# ETSI TS 103 523-3 V1.2.1 (2019-03)

TECHNICAL SPECIFICATION

**CYBER;**
**Middlebox Security Protocol;**
**Part 3: Enterprise Transport Security**

Reference

RTS/CYBER-0027-3v121

Keywords

cyber security

*ETSI*

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

# Contents

# Intellectual Property Rights

### Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

### Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

# Foreword

This Technical Specification (TS) has been produced by ETSI Technical Committee Cyber Security (CYBER).

The present document is part 3 of a multi-part deliverable. Full details of the entire series can be found in part 1 [i.1].

# Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

# Executive summary

Requirements - such as legal mandates and service agreements - exist for enterprise network and data centre operators and service providers, organizations, and small businesses to be able to observe and audit the content and metadata of encrypted sessions transported across their infrastructures [i.2]. The original TLS protocol standards adopted in the 1986-1995 period in multiple bodies and IETF versions up to and including TLS 1.2, provided for these capabilities [i.3] and [1]. The latest version of the protocol, TLS 1.3, does not provide for these capabilities [2]. Where these capabilities do not exist, this new encryption protocol could be blocked altogether at the enterprise gateway, forcing users to revert to older, less secure TLS protocols.

The present document is one of a series of MSP implementation profiles that support these capabilities, while allowing enterprise operators and users to stay in control of access to their data. It sets forth an MSP profile, "Enterprise Transport Security", for use in enterprise networks and data centres that meets mandatory capabilities for the Middlebox Security Protocol (MSP) [i.1].

# Introduction

The present document specifies an MSP profile for enterprise network and data centre domains: Enterprise Transport Security. This is an implementation variant of Transport Layer Security (TLS) protocol version 1.3 [2].

TLS 1.3 is a recent version in a series of TLS protocol standards [i.6] and [i.3]. TLS 1.3 [2] introduces several significant changes compared with TLS 1.2 [1]. One of these changes is the removal of support for RSA key exchange and static Diffie-Hellman key exchange. The primary key exchange mechanism in TLS 1.3 is ephemeral Diffie-Hellman. Ephemeral Diffie-Hellman prevents passive decryption of TLS 1.3 sessions at any scale. However, there are operational circumstances where passive decryption of TLS sessions by authorized entities is a requirement. The decryption may need to be performed in real-time, or the packets may need to be stored and decrypted post-capture.

Situations requiring passive decryption of TLS sessions generally occur in environments where both the client and server, and by inference the data being exchanged over the TLS session, are under the control of the same entity. TLS encryption is often stipulated by internal or external security policies, but access to the unencrypted packet data is required for operational reasons, including:

- Application health monitoring and troubleshooting.
- Intrusion detection.
- Detection of malware activity, e.g. lateral movement, command and control, and data exfiltration traffic.
- Detection of advanced Distributed Denial Of Service (DDOS) attacks.
- Compliance audits.

One possible approach to passively decrypting TLS 1.3 sessions is to export the ephemeral keys generated for each TLS session to middleboxes. However, this approach has several significant limitations. Firstly, it is very difficult to ensure that the exported ephemeral keys will arrive at the middlebox in sufficient time to allow decryption in real-time. Secondly, the keys need to be correlated with every stored packet session in anticipation of post-capture decryption. For these reasons, this approach does not scale to the needs of a data centre.

The Enterprise Transport Security profile therefore uses longer-lived static Diffie-Hellman keys that are re-used across multiple sessions; enterprises could implement automated key rotation in order to reduce the rotation cycle time. This ensures that the keys can be distributed to real-time decryption middleboxes in advance, and it greatly reduces the number of keys to be stored and correlated with packet storage systems.

The Enterprise Transport Security profile also requires the server to report visibility information in its certificate, to indicate to the client that Transport Layer Security is in use with a particular static Diffie-Hellman public key, and to describe the set of entities or roles or domains, or any combination of these, for which the policy of the party signing the certificate allows sharing of the corresponding private key.

