NETCONF Working Group
Internet-Draft
Intended status: Standards Track
Expires: 17 September 2024

K. Watsen
Watsen Networks

16 March 2024

YANG Groupings for TLS Clients and TLS Servers
draft-ietf-netconf-tls-client-server-41

Abstract

This document presents four YANG 1.1 modules. Three IETF modules, and one supporting IANA module.

The three IETF modules are: ietf-tls-common, ietf-tls-client, and ietf-tls-server. The "ietf-tls-client" and "ietf-tls-server" modules are the primary productions of this work, supporting the configuration and monitoring of TLS clients and servers.

The IANA module is: iana-tls-cipher-suite-algs. This module defines YANG enumerations providing support for an IANA-maintained algorithm registry.

Editorial Note (To be removed by RFC Editor)

This draft contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

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

* AAAA --> the assigned RFC value for draft-ietf-netconf-crypto-types
* BBBB --> the assigned RFC value for draft-ietf-netconf-trust-anchors
* CCCC --> the assigned RFC value for draft-ietf-netconf-keystore
* DDDD --> the assigned RFC value for draft-ietf-netconf-tcp-client-server
* FFFF --> the assigned RFC value for this draft

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

* 2024-03-16 --> the publication date of this draft

The "Relation to other RFCs" section Section 1.2 contains the text "one or more YANG modules" and, later, "modules". This text is sourced from a file in a context where it is unknown how many modules a draft defines. The text is not wrong as is, but it may be improved by stating more directly how many modules are defined.

The "Relation to other RFCs" section Section 1.2 contains a self-reference to this draft, along with a corresponding reference in the Appendix. Please replace the self-reference in this section with "This RFC" (or similar) and remove the self-reference in the "Normative/Informative References" section, whichever it is in.

Tree-diagrams in this draft may use the '\' line-folding mode defined in RFC 8792. However, nicer-to-the-eye is when the '\\' line-folding mode is used. The AD suggested suggested putting a request here for the RFC Editor to help convert "ugly" '\' folded examples to use the '\\' folding mode. "Help convert" may be interpreted as, identify what looks ugly and ask the authors to make the adjustment.

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

* Appendix A.1. Initial Module for the "TLS Cipher Suites" Registry

* Appendix B. Change Log

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 17 September 2024.

Table of Contents

1. Introduction

This document presents four YANG 1.1 [RFC7950] modules. Three "IETF" modules and one "IANA" module.

The three IETF modules are ietf-tls-common (Section 2), ietf-tls-client (Section 3), and ietf-tls-server (Section 4). The "ietf-tls-client" and "ietf-tls-server" modules are the primary productions of this work, supporting the configuration and monitoring of TLS clients and servers.

The groupings defined in this document are expected to be used in conjunction with the groupings defined in an underlying transport-level module, such as the groupings defined in [I-D.ietf-netconf-tcp-client-server]. The transport-level data model enables the configuration of transport-level values such as a remote address, a remote port, a local address, and a local port.

The IANA module is iana-tls-cipher-suite-algs (Appendix A.1). This module defines YANG enumerations providing support for an IANA-maintained algorithm registry.

This document assumes that the IANA module exists, and presents a script in Appendix A that IANA may use to generate the YANG module. This document does not publish initial version of this module. IANA publishes this module.

1.1. Regarding the IETF Modules

The three IETF modules define features and groupings to model "generic" TLS clients and TLS servers, where "generic" should be interpreted as "least common denominator" rather than "complete." Basic TLS protocol support is afforded by these modules, leaving configuration of advance features to augmentations made by consuming modules.

It is intended that the YANG groupings will be used by applications needing to configure TLS client and server protocol stacks. For instance, these groupings are used to help define the data model for HTTPS [RFC2818] and NETCONF over TLS [RFC7589] based clients and servers in [I-D.ietf-netconf-http-client-server] and [I-D.ietf-netconf-netconf-client-server] respectively.

The ietf-tls-client and ietf-tls-server YANG modules each define one grouping, which is focused on just TLS-specific configuration, and specifically avoids any transport-level configuration, such as what ports to listen-on or connect-to. This affords applications the opportunity to define their own strategy for how the underlying TCP

connection is established. For instance, applications supporting NETCONF Call Home [RFC8071] could use the "tls-server-grouping" grouping for the TLS parts it provides, while adding data nodes for the TCP-level call-home configuration.

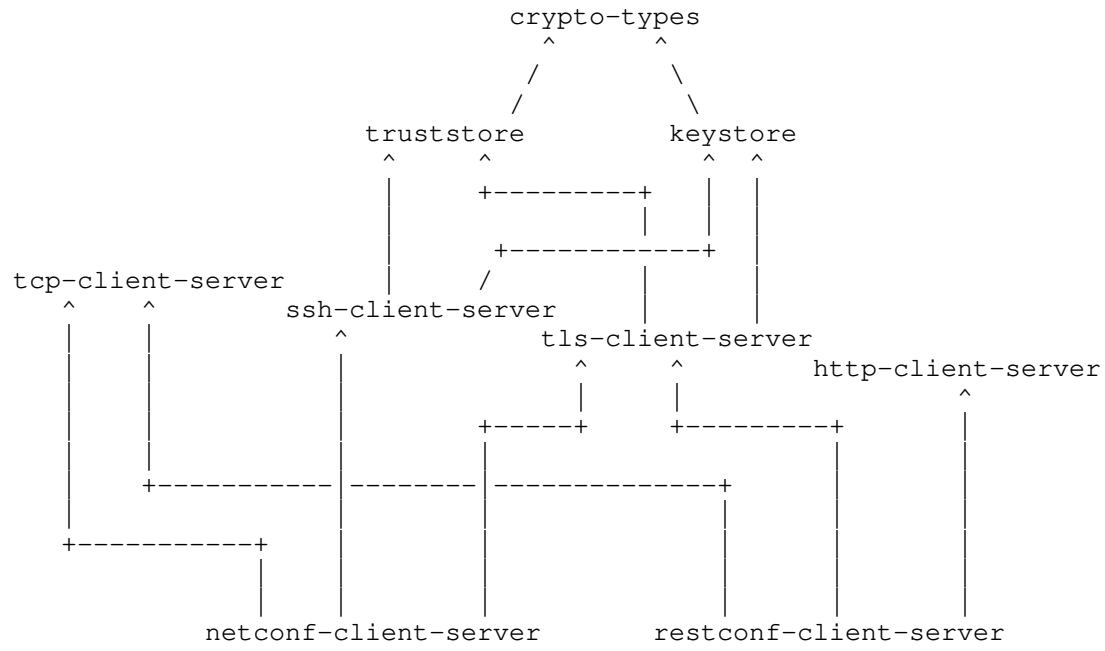
Both TLS 1.2 and TLS 1.3 may be configured. TLS 1.2 [RFC5246] is obsoleted by TLS 1.3 [RFC8446] but still in common use, and hence its "feature" statement is marked "status deprecated".

1.2. Relation to other RFCs

This document presents one or more YANG modules [RFC7950] that are part of a collection of RFCs that work together to, ultimately, support the configuration of both the clients and servers of both the NETCONF [RFC6241] and RESTCONF [RFC8040] protocols.

The dependency relationship between the primary YANG groupings defined in the various RFCs is presented in the below diagram. In some cases, a draft may define secondary groupings that introduce dependencies not illustrated in the diagram. The labels in the diagram are a shorthand name for the defining RFC. The citation reference for shorthand name is provided below the diagram.

Please note that the arrows in the diagram point from referencer to referenced. For example, the "crypto-types" RFC does not have any dependencies, whilst the "keystore" RFC depends on the "crypto-types" RFC.



Label in Diagram	Originating RFC
crypto-types	[I-D.ietf-netconf-crypto-types]
truststore	[I-D.ietf-netconf-trust-anchors]
keystore	[I-D.ietf-netconf-keystore]
tcp-client-server	[I-D.ietf-netconf-tcp-client-server]
ssh-client-server	[I-D.ietf-netconf-ssh-client-server]
tls-client-server	[I-D.ietf-netconf-tls-client-server]
http-client-server	[I-D.ietf-netconf-http-client-server]
netconf-client-server	[I-D.ietf-netconf-netconf-client-server]
restconf-client-server	[I-D.ietf-netconf-restconf-client-server]

Table 1: Label in Diagram to RFC Mapping

1.3. Specification Language

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

1.4. Adherence to the NMDA

This document is compliant with the Network Management Datastore Architecture (NMDA) [RFC8342]. For instance, as described in [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], trust anchors and keys installed during manufacturing are expected to appear in <operational> (Section 5.3 of [RFC8342]), and <system> [I-D.ietf-netmod-system-config], if implemented.

1.5. Conventions

Various examples in this document use "BASE64VALUE=" as a placeholder value for binary data that has been base64 encoded (per Section 9.8 of [RFC7950]). This placeholder value is used because real base64 encoded structures are often many lines long and hence distracting to the example being presented.

2. The "ietf-tls-common" Module

The TLS common model presented in this section contains features and groupings common to both TLS clients and TLS servers. The "hello-params-grouping" grouping can be used to configure the list of TLS algorithms permitted by the TLS client or TLS server. The lists of algorithms are ordered such that, if multiple algorithms are permitted by the client, the algorithm that appears first in its list that is also permitted by the server is used for the TLS transport layer connection. The ability to restrict the algorithms allowed is provided in this grouping for TLS clients and TLS servers that are capable of doing so and may serve to make TLS clients and TLS servers compliant with local security policies. This model supports both TLS 1.2 [RFC5246] and TLS 1.3 [RFC8446].

Thus, in order to support both TLS1.2 and TLS1.3, the cipher-suites part of the "hello-params-grouping" grouping should include three parameters for configuring its permitted TLS algorithms, which are: TLS Cipher Suites, TLS SignatureScheme, TLS Supported Groups.

2.1. Data Model Overview

This section provides an overview of the "ietf-tls-common" module in terms of its features, identities, and groupings.

2.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tls-common" module:

```
Features:
  +-- tls12
  +-- tls13
  +-- hello-params
  +-- asymmetric-key-pair-generation
  +-- supported-algorithms
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

Please refer to the YANG module for a description of each feature.

2.1.2. Identities

The following diagram illustrates the relationship amongst the "identity" statements defined in the "ietf-tls-common" module:

```
Identities:
  +-- tls-version-base
     +-- tls12
     +-- tls13
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

Comments:

* The diagram shows that there are two base identities.
* One base identity is used to specific TLS versions, while the other is used to specify cipher-suites.
* These base identities are "abstract", in the object oriented programming sense, in that they only define a "class" of things, rather than a specific thing.

2.1.3. Groupings

The "ietf-tls-common" module defines the following "grouping" statement:

* hello-params-grouping

This grouping is presented in the following subsection.

2.1.3.1. The "hello-params-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "hello-params-grouping" grouping:

```
  grouping hello-params-grouping:
    +-- tls-versions
    |  +-- min?   identityref
    |  +-- max?   identityref
    +-- cipher-suites
       +-- cipher-suite*   tlscsa:tls-cipher-suite-algorithm
```

Comments:

* This grouping is used by both the "tls-client-grouping" and the "tls-server-grouping" groupings defined in Section 3.1.2.1 and Section 4.1.2.1, respectively.

* This grouping enables client and server configurations to specify the TLS versions and cipher suites that are to be used when establishing TLS sessions.

* The "cipher-suites" list is "ordered-by user".

2.1.4. Protocol-accessible Nodes

The following tree diagram [RFC8340] lists all the protocol-accessible nodes defined in the "ietf-tls-common" module, without expanding the "grouping" statements:

```
module: ietf-tls-common
  +--ro supported-algorithms {algorithm-discovery}?
     +--ro supported-algorithm*   tlscsa:tls-cipher-suite-algorithm

  rpcs:
    +---x generate-asymmetric-key-pair
            {asymmetric-key-pair-generation}?
       +---w input
       |  +---w algorithm
       |  |       tlscsa:tls-cipher-suite-algorithm
       |  +---w num-bits?               uint16
       |  +---w private-key-encoding
       |     +---w (private-key-encoding)
       |        +--:(cleartext) {ct:cleartext-private-keys}?
       |        |  +---w cleartext?   empty
       |        +--:(encrypted) {ct:encrypted-private-keys}?
       |        |  +---w encrypted
       |        |     +---w ks:encrypted-by-grouping
       |        +--:(hidden) {ct:hidden-private-keys}?
       |           +---w hidden?      empty
       +--ro output
          +--ro (key-or-hidden)?
             +--:(key)
             |  +---u ct:asymmetric-key-pair-grouping
             +--:(hidden)
                +--ro location?
                        instance-identifier
```

Comments:

* Protocol-accessible nodes are those nodes that are accessible when the module is "implemented", as described in Section 5.6.5 of [RFC7950].

* The protocol-accessible nodes for the "ietf-tls-common" module are limited to the "supported-algorithms" container, which is constrained by the "algorithm-discovery" feature, and the RPC "generate-asymmetric-key-pair", which is constrained by the "asymmetric-key-pair-generation" feature.

* The "encrypted-by-grouping" grouping is discussed in Section 2.1.3.1 of [I-D.ietf-netconf-keystore].

* The "asymmetric-key-pair-grouping" grouping is discussed in Section 2.1.4.6 of [I-D.ietf-netconf-crypto-types].

2.2. Example Usage

The following example illustrates the "hello-params-grouping' grouping when populated with some data.

```
<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<hello-params
   xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common"
   xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common">
  <tls-versions>
    <min>tlscmn:tls12</min>
    <max>tlscmn:tls13</max>
  </tls-versions>
  <cipher-suites>
    <cipher-suite>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</cipher-suite>
    <cipher-suite>TLS_DHE_RSA_WITH_AES_128_CBC_SHA256</cipher-suite>
    <cipher-suite>TLS_RSA_WITH_3DES_EDE_CBC_SHA</cipher-suite>
  </cipher-suites>
</hello-params>
```

The following example illustrates operational state data indicating the TLS algorithms supported by the server.

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

   <supported-algorithms
     xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common">
     <supported-algorithm>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA</support\
   ed-algorithm>
     <supported-algorithm>TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384</supp\
   orted-algorithm>
     <supported-algorithm>TLS_DHE_RSA_WITH_AES_128_CBC_SHA256</supporte\
   d-algorithm>
     <supported-algorithm>TLS_RSA_WITH_3DES_EDE_CBC_SHA</supported-algo\
   rithm>
     <supported-algorithm>TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384</suppor\
   ted-algorithm>
     <supported-algorithm>TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256</su\
   pported-algorithm>
     <supported-algorithm>TLS_ECCPWD_WITH_AES_256_GCM_SHA384</supported\
   -algorithm>
     <supported-algorithm>TLS_PSK_WITH_AES_256_CCM</supported-algorithm>
     <supported-algorithm>TLS_PSK_WITH_AES_256_CCM_8</supported-algorit\
   hm>
     <supported-algorithm>TLS_DHE_PSK_WITH_CAMELLIA_256_CBC_SHA384</sup\
   ported-algorithm>
     <supported-algorithm>TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384</support\
   ed-algorithm>
     <supported-algorithm>TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA</supported\
   -algorithm>
     <supported-algorithm>TLS_DH_DSS_WITH_AES_128_GCM_SHA256</supported\
   -algorithm>
   </supported-algorithms>
```

The following example illustrates the "generate-asymmetric-key-pair" RPC.

REQUEST

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

   <rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <generate-asymmetric-key-pair
       xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-common">
       <algorithm>TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256</algorithm>
       <num-bits>521</num-bits>
       <private-key-encoding>
         <encrypted>
           <asymmetric-key-ref>hidden-asymmetric-key</asymmetric-key-re\
   f>
         </encrypted>
       </private-key-encoding>
     </generate-asymmetric-key-pair>
   </rpc>

   RESPONSE

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

   <rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
     xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types"
     xmlns:tlscmn="urn:ietf:params:xml:ns:yang:ietf-tls-common">
     <tlscmn:public-key-format>ct:subject-public-key-info-format</tlscm\
   n:public-key-format>
     <tlscmn:public-key>BASE64VALUE=</tlscmn:public-key>
     <tlscmn:private-key-format>ct:ec-private-key-format</tlscmn:privat\
   e-key-format>
     <tlscmn:cleartext-private-key>BASE64VALUE=</tlscmn:cleartext-priva\
   te-key>
   </rpc-reply>
```

2.3. YANG Module

This YANG module has a normative references to [RFC5288], [RFC5289], [RFC8422], and FIPS PUB 180-4.

