RFC 9456
Updates to the TLS Transport Model for SNMP

Abstract

This document updates RFC 6353 (“Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP”) to reflect changes necessary to support Transport Layer Security version 1.3 (TLS 1.3) and Datagram Transport Layer Security version 1.3 (DTLS 1.3), which are jointly known as “(D)TLS 1.3”. This document is compatible with (D)TLS 1.2 and is intended to be compatible with future versions of SNMP and (D)TLS.

This document updates the SNMP-TLS-TM-MIB as defined in RFC 6353.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9456.

Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction 2
   1.1. The Internet-Standard Management Framework 3
   1.2. Conventions 3
2. Changes from RFC 6353 4
   2.1. TLSTM Fingerprint 4
   2.2. Security Level 5
   2.3. (D)TLS Version 5
3. Additional Rules for TLS 1.3 5
   3.1. Zero Round-Trip Time Resumption (0-RTT) 5
   3.2. TLS Cipher Suites, Extensions, and Protocol Invariants 6
4. MIB Module Definitions 6
5. Security Considerations 26
6. IANA Considerations 26
7. References 27
   7.1. Normative References 27
   7.2. Informative References 29
Acknowledgements 30
Author's Address 30

1. Introduction

This document updates and clarifies how the rules of [RFC6353] apply when using Transport Layer Security (TLS) or Datagram Transport Layer Security (DTLS) versions later than 1.2. This document jointly refers to these two protocols as "(D)TLS". The update also emphasizes the requirement in [RFC8996] prohibiting the use of TLS versions prior to TLS 1.2 [RFC5246] when
using SNMP. Although the text of this document specifically references SNMPv3 and (D)TLS 1.3, this document may be applicable to future versions of these protocols and is backwards compatible with (D)TLS 1.2.

1.1. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to .

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in RFCs 2578, 2579, and 2580 [STD58].

1.2. Conventions

Within this document, the terms "TLS", "DTLS", and "(D)TLS" apply to all versions of the indicated protocols. The term "SNMP" means "SNMPv3" unless a specific version number is indicated. Specific version numbers are used when the text needs to emphasize version numbers.

For consistency with SNMP-related specifications, this document favors terminology as defined in [STD62], rather than favoring terminology that is consistent with non-SNMP specifications. This is consistent with the IESG decision to not require that the SNMP terminology be modified to match the usage of other non-SNMP specifications when SNMP was advanced to an Internet Standard. "Authentication" in this document typically refers to the English meaning of "serving to prove the authenticity of" the message, not data source authentication or peer identity authentication. The terms "manager" and "agent" are not used in this document because, in the architecture defined in RFC 3411 [STD62], all SNMP entities have the capability of acting as manager, agent, or both, depending on the SNMP application types supported in the implementation. Where distinction is necessary, the application names of command generator, command responder, notification originator, notification receiver, and proxy forwarder are used. See "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks" (RFC 3411 [STD62]) for further information.

Throughout this document, the terms "client" and "server" are used to refer to the two ends of the TLS transport connection. The client actively opens the TLS connection, and the server passively listens for the incoming TLS connection. An SNMP entity MAY act as a TLS client, TLS server, or both, depending on the SNMP applications supported.

Throughout this document, the term "session" is used to refer to a secure association between two instances of the TLS Transport Model (TLSTM) that permits the transmission of one or more SNMP messages within the lifetime of the session. The TLS protocol also has an internal notion of a session, and although these two concepts of a session are related, when the term "session" is used, this document is referring to the TLSTM’s specific session and not directly to the TLS protocol’s session.
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.

2. Changes from RFC 6353

This document updates [RFC6353]. The changes from [RFC6353] are defined in the following subsections.

2.1. TLSTM Fingerprint

[RFC6353] defines the SnmpTLSFingerprint textual convention to include the one-octet TLS 1.2 hash algorithm identifier. This one-octet algorithm identifier is only applicable to (D)TLS protocol versions prior to 1.3. The TLS community does not plan to ever add additional values to the "TLS HashAlgorithm" registry [RFC5246], because some might incorrectly infer that using a new hash algorithm with TLS 1.2 would overcome the limitations of TLS 1.2. However, there is still a need within TLSTM to support new values as they are developed.

This document updates the definition of SnmpTLSFingerprint to clarify that the one-octet algorithm identifier uses the values in the IANA "SNMP-TLSTM HashAlgorithms" registry; this registry is consistent with the IANA "TLS HashAlgorithm" registry for its initial values but can be extended as needed to support new hashing algorithms without implying that the new values can be used by TLS version 1.2. This change allows the reuse of the existing fingerprint textual convention and minimizes the impact to [RFC6353].

A "Y" in the "Recommended" column (Table 1) indicates that the registered value has been recommended through a formal Standards Action [RFC8126]. Not all parameters defined in Standards Track documents are necessarily marked as "Recommended".

An "N" in the "Recommended" column does not necessarily mean that the value is flawed; rather, it indicates that the item either has not been through the IETF consensus process, has limited applicability, or is intended only for specific use cases.

The initial values for the "SNMP-TLSTM HashAlgorithms" registry are defined below:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Recommended</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>none</td>
<td>N</td>
<td>[RFC5246]</td>
</tr>
<tr>
<td>1</td>
<td>md5</td>
<td>N</td>
<td>[RFC5246]</td>
</tr>
<tr>
<td>2</td>
<td>sha1</td>
<td>N</td>
<td>[RFC5246]</td>
</tr>
<tr>
<td>3</td>
<td>sha224</td>
<td>Y</td>
<td>[RFC5246]</td>
</tr>
<tr>
<td>4</td>
<td>sha256</td>
<td>Y</td>
<td>[RFC5246]</td>
</tr>
</tbody>
</table>
Values 0 through 2 MUST NOT be used by implementations of this document but are listed for historical consistency.

### 2.2. Security Level

The architecture defined in RFC 3411 [STD62] recognizes three levels of security:

- without authentication and without privacy (noAuthNoPriv)
- with authentication but without privacy (authNoPriv)
- with authentication and with privacy (authPriv)

Cipher suites for (D)TLS 1.3 defined in [RFC8446] provide both authentication and privacy. Cipher suites defined in [RFC9150] for (D)TLS 1.3 provide only authentication, without any privacy protection. Implementations MAY choose to force (D)TLS 1.3 to only allow cipher suites that provide both authentication and privacy.