There are circumstances in which visibility information is not suitable and in which the client operator has been informed by other means that connections can be inspected; in such circumstances, annex A can be used. annex A, which is optional, specifies a variant of the Enterprise Transport Security profile where the visibility information is not sent.

EXAMPLE: Annex A can be used when the client and server are wholly within a private enterprise network and the client operator has already been notified by alternative means, such as a condition of access to the network, that connections can be inspected.

The Enterprise Transport Security profile is compatible with any TLS 1.3 compliant client. Annex B, which is optional, defines the concept of an "Enterprise Transport Security aware client" whereby a TLS 1.3 client provides additional capabilities in relation to Enterprise Transport Security visibility.

# 1 Scope

The present document specifies the "Enterprise Transport Security" profile to enable secure communication sessions between network endpoints whilst enabling network operations. The Enterprise Transport Security (ETS) profile enables use of Transport Layer Security (TLS) version 1.3 [2] in, for example, compliance constrained environments.

The present document describes three Enterprise Transport Security architectures:

- In the first architecture, both the TLS 1.3 client and the Enterprise Transport Security server are located inside the enterprise.
- In the second architecture, the server is an Enterprise Transport Security server inside the enterprise and the TLS 1.3 client is external to the enterprise. TLS 1.3 is terminated at the enterprise edge such that Enterprise Transport Security is used only inside the enterprise.
- In the third architecture, the TLS 1.3 server is external to the enterprise and the TLS 1.3 client is internal to the enterprise. TLS 1.3 is again terminated at the network edge such that Enterprise Transport Security is used only inside the enterprise.

The Diffie-Hellman key exchange and visibility information for indicating the Enterprise Transport Security profile setup is specified.

The actions of the client on receiving the visibility information and structure of the policy included in the visibility information are not normatively defined; however, capabilities for an "Enterprise Transport Security aware client" are defined in annex B, which is optional. The means by which the Enterprise Transport Security endpoints share the Diffie-Hellman key with key consumers is specified, and examples are provided.

The present document describes a variant of the Enterprise Transport Security profile in annex A for circumstances in which visibility information is not suitable and in which the client operator has been informed by other means that connections can be inspected. The means by which the client operator is informed is out of scope.

The present document also includes the security assurances made by the Enterprise Transport Security profile, based on the security assurances of TLS 1.3. Annex C gives details of the MSP profile capabilities that are applicable to the Enterprise Transport Security profile, taken from the draft specification of ETSI TS 103 523-1 [i.1], such that this MSP Part may be a standalone document. A final mapping of MSP profile capabilities to the Enterprise Transport Security profile is left to a future version of the present document.

# 2 References

## 2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at https://docbox.etsi.org/Reference/.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2".

[2] IETF RFC 8446: "The Transport Layer Security (TLS) Protocol Version 1.3".

[3] IETF RFC 5958: "Asymmetric Key Packages".

[4] IETF RFC 7906: "NSA's Cryptographic Message Syntax (CMS) Key Management Attributes".

[5] IETF RFC 3279: "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[6] IETF RFC 5480: "Elliptic Curve Cryptography Subject Public Key Information".

[7] IETF RFC 5915: "Elliptic Curve Private Key Structure".

[8] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[9] Recommendation ITU-T X.509 (10/2016) | ISO/IEC 9594-8: "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks".

[10] IETF RFC 2818: "HTTP Over TLS".

[11] FIPS 180-4: "Secure Hash Standard (SHS)".

[12] IETF RFC 7231: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content".


## 2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.


[i.1] ETSI TS 103 523-1: "CYBER; Middlebox Security Protocol; Part 1: Capability Requirements".

[i.2] S. Fenter: "Why Enterprises Need Out-of-Band TLS Decryption", IETF, 2018.

[i.3] Recommendation ITU-T X.274 (1994) | ISO/IEC 10736:1995: "Transport Layer Security Protocol".

[i.4] IETF RFC 5652: "Cryptographic Message Syntax (CMS)".

[i.5] IETF RFC 5083: "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type".