This YANG module has a informative references to [RFC5246], and [RFC8446].

```
   <CODE BEGINS> file "ietf-tls-common@2024-03-16.yang"
```

```
   module ietf-tls-common {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-tls-common";
     prefix tlscmn;

     import iana-tls-cipher-suite-algs {
       prefix tlscsa;
       reference
         "RFC FFFF: YANG Groupings for TLS Clients and SSH Servers";
     }

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

     import ietf-keystore {
       prefix ks;
       reference
         "RFC CCCC: A YANG Data Model for a Keystore";
     }

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

     contact
       "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        WG Web:   https://datatracker.ietf.org/wg/netconf
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Jeff Hartley <mailto:intensifysecurity@gmail.com>
        Author:   Gary Wu <mailto:garywu@cisco.com>";

      description
       "This module defines a common features and groupings for
        Transport Layer Security (TLS).

        Copyright (c) 2024 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC FFFF
```

```
        (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
        itself for full legal notices.

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

      revision 2024-03-16 {
        description
          "Initial version";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      // Features

      feature tls12 {
        description
          "TLS Protocol Version 1.2 is supported. TLS 1.2 is obsolete
           and thus it is NOT RECOMMENDED to enable this feature.";
        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
                     Version 1.2";
      }

      feature tls13 {
        description
          "TLS Protocol Version 1.3 is supported.";
        reference
          "RFC 8446: The Transport Layer Security (TLS)
                     Protocol Version 1.3";
      }

      feature hello-params {
        description
          "TLS hello message parameters are configurable.";
      }

      feature algorithm-discovery {
        description
          "Indicates that the server implements the
           'supported-algorithms' container.";
      }

      feature asymmetric-key-pair-generation {
```

```
      description
        "Indicates that the server implements the
         'generate-asymmetric-key-pair' RPC.";
    }

    // Identities

    identity tls-version-base {
      description
        "Base identity used to identify TLS protocol versions.";
    }

    identity tls12 {
      if-feature "tls12";
      base tls-version-base;
      description
        "TLS Protocol Version 1.2.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    identity tls13 {
      if-feature "tls13";
      base tls-version-base;
      description
        "TLS Protocol Version 1.3.";
      reference
        "RFC 8446: The Transport Layer Security (TLS)
                   Protocol Version 1.3";
    }

    // Typedefs

    typedef epsk-supported-hash {
      type enumeration {
        enum sha-256 {
          description
            "The SHA-256 Hash.";
        }
        enum sha-384 {
          description
            "The SHA-384 Hash.";
        }
      }
      description
        "As per Section 4.2.11 of RFC 8446, the hash algorithm
         supported by an instance of an External Pre-Shared
```

```
          Key (EPSK).";
      reference
        "RFC 8446: The Transport Layer Security (TLS)
                   Protocol Version 1.3";
    }


    // Groupings

    grouping hello-params-grouping {
      description
        "A reusable grouping for TLS hello message parameters.";
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2
         RFC 8446: The Transport Layer Security (TLS) Protocol
                   Version 1.3";
      container tls-versions {
        description
          "Parameters limiting which TLS versions, amongst
           those enabled by 'features', are presented during
           the TLS handshake.";
        leaf min {
          type identityref {
            base tls-version-base;
          }
          description
            "If not specified, then there is no configured
             minimum version.";
        }
        leaf max {
          type identityref {
            base tls-version-base;
          }
          description
            "If not specified, then there is no configured
             maximum version.";
        }
      }
      container cipher-suites {
        description
          "Parameters regarding cipher suites.";
        leaf-list cipher-suite {
          type tlscsa:tls-cipher-suite-algorithm;
          ordered-by user;
          description
            "Acceptable cipher suites in order of descending
             preference.  The configured host key algorithms should
```

```
            be compatible with the algorithm used by the configured
            private key.  Please see Section 5 of RFC FFFF for
            valid combinations.

            If this leaf-list is not configured (has zero elements)
            the acceptable cipher suites are implementation-
            defined.";
          reference
            "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
        }
      }
    } // hello-params-grouping


    // Protocol-accessible Nodes

    container supported-algorithms {
      if-feature "algorithm-discovery";
      config false;
      description
        "A container for a list of cipher suite algorithms supported
         by the server.";
      leaf-list supported-algorithm {
        type tlscsa:tls-cipher-suite-algorithm;
        description
          "A cipher suite algorithm supported by the server.";
      }
    }

    rpc generate-asymmetric-key-pair {
      if-feature "asymmetric-key-pair-generation";
      description
        "Requests the device to generate an asymmetric-key-pair
         key using the specified key algorithm.";
      input {
        leaf algorithm {
          type tlscsa:tls-cipher-suite-algorithm;
          mandatory true;
          description
            "The cipher suite algorithm that the generated key is
             to work with.  Implementations derive the public key
             algorithm from the cipher suite algorithm.  Example:
             cipher suite 'tls-rsa-with-aes-256-cbc-sha256' maps
             to the RSA public key.";
        }
        leaf num-bits {
          type uint16;
          description
```

```
            "Specifies the number of bits in the key to create.
             For RSA keys, the minimum size is 1024 bits and
             the default is 3072 bits. Generally, 3072 bits is
             considered sufficient. DSA keys must be exactly 1024
             bits as specified by FIPS 186-2.  For elliptical
             keys, the 'num-bits' value determines the key length
             of the curve (e.g., 256, 384 or 521), where valid
             values supported by the server are conveyed via an
             unspecified mechanism.  For some public algorithms,
             the keys have a fixed length and thus the 'num-bits'
             value is not specified.";
          }
          container private-key-encoding {
            description
              "Indicates how the private key is to be encoded.";
            choice private-key-encoding {
              mandatory true;
              description
                "A choice amongst optional private key handling.";
              case cleartext {
                if-feature "ct:cleartext-private-keys";
                leaf cleartext {
                  type empty;
                  description
                    "Indicates that the private key is to be returned
                     as a cleartext value.";
                }
              }
              case encrypted {
                if-feature "ct:encrypted-private-keys";
                container encrypted {
                  description
                     "Indicates that the key is to be encrypted using
                      the specified symmetric or asymmetric key.";
                  uses ks:encrypted-by-grouping;
                }
              }
              case hidden {
                if-feature "ct:hidden-private-keys";
                leaf hidden {
                  type empty;
                  description
                    "Indicates that the private key is to be hidden.

                     Unlike the 'cleartext' and 'encrypt' options, the
                     key returned is a placeholder for an internally
                     stored key.  See the 'Support for Built-in Keys'
                     section in RFC CCCC for information about hidden
```

```
                    keys.";
                }
              }
            }
          }
        }
        output {
          choice key-or-hidden {
            case key {
              uses ct:asymmetric-key-pair-grouping;
            }
            case hidden {
              leaf location {
                type instance-identifier;
                description
                  "The location to where a hidden key was created.";
              }
            }
            description
              "The output can be either a key (for cleartext and
               encrypted keys) or the location to where the key
               was created (for hidden keys).";
          }
        }
      } // end generate-asymmetric-key-pair

    }
```

<CODE ENDS>

3. The "ietf-tls-client" Module

This section defines a YANG 1.1 [RFC7950] module called "ietf-tls-client". A high-level overview of the module is provided in Section 3.1. Examples illustrating the module's use are provided in Examples (Section 3.2). The YANG module itself is defined in Section 3.3.

3.1. Data Model Overview

This section provides an overview of the "ietf-tls-client" module in terms of its features and groupings.

3.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tls-client" module:

```
Features:
  +-- tls-client-keepalives
  +-- client-ident-x509-cert
  +-- client-ident-raw-public-key
  +-- client-ident-psk
  +-- server-auth-x509-cert
  +-- server-auth-raw-public-key
  +-- server-auth-psk
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

Please refer to the YANG module for a description of each feature.

3.1.2. Groupings

The "ietf-tls-client" module defines the following "grouping" statement:

* tls-client-grouping

This grouping is presented in the following subsection.

3.1.2.1. The "tls-client-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tls-client-grouping" grouping:

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

     grouping tls-client-grouping:
       +-- client-identity!
       |  +-- (auth-type)
       |     +--:(certificate) {client-ident-x509-cert}?
       |     |  +-- certificate
       |     |     +---u ks:inline-or-keystore-end-entity-cert-with-key\
   -grouping
       |     +--:(raw-public-key) {client-ident-raw-public-key}?
       |     |  +-- raw-private-key
       |     |     +---u ks:inline-or-keystore-asymmetric-key-grouping
       |     +--:(tls12-psk) {client-ident-tls12-psk}?
       |     |  +-- tls12-psk
       |     |     +---u ks:inline-or-keystore-symmetric-key-grouping
       |     |     +-- id?
       |     |              string
       |     +--:(tls13-epsk) {client-ident-tls13-epsk}?
       |        +-- tls13-epsk
       |           +---u ks:inline-or-keystore-symmetric-key-grouping
       |           +-- external-identity
       |           |       string
       |           +-- hash?
       |           |       tlscmn:epsk-supported-hash
       |           +-- context?
       |           |       string
       |           +-- target-protocol?
       |           |       uint16
       |           +-- target-kdf?
       |                   uint16
       +-- server-authentication
       |  +-- ca-certs! {server-auth-x509-cert}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- ee-certs! {server-auth-x509-cert}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- raw-public-keys! {server-auth-raw-public-key}?
       |  |  +---u ts:inline-or-truststore-public-keys-grouping
       |  +-- tls12-psks?        empty {server-auth-tls12-psk}?
       |  +-- tls13-epsks?       empty {server-auth-tls13-epsk}?
       +-- hello-params {tlscmn:hello-params}?
       |  +---u tlscmn:hello-params-grouping
       +-- keepalives {tls-client-keepalives}?
          +-- peer-allowed-to-send?   empty
          +-- test-peer-aliveness!
             +-- max-wait?       uint16
             +-- max-attempts?   uint8
```

Comments:

* The "client-identity" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures identity credentials, each enabled by a "feature" statement defined in Section 3.1.1.

* The "server-authentication" node configures trust anchors for authenticating the TLS server, with each option enabled by a "feature" statement.

* The "hello-params" node, which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a "presence" container for testing the aliveness of the TLS server. The aliveness-test occurs at the TLS protocol layer.

* For the referenced grouping statement(s):

  - The "inline-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "inline-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "inline-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "inline-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-trust-anchors].
  - The "inline-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

3.1.3. Protocol-accessible Nodes

The "ietf-tls-client" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not itself define any protocol-accessible nodes.

3.2. Example Usage

This section presents two examples showing the "tls-client-grouping" grouping populated with some data. These examples are effectively the same except the first configures the client identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

The following configuration example uses inline-definitions for the client identity and server authentication:

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

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tls-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-client"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

    <!-- how this client will authenticate itself to the server -->
    <client-identity>
        <certificate>
            <inline-definition>
                <private-key-format>ct:rsa-private-key-format</priva\
te-key-format>
                <cleartext-private-key>BASE64VALUE=</cleartext-priva\
te-key>
                <cert-data>BASE64VALUE=</cert-data>
            </inline-definition>
        </certificate>
    </client-identity>

    <!-- which certificates will this client trust -->
    <server-authentication>
        <ca-certs>
            <inline-definition>
                <certificate>
                    <name>Server Cert Issuer #1</name>
                    <cert-data>BASE64VALUE=</cert-data>
                </certificate>
                <certificate>
                    <name>Server Cert Issuer #2</name>
                    <cert-data>BASE64VALUE=</cert-data>
                </certificate>
            </inline-definition>
```

```
          </ca-certs>
          <ee-certs>
              <inline-definition>
                  <certificate>
                      <name>My Application #1</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
                  <certificate>
                      <name>My Application #2</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
              </inline-definition>
          </ee-certs>
          <raw-public-keys>
              <inline-definition>
                  <public-key>
                      <name>corp-fw1</name>
                      <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
                  </public-key>
                  <public-key>
                      <name>corp-fw2</name>
                      <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
                  </public-key>
              </inline-definition>
          </raw-public-keys>
          <tls12-psks/>
          <tls13-epsks/>
      </server-authentication>

      <keepalives>
          <test-peer-aliveness>
              <max-wait>30</max-wait>
              <max-attempts>3</max-attempts>
          </test-peer-aliveness>
      </keepalives>

  </tls-client>
```

The following configuration example uses central-keystore-references for the client identity and central-truststore-references for server authentication: from the keystore:

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

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <tls-client xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-client">

       <!-- how this client will authenticate itself to the server -->
       <client-identity>
           <certificate>
               <central-keystore-reference>
                   <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                   <certificate>ex-rsa-cert</certificate>
               </central-keystore-reference>
           </certificate>
       </client-identity>

       <!-- which certificates will this client trust -->
       <server-authentication>
           <ca-certs>
               <central-truststore-reference>trusted-server-ca-certs</c\
   entral-truststore-reference>
           </ca-certs>
           <ee-certs>
               <central-truststore-reference>trusted-server-ee-certs</c\
   entral-truststore-reference>
           </ee-certs>
           <raw-public-keys>
               <central-truststore-reference>Raw Public Keys for TLS Se\
   rvers</central-truststore-reference>
           </raw-public-keys>
           <tls12-psks/>
           <tls13-epsks/>
       </server-authentication>

       <keepalives>
           <test-peer-aliveness>
               <max-wait>30</max-wait>
               <max-attempts>3</max-attempts>
           </test-peer-aliveness>
       </keepalives>

   </tls-client>
```

3.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], and Informative references to [RFC5246], [RFC8446], [RFC9258] and [RFC9257].