### 2.3. (D)TLS Version

[RFC6353] states that TLSTM clients and servers MUST NOT request, offer, or use SSL 2.0. [RFC8996] prohibits the use of (D)TLS versions prior to version 1.2. TLSTM MUST only be used with (D)TLS versions 1.2 and later.

### 3. Additional Rules for TLS 1.3

This document specifies additional rules and clarifications for the use of TLS 1.3. These rules may additionally apply to future versions of TLS.

#### 3.1. Zero Round-Trip Time Resumption (0-RTT)

TLS 1.3 implementations for SNMP MUST NOT enable the 0-RTT mode of session resumption (either sending or accepting) and MUST NOT automatically resend 0-RTT data if it is rejected by the server. 0-RTT is disallowed because there are no "safe" SNMP messages that, if replayed, will...
be guaranteed to cause no harm at the server side: all incoming notifications or command responses are meant to be acted upon only once. See Section 5 ("Security Considerations") for further details.

TLSTM clients and servers MUST NOT request, offer, or use the 0-RTT mode of TLS 1.3. [RFC8446] removed the renegotiation supported in TLS 1.2 [RFC5246]; for session resumption, it introduced a zero-RTT (0-RTT) mode, saving a round trip at connection setup at the cost of increased risk of replay attacks (it is possible for servers to guard against this attack by keeping track of all the messages received). [RFC8446] requires that a profile be written for any application that wants to use 0-RTT, specifying which messages are "safe to use" with this mode. Within SNMP, there are no messages that are "safe to use" with this mode.

Renegotiation of sessions is not supported, as it is not supported by TLS 1.3. If a future version of TLS supports renegotiation, this RFC should be updated to indicate whether there are any additional requirements related to its use.

3.2. TLS Cipher Suites, Extensions, and Protocol Invariants

Section 9 of [RFC8446] requires that, in the absence of application profiles, certain cipher suites, TLS extensions, and TLS protocol invariants be mandatory to implement. This document does not specify an application profile; hence, all the compliance requirements in [RFC8446] apply.