[i.6] Nelson & Heimann: "SDNS Architecture and End-to-End Encryption", CRYPTO' 89 Proceedings, pp 356-366; Ruth Nelson, SDNS Services and Architecture, Proceedings of the 10th National Computer Security Conference, Sept. 1987, pp 153-157.


# 3 Definition of terms, symbols and abbreviations

## 3.1 Terms

For the purposes of the present document, the following terms apply:

**1-sided:** middlebox traffic observability enabled unilaterally by one endpoint such that the other endpoint is not able to reject or negotiate the traffic observability, other than by ceasing the communication

**Enterprise Transport Security (ETS):** MSP profile described in the present document that instantiates an Enterprise Transport Security session

**single-context:** access is granted, or not granted, only to the entire data stream, not to portions of the data stream

## 3.2 Symbols

Void.

## 3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

| | |
|---|---|
| ASN | Abstract Syntax Notation |
| CMS | Cryptographic Message Syntax |
| DDOS | Distributed Denial Of Service |
| DER | Distinguished Encoding Rules |
| ETS | Enterprise Transport Security |
| FIPS | Federal Information Processing Standards |
| HSM | Hardware Security Module |
| HTTP | HyperText Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| MSP | Middlebox Security Protocol |
| RSA | Rivest-Shamir-Adleman |
| TLS | Transport Layer Security |

# 4 Enterprise Transport Security for the MSP framework

## 4.1 MSP requirements mapping

MSP Part 1 [i.1] defines several Capability Requirements that are demanded of a profile wishing to comply with the MSP framework. The full and complete mapping of MSP Part 1 [i.1] requirements to the Enterprise Transport Security profile described in the current document is left to a future revision of the present document when MSP Part 1 [i.1] is finalized. However, desired capabilities of MSP profiles are mapped to relevant properties of the Enterprise Transport Security profile in annex C.

For any mapping, an MSP profile needs categorizing as a 1-sided or 2-sided profile with single or fine-grained context, as defined in the planned MSP Part 1 [i.1]. This categorization determines the mandatory and optional requirements that the MSP profile needs to satisfy.

Enterprise Transport Security is a 1-sided MSP profile, as only one endpoint will be using a static Diffie-Hellman key, and so that endpoint unilaterally enables traffic observability. The Enterprise Transport Security profile is also single-context, as access is granted, or not granted, only to the entire data stream.

## 4.2 Enterprise Transport Security architectures

### 4.2.1 Enterprise Transport Security with enterprise clients and servers

Figure 4.1 depicts the Enterprise Transport Security implementation architecture when both the TLS 1.3 [2] client and the Enterprise Transport Security server are located in the Enterprise.

Middlebox A is authorized to inspect the traffic flowing between the firewall and the web server. It therefore receives a passive copy of these packets along with a copy of the static Diffie-Hellman public/private key pair (A) used by the web server.

EXAMPLE 1: Middlebox A decrypts the traffic in real-time to perform intrusion detection.

The web server acts as a TLS client in its connection to the application server. In this case, Middlebox B is authorized to inspect the traffic flowing between the two servers, and it decrypts the TLS sessions using the application server's static Diffie-Hellman public/private key pair (B).

EXAMPLE 2: Middlebox B decrypts the traffic in real-time to provide application health monitoring, but also stores the encrypted packets so they can be decrypted at a later date for compliance and auditing purposes.