<CODE BEGINS> file "ietf-tls-client@2024-03-16.yang"

```
module ietf-tls-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-client";
  prefix tlsc;

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

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

  import ietf-truststore {
    prefix ts;
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
  }

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-tls-common {
    prefix tlscmn;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
```

```
       "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
        WG Web:   https://datatracker.ietf.org/wg/netconf
        Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
        Author:   Jeff Hartley <mailto:intensifysecurity@gmail.com>";

     description
       "This module defines reusable groupings for TLS clients that
        can be used as a basis for specific TLS client instances.

        Copyright (c) 2024 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

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

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

     revision 2024-03-16 {
       description
         "Initial version";
       reference
         "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
     }

     // Features

     feature tls-client-keepalives {
       description
         "Per socket TLS keepalive parameters are configurable for
          TLS clients on the server implementing this feature.";
     }

     feature client-ident-x509-cert {
       description
         "Indicates that the client supports identifying itself
```

```
          using X.509 certificates.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile";
      }

      feature client-ident-raw-public-key {
        description
          "Indicates that the client supports identifying itself
           using raw public keys.";
        reference
          "RFC 7250:
             Using Raw Public Keys in Transport Layer Security (TLS)
             and Datagram Transport Layer Security (DTLS)";
      }

      feature client-ident-tls12-psk {
        if-feature "tlscmn:tls12";
        description
          "Indicates that the client supports identifying itself
           using TLS-1.2 PSKs (pre-shared or pairwise-symmetric keys).";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }

      feature client-ident-tls13-epsk {
        if-feature "tlscmn:tls13";
        description
          "Indicates that the client supports identifying itself
           using TLS-1.3 External PSKs (pre-shared keys).";
        reference
          "RFC 8446:
             The Transport Layer Security (TLS) Protocol Version 1.3";
      }

      feature server-auth-x509-cert {
        description
          "Indicates that the client supports authenticating servers
           using X.509 certificates.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile";
      }
```

```
    feature server-auth-raw-public-key {
      description
        "Indicates that the client supports authenticating servers
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }

    feature server-auth-tls12-psk {
      description
        "Indicates that the client supports authenticating servers
         using PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature server-auth-tls13-epsk {
      description
        "Indicates that the client supports authenticating servers
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    // Groupings

    grouping tls-client-grouping {
      description
        "A reusable grouping for configuring a TLS client without
         any consideration for how an underlying TCP session is
         established.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tls-client-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container client-identity {
        nacm:default-deny-write;
```

```
        presence
          "Indicates that a TLS-level client identity has been
           configured.  This statement is present so the mandatory
           descendant do not imply that this node must be configured.";
        description
          "Identity credentials the TLS client MAY present when
           establishing a connection to a TLS server.  If not
           configured, then client authentication is presumed to
           occur in a protocol layer above TLS.  When configured,
           and requested by the TLS server when establishing a
           TLS session, these credentials are passed in the
           Certificate message defined in Section 7.4.2 of
           RFC 5246 and Section 4.4.2 in RFC 8446.";
        reference
          "RFC 5246: The Transport Layer Security (TLS)
                     Protocol Version 1.2
           RFC 8446: The Transport Layer Security (TLS)
                     Protocol Version 1.3
           RFC CCCC: A YANG Data Model for a Keystore";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst authentication types, of which one must
             be enabled (via its associated 'feature') and selected.";
          case certificate {
            if-feature "client-ident-x509-cert";
            container certificate {
              description
                "Specifies the client identity using a certificate.";
              uses
                "ks:inline-or-keystore-end-entity-cert-with-key-"
                + "grouping" {
                refine "inline-or-keystore/inline/inline-definition" {
                  must 'not(public-key-format) or derived-from-or-self'
                     + '(public-key-format, "ct:subject-public-key-'
                     + 'info-format")';
                }
                refine "inline-or-keystore/central-keystore/"
                     + "central-keystore-reference/asymmetric-key" {
                  must 'not(deref(.)/../ks:public-key-format) or '
                     + 'derived-from-or-self(deref(.)/../ks:public-'
                     + 'key-format, "ct:subject-public-key-info-'
                     + 'format")';
                }
              }
            }
          }
          case raw-public-key {
```

```
            if-feature "client-ident-raw-public-key";
            container raw-private-key {
              description
                "Specifies the client identity using a raw
                 private key.";
              uses ks:inline-or-keystore-asymmetric-key-grouping {
                refine "inline-or-keystore/inline/inline-definition" {
                  must 'not(public-key-format) or derived-from-or-self'
                     + '(public-key-format, "ct:subject-public-key-'
                     + 'info-format")';
                }
                refine "inline-or-keystore/central-keystore/"
                     + "central-keystore-reference" {
                  must 'not(deref(.)/../ks:public-key-format) or '
                     + 'derived-from-or-self(deref(.)/../ks:public-'
                     + 'key-format, "ct:subject-public-key-info-'
                     + 'format")';
                }
              }
            }
          }
          case tls12-psk {
            if-feature "client-ident-tls12-psk";
            container tls12-psk {
              description
                "Specifies the client identity using a PSK (pre-shared
                 or pairwise-symmetric key).";
              uses ks:inline-or-keystore-symmetric-key-grouping;
              leaf id {
                type string;
                description
                  "The key 'psk_identity' value used in the TLS
                   'ClientKeyExchange' message.";
                reference
                  "RFC 4279: Pre-Shared Key Ciphersuites for
                             Transport Layer Security (TLS)";
              }
            }
          }
          case tls13-epsk {
            if-feature "client-ident-tls13-epsk";
            container tls13-epsk {
              description
                "An External Pre-Shared Key (EPSK) is established
                or provisioned out-of-band, i.e., not from a TLS
                connection.  An EPSK is a tuple of (Base Key,
                External Identity, Hash).  External PSKs MUST NOT
                be imported for (D)TLS 1.2 or prior versions.  When
```

```
            PSKs are provisioned out of band, the PSK identity
            and the KDF hash algorithm to be used with the PSK
            MUST also be provisioned.

            The structure of this container is designed to
            satisfy the requirements of RFC 8446 Section
            4.2.11, the recommendations from Section 6 in
            RFC 9257, and the EPSK input fields detailed in
            Section 5.1 in RFC 9258.  The base-key is based
            upon ks:inline-or-keystore-symmetric-key-grouping
            in order to provide users with flexible and
            secure storage options.";
          reference
            "RFC 8446: The Transport Layer Security (TLS)
                       Protocol Version 1.3
             RFC 9257: Guidance for External Pre-Shared Key
                       (PSK) Usage in TLS
             RFC 9258: Importing External Pre-Shared Keys
                       (PSKs) for TLS 1.3";
          uses ks:inline-or-keystore-symmetric-key-grouping;
          leaf external-identity {
            type string;
            mandatory true;
            description
              "As per Section 4.2.11 of RFC 8446, and Section 4.1
               of RFC 9257, a sequence of bytes used to identify
               an EPSK. A label for a pre-shared key established
               externally.";
            reference
              "RFC 8446: The Transport Layer Security (TLS)
                         Protocol Version 1.3
               RFC 9257: Guidance for External Pre-Shared Key
                         (PSK) Usage in TLS";
          }
          leaf hash {
            type tlscmn:epsk-supported-hash;
            default sha-256;
            description
              "As per Section 4.2.11 of RFC 8446, for externally
               established PSKs, the Hash algorithm MUST be set
               when the PSK is established or default to SHA-256
               if no such algorithm is defined.  The server MUST
               ensure that it selects a compatible PSK (if any)
               and cipher suite.  Each PSK MUST only be used with
               a single hash function.";
            reference
              "RFC 8446: The Transport Layer Security (TLS)
                         Protocol Version 1.3";
```

```
              }
              leaf context {
                type string;
                description
                  "Per Section 5.1 of RFC 9258, context MUST include
                   the context used to determine the EPSK, if
                   any exists. For example, context may include
                   information about peer roles or identities
                   to mitigate Selfie-style reflection attacks.
                   Since the EPSK is a key derived from an external
                   protocol or sequence of protocols, context MUST
                   include a channel binding for the deriving
                   protocols [RFC5056].  The details of this
                   binding are protocol specfic and out of scope
                   for this document.";
                reference
                  "RFC 9258: Importing External Pre-Shared Keys
                              (PSKs) for TLS 1.3";
              }
              leaf target-protocol {
                type uint16;
                description
                  "As per Section 3 of RFC 9258, the protocol
                   for which a PSK is imported for use.";
                reference
                  "RFC 9258: Importing External Pre-Shared Keys
                              (PSKs) for TLS 1.3";
              }
              leaf target-kdf {
                type uint16;
                description
                  "As per Section 3 of RFC 9258, the KDF for
                   which a PSK is imported for use.";
                reference
                  "RFC 9258: Importing External Pre-Shared Keys
                              (PSKs) for TLS 1.3";
              }
            }
          }
        }
      } // container client-identity

      container server-authentication {
        nacm:default-deny-write;
        must 'ca-certs or ee-certs or raw-public-keys or tls12-psks
          or tls13-epsks';
        description
          "Specifies how the TLS client can authenticate TLS servers.
```

```
           Any combination of credentials is additive and unordered.

           Note that no configuration is required for PSK (pre-shared
           or pairwise-symmetric key) based authentication as the key
           is necessarily the same as configured in the '../client-
           identity' node.";
         container ca-certs {
           if-feature "server-auth-x509-cert";
           presence
             "Indicates that CA certificates have been configured.
              This statement is present so the mandatory descendant
              nodes do not imply that this node must be configured.";
           description
             "A set of certificate authority (CA) certificates used by
              the TLS client to authenticate TLS server certificates.
              A server certificate is authenticated if it has a valid
              chain of trust to a configured CA certificate.";
           reference
             "RFC BBBB: A YANG Data Model for a Truststore";
           uses ts:inline-or-truststore-certs-grouping;
         }
         container ee-certs {
           if-feature "server-auth-x509-cert";
           presence
             "Indicates that EE certificates have been configured.
              This statement is present so the mandatory descendant
              nodes do not imply that this node must be configured.";
           description
             "A set of server certificates (i.e., end entity
              certificates) used by the TLS client to authenticate
              certificates presented by TLS servers.  A server
              certificate is authenticated if it is an exact
              match to a configured server certificate.";
           reference
             "RFC BBBB: A YANG Data Model for a Truststore";
           uses ts:inline-or-truststore-certs-grouping;
         }
         container raw-public-keys {
           if-feature "server-auth-raw-public-key";
           presence
             "Indicates that raw public keys have been configured.
              This statement is present so the mandatory descendant
              nodes do not imply that this node must be configured.";
           description
             "A set of raw public keys used by the TLS client to
              authenticate raw public keys presented by the TLS
              server.  A raw public key is authenticated if it
              is an exact match to a configured raw public key.";
```

```
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-public-keys-grouping {
            refine "inline-or-truststore/inline/inline-definition/"
                 + "public-key" {
              must 'derived-from-or-self(public-key-format,'
                 + ' "ct:subject-public-key-info-format")';
            }
            refine "inline-or-truststore/central-truststore/"
                 + "central-truststore-reference" {
              must 'not(deref(.)/../ts:public-key/ts:public-key-'
                 + 'format[not(derived-from-or-self(., "ct:subject-'
                 + 'public-key-info-format"))])';
            }
          }
        }
        leaf tls12-psks {
          if-feature "server-auth-tls12-psk";
          type empty;
          description
            "Indicates that the TLS client can authenticate TLS servers
             using configured PSKs (pre-shared or pairwise-symmetric
             keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'client-identity'
             node.";
        }
        leaf tls13-epsks {
          if-feature "server-auth-tls13-epsk";
          type empty;
          description
            "Indicates that the TLS client can authenticate TLS servers
             using configured external PSKs (pre-shared keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'client-identity'
             node.";
        }
      } // container server-authentication

      container hello-params {
        nacm:default-deny-write;
        if-feature "tlscmn:hello-params";
        uses tlscmn:hello-params-grouping;
        description
          "Configurable parameters for the TLS hello message.";
      } // container hello-params
```

```
      container keepalives {
        nacm:default-deny-write;
        if-feature "tls-client-keepalives";
        description
          "Configures the keepalive policy for the TLS client.";
        leaf peer-allowed-to-send {
          type empty;
          description
            "Indicates that the remote TLS server is allowed to send
             HeartbeatRequest messages, as defined by RFC 6520
             to this TLS client.";
          reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
        }
        container test-peer-aliveness {
          presence
            "Indicates that the TLS client proactively tests the
             aliveness of the remote TLS server.";
          description
            "Configures the keep-alive policy to proactively test
             the aliveness of the TLS server.  An unresponsive
             TLS server is dropped after approximately max-wait
             * max-attempts seconds.  The TLS client MUST send
             HeartbeatRequest messages, as defined by RFC 6520.";
          reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
          leaf max-wait {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            default "30";
            description
              "Sets the amount of time in seconds after which if
               no data has been received from the TLS server, a
               TLS-level message will be sent to test the
               aliveness of the TLS server.";
          }
          leaf max-attempts {
            type uint8;
            default "3";
            description
              "Sets the maximum number of sequential keep-alive
               messages that can fail to obtain a response from
               the TLS server before assuming the TLS server is
               no longer alive.";
```

```
          }
        }
      }
    } // grouping tls-client-grouping

  }
```

<CODE ENDS>

4. The "ietf-tls-server" Module

This section defines a YANG 1.1 module called "ietf-tls-server". A high-level overview of the module is provided in Section 4.1. Examples illustrating the module's use are provided in Examples (Section 4.2). The YANG module itself is defined in Section 4.3.

4.1. Data Model Overview

This section provides an overview of the "ietf-tls-server" module in terms of its features and groupings.

4.1.1. Features

The following diagram lists all the "feature" statements defined in the "ietf-tls-server" module:

```
Features:
  +-- tls-server-keepalives
  +-- server-ident-x509-cert
  +-- server-ident-raw-public-key
  +-- server-ident-psk
  +-- client-auth-supported
  +-- client-auth-x509-cert
  +-- client-auth-raw-public-key
  +-- client-auth-psk
```

The diagram above uses syntax that is similar to but not defined in [RFC8340].

Please refer to the YANG module for a description of each feature.

4.1.2. Groupings

The "ietf-tls-server" module defines the following "grouping" statement:

* tls-server-grouping

This grouping is presented in the following subsection.

4.1.2.1. The "tls-server-grouping" Grouping

The following tree diagram [RFC8340] illustrates the "tls-server-grouping" grouping:

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

     grouping tls-server-grouping:
       +-- server-identity
       |  +-- (auth-type)
       |     +--:(certificate) {server-ident-x509-cert}?
       |     |  +-- certificate
       |     |     +---u ks:inline-or-keystore-end-entity-cert-with-key\
   -grouping
       |     +--:(raw-private-key) {server-ident-raw-public-key}?
       |     |  +-- raw-private-key
       |     |     +---u ks:inline-or-keystore-asymmetric-key-grouping
       |     +--:(tls12-psk) {server-ident-tls12-psk}?
       |     |  +-- tls12-psk
       |     |     +---u ks:inline-or-keystore-symmetric-key-grouping
       |     |     +-- id-hint?
       |     |           string
       |     +--:(tls13-epsk) {server-ident-tls13-epsk}?
       |        +-- tls13-epsk
       |           +---u ks:inline-or-keystore-symmetric-key-grouping
       |           +-- external-identity
       |           |       string
       |           +-- hash?
       |           |       tlscmn:epsk-supported-hash
       |           +-- context?
       |           |       string
       |           +-- target-protocol?
       |           |       uint16
       |           +-- target-kdf?
       |                   uint16
       +-- client-authentication! {client-auth-supported}?
       |  +-- ca-certs! {client-auth-x509-cert}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- ee-certs! {client-auth-x509-cert}?
       |  |  +---u ts:inline-or-truststore-certs-grouping
       |  +-- raw-public-keys! {client-auth-raw-public-key}?
       |  |  +---u ts:inline-or-truststore-public-keys-grouping
       |  +-- tls12-psks?        empty {client-auth-tls12-psk}?
       |  +-- tls13-epsks?       empty {client-auth-tls13-epsk}?
       +-- hello-params {tlscmn:hello-params}?
       |  +---u tlscmn:hello-params-grouping
       +-- keepalives {tls-server-keepalives}?
          +-- peer-allowed-to-send?   empty
          +-- test-peer-aliveness!
             +-- max-wait?       uint16
             +-- max-attempts?   uint8
```

Comments:

* The "server-identity" node configures identity credentials, each of which is enabled by a "feature".

* The "client-authentication" node, which is optionally configured (as client authentication MAY occur at a higher protocol layer), configures trust anchors for authenticating the TLS client, with each option enabled by a "feature" statement.

* The "hello-params" node, which must be enabled by a feature, configures parameters for the TLS sessions established by this configuration.

* The "keepalives" node, which must be enabled by a feature, configures a flag enabling the TLS client to test the aliveness of the TLS server, as well as a "presence" container for testing the aliveness of the TLS client. The aliveness-tests occurs at the TLS protocol layer.

* For the referenced grouping statement(s):

  - The "inline-or-keystore-end-entity-cert-with-key-grouping" grouping is discussed in Section 2.1.3.6 of [I-D.ietf-netconf-keystore].
  - The "inline-or-keystore-asymmetric-key-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-keystore].
  - The "inline-or-keystore-symmetric-key-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-keystore].
  - The "inline-or-truststore-public-keys-grouping" grouping is discussed in Section 2.1.3.4 of [I-D.ietf-netconf-trust-anchors].
  - The "inline-or-truststore-certs-grouping" grouping is discussed in Section 2.1.3.3 of [I-D.ietf-netconf-trust-anchors].
  - The "hello-params-grouping" grouping is discussed in Section 2.1.3.1 in this document.

4.1.3. Protocol-accessible Nodes

The "ietf-tls-server" module defines only "grouping" statements that are used by other modules to instantiate protocol-accessible nodes. Thus this module, when implemented, does not itself define any protocol-accessible nodes.

4.2. Example Usage

This section presents two examples showing the "tls-server-grouping" grouping populated with some data. These examples are effectively the same except the first configures the server identity using a local key while the second uses a key configured in a keystore. Both examples are consistent with the examples presented in Section 2.2.1 of [I-D.ietf-netconf-trust-anchors] and Section 2.2.1 of [I-D.ietf-netconf-keystore].

The following configuration example uses inline-definitions for the server identity and client authentication:

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

<!-- The outermost element below doesn't exist in the data model. -->
<!--  It simulates if the "grouping" were a "container" instead.  -->

<tls-server
  xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server"
  xmlns:ct="urn:ietf:params:xml:ns:yang:ietf-crypto-types">

    <!-- how this server will authenticate itself to the client -->
    <server-identity>
        <certificate>
          <inline-definition>
              <private-key-format>ct:rsa-private-key-format</private\
-key-format>
              <cleartext-private-key>BASE64VALUE=</cleartext-private\
-key>
              <cert-data>BASE64VALUE=</cert-data>
            </inline-definition>
        </certificate>
    </server-identity>

    <!-- which certificates will this server trust -->
    <client-authentication>
        <ca-certs>
            <inline-definition>
                <certificate>
                    <name>Identity Cert Issuer #1</name>
                    <cert-data>BASE64VALUE=</cert-data>
                </certificate>
                <certificate>
                    <name>Identity Cert Issuer #2</name>
                    <cert-data>BASE64VALUE=</cert-data>
                </certificate>
            </inline-definition>
```

```
          </ca-certs>
          <ee-certs>
              <inline-definition>
                  <certificate>
                      <name>Application #1</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
                  <certificate>
                      <name>Application #2</name>
                      <cert-data>BASE64VALUE=</cert-data>
                  </certificate>
              </inline-definition>
          </ee-certs>
          <raw-public-keys>
              <inline-definition>
                  <public-key>
                      <name>User A</name>
                      <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
                  </public-key>
                  <public-key>
                      <name>User B</name>
                      <public-key-format>ct:subject-public-key-info-fo\
rmat</public-key-format>
                      <public-key>BASE64VALUE=</public-key>
                  </public-key>
              </inline-definition>
          </raw-public-keys>
          <tls12-psks/>
          <tls13-epsks/>
      </client-authentication>

      <keepalives>
          <peer-allowed-to-send/>
      </keepalives>

  </tls-server>
```