4. MIB Module Definitions

This SNMP-TLS-TM-MIB module imports items from RFCs 2578, 2579, and 2580 [STD58], as well as RFCs 3411 and 3413 [STD62]. It also references [RFC1123], [RFC5246], [RFC5280], [RFC5591], [RFC5890], [RFC5952], [RFC5953], [RFC6353], and RFC 2579 [STD58].

```<CODE BEGINS> file "SNMP-TLS-TM-MIB"
SNMP-TLS-TM-MIB DEFINITIONS ::= BEGIN
IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    OBJECT-IDENTITY, mib-2, snmpDomains,
    Counter32, Unsigned32, Gauge32, NOTIFICATION-TYPE
    FROM SNMPv2-SMI            -- RFC 2578 or any update thereof
    TEXTUAL-CONVENTION, TimeStamp, RowStatus, StorageType,
    AutonomousType
    FROM SNMPv2-TC             -- RFC 2579 or any update thereof
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
    FROM SNMPv2-CONF           -- RFC 2580 or any update thereof
    SnmpAdminString
    FROM SNMP-FRAMEWORK-MIB    -- RFC 3411 or any update thereof
    snmpTargetParamsName, snmpTargetAddrName
    FROM SNMP-TARGET-MIB       -- RFC 3413 or any update thereof
;
snmpTlstmMIB MODULE-IDENTITY
LAST-UPDATED "202311080000Z"
</CODE BEGINS>
This is the MIB module for the TLS Transport Model (TLSTM).

Copyright (c) 2023 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 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.

This version of this MIB module is part of RFC 9456; see the RFC itself for full legal notices. This version does the following:

1) Updates the definition of SnmpTLSFingerprint to clarify the registry used for the one-octet hash algorithm identifier.
2) Capitalizes key words in conformance with BCP 14.
3) Replaces 'may not' with 'MUST NOT' to clarify intent in several locations.
4) Replaces 'may not' with a clarification within the definition of SnmpTLSAddress.
5) Applies cosmetic grammar improvements and reformatting causing whitespace changes.

This version of this MIB module is part of RFC 6353; see the RFC itself for full legal notices. The only change was to introduce new wording to reflect required changes for Internationalized Domain Names for Applications (IDNA) addresses in the SnmpTLSAddress textual convention (TC).
DESCRIPTION

"This version of this MIB module is part of
RFC 5953; see the RFC itself for full legal
notices."
::= { mib-2 198 }

-- ************************************************************
-- subtrees of the SNMP-TLS-TM-MIB
-- ************************************************************

snmpTlstmNotifications OBJECT IDENTIFIER ::= { snmpTlstmMIB 0 }
snmpTlstmIdentities OBJECT IDENTIFIER ::= { snmpTlstmMIB 1 }
snmpTlstmObjects OBJECT IDENTIFIER ::= { snmpTlstmMIB 2 }
snmpTlstmConformance OBJECT IDENTIFIER ::= { snmpTlstmMIB 3 }

snmpTlstmHashAlgorithms OBJECT-IDENTITY

STATUS current

DESCRIPTION

"A node used to register hashing algorithm identifiers
recorded in the IANA 'SNMP-TLSTM HashAlgorithms' registry."
::= { snmpTlstmMIB 4 }

-- ************************************************************
-- snmpTlstmObjects - Objects
-- ************************************************************

snmpTLSTCPDomain OBJECT-IDENTITY

STATUS current

DESCRIPTION

"The OBJECT IDENTIFIER representing the TDomain for the
SNMP over TLS via TCP transport domain. The
corresponding transport address is of type SnmpTLSAddress.

The securityName prefix to be associated with the
snmpTLSTCPDomain is 'tls'. This prefix MAY be used by
security models or other components to identify which secure
transport infrastructure authenticated a securityName."

REFERENCE

"TDomain, as defined in RFC 2579: Textual Conventions
for SMIV2"
::= { snmpDomains 8 }

snmpDTLSUDPDomain OBJECT-IDENTITY

STATUS current

DESCRIPTION

"The OBJECT IDENTIFIER representing the TDomain for the
SNMP over DTLS via UDP transport domain. The
corresponding transport address is of type SnmpTLSAddress.

The securityName prefix to be associated with the
snmpDTLSUDPDomain is 'dtls'. This prefix MAY be used by
security models or other components to identify which secure
transport infrastructure authenticated a securityName."

REFERENCE

"TDomain, as defined in RFC 2579: Textual Conventions
for SMIV2"
::= { snmpDomains 9 }
SnmpTLSAddress ::= TEXTUAL-CONVENTION
DISPLAY-HINT "1a"
STATUS current
DESCRIPTION
'Represents an IPv4 address, an IPv6 address, or an ASCII-encoded host name and port number.

An IPv4 address MUST be in dotted decimal format followed by a colon ':' (ASCII character 0x3A) and a decimal port number in ASCII.

An IPv6 address MUST be a colon-separated format (as described in RFC 5952), surrounded by square brackets ('[', ASCII character 0x5B, and ']', ASCII character 0x5D), followed by a colon ':' (ASCII character 0x3A) and a decimal port number in ASCII.

A host name MUST be in ASCII (as per RFC 1123); internationalized host names MUST be encoded as A-labels as specified in RFC 5890. The host name is followed by a colon ':' (ASCII character 0x3A) and a decimal port number in ASCII. The name SHOULD be fully qualified whenever possible.

Values of this textual convention are not guaranteed to be directly usable as transport-layer addressing information, potentially requiring additional processing, such as run-time resolution. As such, applications that write them MUST be prepared for handling errors if such values are not supported or cannot be resolved (if resolution occurs at the time of the management operation).

The DESCRIPTION clause of TransportAddress objects that may have SnmpTLSAddress values MUST fully describe how (and when) such names are to be resolved to IP addresses and vice versa.

This textual convention SHOULD NOT be used directly in object definitions, since it restricts addresses to a specific format. However, if it is used, it MAY be used either on its own or in conjunction with TransportAddressType or TransportDomain as a pair.

When this textual convention is used as a syntax of an index object, there may be issues with the limit of 128 sub-identifiers specified in SMIv2 (STD 58). It is RECOMMENDED that all MIB documents using this textual convention make explicit any limitations on index component lengths that management software MUST observe. This MAY be done by either 1) including SIZE constraints on the index components or 2) specifying applicable constraints in the conceptual row's DESCRIPTION clause or in the surrounding documentation.

REFERENCE
'RFC 1123: Requirements for Internet Hosts - Application and Support
RFC 5890: Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework
RFC 5952: A Recommendation for IPv6 Address Text Representation

SYNTAX OCTET STRING (SIZE (1..255))

SnmpTLSFingerprint ::= TEXTUAL-CONVENTION
DISPLAY-HINT "1x:1x"
STATUS current
DESCRIPTION "A fingerprint value that can be used to uniquely reference other data of potentially arbitrary length.

An SnmpTLSFingerprint value is composed of a one-octet hashing algorithm identifier followed by the fingerprint value. The one-octet identifier value encoded is taken from the IANA 'SNMP-TLSTM HashAlgorithms' registry. The remaining octets of the SnmpTLSFingerprint value are filled using the results of the hashing algorithm.

Historically, the one-octet hashing algorithm identifier was based on the IANA 'TLS HashAlgorithm' registry (RFC 5246); however, this registry is no longer in use for TLS 1.3 and above and is not expected to have any new registrations added to it. To allow the fingerprint algorithm to support additional hashing algorithms that might be used by later versions of (D)TLS, the octet value encoded is now taken from the IANA 'SNMP-TLSTM HashAlgorithms' registry. The initial values within this registry are identical to the values in the 'TLS HashAlgorithm' registry but can be extended to support new hashing algorithms as needed.