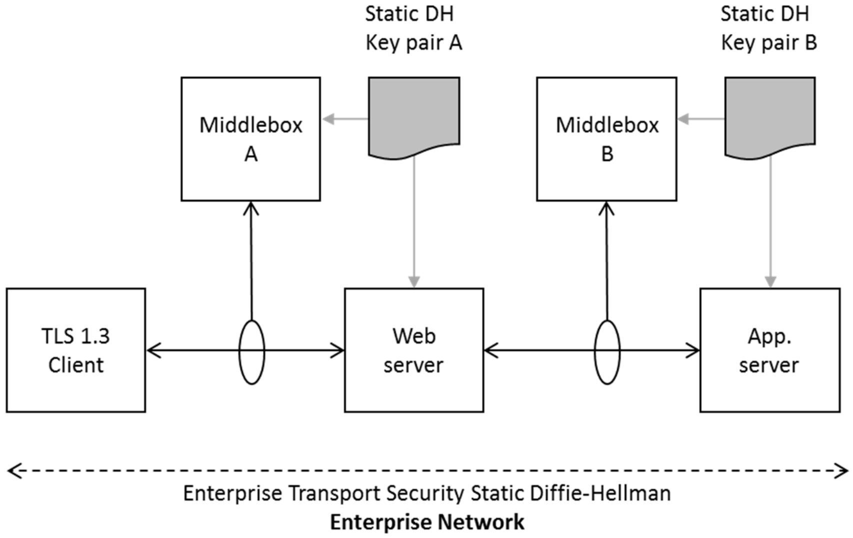



**Figure 4.1: Enterprise Transport Security architecture with enterprise client and server**

## 4.2.2 Enterprise Transport Security with enterprise servers

Figure 4.2 depicts the Enterprise Transport Security implementation architecture when the TLS 1.3 client is outside the enterprise and the Enterprise Transport Security server is located in the Enterprise. TLS connections to clients that are external to an enterprise network or data centre may be made using TLS 1.3 [2], using forward secrecy.

The firewall terminates the Internet TLS 1.3 sessions and uses the Enterprise Transport Security profile between the firewall and the web server, with the firewall acting as the TLS client. Middlebox A is authorized to inspect the traffic flowing between the firewall and the web server. It therefore receives a passive copy of these packets along with a copy of the static Diffie-Hellman public/private key pair (A) used by the web server.

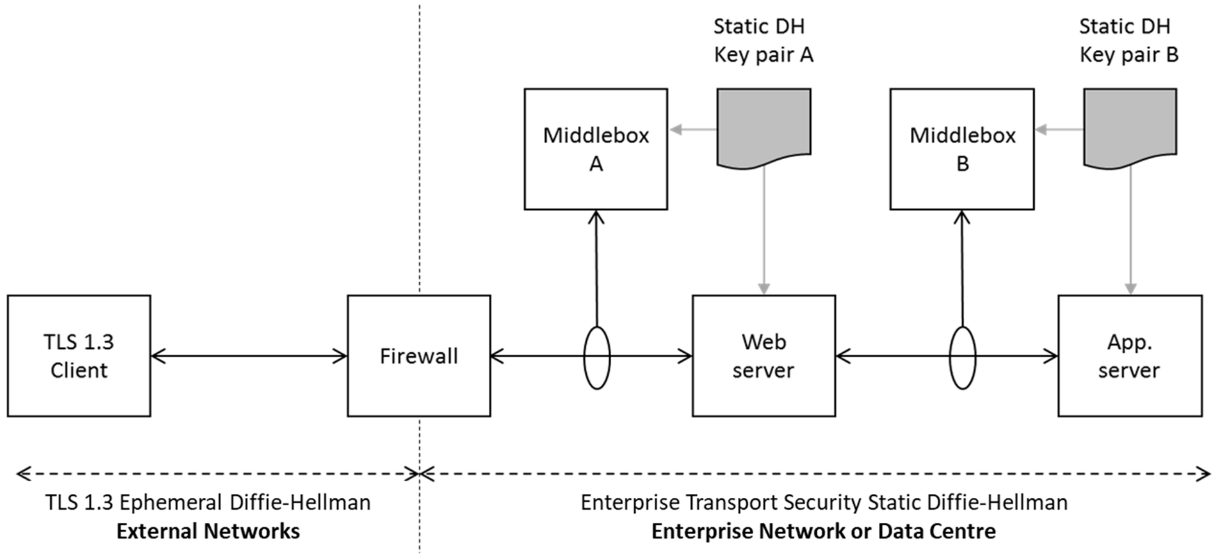



**Figure 4.2: Enterprise Transport Security architecture with enterprise servers**

### 4.2.3 Enterprise Transport Security with enterprise clients

Figure 4.3 depicts the Enterprise Transport Security implementation architecture when enterprise clients are used with a TLS 1.3 server outside the enterprise. TLS connections to servers that are external to an enterprise network may be made using TLS 1.3 [2], using forward secrecy.