The following configuration example uses central-keystore-references for the server identity and central-truststore-references for client authentication: from the keystore:

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

   <!-- The outermost element below doesn't exist in the data model. -->
   <!--  It simulates if the "grouping" were a "container" instead.  -->

   <tls-server xmlns="urn:ietf:params:xml:ns:yang:ietf-tls-server">

       <!-- how this server will authenticate itself to the client -->
       <server-identity>
           <certificate>
               <central-keystore-reference>
                   <asymmetric-key>rsa-asymmetric-key</asymmetric-key>
                   <certificate>ex-rsa-cert</certificate>
               </central-keystore-reference>
           </certificate>
       </server-identity>

       <!-- which certificates will this server trust -->
       <client-authentication>
           <ca-certs>
               <central-truststore-reference>trusted-client-ca-certs</c\
   entral-truststore-reference>
           </ca-certs>
           <ee-certs>
               <central-truststore-reference>trusted-client-ee-certs</c\
   entral-truststore-reference>
           </ee-certs>
           <raw-public-keys>
               <central-truststore-reference>Raw Public Keys for TLS Cl\
   ients</central-truststore-reference>
           </raw-public-keys>
           <tls12-psks/>
           <tls13-epsks/>
       </client-authentication>

       <keepalives>
           <peer-allowed-to-send/>
       </keepalives>

   </tls-server>
```

4.3. YANG Module

This YANG module has normative references to [I-D.ietf-netconf-trust-anchors] and [I-D.ietf-netconf-keystore], and Informative references to [RFC5246], [RFC8446], [RFC9258] and [RFC9257].

```
<CODE BEGINS> file "ietf-tls-server@2024-03-16.yang"

module ietf-tls-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-tls-server";
  prefix tlss;

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

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

  import ietf-truststore {
    prefix ts;
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
  }

  import ietf-keystore {
    prefix ks;
    reference
      "RFC CCCC: A YANG Data Model for a Keystore";
  }

  import ietf-tls-common {
    prefix tlscmn;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

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

  contact
    "WG List:  NETCONF WG list <mailto:netconf@ietf.org>
     WG Web:   https://datatracker.ietf.org/wg/netconf
     Author:   Kent Watsen <mailto:kent+ietf@watsen.net>
     Author:   Jeff Hartley <mailto:intensifysecurity@gmail.com>";

  description
    "This module defines reusable groupings for TLS servers that
```

```
        can be used as a basis for specific TLS server instances.

        Copyright (c) 2024 IETF Trust and the persons identified
        as authors of the code. All rights reserved.

        Redistribution and use in source and binary forms, with
        or without modification, is permitted pursuant to, and
        subject to the license terms contained in, the Revised
        BSD License set forth in Section 4.c of the IETF Trust's
        Legal Provisions Relating to IETF Documents
        (https://trustee.ietf.org/license-info).

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

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

      revision 2024-03-16 {
        description
          "Initial version";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      // Features

      feature tls-server-keepalives {
        description
          "Per socket TLS keepalive parameters are configurable for
           TLS servers on the server implementing this feature.";
      }

      feature server-ident-x509-cert {
        description
          "Indicates that the server supports identifying itself
           using X.509 certificates.";
        reference
          "RFC 5280:
             Internet X.509 Public Key Infrastructure Certificate
             and Certificate Revocation List (CRL) Profile";
      }
```

```
    feature server-ident-raw-public-key {
      description
        "Indicates that the server supports identifying itself
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }

    feature server-ident-tls12-psk {
      if-feature "tlscmn:tls12";
      description
        "Indicates that the server supports identifying itself
         using TLS-1.2 PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature server-ident-tls13-epsk {
      if-feature "tlscmn:tls13";
      description
        "Indicates that the server supports identifying itself
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    feature client-auth-supported {
      description
        "Indicates that the configuration for how to authenticate
         clients can be configured herein.  TLS-level client
         authentication may not be needed when client authentication
         is expected to occur only at another protocol layer.";
    }

    feature client-auth-x509-cert {
      description
        "Indicates that the server supports authenticating clients
         using X.509 certificates.";
      reference
        "RFC 5280:
           Internet X.509 Public Key Infrastructure Certificate
           and Certificate Revocation List (CRL) Profile";
    }
```

```
    feature client-auth-raw-public-key {
      description
        "Indicates that the server supports authenticating clients
         using raw public keys.";
      reference
        "RFC 7250:
           Using Raw Public Keys in Transport Layer Security (TLS)
           and Datagram Transport Layer Security (DTLS)";
    }

    feature client-auth-tls12-psk {
      description
        "Indicates that the server supports authenticating clients
         using PSKs (pre-shared or pairwise-symmetric keys).";
      reference
        "RFC 4279:
           Pre-Shared Key Ciphersuites for Transport Layer Security
           (TLS)";
    }

    feature client-auth-tls13-epsk {
      description
        "Indicates that the server supports authenticating clients
         using TLS-1.3 External PSKs (pre-shared keys).";
      reference
        "RFC 8446:
           The Transport Layer Security (TLS) Protocol Version 1.3";
    }

    // Groupings

    grouping tls-server-grouping {
      description
        "A reusable grouping for configuring a TLS server without
         any consideration for how underlying TCP sessions are
         established.

         Note that this grouping uses fairly typical descendant
         node names such that a stack of 'uses' statements will
         have name conflicts.  It is intended that the consuming
         data model will resolve the issue (e.g., by wrapping
         the 'uses' statement in a container called
         'tls-server-parameters').  This model purposely does
         not do this itself so as to provide maximum flexibility
         to consuming models.";

      container server-identity {
        nacm:default-deny-write;
```

```
        description
          "A locally-defined or referenced end-entity certificate,
           including any configured intermediate certificates, the
           TLS server will present when establishing a TLS connection
           in its Certificate message, as defined in Section 7.4.2
           in RFC 5246 and Section 4.4.2 in RFC 8446.";
        reference
          "RFC 5246: The Transport Layer Security (TLS) Protocol
                     Version 1.2
           RFC 8446: The Transport Layer Security (TLS) Protocol
                     Version 1.3
           RFC CCCC: A YANG Data Model for a Keystore";
        choice auth-type {
          mandatory true;
          description
            "A choice amongst authentication types, of which one must
             be enabled (via its associated 'feature') and selected.";
          case certificate {
            if-feature "server-ident-x509-cert";
            container certificate {
              description
                "Specifies the server identity using a certificate.";
              uses
                "ks:inline-or-keystore-end-entity-cert-with-key-"
                + "grouping" {
                refine "inline-or-keystore/inline/inline-definition" {
                  must 'not(public-key-format) or derived-from-or-self'
                     + '(public-key-format,' + ' "ct:subject-public-'
                     + 'key-info-format")';
                }
                refine "inline-or-keystore/central-keystore/"
                     + "central-keystore-reference/asymmetric-key" {
                  must 'not(deref(.)/../ks:public-key-format) or '
                     + 'derived-from-or-self(deref(.)/../ks:public-key'
                     + '-format, "ct:subject-public-key-info-format")';
                }
              }
            }
          }
          case raw-private-key {
            if-feature "server-ident-raw-public-key";
            container raw-private-key {
              description
                "Specifies the server identity using a raw
                 private key.";
              uses ks:inline-or-keystore-asymmetric-key-grouping {
                refine "inline-or-keystore/inline/inline-definition" {
                  must 'not(public-key-format) or derived-from-or-self'
```

```
                    + '(public-key-format,' + ' "ct:subject-public-'
                    + 'key-info-format")';
                }
                refine "inline-or-keystore/central-keystore/"
                    + "central-keystore-reference" {
                  must 'not(deref(.)/../ks:public-key-format) or '
                    + 'derived-from-or-self(deref(.)/../ks:public-key'
                    + '-format, "ct:subject-public-key-info-format")';
                }
              }
            }
          }
          case tls12-psk {
            if-feature "server-ident-tls12-psk";
            container tls12-psk {
              description
                "Specifies the server identity using a PSK (pre-shared
                 or pairwise-symmetric key).";
              uses ks:inline-or-keystore-symmetric-key-grouping;
              leaf id-hint {
                type string;
                description
                  "The key 'psk_identity_hint' value used in the TLS
                   'ServerKeyExchange' message.";
                reference
                  "RFC 4279: Pre-Shared Key Ciphersuites for
                             Transport Layer Security (TLS)";
              }
            }
          }
          case tls13-epsk {
            if-feature "server-ident-tls13-epsk";
            container tls13-epsk {
              description
                "An External Pre-Shared Key (EPSK) is established
                or provisioned out-of-band, i.e., not from a TLS
                connection.  An EPSK is a tuple of (Base Key,
                External Identity, Hash).  External PSKs MUST
                NOT be imported for (D)TLS 1.2 or prior versions.
                When PSKs are provisioned out of band, the PSK
                identity and the KDF hash algorithm to be used
                with the PSK MUST also be provisioned.

                The structure of this container is designed to
                satisfy the requirements of RFC 8446 Section
                4.2.11, the recommendations from Section 6 in
                RFC 9257, and the EPSK input fields detailed in
                Section 5.1 in RFC 9258.  The base-key is based
```

```
            upon ks:inline-or-keystore-symmetric-key-grouping
            in order to provide users with flexible and
            secure storage options.";
          reference
            "RFC 8446: The Transport Layer Security (TLS)
                       Protocol Version 1.3
             RFC 9257: Guidance for External Pre-Shared Key
                       (PSK) Usage in TLS
             RFC 9258: Importing External Pre-Shared Keys
                       (PSKs) for TLS 1.3";
          uses ks:inline-or-keystore-symmetric-key-grouping;
          leaf external-identity {
            type string;
            mandatory true;
            description
              "As per Section 4.2.11 of RFC 8446, and Section 4.1
               of RFC 9257, a sequence of bytes used to identify
               an EPSK. A label for a pre-shared key established
               externally.";
            reference
              "RFC 8446: The Transport Layer Security (TLS)
                         Protocol Version 1.3
               RFC 9257: Guidance for External Pre-Shared Key
                         (PSK) Usage in TLS";
          }
          leaf hash {
            type tlscmn:epsk-supported-hash;
            default sha-256;
            description
              "As per Section 4.2.11 of RFC 8446, for externally
               established PSKs, the Hash algorithm MUST be set
               when the PSK is established or default to SHA-256
               if no such algorithm is defined.  The server MUST
               ensure that it selects a compatible PSK (if any)
               and cipher suite.  Each PSK MUST only be used
               with a single hash function.";
            reference
              "RFC 8446: The Transport Layer Security (TLS)
                         Protocol Version 1.3";
          }
          leaf context {
            type string;
            description
              "Per Section 5.1 of RFC 9258, context MUST include
               the context used to determine the EPSK, if
               any exists. For example, context may include
               information about peer roles or identities
               to mitigate Selfie-style reflection attacks.
```

```
                    Since the EPSK is a key derived from an external
                    protocol or sequence of protocols, context MUST
                    include a channel binding for the deriving
                    protocols [RFC5056].  The details of this
                    binding are protocol specfic and out of scope
                    for this document.";
                  reference
                    "RFC 9258: Importing External Pre-Shared Keys
                               (PSKs) for TLS 1.3";
                }
                leaf target-protocol {
                  type uint16;
                  description
                    "As per Section 3.1 of RFC 9258, the protocol
                     for which a PSK is imported for use.";
                  reference
                    "RFC 9258: Importing External Pre-Shared Keys
                               (PSKs) for TLS 1.3";
                }
                leaf target-kdf {
                  type uint16;
                  description
                    "As per Section 3 of RFC 9258, the KDF for
                     which a PSK is imported for use.";
                  reference
                    "RFC 9258: Importing External Pre-Shared Keys
                               (PSKs) for TLS 1.3";
                }
              }
            }
          }
        } // container server-identity

        container client-authentication {
          if-feature "client-auth-supported";
          nacm:default-deny-write;
          must 'ca-certs or ee-certs or raw-public-keys or tls12-psks
            or tls13-epsks';
          presence
            "Indicates that client authentication is supported (i.e.,
             that the server will request clients send certificates).
             If not configured, the TLS server SHOULD NOT request the
             TLS clients provide authentication credentials.";
          description
            "Specifies how the TLS server can authenticate TLS clients.
             Any combination of credentials is additive and unordered.

             Note that no configuration is required for PSK (pre-shared
```

```
           or pairwise-symmetric key) based authentication as the key
           is necessarily the same as configured in the '../server-
           identity' node.";
        container ca-certs {
          if-feature "client-auth-x509-cert";
          presence
            "Indicates that CA certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of certificate authority (CA) certificates used by
             the TLS server to authenticate TLS client certificates.
             A client certificate is authenticated if it has a valid
             chain of trust to a configured CA certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-certs-grouping;
        }
        container ee-certs {
          if-feature "client-auth-x509-cert";
          presence
            "Indicates that EE certificates have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of client certificates (i.e., end entity
             certificates) used by the TLS server to authenticate
             certificates presented by TLS clients. A client
             certificate is authenticated if it is an exact
             match to a configured client certificate.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-certs-grouping;
        }
        container raw-public-keys {
          if-feature "client-auth-raw-public-key";
          presence
            "Indicates that raw public keys have been configured.
             This statement is present so the mandatory descendant
             nodes do not imply that this node must be configured.";
          description
            "A set of raw public keys used by the TLS server to
             authenticate raw public keys presented by the TLS
             client.  A raw public key is authenticated if it
             is an exact match to a configured raw public key.";
          reference
            "RFC BBBB: A YANG Data Model for a Truststore";
          uses ts:inline-or-truststore-public-keys-grouping {
```

```
            refine "inline-or-truststore/inline/inline-definition/"
                 + "public-key" {
              must 'derived-from-or-self(public-key-format,'
                 + ' "ct:subject-public-key-info-format")';
            }
            refine "inline-or-truststore/central-truststore/"
                 + "central-truststore-reference" {
              must 'not(deref(.)/../ts:public-key/ts:public-key-'
                 + 'format[not(derived-from-or-self(., "ct:subject-'
                 + 'public-key-info-format"))])';
            }
          }
        }
        leaf tls12-psks {
          if-feature "client-auth-tls12-psk";
          type empty;
          description
            "Indicates that the TLS server can authenticate TLS clients
             using configured PSKs (pre-shared or pairwise-symmetric
             keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'server-identity'
             node.";
        }
        leaf tls13-epsks {
          if-feature "client-auth-tls13-epsk";
          type empty;
          description
            "Indicates that the TLS 1.3 server can authenticate TLS
             clients using configured external PSKs (pre-shared keys).