This textual convention allows for a zero-length (blank) SnmpTLSFingerprint value for use in tables where the fingerprint value MAY be optional. MIB definitions or implementations MAY refuse to accept a zero-length value as appropriate."

REFERENCE
https://www.iana.org/assignments/smi-numbers/
SYNTAX OCTET STRING (SIZE (0..255))

-- Identities for use in the snmpTlstmCertToTSNTable

snmpTlstmCertToTSNMIdentities OBJECT IDENTIFIER ::= { snmpTlstmIdentities 1 }

snmpTlstmCertSpecified OBJECT-IDENTITY
STATUS current
DESCRIPTION "Directly specifies the tmSecurityName to be used for this certificate. The value of the tmSecurityName to use is specified in the 'snmpTlstmCertToTSNData' column. The 'snmpTlstmCertToTSNData' column MUST contain a non-zero-length SnmpAdminString-compliant value, or the mapping described in this row MUST be considered a failure."
::= { snmpTlstmCertToTSNMIdentities 1 }

Vaughn Standards Track Page 10
snmpTlstmCertSANRFC822Name OBJECT-IDENTITY
STATUS current
DESCRIPTION
"Maps a subjectAltName’s rfc822Name to a tmSecurityName. The local-part of the rfc822Name is passed unaltered, but the domain of the name MUST be passed in lowercase. This mapping results in a 1:1 correspondence between equivalent subjectAltName rfc822Name values and tmSecurityName values, except that the domain of the name MUST be passed in lowercase. Example rfc822Name field: FooBar@Example.COM is mapped to tmSecurityName: FooBar@example.com."
 ::= { snmpTlstmCertToTSNMIdentities 2 }

snmpTlstmCertSANDNSName OBJECT-IDENTITY
STATUS current
DESCRIPTION
"Maps a subjectAltName’s dNSName to a tmSecurityName after first converting it to all lowercase (RFC 5280 does not specify converting to lowercase, so this involves an extra step). This mapping results in a 1:1 correspondence between subjectAltName dNSName values and the tmSecurityName values."
REFERENCE
"RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile"
 ::= { snmpTlstmCertToTSNMIdentities 3 }

snmpTlstmCertSANIPAddress OBJECT-IDENTITY
STATUS current
DESCRIPTION
"Maps a subjectAltName’s iPAddress to a tmSecurityName by transforming the binary-encoded address as follows:

1) For IPv4, the value is converted into a decimal-dotted quad address (e.g., ‘192.0.2.1’).

2) For IPv6 addresses, the value is converted into a 32-character all-lowercase hexadecimal string without any colon separators.

This mapping results in a 1:1 correspondence between subjectAltName iPAddress values and the tmSecurityName values.

The resulting length of an encoded IPv6 address is the maximum length supported by the View-based Access Control Model (VACM). Using an IPv6 address while the value of snmpTsmConfigurationUsePrefix is ‘true’ (see the SNMP-TSM-MIB, as defined in RFC 5591) will result in securityName lengths that exceed what the VACM can handle."
REFERENCE
 ::= { snmpTlstmCertToTSNMIdentities 4 }
The snmpTlstmCertSANAny OBJECT-IDENTITY

STATUS current
DESCRIPTION
"Maps any of the following fields using the corresponding mapping algorithms:

| Type       | Algorithm                           |
|------------+------------------------------------|
| rfc822Name | snmpTlstmCertSANRFC822Name          |
| dNSName    | snmpTlstmCertSANDNSName             |
| iPAddress  | snmpTlstmCertSANIpAddress           |

The first subjectAltName value contained in the certificate that matches any of the above types MUST be used when deriving the tmSecurityName. The mapping algorithm specified in the 'Algorithm' column of the corresponding row MUST be used to derive the tmSecurityName.

This mapping results in a 1:1 correspondence between subjectAltName values and tmSecurityName values. The three sub-mapping algorithms produced by this combined algorithm cannot produce conflicting results between themselves.

::= { snmpTlstmCertToTSNMIIdentities 5 }

The snmpTlstmCertCommonName OBJECT-IDENTITY

STATUS current
DESCRIPTION
"Maps a certificate’s CommonName to a tmSecurityName after converting it to a UTF-8 encoding. The usage of CommonNames is deprecated, and users are encouraged to use subjectAltName mapping methods instead. This mapping results in a 1:1 correspondence between certificate CommonName values and tmSecurityName values."

::= { snmpTlstmCertToTSNMIIdentities 6 }

-- The snmpTlstmSession Group

The snmpTlstmSession OBJECT IDENTIFIER ::= { snmpTlstmObjects 1 }

The snmpTlstmSessionOpens OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of times an openSession() request has been executed as a (D)TLS client, regardless of whether it succeeded or failed."

::= { snmpTlstmSession 1 }

The snmpTlstmSessionClientCloses OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of times a closeSession() request has been executed as a (D)TLS client, regardless of whether it succeeded or failed."
::= { snmpTlstmSession 2 }

snmpTlstmSessionOpenErrors OBJECT-TYPE
SYNTAX     Counter32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"The number of times an openSession() request failed to open a session as a (D)TLS client, for any reason."
::= { snmpTlstmSession 3 }

snmpTlstmSessionAccepts OBJECT-TYPE
SYNTAX     Counter32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"The number of times a (D)TLS server has accepted a new connection from a client and has received at least one SNMP message through it."
::= { snmpTlstmSession 4 }

snmpTlstmSessionServerCloses OBJECT-TYPE
SYNTAX     Counter32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"The number of times a closeSession() request has been executed as a (D)TLS server, regardless of whether it succeeded or failed."
::= { snmpTlstmSession 5 }

snmpTlstmSessionNoSessions OBJECT-TYPE
SYNTAX     Counter32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"The number of times an outgoing message was dropped because the session associated with the passed tmStateReference was no longer (or never) available."
::= { snmpTlstmSession 6 }

snmpTlstmSessionInvalidClientCertificates OBJECT-TYPE
SYNTAX     Counter32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"The number of times an incoming session was not established on a (D)TLS server because the presented client certificate was invalid. Reasons for invalidation include, but are not limited to, cryptographic validation failures or lack of a suitable mapping row in the snmpTlstmCertToTSNTable."
::= { snmpTlstmSession 7 }

snmpTlstmSessionUnknownServerCertificate OBJECT-TYPE
 The number of times an outgoing session was not established on a (D)TLS client because the server certificate presented by an SNMP over (D)TLS server was invalid because no configured fingerprint or Certification Authority (CA) was acceptable to validate it. This may result because there was no entry in the snmpTlstmAddrTable or because no path to a known CA could be found.

::= { snmpTlstmSession 8 }

snmpTlstmSessionInvalidServerCertificates OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of times an outgoing session was not established on a (D)TLS client because the server certificate presented by an SNMP over (D)TLS server could not be validated even if the fingerprint or expected validation path was known. That is, a cryptographic validation error occurred during certificate validation processing.

Reasons for invalidation include, but are not limited to, cryptographic validation failures."

::= { snmpTlstmSession 9 }

snmpTlstmSessionInvalidCaches OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of outgoing messages dropped because the tmStateReference referred to an invalid cache."

::= { snmpTlstmSession 10 }

-- Configuration Objects

snmpTlstmConfig OBJECT IDENTIFIER ::= { snmpTlstmObjects 2 }

-- Certificate mapping

snmpTlstmCertificateMapping OBJECT IDENTIFIER ::= { snmpTlstmConfig 1 }