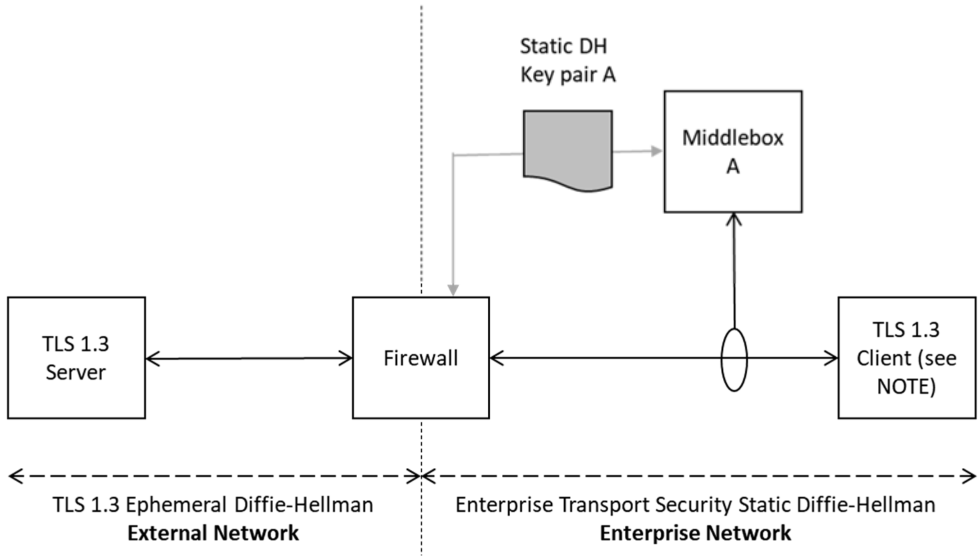



**Figure 4.3: Enterprise Transport Security architecture with enterprise clients**

The firewall terminates the enterprise network sessions that are using Enterprise Transport Security, with the firewall acting as the Enterprise Transport Security server. The firewall then acts as the TLS 1.3 client and uses TLS 1.3 to communicate to the TLS 1.3 server on the external network. Middlebox A is authorized to inspect the traffic flowing between the client and the firewall. It therefore receives a passive copy of these packets along with a copy of the static Diffie-Hellman public/private key pair (A) used by the firewall, in its role as the Enterprise Transport Security server.

NOTE: Although the client is participating in an Enterprise Transport Security connection, it is a TLS 1.3 compliant client.

EXAMPLE: Middlebox A decrypts the traffic in real-time to perform malware detection or data loss prevention.

## 4.3 The Enterprise Transport Security profile

### 4.3.1 Normal TLS 1.3 Diffie-Hellman key exchange

For reference, a description of the normal TLS 1.3 Diffie-Hellman key exchange is included. Unless a pre-shared key is in use, the TLS 1.3 key exchange mechanism proceeds at the start of a new session as follows [2]:

1) The client generates an ephemeral Diffie-Hellman public and private key. The public key is transmitted to the server in a "key_share" message with a random client nonce.

2) The server generates an ephemeral Diffie-Hellman public and private key. The public key is transmitted to the client in a "key_share" message with a random server nonce.

3) The client and server each use a combination of their own private keys and the public key received from the other side of the connection to generate a shared secret.

4) The client and server then use the shared secret along with the initial handshake messages, which include the nonce, to generate a set of handshake traffic keys for encryption of the remainder of the handshake.

5) During the remainder of the handshake, the server sends its certificate encrypted using a handshake traffic key.

6) Upon completion of the handshake, the client and server then use the shared secret along with various elements of the handshake messages, which again include the nonce, to generate a set of application traffic keys for the session.

7) The application traffic keys are used to encrypt further data exchanged between the client and server.