             No configuration is required since the PSK value is the
             same as PSK value configured in the 'server-identity'
             node.";
        }
      } // container client-authentication

      container hello-params {
        nacm:default-deny-write;
        if-feature "tlscmn:hello-params";
        uses tlscmn:hello-params-grouping;
        description
          "Configurable parameters for the TLS hello message.";
      } // container hello-params

      container keepalives {
        nacm:default-deny-write;
```

```
        if-feature "tls-server-keepalives";
        description
          "Configures the keepalive policy for the TLS server.";
        leaf peer-allowed-to-send {
          type empty;
          description
            "Indicates that the remote TLS client is allowed to send
             HeartbeatRequest messages, as defined by RFC 6520
             to this TLS server.";
          reference
            "RFC 6520: Transport Layer Security (TLS) and Datagram
             Transport Layer Security (DTLS) Heartbeat Extension";
        }
        container test-peer-aliveness {
          presence
            "Indicates that the TLS server proactively tests the
             aliveness of the remote TLS client.";
          description
            "Configures the keep-alive policy to proactively test
             the aliveness of the TLS client.  An unresponsive
             TLS client is dropped after approximately max-wait
             * max-attempts seconds.";
          leaf max-wait {
            type uint16 {
              range "1..max";
            }
            units "seconds";
            default "30";
            description
              "Sets the amount of time in seconds after which if
               no data has been received from the TLS client, a
               TLS-level message will be sent to test the
               aliveness of the TLS client.";
          }
          leaf max-attempts {
            type uint8;
            default "3";
            description
              "Sets the maximum number of sequential keep-alive
               messages that can fail to obtain a response from
               the TLS client before assuming the TLS client is
               no longer alive.";
          }
        }
      } // container keepalives
    } // grouping tls-server-grouping

  }
```

<CODE ENDS>

5. Security Considerations

The three IETF YANG modules in this document define groupings and will not be deployed as standalone modules. Their security implications may be context dependent based on their use in other modules. The designers of modules which import these grouping must conduct their own analysis of the security considerations.

5.1. Considerations for the "iana-tls-cipher-suite-algs" Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

The "iana-tls-cipher-suite-algs" YANG module defines a data model that is designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

This YANG module defines YANG enumerations, for a public IANA-maintained registry.

YANG enumerations are not security-sensitive, as they are statically defined in the publicly-accessible YANG module. IANA MAY deprecate and/or obsolete enumerations over time as needed to address security issues found in the algorithms.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.2. Considerations for the "ietf-tls-common" YANG Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

The "ietf-tls-common" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Please be aware that this YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

None of the writable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-write" extension has not been set for any data nodes defined in this module.

This module defines the RPC "generate-asymmetric-key-pair" that may, if the "ct:cleartext-private-keys" feature is enabled, and the client requests it, return the private clear in cleartext form. It is NOT RECOMMENDED for private keys to pass the server's security perimeter.

This module does not define any actions or notifications, and thus the security consideration for such is not provided here.

5.3. Considerations for the "ietf-tls-client" YANG Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

The "ietf-tls-client" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Please be aware that this YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

5.4. Considerations for the "ietf-tls-server" YANG Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

The "ietf-tls-server" YANG module defines "grouping" statements that are designed to be accessed via YANG based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. Both of these protocols have mandatory-to-implement secure transport layers (e.g., SSH, TLS) with mutual authentication.

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Please be aware that this YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

None of the readable data nodes defined in this YANG module are considered sensitive or vulnerable in network environments. The NACM "default-deny-all" extension has not been set for any data nodes defined in this module.

Please be aware that this module uses the "key" and "private-key" nodes from the "ietf-crypto-types" module [I-D.ietf-netconf-crypto-types], where said nodes have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, any modification to a key or reference to a key may dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

6. IANA Considerations

6.1. The "IETF XML" Registry

This document registers four URIs in the "ns" subregistry of the IETF XML Registry [RFC3688]. Following the format in [RFC3688], the following registrations are requested:

URI: urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-common
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-client
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-tls-server
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace.

6.2. The "YANG Module Names" Registry

This document registers four YANG modules in the YANG Module Names registry [RFC6020]. Following the format in [RFC6020], the following registrations are requested:

```
name:         iana-tls-cipher-suite-algs
namespace:    urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs
prefix:       tlscsa
reference:    RFC FFFF

name:         ietf-tls-common
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-common
prefix:       tlscmn
reference:    RFC FFFF

name:         ietf-tls-client
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-client
prefix:       tlsc
reference:    RFC FFFF

name:         ietf-tls-server
namespace:    urn:ietf:params:xml:ns:yang:ietf-tls-server
prefix:       tlss
reference:    RFC FFFF
```

6.3. Considerations for the "iana-tls-cipher-suite-algs" Module

This section follows the template defined in Section 4.30.3.1 of [I-D.ietf-netmod-rfc8407bis].

This document presents a script (see Appendix A) for IANA to use to generate the IANA-maintained "iana-tls-cipher-suite-algs" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [IANA-YANG-PARAMETERS].

IANA is requested to add the following note to the registry:

> New values must not be directly added to the "iana-tls-cipher-suite-algs" YANG module. They must instead be added to the "TLS Cipher Suites" sub-registry of the "Transport Layer Security (TLS) Parameters" registry [IANA-CIPHER-ALGS].

When a value is added to the "TLS Cipher Suites" sub-registry, a new "enum" statement must be added to the "iana-tls-cipher-suite-algs" YANG module. The "enum" statement, and sub-statements thereof, should be defined as follows:

enum
: Replicates a name from the registry.

value
: Contains the decimal value of the IANA-assigned value.

status
: Include only if a registration has been deprecated or obsoleted. An IANA "Recommended" maps to YANG status "deprecated". Since the registry is unable to express a logical "MUST NOT" recommendation, there is no mapping to YANG status "obsolete", which is unfortunate given Moving single-DES and IDEA TLS ciphersuites to Historic (https://datatracker.ietf.org/doc/status-change-tls-des-idea-ciphers-to-historic) .

description
: Contains "Enumeration for the 'TLS_FOO' algorithm.", where "TLS_FOO" is a placeholder for the algorithm's name (e.g., "TLS_PSK_WITH_AES_256_CBC_SHA").

reference
: Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-tls-cipher-suite-algs" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements. The "revision" must have a "description" statement explaining why the the update occurred, and must have a "reference" substatement that points to the document defining the registry update that resulted in this change. For instance:

```
revision 2024-02-02 {
    description
        "This update reflect the update made to the underlying
         Foo Bar registry per RFC XXXX.";
    reference
        "RFC XXXX: Extend the Foo Bars Registry
                   to Support Something Important";
}
```

IANA is requested to add the following note to the "TLS Cipher Suites" sub-registry of the "Transport Layer Security (TLS) Parameters" registry [IANA-CIPHER-ALGS].

> When this registry is modified, the YANG module "iana-tls-cipher-suite-algs" [IANA-YANG-PARAMETERS] must be updated as defined in RFC FFFF.

An initial version of this module can be found in Appendix A.1.

7. References

7.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-33, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-33>.

[I-D.ietf-netconf-keystore]
Watsen, K., "A YANG Data Model for a Keystore and Keystore Operations", Work in Progress, Internet-Draft, draft-ietf-netconf-keystore-34, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-keystore-34>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-27, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-27>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2712] Medvinsky, A. and M. Hur, "Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)", RFC 2712, DOI 10.17487/RFC2712, October 1999, <https://www.rfc-editor.org/info/rfc2712>.

[RFC4162] Lee, H.J., Yoon, J.H., and J.I. Lee, "Addition of SEED Cipher Suites to Transport Layer Security (TLS)", RFC 4162, DOI 10.17487/RFC4162, August 2005, <https://www.rfc-editor.org/info/rfc4162>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <https://www.rfc-editor.org/info/rfc4279>.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, DOI 10.17487/RFC4346, April 2006, <https://www.rfc-editor.org/info/rfc4346>.

[RFC4785] Blumenthal, U. and P. Goel, "Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)", RFC 4785, DOI 10.17487/RFC4785, January 2007, <https://www.rfc-editor.org/info/rfc4785>.

[RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, DOI 10.17487/RFC5054, November 2007, <https://www.rfc-editor.org/info/rfc5054>.

[RFC5288] Salowey, J., Choudhury, A., and D. McGrew, "AES Galois Counter Mode (GCM) Cipher Suites for TLS", RFC 5288, DOI 10.17487/RFC5288, August 2008, <https://www.rfc-editor.org/info/rfc5288>.

[RFC5289] Rescorla, E., "TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)", RFC 5289, DOI 10.17487/RFC5289, August 2008, <https://www.rfc-editor.org/info/rfc5289>.

[RFC5469] Eronen, P., Ed., "DES and IDEA Cipher Suites for Transport Layer Security (TLS)", RFC 5469, DOI 10.17487/RFC5469, February 2009, <https://www.rfc-editor.org/info/rfc5469>.

[RFC5487] Badra, M., "Pre-Shared Key Cipher Suites for TLS with SHA-256/384 and AES Galois Counter Mode", RFC 5487, DOI 10.17487/RFC5487, March 2009, <https://www.rfc-editor.org/info/rfc5487>.

[RFC5489] Badra, M. and I. Hajjeh, "ECDHE_PSK Cipher Suites for Transport Layer Security (TLS)", RFC 5489, DOI 10.17487/RFC5489, March 2009, <https://www.rfc-editor.org/info/rfc5489>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <https://www.rfc-editor.org/info/rfc5746>.

[RFC5932] Kato, A., Kanda, M., and S. Kanno, "Camellia Cipher Suites for TLS", RFC 5932, DOI 10.17487/RFC5932, June 2010, <https://www.rfc-editor.org/info/rfc5932>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6209] Kim, W., Lee, J., Park, J., and D. Kwon, "Addition of the ARIA Cipher Suites to Transport Layer Security (TLS)", RFC 6209, DOI 10.17487/RFC6209, April 2011, <https://www.rfc-editor.org/info/rfc6209>.

[RFC6367] Kanno, S. and M. Kanda, "Addition of the Camellia Cipher Suites to Transport Layer Security (TLS)", RFC 6367, DOI 10.17487/RFC6367, September 2011, <https://www.rfc-editor.org/info/rfc6367>.

[RFC6655] McGrew, D. and D. Bailey, "AES-CCM Cipher Suites for Transport Layer Security (TLS)", RFC 6655, DOI 10.17487/RFC6655, July 2012, <https://www.rfc-editor.org/info/rfc6655>.

[RFC7251] McGrew, D., Bailey, D., Campagna, M., and R. Dugal, "AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites for TLS", RFC 7251, DOI 10.17487/RFC7251, June 2014, <https://www.rfc-editor.org/info/rfc7251>.

[RFC7507] Moeller, B. and A. Langley, "TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks", RFC 7507, DOI 10.17487/RFC7507, April 2015, <https://www.rfc-editor.org/info/rfc7507>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <https://www.rfc-editor.org/info/rfc7589>.

[RFC7905] Langley, A., Chang, W., Mavrogiannopoulos, N., Strombergson, J., and S. Josefsson, "ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)", RFC 7905, DOI 10.17487/RFC7905, June 2016, <https://www.rfc-editor.org/info/rfc7905>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8422] Nir, Y., Josefsson, S., and M. Pegourie-Gonnard, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier", RFC 8422, DOI 10.17487/RFC8422, August 2018, <https://www.rfc-editor.org/info/rfc8422>.

[RFC8442] Mattsson, J. and D. Migault, "ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS 1.2 and DTLS 1.2", RFC 8442, DOI 10.17487/RFC8442, September 2018, <https://www.rfc-editor.org/info/rfc8442>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8492] Harkins, D., Ed., "Secure Password Ciphersuites for Transport Layer Security (TLS)", RFC 8492, DOI 10.17487/RFC8492, February 2019, <https://www.rfc-editor.org/info/rfc8492>.

[RFC8998] Yang, P., "ShangMi (SM) Cipher Suites for TLS 1.3", RFC 8998, DOI 10.17487/RFC8998, March 2021, <https://www.rfc-editor.org/info/rfc8998>.

[RFC9150] Cam-Winget, N. and J. Visoky, "TLS 1.3 Authentication and Integrity-Only Cipher Suites", RFC 9150, DOI 10.17487/RFC9150, April 2022, <https://www.rfc-editor.org/info/rfc9150>.

[RFC9189] Smyshlyaev, S., Ed., Belyavsky, D., and E. Alekseev, "GOST Cipher Suites for Transport Layer Security (TLS) Protocol Version 1.2", RFC 9189, DOI 10.17487/RFC9189, March 2022, <https://www.rfc-editor.org/info/rfc9189>.

7.2. Informative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-19, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-19>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-35>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-35, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-35>.

[I-D.ietf-netconf-ssh-client-server]
Watsen, K., "YANG Groupings for SSH Clients and SSH Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-ssh-client-server-39, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-ssh-client-server-39>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-23, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-23>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-40, 1 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-40>.

[I-D.ietf-netmod-rfc8407bis]
Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-09, 28 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-09>.

[I-D.ietf-netmod-system-config]
Ma, Q., Wu, Q., and C. Feng, "System-defined Configuration", Work in Progress, Internet-Draft, draft-ietf-netmod-system-config-05, 21 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-system-config-05>.

[IANA-CIPHER-ALGS]
(IANA), I. A. N. A., "IANA "TLS Cipher Suites" Sub-registry of the "Transport Layer Security (TLS) Parameters" Registry", <https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml#tls-parameters-4>.

[IANA-YANG-PARAMETERS]
"YANG Parameters", n.d., <https://www.iana.org/assignments/yang-parameters>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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, <https://www.rfc-editor.org/info/rfc6241>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8071] Watsen, K., "NETCONF Call Home and RESTCONF Call Home", RFC 8071, DOI 10.17487/RFC8071, February 2017, <https://www.rfc-editor.org/info/rfc8071>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC9257] Housley, R., Hoyland, J., Sethi, M., and C. A. Wood, "Guidance for External Pre-Shared Key (PSK) Usage in TLS", RFC 9257, DOI 10.17487/RFC9257, July 2022, <https://www.rfc-editor.org/info/rfc9257>.

[RFC9258] Benjamin, D. and C. A. Wood, "Importing External Pre-Shared Keys (PSKs) for TLS 1.3", RFC 9258, DOI 10.17487/RFC9258, July 2022, <https://www.rfc-editor.org/info/rfc9258>.

Appendix A. Script to Generate IANA-Maintained YANG Modules

This section is not Normative.

The Python https://www.python.org script contained in this section will create the IANA-maintained module described in this document.

Run the script using the command 'python gen-yang-modules.py', to produce the YANG module file in the current directory.

Be aware that the script does not attempt to copy the "revision" statements from the previous/current YANG module. Copying the revision statements must be done manually.