snmpTlstmCertToTsnCount OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION "A count of the number of entries in the snmpTlstmCertToTsnTable."

::= { snmpTlstmCertificateMapping 1 }
snmpTlstmCertToTSNTableLastChanged OBJECT-TYPE
SYNTAX      TimeStamp
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"The value of sysUpTime.0 when the snmpTlstmCertToTSNTable
was last modified through any means, or 0 if it has not
been modified since the command responder was started."
::= { snmpTlstmCertificateMapping 2 }

snmpTlstmCertToTSNTable OBJECT-TYPE
SYNTAX      SEQUENCE OF SnmpTlstmCertToTSNEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"This table is used by a (D)TLS server to map the (D)TLS
client's presented X.509 certificate to a tmSecurityName.

On an incoming (D)TLS/SNMP connection, the client's
presented certificate either MUST be validated based on an
established trust anchor or MUST directly match a
fingerprint in this table. This table does not provide
any mechanisms for configuring the trust anchors; the
transfer of any needed trusted certificates for path
validation is expected to occur through an out-of-band
transfer.

Once the certificate has been found acceptable (either via
path validation or by directly matching a fingerprint in
this table), this table is consulted to determine the
appropriate tmSecurityName to identify with the remote
connection. This is done by considering each active row
from this table in prioritized order according to its
snmpTlstmCertToTSNID value. Each row's
snmpTlstmCertToTSNID value determines whether the
row is a match for the incoming connection:

1) If the row's snmpTlstmCertToTSNID value
identifies the presented certificate, then consider
the row as a successful match.

2) If the row's snmpTlstmCertToTSNID value
identifies a locally held copy of a trusted CA
certificate and that CA certificate was used to
validate the path to the presented certificate, then
consider the row as a successful match.

Once a matching row has been found, the
snmpTlstmCertToTSNMapType value can be used to determine
how the tmSecurityName to associate with the session
should be determined. See the 'snmpTlstmCertToTSNMapType'
column's DESCRIPTION clause for details on determining the
tmSecurityName value. If it is impossible to determine a
tmSecurityName from the row's data combined with the data
presented in the certificate, then additional rows MUST be
searched to look for another potential match. If a
resulting tmSecurityName mapped from a given row is not
compatible with the needed requirements of a
tmSecurityName (e.g., the VACM imposes a 32-octet-maximum length and the certificate-derived securityName could be longer), then it MUST be considered an invalid match and additional rows MUST be searched to look for another potential match.

If no matching and valid row can be found, the connection MUST be closed and SNMP messages MUST NOT be accepted over it.

Missing values of snmpTlstmCertToTSNID are acceptable, and implementations SHOULD continue to the next-highest-numbered row. It is RECOMMENDED that administrators skip index values to leave room for the insertion of future rows (for example, use values of 10 and 20 when creating initial rows).

Users are encouraged to make use of certificates with subjectAltName fields that can be used as tmSecurityNames. This allows all child certificates of a single root CA certificate to include a subjectAltName that maps directly to a tmSecurityName via a 1:1 transformation. However, this table is flexible, to allow for situations where existing deployed certificate infrastructures do not provide adequate subjectAltName values for use as tmSecurityNames. Certificates MAY also be mapped to tmSecurityNames using the CommonName portion of the Subject field. However, the usage of the CommonName field is deprecated, and thus this usage is NOT RECOMMENDED. Direct mapping from each individual certificate fingerprint to a tmSecurityName is also possible but requires one entry in the table per tmSecurityName and requires more management operations to completely configure a device.

::= { snmpTlstmCertificateMapping 3 }

---

<table>
<thead>
<tr>
<th>SnmpTlstmCertToTSNEntry OBJECT-TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNTAX SnmpTlstmCertToTSNEntry</td>
</tr>
<tr>
<td>MAX-ACCESS not-accessible</td>
</tr>
<tr>
<td>STATUS current</td>
</tr>
<tr>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>&quot;A row in the snmpTlstmCertToTSNTable that specifies a mapping for an incoming (D)TLS certificate to a tmSecurityName to use for a connection.&quot;</td>
</tr>
<tr>
<td>INDEX { snmpTlstmCertToTSNID }</td>
</tr>
<tr>
<td>::= { snmpTlstmCertToTSNTable 1 }</td>
</tr>
</tbody>
</table>

SnmpTlstmCertToTSNEntry ::= SEQUENCE {
    snmpTlstmCertToTSNID           Unsigned32,
    snmpTlstmCertToTSNFingerprint  SnmpTLSFingerprint,
    snmpTlstmCertToTSNMapType      AutonomousType,
    snmpTlstmCertToTSNData         OCTET STRING,
    snmpTlstmCertToTSNStorageType  StorageType,
    snmpTlstmCertToTSNRowStatus    RowStatus
}

snmpTlstmCertToTSNID OBJECT-TYPE
SYNTAX   Unsigned32 (1..4294967295)
MAX-ACCESS not-accessible
STATUS      current
DESCRIPTION
"A unique, prioritized index for the given entry. Lower numbers indicate a higher priority."
::= { snmpTlstmCertToTSNEntry 1 }

snmpTlstmCertToTSNFingerprint OBJECT-TYPE
SYNTAX      SnmpTLSFingerprint (SIZE (1..255))
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"A cryptographic hash of an X.509 certificate. The results of a successful matching fingerprint to either the trusted CA in the certificate validation path or the certificate itself is dictated by the 'snmpTlstmCertToTSNMapType' column."
::= { snmpTlstmCertToTSNEntry 2 }

snmpTlstmCertToTSNMapType OBJECT-TYPE
SYNTAX      AutonomousType
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"Specifies the mapping type for deriving a tmSecurityName from a certificate. Details for mapping of a particular type SHALL be specified in the DESCRIPTION clause of the OBJECT-IDENTITY that describes the mapping. If a mapping succeeds, it will return a tmSecurityName for use by the TLSTM and processing will stop.

If the resulting mapped value is not compatible with the needed requirements of a tmSecurityName (e.g., the VACM imposes a 32-octet-maximum length and the certificate-derived securityName could be longer), then future rows MUST be searched for additional snmpTlstmCertToTSNFingerprint matches to look for a mapping that succeeds.

Suitable values for assigning to this object that are defined within the SNMP-TLS-TM-MIB can be found in the snmpTlstmCertToTSNIdentities portion of the MIB tree."
DEFVAL { snmpTlstmCertSpecified }
::= { snmpTlstmCertToTSNEntry 3 }

snmpTlstmCertToTSNData OBJECT-TYPE
SYNTAX      OCTET STRING (SIZE (0..1024))
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"Auxiliary data used as optional configuration information for a given mapping specified by the 'snmpTlstmCertToTSNMapType' column. Only some mapping systems will make use of this column. The value in this column MUST be ignored for any mapping type that does not require that data be present in this column."
DEFVAL { "" }
::= { snmpTlstmCertToTSNEntry 4 }
snmpTlstmCertToTSNStorageType OBJECT-TYPE
SYNTAX     StorageType
MAX-ACCESS read-create
STATUS     current
DESCRIPTION
"The storage type for this conceptual row. Conceptual rows
having the value 'permanent' need not allow write-access
to any columnar objects in the row."
DEFVAL     { nonVolatile }
 ::= { snmpTlstmCertToTSNEntry 5 }

snmpTlstmCertToTSNRowStatus OBJECT-TYPE
SYNTAX     RowStatus
MAX-ACCESS read-create
STATUS     current
DESCRIPTION