### 4.3.2 Enterprise Transport Security Diffie-Hellman key exchange

The Enterprise Transport Security key exchange shall use exactly the same messages and procedures to establish a set of session keys as a TLS 1.3 ephemeral Diffie-Hellman key exchange, except for two differences [2]:

1) the server shall use a static public/private key pair at Step 2 in clause 4.3.1; and

2) the server's certificate at Step 5 shall contain visibility information as defined in clause 4.3.3 to indicate to the client that Enterprise Transport Security is in use.

NOTE: Neither the static public key nor the visibility information affects the operation of a TLS 1.3 compliant client, so an Enterprise Transport Security server is therefore fully interoperable with TLS 1.3 compliant clients.

The Enterprise Transport Security server shall be provisioned with a static key pair for each elliptic curve (or finite field length) supported by the server. These key pairs may be shared with middleboxes that are authorized to decrypt sessions from the server as shown in Figure 4.2.

The static key pair shall be:

1) installed directly on the server, or generated on an associated hardware security module (HSM), and rotated as necessary, described in clause 4.3.4; or

2) downloaded from a central key manager and updated as necessary, described in clause 4.3.5.

### 4.3.3 Visibility information

With the Enterprise Transport Security profile, the server certificate, as defined in Recommendation ITU-T X.509 [9], shall send visibility information in its certificate that shall:

1) Indicate that Enterprise Transport Security is being used.

2) Be bound to the static Diffie-Hellman public/private key pair for Step 2 of the Enterprise Transport Security profile, described in clause 4.3.2. This public/private key pair shall not be the same as the certificate subject's public/private key pair (defined in clause 4.1.2.7 Subject Public Key Info of IETF RFC 5280 [8]), which is used for the signature in the CertificateVerify message (defined in clause 4.4.3 Certificate Verify of IETF RFC 8446 [2]).

3) Identify, either generally or specifically, the controlling or authorizing entities or roles or domains, or any combination of these, of any middleboxes that may be allowed access to the Enterprise Transport Security static Diffie-Hellman private key described in clause 4.3.2 of the present document.

The above 3 points describe the Visibility Information field, which shall be defined by the following ASN.1 type:

```
VisibilityInformation ::= SEQUENCE {
     fingerprint         OCTET STRING (SIZE(10)),
     accessDescription   UTF8String }
```

where the SHA-256 digest of the static Diffie-Hellman public key as transmitted in the key_share extension of the ServerHello message shall be represented as the vector of 32-bit words ($H_0$, $H_1$,…, $H_7$) as defined in FIPS 180-4 [11]. The `fingerprint` field shall be set to $H_0||H_1||(H_2>>16)$, which is the first 80 bits of the digest vector read in big-endian format. The `accessDescription` field shall be a human-readable text string that identifies, either generally or specifically, the controlling or authorizing entities or roles or domains, or any combination of these, of any middleboxes that may be allowed access to the Enterprise Transport Security static Diffie-Hellman private key.

The Visibility Information field shall be sent as an `otherName` field entry of the `subjectAltName` as defined in clause 4.2.1.6 Subject Alternative Name of IETF RFC 5280 [8]. The `type-id` shall be set to the Object Identifier 0.4.0.3523.3.1 {itu-t(0) identified-organization(4) etsi(0) msp(3523) ets(3) visibility(1)} and the contents of `value` shall be set to the DER encoding of `VisibilityInformation`.

Thus `fingerprint`, in conjunction with the certificate's validity period, binds the certificate to a particular static Diffie-Hellman public/private key pair. The `accessDescription` allows the endpoints to identify, either generally or specifically, the controlling or authorizing entities or roles or domains, or any combination of these, of any middleboxes that may be given the ability to decrypt data protected by the corresponding Enterprise Transport Security key exchange.

The `accessDescription` field shall be accurate; a structure for the description may be introduced in a future version of the Enterprise Transport Security profile.

The server may bind multiple static Diffie-Hellman public/private key pairs to a single certificate by including a `subjectAltName` containing `VisibilityInformation` for each key pair.