```
<CODE BEGINS>
=============== NOTE: '\\' line wrapping per RFC 8792 ===============

import re
import csv
import requests
import textwrap
import requests_cache
from io import StringIO
from datetime import datetime

# Metadata for the one YANG module produced by this script
MODULES = [
    {
```

```
        "csv_url": "https://www.iana.org/assignments/tls-parameters/\
  \tls-parameters-4.csv",
        "spaced_name": "cipher-suite",
        "hypenated_name": "cipher-suite",
        "prefix": "tlscsa",
    }
  ]


  def create_module_begin(module, f):

    # Define template for all four modules
    PREAMBLE_TEMPLATE="""
  module iana-tls-HNAME-algs {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:iana-tls-HNAME-algs";
    prefix PREFIX;

    organization
      "Internet Assigned Numbers Authority (IANA)";

    contact
      "Postal: ICANN
               12025 Waterfront Drive, Suite 300
               Los Angeles, CA  90094-2536
               United States of America
       Tel:    +1 310 301 5800
       Email:  iana@iana.org";

    description
      "This module defines enumerations for the Cipher Suite
       algorithms defined in the 'TLS Cipher Suites' sub-registry
       of the 'Transport Layer Security (TLS) Parameters' registry
       maintained by IANA.

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

       Redistribution and use in source and binary forms, with
       or without modification, is permitted pursuant to, and
       subject to the license terms contained in, the Revised
       BSD License set forth in Section 4.c of the IETF Trust's
       Legal Provisions Relating to IETF Documents
       (https://trustee.ietf.org/license-info).

       The initial version of this YANG module is part of RFC FFFF
       (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
       itself for full legal notices.
```

```
        All versions of this module are published by IANA at
        https://www.iana.org/assignments/yang-parameters.";

    revision DATE {
      description
        "This initial version of the module was created using
         the script defined in RFC FFFF to reflect the contents
         of the SNAME algorithms registry maintained by IANA.";
      reference
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
    }

    typedef tls-HNAME-algorithm {
      type enumeration {
"""
    # Replacements
    rep = {
      "DATE": datetime.today().strftime('%Y-%m-%d'),
      "YEAR": datetime.today().strftime('%Y'),
      "SNAME": module["spaced_name"],
      "HNAME": module["hypenated_name"],
      "PREFIX": module["prefix"]
    }

    # Do the replacement
    rep = dict((re.escape(k), v) for k, v in rep.items())
    pattern = re.compile("|".join(rep.keys()))
    text = pattern.sub(lambda m: rep[re.escape(m.group(0))], PREAMBL\
\E_TEMPLATE)

    # Write preamble into the file
    f.write(text)


def create_module_body(module, f):

    # Fetch the current CSV file from IANA
    r = requests.get(module["csv_url"])
    assert r.status_code == 200, "Could not get " + module["csv_url"]

    # Parse each CSV line
    with StringIO(r.text) as csv_file:
        csv_reader = csv.DictReader(csv_file)
        for row in csv_reader:

            # Skip reserved algs
            if row["Description"].startswith("Unassigned"):
                continue
```

```
            # Skip unassigned algs
            if row["Description"].startswith("Reserved"):
                continue

            # Ensure this is the TLS line
            assert row["Description"].startswith("TLS_"), "Unrecogni\
zed description: '" + row["Description"] + "'"

            # Set the 'refs' and 'titles' lists
            if row["Reference"] == "":
                pass # skip when the Reference field is empty

            else:

                # There may be more than one ref
                refs = row["Reference"][1:-1]  # remove the '[' and \
\']' chars
                refs = refs.split("][")
                titles = []
                for ref in refs:

                    # Ascertain the ref's title
                    if ref.startswith("RFC"):

                        # Fetch the current BIBTEX entry
                        bibtex_url="https://datatracker.ietf.org/doc\
\/"+ ref.lower() + "/bibtex/"
                        r = requests.get(bibtex_url)
                        assert r.status_code == 200, "Could not GET \
\" + bibtex_url

                        # Append to 'titles' value from the "title" \
\line
                        for item in r.text.split("\n"):
                            if "title =" in item:
                                title = re.sub('.*{{(.*)}}.*', r'\g<\
\1>', item)
                                if title.startswith("ECDHE\_PSK"):
                                    title = re.sub("ECDHE\\\\_PSK", \
\"ECDHE_PSK", title)
                                titles.append(re.sub('.*{{(.*)}}.*',\
\ r'\g<1>', title))
                                break
                        else:
                            raise Exception("RFC title not found")


                        # Insert a space: "RFCXXXX" --> "RFC XXXX"
```

```
                        index = refs.index(ref)
                        refs[index] = "RFC " + ref[3:]

                    elif ref == "IESG Action 2018-08-16":

                        # Rewrite the ref value
                        index = refs.index(ref)
                        refs[index] = "IESG Action"

                        # Let title be something descriptive
                        titles.append("IESG Action 2018-08-16")

                    elif ref == "draft-irtf-cfrg-aegis-aead-08":

                        # Manually set the draft's title
                        titles.append("The AEGIS Family of Authentic\
   \ated Encryption Algorithms")

                    elif ref:
                        raise Exception(f'ref "{ref}" not found')

                    else:
                        raise Exception(f'ref missing: {row}')

            # Write out the enum
            f.write(f'      enum {row["Description"]} {{\n');
            if row["Recommended"] == 'N':
                f.write(f'        status deprecated;\n')
            f.write(f'        description\n')
            description = f'          "Enumeration for the \'{row["D\
   \escription"]}\' algorithm.";'
            description = textwrap.fill(description, width=69, subse\
   \quent_indent="           ")
            f.write(f'{description}\n')
            f.write('        reference\n')
            f.write('          "')
            if row["Reference"] == "":
                f.write('Missing in IANA registry.')
            else:
                ref_len = len(refs)
                for i in range(ref_len):
                    ref = refs[i]
                    f.write(f'{ref}:\n')
                    title = "             " + titles[i]
                    if i == ref_len - 1:
                        title += '";'
                    title = textwrap.fill(title, width=69, subsequen\
   \t_indent="             ")
```

```
                    f.write(f'{title}')
                    if i != ref_len - 1:
                        f.write('\n           ')
            f.write('\n')
            f.write('      }\n')


   def create_module_end(module, f):

       # Close out the enumeration, typedef, and module
       f.write("    }\n")
       f.write("    description\n")
       f.write(f'      "An enumeration for TLS {module["spaced_name"]} \
   \algorithms.";\n')
       f.write("  }\n")
       f.write('\n')
       f.write('}\n')


   def create_module(module):

       # Install cache for 8x speedup
       requests_cache.install_cache()

       # Ascertain yang module's name
       yang_module_name = "iana-tls-" + module["hypenated_name"] + "-al\
   \gs.yang"

       # Create yang module file
       with open(yang_module_name, "w") as f:
           create_module_begin(module, f)
           create_module_body(module, f)
           create_module_end(module, f)


   def main():
       for module in MODULES:
           create_module(module)


   if __name__ == "__main__":
       main()
   <CODE ENDS>
```

A.1. Initial Module for the "TLS Cipher Suites" Registry

Following are the complete contents to the initial IANA-maintained YANG module. Please note that the date "2024-03-16" reflects the day on which the extraction occurred. Applications SHOULD use the IANA-maintained module, not the module defined in this draft.

This YANG module has normative references to [RFC2712], [RFC4162], [RFC4279], [RFC4346], [RFC4785], [RFC5054], [RFC5246], [RFC5288], [RFC5289], [RFC5469], [RFC5487], [RFC5489], [RFC5746], [RFC5932], [RFC6209], [RFC6367], [RFC6655], [RFC7251], [RFC7507], [RFC7905], [RFC8422], [RFC8442], [RFC8446], [RFC8492], [RFC8998], [RFC9150], [RFC9189], and [RFC8340].

<CODE BEGINS> file "iana-tls-cipher-suite-algs@2024-03-16.yang"

```
module iana-tls-cipher-suite-algs {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-tls-cipher-suite-algs";
  prefix tlscsa;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Postal: ICANN
             12025 Waterfront Drive, Suite 300
             Los Angeles, CA  90094-2536
             United States of America
     Tel:    +1 310 301 5800
     Email:  iana@iana.org";

  description
    "This module defines enumerations for the Cipher Suite
     algorithms defined in the 'TLS Cipher Suites' sub-registry
     of the 'Transport Layer Security (TLS) Parameters' registry
     maintained by IANA.

     Copyright (c) 2024 IETF Trust and the persons identified as
     authors of the code. All rights reserved.

     Redistribution and use in source and binary forms, with
     or without modification, is permitted pursuant to, and
     subject to the license terms contained in, the Revised
     BSD License set forth in Section 4.c of the IETF Trust's
     Legal Provisions Relating to IETF Documents
     (https://trustee.ietf.org/license-info).
```

```
        The initial version of this YANG module is part of RFC FFFF
        (https://www.rfc-editor.org/info/rfcFFFF); see the RFC
        itself for full legal notices.

        All versions of this module are published by IANA at
        https://www.iana.org/assignments/yang-parameters.";

      revision 2024-03-16 {
        description
          "This initial version of the module was created using
           the script defined in RFC FFFF to reflect the contents
           of the cipher-suite algorithms registry maintained by IANA.";
        reference
          "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
      }

      typedef tls-cipher-suite-algorithm {
        type enumeration {
          enum TLS_NULL_WITH_NULL_NULL {
            status deprecated;
            description
              "Enumeration for the 'TLS_NULL_WITH_NULL_NULL' algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_NULL_MD5 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_NULL_MD5' algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_NULL_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_NULL_SHA' algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_EXPORT_WITH_RC4_40_MD5 {
            status deprecated;
            description
```

```
            "Enumeration for the 'TLS_RSA_EXPORT_WITH_RC4_40_MD5'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version 1.1
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_RSA_WITH_RC4_128_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_RC4_128_MD5'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version 1.2
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_RSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version 1.2
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_RSA_WITH_IDEA_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_IDEA_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
```

```
        }
        enum TLS_RSA_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_EXPORT_WITH_DES40_CBC_SHA'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_RSA_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_EXPORT_WITH_DES40_CBC_SHA'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_DH_DSS_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
```

```
        }
        enum TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_EXPORT_WITH_DES40_CBC_SHA'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_DH_RSA_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA' algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
```

```
            1.1";
        }
        enum TLS_DHE_DSS_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA' algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_DHE_RSA_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
```

```
            1.2";
        }
        enum TLS_DH_anon_EXPORT_WITH_RC4_40_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_EXPORT_WITH_RC4_40_MD5'
             algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version 1.1
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_DH_anon_WITH_RC4_128_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_RC4_128_MD5'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version 1.2
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_anon_EXPORT_WITH_DES40_CBC_SHA' algorithm.";
          reference
            "RFC 4346:
               The Transport Layer Security (TLS) Protocol Version
               1.1";
        }
        enum TLS_DH_anon_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 8996:
               Deprecating TLS 1.0 and TLS 1.1";
        }
        enum TLS_DH_anon_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
```

```
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_KRB5_WITH_DES_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_DES_CBC_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_KRB5_WITH_IDEA_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_IDEA_CBC_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_DES_CBC_MD5 {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_DES_CBC_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_3DES_EDE_CBC_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_3DES_EDE_CBC_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_WITH_RC4_128_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_RC4_128_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_KRB5_WITH_IDEA_CBC_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_WITH_IDEA_CBC_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_DES_CBC_40_SHA'
             algorithm.";
          reference
            "RFC 2712:
```

```
            Addition of Kerberos Cipher Suites to Transport Layer
            Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_RC2_CBC_40_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_RC2_CBC_40_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_RC4_40_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_RC4_40_SHA'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_DES_CBC_40_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_RC2_CBC_40_MD5 {
          status deprecated;
          description
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_RC2_CBC_40_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_KRB5_EXPORT_WITH_RC4_40_MD5 {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_KRB5_EXPORT_WITH_RC4_40_MD5'
             algorithm.";
          reference
            "RFC 2712:
               Addition of Kerberos Cipher Suites to Transport Layer
               Security (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_PSK_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_NULL_SHA' algorithm.";
          reference
            "RFC 4785:
               Pre-Shared Key (PSK) Ciphersuites with NULL Encryption
               for Transport Layer Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 4785:
               Pre-Shared Key (PSK) Ciphersuites with NULL Encryption
               for Transport Layer Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 4785:
               Pre-Shared Key (PSK) Ciphersuites with NULL Encryption
               for Transport Layer Security (TLS)";
        }
        enum TLS_RSA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
```

```
          enum TLS_DH_DSS_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_DSS_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_RSA_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DHE_DSS_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DHE_RSA_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_anon_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
```

```
            1.2";
        }
        enum TLS_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DH_DSS_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DH_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DHE_DSS_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
        }
        enum TLS_DHE_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
```

```
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
          }
          enum TLS_DH_anon_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_AES_256_CBC_SHA'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_NULL_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_NULL_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_AES_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_AES_128_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_RSA_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_AES_256_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_DSS_WITH_AES_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_DSS_WITH_AES_128_CBC_SHA256'
```

```
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_DH_RSA_WITH_AES_128_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_RSA_WITH_AES_128_CBC_SHA256'
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_DHE_DSS_WITH_AES_128_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_DSS_WITH_AES_128_CBC_SHA256'
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_RSA_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_128_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA'
```

```
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA' algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA' algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA' algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_CBC_SHA256'
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_DH_DSS_WITH_AES_256_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_DSS_WITH_AES_256_CBC_SHA256'
           algorithm.";
        reference
```

```
            "RFC 5246:
               The Transport Layer Security (TLS) Protocol Version
               1.2";
          }
          enum TLS_DH_RSA_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_AES_256_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_AES_256_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_anon_WITH_AES_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_AES_128_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5246:
                 The Transport Layer Security (TLS) Protocol Version
                 1.2";
          }
          enum TLS_DH_anon_WITH_AES_256_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_AES_256_CBC_SHA256'
```

```
           algorithm.";
        reference
          "RFC 5246:
             The Transport Layer Security (TLS) Protocol Version
             1.2";
      }
      enum TLS_RSA_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_256_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA'
           algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA' algorithm.";
        reference
          "RFC 5932:
             Camellia Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA' algorithm.";
        reference
```

```
            "RFC 5932:
               Camellia Cipher Suites for TLS";
          }
          enum TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA' algorithm.";
            reference
              "RFC 5932:
                 Camellia Cipher Suites for TLS";
          }
          enum TLS_PSK_WITH_RC4_128_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_RC4_128_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)
               RFC 6347:
                 Datagram Transport Layer Security Version 1.2";
          }
          enum TLS_PSK_WITH_3DES_EDE_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_3DES_EDE_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_PSK_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_PSK_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_AES_256_CBC_SHA'
```

```
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }
      enum TLS_DHE_PSK_WITH_RC4_128_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_PSK_WITH_RC4_128_SHA'
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)
           RFC 6347:
             Datagram Transport Layer Security Version 1.2";
      }
      enum TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_PSK_WITH_3DES_EDE_CBC_SHA'
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }
      enum TLS_DHE_PSK_WITH_AES_128_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_PSK_WITH_AES_128_CBC_SHA'
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }
      enum TLS_DHE_PSK_WITH_AES_256_CBC_SHA {
        status deprecated;
        description
          "Enumeration for the 'TLS_DHE_PSK_WITH_AES_256_CBC_SHA'
           algorithm.";
        reference
          "RFC 4279:
             Pre-Shared Key Ciphersuites for Transport Layer Security
             (TLS)";
      }
```

```
          enum TLS_RSA_PSK_WITH_RC4_128_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_RC4_128_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)
               RFC 6347:
                 Datagram Transport Layer Security Version 1.2";
          }
          enum TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_3DES_EDE_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_RSA_PSK_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_RSA_PSK_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_AES_256_CBC_SHA'
               algorithm.";
            reference
              "RFC 4279:
                 Pre-Shared Key Ciphersuites for Transport Layer Security
                 (TLS)";
          }
          enum TLS_RSA_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
```