"The status of this conceptual row. This object MAY be
used to create or remove rows from this table.

To create a row in this table, an administrator MUST set
this object to either createAndGo(4) or createAndWait(5).

Until instances of all corresponding columns are
appropriately configured, the value of the corresponding
instance of the 'snmpTlstmParamsRowStatus' column is
notReady(3).

In particular, a newly created row cannot be made active
until the corresponding 'snmpTlstmCertToTSNFingerprint',
'snmpTlstmCertToTSNMapType', and 'snmpTlstmCertToTSNData'
columns have been set.

The following objects MUST NOT be modified while the
value of this object is active(1):

- snmpTlstmCertToTSNFingerprint
- snmpTlstmCertToTSNMapType
- snmpTlstmCertToTSNData

An attempt to set these objects while the value of
snmpTlstmParamsRowStatus is active(1) will result in
an inconsistentValue error."
 ::= { snmpTlstmCertToTSNEntry 6 }

-- Maps tmSecurityNames to certificates for use by the
-- SNMP-TARGET-MIB

snmpTlstmParamsCount OBJECT-TYPE
SYNTAX     Gauge32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"A count of the number of entries in the
snmpTlstmParamsTable."
 ::= { snmpTlstmCertificateMapping 4 }

snmpTlstmParamsTableLastChanged OBJECT-TYPE
SYNTAX     TimeStamp

Vaughn Standards Track Page 18
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"The value of sysUpTime.0 when the snmpTlstmParamsTable
was last modified through any means, or 0 if it has not
been modified since the command responder was started."
::= { snmpTlstmCertificateMapping 5 }

snmpTlstmParamsTable OBJECT-TYPE
SYNTAX      SEQUENCE OF SnmpTlstmParamsEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"This table is used by a (D)TLS client when a (D)TLS
connection is being set up using an entry in the
SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's
snmpTargetParamsTable with a fingerprint of a certificate
to use when establishing such a (D)TLS connection."
::= { snmpTlstmCertificateMapping 6 }

SnmpTlstmParamsEntry OBJECT-TYPE
SYNTAX      SnmpTlstmParamsEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A conceptual row containing a fingerprint hash of a
locally held certificate for a given
snmpTargetParamsEntry. The values in this row SHOULD be
ignored if the connection that needs to be established, as
indicated by the SNMP-TARGET-MIB infrastructure, is not a
certificate-based and (D)TLS-based connection. The
connection SHOULD NOT be established if the certificate
fingerprint stored in this entry does not point to a valid
locally held certificate or if it points to an unusable
certificate (such as might happen when the certificate's
expiration date has been reached)."
INDEX    { IMPLIED snmpTargetParamsName }
::= { snmpTlstmParamsTable 1 }

SnmpTlstmParamsEntry ::= SEQUENCE {
    snmpTlstmParamsClientFingerprint SnmpTLSFingerprint,
    snmpTlstmParamsStorageType       StorageType,
    snmpTlstmParamsRowStatus         RowStatus
}

snmpTlstmParamsClientFingerprint OBJECT-TYPE
SYNTAX      SnmpTLSFingerprint
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"This object stores the hash of the public portion of a
locally held X.509 certificate. The X.509 certificate,
its public key, and the corresponding private key will be
used when initiating a (D)TLS connection as a (D)TLS
client."
::= { snmpTlstmParamsEntry 1 }

SnmpTlstmParamsStorageType OBJECT-TYPE
SYNTAX StorageType
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"The storage type for this conceptual row. Conceptual rows having the value 'permanent' need not allow write-access to any columnar objects in the row."
DEFVAL { nonVolatile }
::= { snmpTlstmParamsEntry 2 }

snmpTlstmParamsRowStatus OBJECT-TYPE
SYNTAX RowStatus
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"The status of this conceptual row. This object MAY be used to create or remove rows from this table.

To create a row in this table, an administrator MUST set this object to either createAndGo(4) or createAndWait(5).

Until instances of all corresponding columns are appropriately configured, the value of the corresponding instance of the 'snmpTlstmParamsRowStatus' column is notReady(3).

In particular, a newly created row cannot be made active until the corresponding 'snmpTlstmParamsClientFingerprint' column has been set.

The snmpTlstmParamsClientFingerprint object MUST NOT be modified while the value of this object is active(1).

An attempt to set these objects while the value of snmpTlstmParamsRowStatus is active(1) will result in an inconsistentValue error."
::= { snmpTlstmParamsEntry 3 }

snmpTlstmAddrCount OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A count of the number of entries in the snmpTlstmAddrTable."
::= { snmpTlstmCertificateMapping 7 }

snmpTlstmAddrTableLastChanged OBJECT-TYPE
SYNTAX TimeStamp
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The value of sysUpTime.0 when the snmpTlstmAddrTable was last modified through any means, or 0 if it has not been modified since the command responder was started."
::= { snmpTlstmCertificateMapping 8 }

snmpTlstmAddrTable OBJECT-TYPE
SYNTAX      SEQUENCE OF SnmpTlstmAddrEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"This table is used by a (D)TLS client when a (D)TLS connection is being set up using an entry in the SNMP-TARGET-MIB. It extends the SNMP-TARGET-MIB's snmpTargetAddrTable so that the client can verify that the correct server has been reached. This verification can use either 1) a certificate fingerprint or 2) an identity authenticated via certification path validation.

If there is an active row in this table corresponding to the entry in the SNMP-TARGET-MIB that was used to establish the connection and the row's 'snmpTlstmAddrServerFingerprint' column has a non-empty value, then the server's presented certificate is compared with the snmpTlstmAddrServerFingerprint value (and the 'snmpTlstmAddrServerIdentity' column is ignored). If the fingerprint matches, the verification has succeeded. If the fingerprint does not match, then the connection MUST be closed.

If the server's presented certificate has passed certification path validation (RFC 5280) to a configured trust anchor and an active row exists with a zero-length snmpTlstmAddrServerFingerprint value, then the 'snmpTlstmAddrServerIdentity' column contains the expected host name. This expected host name is then compared against the server’s certificate as follows:

- Implementations MUST support matching the expected host name against a dNSName in the subjectAltName extension field and MAY support checking the name against the CommonName portion of the subject distinguished name.

- The '*' (ASCII 0x2A) wildcard character is allowed in the dNSName of the subjectAltName extension (and in CommonName, if used to store the host name), but only as the leftmost (least significant) DNS label in that value. This wildcard matches any leftmost DNS label in the server name. That is, the subject *.example.com matches the server names a.example.com and b.example.com but does not match example.com or a.b.example.com. Implementations MUST support wildcards in certificates as specified above but MAY provide a configuration option to disable them.

- If the locally configured name is an internationalized domain name, conforming implementations MUST convert it to the ASCII Compatible Encoding (ACE) format for performing comparisons, as specified in Section 7 of RFC 5280.

If the expected host name fails these conditions, then the connection MUST be closed.
If there is no row in this table corresponding to the entry in the SNMP-TARGET-MIB and the server can be authorized by another, implementation-dependent means, then the connection MAY still proceed.

