-id" {
          must ". == 'ospf-algid-md5' or " +
               ". == 'ospf-algid-hmac-md5' or " +
               ". == 'ospf-algid-sha-1' or " +
               ". == 'ospf-algid-hmac-sha-1' or " +
               ". == 'ospf-algid-hmac-sha-1-12' or " +
               ". == 'ospf-algid-hmac-sha-256' or " +
               ". == 'ospf-algid-hmac-sha-384' or " +
               ". == 'ospf-algid-hmac-sha-512' or " {
            description
              "Only ospf-cryptographic-algorithms are valid.";
          }
        }
        refine "direction" {
          must ". == 'keytable:both'" {
            description
              "Key applies to both directions.";
          }
        }
      }
      description
        "Customize OSPF protocol specific attributes";
    }

    augment "/keytable:key-table" +
            "/keytable:security-association-entry" +
            "/keytable:peers" {
      when "../keytable:protocol == 'rt:ospfv2' or " +
           "../keytable:protocol == 'rt:ospfv3'" {
        description
          "Applies only to OSPFv2 and OSPFv3.";
      }
      description
        "Reference to the apropriate key chain";
      leaf key-chain {
        type leafref {
          path "/example-ospf-key-chains/ospf-key-chain/name";
        }
```

```
        description
          "The name of the key chain";
      }
      leaf security-association {
        type leafref {
          path "/example-ospf-key-chains" +
               "/ospf-key-chain[name = current()/../key-chain]" +
               "/ospf-security-association-entry/name";
```



```
        }
        description
          "The security association within the key chain";
      }
    }

    container example-ospf-key-chains {
      description
        "A container of OSPF key chains modeled after " +
        "ietf-key-chains";
      list ospf-key-chain {
        key "name";
        leaf name {
          type string;
          description
            "Name of the key chain";
        }
        list ospf-security-association-entry {
          key "name";
          leaf name {
            type string;
            description
              "The name of the security assocation";
          }
          leaf ospf-local-key-name {
            type uint8;
            mandatory true;
            description
              "The 8-bit key ID for sending a message, " +
              "as defined in RFC 2328 Appendix D.3";
          }
          leaf ospf-peer-key-name {
            type leafref {
```

```
            path "../ospf-local-key-name";
          }
          description
            "The 8-bit key ID when receiving a message, " +
            "as defined in RFC 2328 Appendix D.3. " +
            "Because OSPF uses the same key for sending " +
            "and receiving, the value of this leaf " +
            "should be identical to the value of " +
            "ospf-local-key-name.";
        }

        description
          "An OSPF security association, i.e. key";
      }
      description
```



```
        "An OSPF key chain";
    }
  } }
```

6. Security Consideration.

TBD.

7. IANA Considerations

TBD.

8. References

8.1. Normative References

[RFC7210] Housley, R., Polk, T., Hartman, S., and D. Zhang, "Database of Long-Lived Symmetric Cryptographic Keys", RFC 7210, April 2014, <http://www.rfc-editor.org/info/rfc7210>.

[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>.

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>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

[I-D.acee-rtg-yang-key-chain] Lindem, A., Qu, Y., Yeung, D., Chen, I., Zhang, J., and Y. Yang, "Key Chain YANG Data Model", draft-acee-rtg-yang-key-chain-09 (work in progress), October 2015.

[I-D.ospf-yang] Yeung, D., Qu, Y., Zhang, J., Bogdanovic, D., and K. Sreenivasa, "draft-ietf-ospf-yang-02 (work in progress)", September 2015.

Author's Address

I. Chen
Ericsson
Email: ing-wher.chen@ericsson.com