```
            "RFC 4162:
               Addition of SEED Cipher Suites to Transport Layer
               Security (TLS)";
          }
          enum TLS_DH_DSS_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_DSS_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
              "RFC 4162:
                 Addition of SEED Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DH_RSA_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
              "RFC 4162:
                 Addition of SEED Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
              "RFC 4162:
                 Addition of SEED Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DHE_RSA_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_RSA_WITH_SEED_CBC_SHA'
               algorithm.";
            reference
              "RFC 4162:
                 Addition of SEED Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DH_anon_WITH_SEED_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_anon_WITH_SEED_CBC_SHA'
```

```
           algorithm.";
        reference
          "RFC 4162:
             Addition of SEED Cipher Suites to Transport Layer
             Security (TLS)";
      }
      enum TLS_RSA_WITH_AES_128_GCM_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_WITH_AES_128_GCM_SHA256'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
      }
      enum TLS_RSA_WITH_AES_256_GCM_SHA384 {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_WITH_AES_256_GCM_SHA384'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 {
        description
          "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_GCM_SHA256'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
      }
      enum TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 {
        description
          "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_GCM_SHA384'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
      }
      enum TLS_DH_RSA_WITH_AES_128_GCM_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_DH_RSA_WITH_AES_128_GCM_SHA256'
           algorithm.";
        reference
          "RFC 5288:
             AES Galois Counter Mode (GCM) Cipher Suites for TLS";
```

```
        }
        enum TLS_DH_RSA_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DHE_DSS_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DH_DSS_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DH_DSS_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DH_anon_WITH_AES_128_GCM_SHA256 {
          status deprecated;
```

```
          description
            "Enumeration for the 'TLS_DH_anon_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_DH_anon_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5288:
               AES Galois Counter Mode (GCM) Cipher Suites for TLS";
        }
        enum TLS_PSK_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_PSK_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 {
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_AES_256_GCM_SHA384 {
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_256_GCM_SHA384'
```

```
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_RSA_PSK_WITH_AES_128_GCM_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_PSK_WITH_AES_128_GCM_SHA256'
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_RSA_PSK_WITH_AES_256_GCM_SHA384 {
        status deprecated;
        description
          "Enumeration for the 'TLS_RSA_PSK_WITH_AES_256_GCM_SHA384'
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_PSK_WITH_AES_128_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the 'TLS_PSK_WITH_AES_128_CBC_SHA256'
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_PSK_WITH_AES_256_CBC_SHA384 {
        status deprecated;
        description
          "Enumeration for the 'TLS_PSK_WITH_AES_256_CBC_SHA384'
           algorithm.";
        reference
          "RFC 5487:
             Pre-Shared Key Cipher Suites for TLS with SHA-256/384
             and AES Galois Counter Mode";
      }
      enum TLS_PSK_WITH_NULL_SHA256 {
        status deprecated;
```

```
          description
            "Enumeration for the 'TLS_PSK_WITH_NULL_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_PSK_WITH_NULL_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_NULL_SHA384'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
        enum TLS_DHE_PSK_WITH_NULL_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_NULL_SHA256'
             algorithm.";
          reference
            "RFC 5487:
               Pre-Shared Key Cipher Suites for TLS with SHA-256/384
               and AES Galois Counter Mode";
        }
```

```
          enum TLS_DHE_PSK_WITH_NULL_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_PSK_WITH_NULL_SHA384'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
                 and AES Galois Counter Mode";
          }
          enum TLS_RSA_PSK_WITH_AES_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_AES_128_CBC_SHA256'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
                 and AES Galois Counter Mode";
          }
          enum TLS_RSA_PSK_WITH_AES_256_CBC_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_AES_256_CBC_SHA384'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
                 and AES Galois Counter Mode";
          }
          enum TLS_RSA_PSK_WITH_NULL_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_NULL_SHA256'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
                 and AES Galois Counter Mode";
          }
          enum TLS_RSA_PSK_WITH_NULL_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_RSA_PSK_WITH_NULL_SHA384'
               algorithm.";
            reference
              "RFC 5487:
                 Pre-Shared Key Cipher Suites for TLS with SHA-256/384
```

```
            and AES Galois Counter Mode";
        }
        enum TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA256 {
```

```
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_anon_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_256_CBC_SHA256'
             algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
          reference
            "RFC 5932:
               Camellia Cipher Suites for TLS";
        }
        enum TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
```

```
          'TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
        reference
          "RFC 5932:
            Camellia Cipher Suites for TLS";
      }
      enum TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA256 {
        status deprecated;
        description
          "Enumeration for the
           'TLS_DH_anon_WITH_CAMELLIA_256_CBC_SHA256' algorithm.";
        reference
          "RFC 5932:
            Camellia Cipher Suites for TLS";
      }
      enum TLS_SM4_GCM_SM3 {
        status deprecated;
        description
          "Enumeration for the 'TLS_SM4_GCM_SM3' algorithm.";
        reference
          "RFC 8998:
            ShangMi (SM) Cipher Suites for TLS 1.3";
      }
      enum TLS_SM4_CCM_SM3 {
        status deprecated;
        description
          "Enumeration for the 'TLS_SM4_CCM_SM3' algorithm.";
        reference
          "RFC 8998:
            ShangMi (SM) Cipher Suites for TLS 1.3";
      }
      enum TLS_EMPTY_RENEGOTIATION_INFO_SCSV {
        status deprecated;
        description
          "Enumeration for the 'TLS_EMPTY_RENEGOTIATION_INFO_SCSV'
           algorithm.";
        reference
          "RFC 5746:
            Transport Layer Security (TLS) Renegotiation Indication
            Extension";
      }
      enum TLS_AES_128_GCM_SHA256 {
        description
          "Enumeration for the 'TLS_AES_128_GCM_SHA256' algorithm.";
        reference
          "RFC 8446:
            The Transport Layer Security (TLS) Protocol Version
            1.3";
      }
```

```
        enum TLS_AES_256_GCM_SHA384 {
          description
            "Enumeration for the 'TLS_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 8446:
               The Transport Layer Security (TLS) Protocol Version
               1.3";
        }
        enum TLS_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the 'TLS_CHACHA20_POLY1305_SHA256'
             algorithm.";
          reference
            "RFC 8446:
               The Transport Layer Security (TLS) Protocol Version
               1.3";
        }
        enum TLS_AES_128_CCM_SHA256 {
          description
            "Enumeration for the 'TLS_AES_128_CCM_SHA256' algorithm.";
          reference
            "RFC 8446:
               The Transport Layer Security (TLS) Protocol Version
               1.3";
        }
        enum TLS_AES_128_CCM_8_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_AES_128_CCM_8_SHA256'
             algorithm.";
          reference
            "RFC 8446:
               The Transport Layer Security (TLS) Protocol Version 1.3
             IESG Action:
               IESG Action 2018-08-16";
        }
        enum TLS_AEGIS_256_SHA512 {
          status deprecated;
          description
            "Enumeration for the 'TLS_AEGIS_256_SHA512' algorithm.";
          reference
            "draft-irtf-cfrg-aegis-aead-08:
               The AEGIS Family of Authenticated Encryption
               Algorithms";
        }
        enum TLS_AEGIS_128L_SHA256 {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_AEGIS_128L_SHA256' algorithm.";
          reference
            "draft-irtf-cfrg-aegis-aead-08:
               The AEGIS Family of Authenticated Encryption
               Algorithms";
        }
        enum TLS_FALLBACK_SCSV {
          status deprecated;
          description
            "Enumeration for the 'TLS_FALLBACK_SCSV' algorithm.";
          reference
            "RFC 7507:
               TLS Fallback Signaling Cipher Suite Value (SCSV) for
               Preventing Protocol Downgrade Attacks";
        }
        enum TLS_ECDH_ECDSA_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_ECDSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and Earlier
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
```

```
        }
        enum TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_ECDSA_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_ECDSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and Earlier
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
```

```
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA' algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_RSA_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_RSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
```

```
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and Earlier
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
          }
          enum TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_RSA_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_RSA_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_256_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDHE_RSA_WITH_NULL_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDHE_RSA_WITH_NULL_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
```

```
            Earlier";
        }
        enum TLS_ECDHE_RSA_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_RSA_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and Earlier
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_ECDH_anon_WITH_NULL_SHA {
```

```
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_NULL_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_anon_WITH_RC4_128_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_RC4_128_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and Earlier
               RFC 6347:
                 Datagram Transport Layer Security Version 1.2";
          }
          enum TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_anon_WITH_AES_128_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_AES_128_CBC_SHA'
               algorithm.";
            reference
              "RFC 8422:
                 Elliptic Curve Cryptography (ECC) Cipher Suites for
                 Transport Layer Security (TLS) Versions 1.2 and
                 Earlier";
          }
          enum TLS_ECDH_anon_WITH_AES_256_CBC_SHA {
            status deprecated;
            description
              "Enumeration for the 'TLS_ECDH_anon_WITH_AES_256_CBC_SHA'
```

```
            algorithm.";
          reference
            "RFC 8422:
               Elliptic Curve Cryptography (ECC) Cipher Suites for
               Transport Layer Security (TLS) Versions 1.2 and
               Earlier";
        }
        enum TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA' algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the
             'TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA' algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
        }
        enum TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5054:
               Using the Secure Remote Password (SRP) Protocol for TLS
               Authentication";
```

```
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5289:
```

```
            TLS Elliptic Curve Cipher Suites with SHA-256/384 and
            AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 {
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 5289:
```

```
            TLS Elliptic Curve Cipher Suites with SHA-256/384 and
            AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 {
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 5289:
```

```
            TLS Elliptic Curve Cipher Suites with SHA-256/384 and
            AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 5289:
               TLS Elliptic Curve Cipher Suites with SHA-256/384 and
               AES Galois Counter Mode (GCM)";
        }
        enum TLS_ECDHE_PSK_WITH_RC4_128_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_RC4_128_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)
             RFC 6347:
               Datagram Transport Layer Security Version 1.2";
        }
        enum TLS_ECDHE_PSK_WITH_3DES_EDE_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_3DES_EDE_CBC_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA256' algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_256_CBC_SHA384' algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_NULL_SHA {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_NULL_SHA'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_NULL_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_NULL_SHA256'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_NULL_SHA384 {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_PSK_WITH_NULL_SHA384'
             algorithm.";
          reference
            "RFC 5489:
               ECDHE_PSK Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_DSS_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_DSS_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
```

```
        }
        enum TLS_DH_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_RSA_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_DSS_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_DSS_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_DSS_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
```

```
            Addition of the ARIA Cipher Suites to Transport Layer
            Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_anon_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_anon_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_ARIA_256_CBC_SHA384' algorithm.";
```

```
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_ARIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_ARIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
```

```
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_ARIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384 {
```

```
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_RSA_WITH_ARIA_256_GCM_SHA384'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DH_RSA_WITH_ARIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_ARIA_128_GCM_SHA256'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DH_RSA_WITH_ARIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DH_RSA_WITH_ARIA_256_GCM_SHA384'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_ARIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_ARIA_128_GCM_SHA256'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_ARIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_DHE_DSS_WITH_ARIA_256_GCM_SHA384'
               algorithm.";
            reference
              "RFC 6209:
                 Addition of the ARIA Cipher Suites to Transport Layer
                 Security (TLS)";
```

```
        }
        enum TLS_DH_DSS_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_DSS_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_DSS_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_anon_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DH_anon_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DH_anon_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_ARIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6209:
```

```
            Addition of the ARIA Cipher Suites to Transport Layer
            Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_ARIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_ARIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_ARIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_ARIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_ARIA_256_GCM_SHA384' algorithm.";
```

```
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_ARIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_ARIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_DHE_PSK_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_ARIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_ARIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_ARIA_256_GCM_SHA384 {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_ARIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_ARIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_ARIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_PSK_WITH_ARIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
```

```
        }
        enum TLS_ECDHE_PSK_WITH_ARIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_ARIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_ARIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_ARIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6209:
               Addition of the ARIA Cipher Suites to Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_CBC_SHA256'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_CBC_SHA384'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
```

```
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
```

```
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_CAMELLIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_RSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_RSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DH_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
```

```
            status deprecated;
            description
              "Enumeration for the
               'TLS_DH_RSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_DH_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DH_RSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_CAMELLIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DHE_DSS_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_DHE_DSS_WITH_CAMELLIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DHE_DSS_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_DH_DSS_WITH_CAMELLIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_DH_DSS_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
```

```
        }
        enum TLS_DH_DSS_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_DSS_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DH_anon_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_anon_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DH_anon_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DH_anon_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CAMELLIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CAMELLIA_256_GCM_SHA384'
             algorithm.";
```

```
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_ECDSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
```

```
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDH_RSA_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_PSK_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_CAMELLIA_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_PSK_WITH_CAMELLIA_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_CAMELLIA_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_CAMELLIA_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_PSK_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_CAMELLIA_256_GCM_SHA384 {
```

```
            status deprecated;
            description
              "Enumeration for the
               'TLS_DHE_PSK_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_RSA_PSK_WITH_CAMELLIA_128_GCM_SHA256 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_RSA_PSK_WITH_CAMELLIA_128_GCM_SHA256' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_RSA_PSK_WITH_CAMELLIA_256_GCM_SHA384 {
            status deprecated;
            description
              "Enumeration for the
               'TLS_RSA_PSK_WITH_CAMELLIA_256_GCM_SHA384' algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_PSK_WITH_CAMELLIA_128_CBC_SHA256 {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_CAMELLIA_128_CBC_SHA256'
               algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
          }
          enum TLS_PSK_WITH_CAMELLIA_256_CBC_SHA384 {
            status deprecated;
            description
              "Enumeration for the 'TLS_PSK_WITH_CAMELLIA_256_CBC_SHA384'
               algorithm.";
            reference
              "RFC 6367:
                 Addition of the Camellia Cipher Suites to Transport
                 Layer Security (TLS)";
```

```
        }
        enum TLS_DHE_PSK_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_PSK_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_DHE_PSK_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_DHE_PSK_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_RSA_PSK_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_RSA_PSK_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_CAMELLIA_128_CBC_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_CAMELLIA_128_CBC_SHA256' algorithm.";
          reference
            "RFC 6367:
```

```
            Addition of the Camellia Cipher Suites to Transport
            Layer Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_CAMELLIA_256_CBC_SHA384 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_CAMELLIA_256_CBC_SHA384' algorithm.";
          reference
            "RFC 6367:
               Addition of the Camellia Cipher Suites to Transport
               Layer Security (TLS)";
        }
        enum TLS_RSA_WITH_AES_128_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_128_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_RSA_WITH_AES_256_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_RSA_WITH_AES_128_CCM {
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_RSA_WITH_AES_256_CCM {
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 6655:
```

```
            AES-CCM Cipher Suites for Transport Layer Security
            (TLS)";
        }
        enum TLS_RSA_WITH_AES_128_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_RSA_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_RSA_WITH_AES_256_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_RSA_WITH_AES_128_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_RSA_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_DHE_RSA_WITH_AES_256_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_WITH_AES_128_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_128_CCM'
             algorithm.";
```

```
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_WITH_AES_256_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_PSK_WITH_AES_128_CCM {
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_128_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_DHE_PSK_WITH_AES_256_CCM {
          description
            "Enumeration for the 'TLS_DHE_PSK_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_WITH_AES_128_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_WITH_AES_256_CCM_8'
             algorithm.";
```

```
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_DHE_WITH_AES_128_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_DHE_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_PSK_DHE_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_PSK_DHE_WITH_AES_256_CCM_8'
             algorithm.";
          reference
            "RFC 6655:
               AES-CCM Cipher Suites for Transport Layer Security
               (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_128_CCM'
             algorithm.";
          reference
            "RFC 7251:
               AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites
               for TLS";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_CCM {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_256_CCM'
             algorithm.";
          reference
            "RFC 7251:
               AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites
               for TLS";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 {
          status deprecated;
          description
```