```snmpTlstmCertificateMapping 9
::= { snmpTlstmAddrTable 1 }

SnmpTlstmAddrEntry ::= SEQUENCE {
  snmpTlstmAddrServerFingerprint    SnmpTLSFingerprint,
  snmpTlstmAddrServerIdentity       SnmpAdminString,
  snmpTlstmAddrStorageType          StorageType,
  snmpTlstmAddrRowStatus            RowStatus
}

snmpTlstmAddrServerFingerprint OBJECT-TYPE
SYNTAX      SnmpTLSFingerprint
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
  "A cryptographic hash of a public X.509 certificate. This object should store the hash of the public X.509 certificate that the remote server should present during the (D)TLS connection setup. The fingerprint of the presented certificate and this hash value MUST match exactly, or the connection MUST NOT be established."
DEFVAL { "" }
::= { snmpTlstmAddrEntry 1 }

snmpTlstmAddrServerIdentity OBJECT-TYPE
SYNTAX      SnmpAdminString
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
  "The reference identity to check against the identity presented by the remote system."
DEFVAL { "" }
::= { snmpTlstmAddrEntry 2 }

snmpTlstmAddrStorageType OBJECT-TYPE
SYNTAX       StorageType
MAX-ACCESS   read-create
STATUS       current
DESCRIPTION
"The storage type for this conceptual row. Conceptual rows
having the value 'permanent' need not allow write-access
to any columnar objects in the row."
DEFVAL      { nonVolatile }
::= { snmpTlstmAddrEntry 3 }

snmpTlstmAddrRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The status of this conceptual row. This object may be
used to create or remove rows from this table.

To create a row in this table, an administrator MUST set
this object to either createAndGo(4) or createAndWait(5).

Until instances of all corresponding columns are
appropriately configured, the value of the corresponding
instance of the 'snmpTlstmAddrRowStatus' column is
notReady(3).

In particular, a newly created row cannot be made active
until the corresponding 'snmpTlstmAddrServerFingerprint'
column has been set.

Rows MUST NOT be active if the
'snmpTlstmAddrServerFingerprint' column is blank and the
snmpTlstmAddrServerIdentity is set to '*', since this
would insecurely accept any presented certificate.

The snmpTlstmAddrServerFingerprint object MUST NOT be
modified while the value of this object is active(1).

An attempt to set these objects while the value of
snmpTlstmAddrRowStatus is active(1) will result in
an inconsistentValue error."
::= { snmpTlstmAddrEntry 4 }

-- ************************************************
--  snmpTlstmNotifications - Notifications Information
-- ************************************************

snmpTlstmServerCertificateUnknown NOTIFICATION-TYPE
OBJECTS { snmpTlstmSessionUnknownServerCertificate }
STATUS  current
DESCRIPTION
"Notification that the server certificate presented by an
SNMP over (D)TLS server was invalid because no configured
fingerprint or CA was acceptable to validate it. This may
be because there was no entry in the snmpTlstmAddrTable or
because no path to a known CA could be found.

To avoid notification loops, this notification MUST NOT be
sent to servers that themselves have triggered the notification."

::= { snmpTlstmNotifications 1 }

snmpTlstmServerInvalidCertificate NOTIFICATION-TYPE
OBJECTS {
  snmpTlstmAddrServerFingerprint,
  snmpTlstmSessionInvalidServerCertificates
} STATUS current
DESCRIPTION
"Notification that the server certificate presented by an
SNMP over (D)TLS server could not be validated even if the
fingerprint or expected validation path was known. That is, a
cryptographic validation error occurred during
certificate validation processing.

To avoid notification loops, this notification MUST NOT be
sent to servers that themselves have triggered the
notification."

::= { snmpTlstmNotifications 2 }

-- ************************************************
-- snmpTlstmCompliances - Conformance Information
-- ************************************************

snmpTlstmCompliances OBJECT IDENTIFIER ::= { snmpTlstmConformance 1 }

snmpTlstmGroups OBJECT IDENTIFIER ::= { snmpTlstmConformance 2 }

-- ************************************************
-- Compliance statements
-- ************************************************

snmpTlstmCompliance MODULE-COMPLIANCE
STATUS current
DESCRIPTION
"The compliance statement for SNMP engines that support the
SNMP-TLS-TM-MIB."
MODULE
  MANDATORY-GROUPS { snmpTlstmStatsGroup,
    snmpTlstmIncomingGroup,
    snmpTlstmOutgoingGroup,
    snmpTlstmNotificationGroup }

::= { snmpTlstmCompliances 1 }

-- ************************************************
-- Units of conformance
-- ************************************************

snmpTlstmStatsGroup OBJECT-GROUP
OBJECTS {
  snmpTlstmSessionOpens,
  snmpTlstmSessionClientCloses,
  snmpTlstmSessionOpenErrors,
  snmpTlstmSessionAccepts,
  snmpTlstmSessionServerCloses,
  snmpTlstmSessionNoSessions,
snmpTlstmSessionInvalidClientCertificates,
snmpTlstmSessionUnknownServerCertificate,
snmpTlstmSessionInvalidServerCertificates,
snmpTlstmSessionInvalidCaches
}

status current
description
"A collection of objects for maintaining statistical
information of an SNMP engine that implements the SNMP
TLSTM."
::= { snmpTlstmGroups 1 }

snmpTlstmIncomingGroup OBJECT-GROUP
OBJECTS {
    snmpTlstmCertToTSNCount,
    snmpTlstmCertToTSNTableLastChanged,
    snmpTlstmCertToTSNFingerprint,
    snmpTlstmCertToTSNMapType,
    snmpTlstmCertToTSNData,
    snmpTlstmCertToTSNStorageType,
    snmpTlstmCertToTSNRowStatus
}

status current
description
"A collection of objects for maintaining incoming
connection certificate mappings to tmSecurityNames of an
SNMP engine that implements the SNMP TLSTM."
::= { snmpTlstmGroups 2 }

snmpTlstmOutgoingGroup OBJECT-GROUP
OBJECTS {
    snmpTlstmParamsCount,
    snmpTlstmParamsTableLastChanged,
    snmpTlstmParamsClientFingerprint,
    snmpTlstmParamsStorageType,
    snmpTlstmParamsRowStatus,
    snmpTlstmAddrCount,
    snmpTlstmAddrTableLastChanged,
    snmpTlstmAddrServerFingerprint,
    snmpTlstmAddrServerIdentity,
    snmpTlstmAddrStorageType,
    snmpTlstmAddrRowStatus
}

status current
description
"A collection of objects for maintaining outgoing
connection certificates to use when opening connections as
a result of SNMP-TARGET-MIB settings."
::= { snmpTlstmGroups 3 }

snmpTlstmNotificationGroup NOTIFICATION-GROUP
NOTIFICATIONS {
    snmpTlstmServerCertificateUnknown,
    snmpTlstmServerInvalidCertificate
}

status current
description
"Notifications."
5. Security Considerations

This document updates a transport model that permits SNMP to utilize (D)TLS security services. The security threats and how the TLSTM mitigates these threats are covered throughout this document and in [RFC6353]. Security considerations for TLS are described in Section 10 and Appendix E of TLS 1.3 [RFC8446]. Security considerations for DTLS are described in Section 11 of DTLS 1.3 [RFC9147].

Implementations should consider the latest recommendations on the use of (DTLS), such as those documented in [RFC9325].

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example, by using IPsec), there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.

It is **RECOMMENDED** that only SNMPv3 messages using the Transport Security Model (TSM) or another secure-transport-aware security model be sent over the TLSTM transport.

6. IANA Considerations

IANA has created a new registry called "SNMP-TLSTM HashAlgorithms" within the "Structure of Management Information (SMI) Numbers (MIB Module Registrations)" group. The description of this registry is "iso.org.dod.internet.mgmt.mib-2.snmpTlstmMIB.snmpTlstmHashAlgorithms (1.3.6.1.2.1.198.4)".

The registry has the following fields: Value, Description, Recommended, and References. The range of values is zero to 255, with initial assignments shown in Section 2.1. The "Recommended" column indicates "Y" for hashing algorithms that are Standards Track and are deemed to be acceptable for widely applicable current use and "N" for hashing algorithms that reflect meanings that are not recommended (e.g., they do not provide sufficient security for modern systems, they are not Standards Track, and they have limited applicability). A blank field indicates that no recommendation is made (e.g., because the value is unassigned or left for private use).

This registry is expected to be updated infrequently; as such, its values are limited to one octet.

The policy for updates to the "SNMP-TLSTM HashAlgorithms" registry is Expert Review [RFC8126]. Registry requests should be sent to the <mailto:snmp-tlstm-reg-review@ietf.org> mailing list. Registration requests sent to the mailing list for review **SHOULD** use an appropriate
subject (e.g., 'Request to register value in "SNMP-TLSTM HashAlgorithms" registry'). In addition, designated experts should consult with the <mailto:tls-reg-review@ietf.org> mailing list to make sure that any new hash algorithms are considered for inclusion in this registry.

Designated experts SHOULD ascertain the existence of suitable documentation that defines a hash algorithm and SHOULD also verify that the request does not conflict with or duplicate other entries in the registry. The experts should also provide a recommendation as to how the "Recommended" column of the registry should be updated. Only publicly available specifications that represent current industry-accepted practices should receive an assignment of "Y" in the "Recommended" column; all other specific assignments in the registry should receive an assignment of "N". Assignments that are nonspecific (e.g., reserved values) SHOULD NOT receive an assigned value for the "Recommended" column.

Within the three-week review period, the designated experts will either approve or deny the registration request, communicating this decision to the review list and IANA. Denials SHOULD include an explanation and, if applicable, suggestions as to how to make the request successful. Registration requests that are undetermined for a period longer than three weeks can be brought to the IESG’s attention (using the <mailto:iesg@ietf.org> mailing list) for resolution.

IANA MUST only accept registry updates from the designated experts and SHOULD direct all requests for registration to the review mailing list. While future additions to the "TLS HashAlgorithm" registry (i.e., the registry from which the "SNMP-TLSTM HashAlgorithms" registry was spawned) are not expected, any future additions to the "TLS HashAlgorithm" registry MUST be consistent with the values assigned in the "SNMP-TLSTM HashAlgorithms" registry.

It is suggested that multiple designated experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly informed reviews of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular expert, that expert SHOULD defer to the judgment of the other experts.

7. References

7.1. Normative References


<https://www.rfc-editor.org/info/std62>

7.2. Informative References


Acknowledgements

This document is based on [RFC6353]. This document was reviewed by the following people, who helped provide useful comments: Michaela Vanderveen, Joe Clarke, Jürgen Schönwälder, and Tom Petch.

Author's Address

Kenneth Vaughn (EDITOR)
Trevilon LLC
1060 Highway 107 South
Del Rio, TN 37727
United States of America
Phone: +1 571 331 5670
Email: kvaughn@trevilon.com