```
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8'
             algorithm.";
          reference
            "RFC 7251:
               AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites
               for TLS";
        }
        enum TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECDHE_ECDSA_WITH_AES_256_CCM_8'
             algorithm.";
          reference
            "RFC 7251:
               AES-CCM Elliptic Curve Cryptography (ECC) Cipher Suites
               for TLS";
        }
        enum TLS_ECCPWD_WITH_AES_128_GCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECCPWD_WITH_AES_128_GCM_SHA256'
             algorithm.";
          reference
            "RFC 8492:
               Secure Password Ciphersuites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECCPWD_WITH_AES_256_GCM_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECCPWD_WITH_AES_256_GCM_SHA384'
             algorithm.";
          reference
            "RFC 8492:
               Secure Password Ciphersuites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECCPWD_WITH_AES_128_CCM_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_ECCPWD_WITH_AES_128_CCM_SHA256'
             algorithm.";
          reference
            "RFC 8492:
               Secure Password Ciphersuites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECCPWD_WITH_AES_256_CCM_SHA384 {
```

```
          status deprecated;
          description
            "Enumeration for the 'TLS_ECCPWD_WITH_AES_256_CCM_SHA384'
             algorithm.";
          reference
            "RFC 8492:
               Secure Password Ciphersuites for Transport Layer
               Security (TLS)";
        }
        enum TLS_SHA256_SHA256 {
          status deprecated;
          description
            "Enumeration for the 'TLS_SHA256_SHA256' algorithm.";
          reference
            "RFC 9150:
               TLS 1.3 Authentication and Integrity-Only Cipher
               Suites";
        }
        enum TLS_SHA384_SHA384 {
          status deprecated;
          description
            "Enumeration for the 'TLS_SHA384_SHA384' algorithm.";
          reference
            "RFC 9150:
               TLS 1.3 Authentication and Integrity-Only Cipher
               Suites";
        }
        enum TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_KUZNYECHIK_CTR_OMAC'
             algorithm.";
          reference
            "RFC 9189:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.2";
        }
        enum TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_MAGMA_CTR_OMAC' algorithm.";
          reference
            "RFC 9189:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.2";
        }
```

```
        enum TLS_GOSTR341112_256_WITH_28147_CNT_IMIT {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_28147_CNT_IMIT' algorithm.";
          reference
            "RFC 9189:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.2";
        }
        enum TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_L {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_L' algorithm.";
          reference
            "RFC 9367:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.3";
        }
        enum TLS_GOSTR341112_256_WITH_MAGMA_MGM_L {
          status deprecated;
          description
            "Enumeration for the 'TLS_GOSTR341112_256_WITH_MAGMA_MGM_L'
             algorithm.";
          reference
            "RFC 9367:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.3";
        }
        enum TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_S {
          status deprecated;
          description
            "Enumeration for the
             'TLS_GOSTR341112_256_WITH_KUZNYECHIK_MGM_S' algorithm.";
          reference
            "RFC 9367:
               GOST Cipher Suites for Transport Layer Security (TLS)
               Protocol Version 1.3";
        }
        enum TLS_GOSTR341112_256_WITH_MAGMA_MGM_S {
          status deprecated;
          description
            "Enumeration for the 'TLS_GOSTR341112_256_WITH_MAGMA_MGM_S'
             algorithm.";
          reference
            "RFC 9367:
               GOST Cipher Suites for Transport Layer Security (TLS)
```

```
            Protocol Version 1.3";
        }
        enum TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256'
             algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_DHE_RSA_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_PSK_WITH_CHACHA20_POLY1305_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_PSK_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
```

```
        }
        enum TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256 {
          description
            "Enumeration for the
             'TLS_DHE_PSK_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_RSA_PSK_WITH_CHACHA20_POLY1305_SHA256' algorithm.";
          reference
            "RFC 7905:
               ChaCha20-Poly1305 Cipher Suites for Transport Layer
               Security (TLS)";
        }
        enum TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256 {
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_128_GCM_SHA256' algorithm.";
          reference
            "RFC 8442:
               ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS
               1.2 and DTLS 1.2";
        }
        enum TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384 {
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_256_GCM_SHA384' algorithm.";
          reference
            "RFC 8442:
               ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS
               1.2 and DTLS 1.2";
        }
        enum TLS_ECDHE_PSK_WITH_AES_128_CCM_8_SHA256 {
          status deprecated;
          description
            "Enumeration for the
             'TLS_ECDHE_PSK_WITH_AES_128_CCM_8_SHA256' algorithm.";
          reference
            "RFC 8442:
               ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS
               1.2 and DTLS 1.2";
        }
```

```
          enum TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256 {
            description
              "Enumeration for the
               'TLS_ECDHE_PSK_WITH_AES_128_CCM_SHA256' algorithm.";
            reference
              "RFC 8442:
                 ECDHE_PSK with AES-GCM and AES-CCM Cipher Suites for TLS
                 1.2 and DTLS 1.2";
          }
        }
        description
          "An enumeration for TLS cipher-suite algorithms.";
      }

    }

    <CODE ENDS>
```

Appendix B. Change Log

B.1. 00 to 01

* Noted that '0.0.0.0' and '::' might have special meanings.

* Renamed "keychain" to "keystore".

B.2. 01 to 02

* Removed the groupings containing transport-level configuration. Now modules contain only the transport-independent groupings.

* Filled in previously incomplete 'ietf-tls-client' module.

* Added cipher suites for various algorithms into new 'ietf-tls-common' module.

B.3. 02 to 03

* Added a 'must' statement to container 'server-auth' asserting that at least one of the various auth mechanisms must be specified.

* Fixed description statement for leaf 'trusted-ca-certs'.

B.4. 03 to 04

* Updated title to "YANG Groupings for TLS Clients and TLS Servers"

* Updated leafref paths to point to new keystore path

* Changed the YANG prefix for ietf-tls-common from 'tlscom' to 'tlscmn'.

* Added TLS protocol verions 1.0 and 1.1.

* Made author lists consistent

* Now tree diagrams reference ietf-netmod-yang-tree-diagrams

* Updated YANG to use typedefs around leafrefs to common keystore paths

* Now inlines key and certificates (no longer a leafref to keystore)

B.5. 04 to 05

* Merged changes from co-author.

B.6. 05 to 06

* Updated to use trust anchors from trust-anchors draft (was keystore draft)

* Now Uses new keystore grouping enabling asymmetric key to be either locally defined or a reference to the keystore.

B.7. 06 to 07

* factored the tls-[client|server]-groupings into more reusable groupings.

* added if-feature statements for the new "x509-certificates" feature defined in draft-ietf-netconf-trust-anchors.

B.8. 07 to 08

* Added a number of compatibility matrices to Section 5 (thanks Frank!)

* Clarified that any configured "cipher-suite" values need to be compatible with the configured private key.

B.9. 08 to 09

* Updated examples to reflect update to groupings defined in the keystore draft.

* Add TLS keepalives features and groupings.

* Prefixed top-level TLS grouping nodes with 'tls-' and support mashups.

* Updated copyright date, boilerplate template, affiliation, and folding algorithm.

B.10. 09 to 10

* Reformatted the YANG modules.

B.11. 10 to 11

* Collapsed all the inner groupings into the top-level grouping.

* Added a top-level "demux container" inside the top-level grouping.

* Added NACM statements and updated the Security Considerations section.

* Added "presence" statements on the "keepalive" containers, as was needed to address a validation error that appeared after adding the "must" statements into the NETCONF/RESTCONF client/server modules.

* Updated the boilerplate text in module-level "description" statement to match copyeditor convention.

B.12. 11 to 12

* In server model, made 'client-authentication' a 'presence' node indicating that the server supports client authentication.

* In the server model, added a 'required-or-optional' choice to 'client-authentication' to better support protocols such as RESTCONF.

* In the server model, added a 'inline-or-external' choice to 'client-authentication' to better support consuming data models that prefer to keep client auth with client definitions than in a model principally concerned with the "transport".

* In both models, removed the "demux containers", floating the nacm:default-deny-write to each descendant node, and adding a note to model designers regarding the potential need to add their own demux containers.

* Fixed a couple references (section 2 --> section 3)

B.13. 12 to 13

* Updated to reflect changes in trust-anchors drafts (e.g., s/trust-anchors/truststore/g + s/pinned.//)

B.14. 12 to 13

* Removed 'container' under 'client-identity' to match server model.

* Updated examples to reflect change grouping in keystore module.

B.15. 13 to 14

* Removed the "certificate" container from "client-identity" in the ietf-tls-client module.

* Updated examples to reflect ietf-crypto-types change (e.g., identities --> enumerations)

B.16. 14 to 15

* Updated "server-authentication" and "client-authentication" nodes from being a leaf of type "ts:certificates-ref" to a container that uses "ts:inline-or-truststore-certs-grouping".

B.17. 15 to 16

* Removed unnecessary if-feature statements in the -client and -server modules.

* Cleaned up some description statements in the -client and -server modules.

* Fixed a canonical ordering issue in ietf-tls-common detected by new pyang.

B.18. 16 to 17

* Removed choice inline-or-external by removing the 'external' case and flattening the 'local' case and adding a "client-auth-supported" feature.

* Removed choice required-or-optional.

* Updated examples to include the "*-key-format" nodes.

* Augmented-in "must" expressions ensuring that locally-defined public-key-format are "ct:tls-public-key-format" (must expr for ref'ed keys are TBD).

B.19. 17 to 18

* Removed the unused "external-client-auth-supported" feature.

* Made client-indentity optional, as there may be over-the-top auth instead.

* Added augment to uses of inline-or-keystore-symmetric-key-grouping for a psk "id" node.

* Added missing presence container "psks" to ietf-tls-server's "client-authentication" container.

* Updated examples to reflect new "bag" addition to truststore.

* Removed feature-limited caseless 'case' statements to improve tree diagram rendering.

* Refined truststore/keystore groupings to ensure the key formats "must" be particular values.

* Switched to using truststore's new "public-key" bag (instead of separate "ssh-public-key" and "raw-public-key" bags).

* Updated client/server examples to cover ALL cases (local/ref x cert/raw-key/psk).

B.20. 18 to 19

* Updated the "keepalives" containers in part to address Michal Vasko's request to align with RFC 8071, and in part to better align to RFC 6520.

* Removed algorithm-mapping tables from the "TLS Common Model" section

* Removed the 'algorithm' node from the examples.

* Renamed both "client-certs" and "server-certs" to "ee-certs"

* Added a "Note to Reviewers" note to first page.

B.21. 19 to 20

* Modified the 'must' expression in the "ietf-tls-client:server-authention" node to cover the "raw-public-keys" and "psks" nodes also.

* Added a "must 'ca-certs or ee-certs or raw-public-keys or psks'" statement to the ietf-tls-server:client-authentication" node.

* Added "mandatory true" to "choice auth-type" and a "presence" statement to its ancestor.

* Expanded "Data Model Overview section(s) [remove "wall" of tree diagrams].

* Moved the "ietf-tls-common" module section to proceed the other two module sections.

* Updated the Security Considerations section.

B.22. 20 to 21

* Updated examples to reflect new "cleartext-" prefix in the crypto-types draft.

B.23. 21 to 22

* In both the "client-authentication" and "server-authentication" subtrees, replaced the "psks" node from being a P-container to a leaf of type "empty".

* Cleaned up examples (e.g., removed FIXMEs)

* Fixed issues found by the SecDir review of the "keystore" draft.

* Updated the "psk" sections in the "ietf-tls-client" and "ietf-tls-server" modules to more correctly reflect RFC 4279.

B.24. 22 to 23

* Addressed comments raised by YANG Doctor in the ct/ts/ks drafts.

B.25. 23 to 24

* Added missing reference to "FIPS PUB 180-4".

* Added identity "tls-1.3" and updated description statement in other identities indicating that the protocol version is obsolete and enabling the feature is NOT RECOMMENDED.

* Added XML-comment above examples explaining the reason for the unexpected top-most element's presence.

* Added missing "client-ident-raw-public-key" and "client-ident-psk" featutes.

* Aligned modules with 'pyang -f' formatting.

* Fixed nits found by YANG Doctor reviews.

* Added a 'Contributors' section.

B.26. 24 to 25

* Added TLS 1.3 references.

* Clarified support for various TLS protocol versions.

* Moved algorithms in ietf-tls-common (plus more) to IANA-maintained modules

* Added "config false" lists for algorithms supported by the server.

* Fixed issues found during YANG Doctor review.

B.27. 25 to 26

* Replaced "base64encodedvalue==" with "BASE64VALUE=" in examples.

* Minor editorial nits

B.28. 26 to 27

* Fixed up the 'WG Web' and 'WG List' lines in YANG module(s)

* Fixed up copyright (i.e., s/Simplified/Revised/) in YANG module(s).

* Created identityref-based typedef for the IANA alg identity base.

* Major update to support TLS 1.3.

B.29. 27 to 28

* Fixed draft text to refer to new "identity" values (e.g., s/tls-1.3/tls13).

* Added ietf-tls-common:generate-public-key() RPC.

B.30. 28 to 29

* Updated modules to IANA-maintained module in Appendix A to 2022-06-16.

B.31. 29 to 30

* Fixed 'must' expressions.

* Added missing 'revision' statement.

B.32. 30 to 31

* Updated per Shepherd reviews impacting the suite of drafts.

B.33. 31 to 32

* Updated per Shepherd reviews impacting the suite of drafts.

B.34. 32 to 33

* Updated per Tom Petch review.

* Added RPC-reply to 'generate-public-key" RPC example.

B.35. 33 to 34

* Addresses AD review comments.

* Added note to Editor to fix line foldings.

* Introduction now more clearly identifies the "ietf-" and "iana-" modules defined.

* Clarified that the modules, when implemented, do not define any protocol-accessible nodes.

* Clarified that IANA may deprecate and/or obsolete identities over time.

* Added Security Consideration for the "generate-public-key" RPC.

* Added Security Considerations text to also look a SC-section from imported modules.

* Added missing if-feature statements.

* Fixed private-key "must" expressions to not require public-key nodes to be present.

* Fixed ident-tls12-psk and ident-tls13-psk YANG and references.

* Renamed leaf from "bits" to "num-bits".

* Added missing "ordered-by user" statement.

* Added container "private-key-encoding" to wrap existing choice.

* Renamed container "encrypt-with" to "encrypted".

* Renamed leaf from "hide" to "hidden".

* Removed "public-key-format" and "public-key" nodes from examples.

B.36. 34 to 35

* Addresses AD review by Rob Wilton.

B.37. 35 to 36

* Complete tls10/tls11 removal and update Jeff’s email.

B.38. 36 to 37

* Addresses 1st-round of IESG reviews.

B.39. 37 to 39

* Addresses issues found in OpsDir review of the ssh-client-server draft.

* Replaced identities with enums in the IANA module.

* Add refs to where the ’operational’ and ’system’ datastores are defined.

* Updated Introduction to read more like the Abstract

* Updated Editor-notes to NOT remove the script (just remove the initial IANA module)

* Renamed Security Considerations section s/Template for/ Considerations for/

* s/defines/presents/ in a few places.

* Renamed script from 'gen-identities.py' to 'gen-yang-module.py'

* Removed the removeInRFC="true" attribute in Appendix sections

B.40. 39 to 40

* Address IESG review comments.

B.41. 40 to 41

* Updated to reflect comments from Paul Wouters.

* Fixed the "generate-asymmetric-key-pair" RPC to return the location to where hidden keys are created.

Acknowledgements

The authors would like to thank the following for lively discussions on list and in the halls (ordered by first name): Alan Luchuk, Andy Bierman, Balázs Kovács, Benoit Claise, Bert Wijnen, David Lamparter, Dhruv Dhody, Éric Vyncke, Gary Wu, Henk Birkholz, Jeff Hartley, Jürgen Schönwälder, Ladislav Lhotka, Liang Xia, Martin Björklund, Martin Thomson, Mehmet Ersue, Michal Vako, Murray Kucherawy, Paul Wouters, Phil Shafer, Qin Wu, Radek Krejci, Rob Wilton, Roman Danyliw, Russ Housley, Sean Turner, Tom Petch, and Thomas Martin.

Contributors

Special acknowledgement goes to Gary Wu who contributed the "ietf-tls-common" module, and Tom Petch who carefully ensured that references were set correctly throughout.

Author's Address

Kent Watsen
Watsen Networks
Email: kent+ietf@watsen.net