NOTE: The ITU-T jointly with ISO/IEC JTC1 may standardize an extension of Recommendation ITU-T X.509 | ISO/IEC 9594-8 [9] which is suitable for `VisibilityInformation`, in which case this clause will be updated to additionally support this new extension.

The recommended actions of the client on receiving the visibility information can be found in annex B, which is optional.

An optional variant of the Enterprise Transport Security profile, where `VisibilityInformation` is not sent, is described in annex A. This variant may be used when visibility information is not suitable. This variant shall not be used unless the client operator has been informed by some means that the connection can be inspected. The means by which the client operator is informed is out of scope for the present document.

EXAMPLE: A client operator is informed through an acceptable use policy for a client on a private enterprise network.

If operation according to annex A is supported, it shall be explicitly enabled; otherwise, the server shall not establish Enterprise Transport Security sessions using certificates that do not include the visibility information defined in the present clause.

### 4.3.4 Directly installed keys

The means by which an Enterprise Transport Security implementation shares the static Diffie-Hellman public/private key pair of clause 4.3.3 between an Enterprise Transport Security server and a middlebox is not specified, however the current clause and subsequent clauses 4.3.5 to 4.3.8 provide possible means without being exhaustive. The current clause describes perhaps the simplest means, which is that the static Diffie-Hellman public/private key pair is directly installed in both the Enterprise Transport Security server and permitted middleboxes.

### 4.3.5 Centrally managed keys

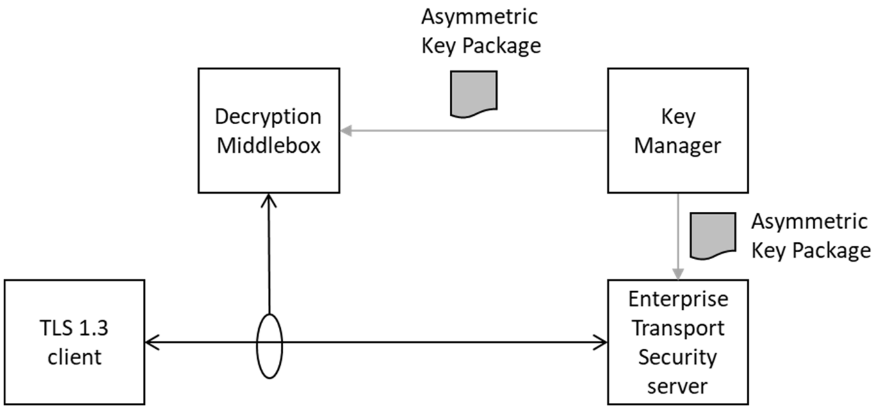
NOTE: In the scenario where a key manager generates the static Diffie-Hellman public/private key pair for installation on both the server and any decryption appliances, it would be natural for the key manager to generate the server certificate at the same time for each key pair.

Figure 4.4 shows the basic architecture for using a central key manager to deploy the Enterprise Transport Security static Diffie-Hellman key pairs to a key consumer.

EXAMPLE 1: An Enterprise Transport Security server is a key consumer.

EXAMPLE 2: A passive decryption middlebox is a key consumer.

The key manager generates the key pair. The key manager can actively push keys to the key consumer or the key consumer can request keys. The key pair package is specified in clause 4.3.6. Protection of the key package is specified in clause 4.3.7 and the transfer mechanism is specified in clause 4.3.8.




NOTE: In the scenario where a key manager generates the static Diffie-Hellman public/private key pair for installation on both the server and any decryption appliances, it would be natural for the key manager to generate the server certificate at the same time for each key pair.

**Figure 4.4: Architecture for centrally managed static Diffie-Hellman key pairs**

### 4.3.6 Asymmetric key package

When an Enterprise Transport Security static Diffie-Hellman public/private key pair are sent from the key manager to a key consumer, they shall be packaged using the Asymmetric Key Package defined in IETF RFC 5958 [3]. Each Asymmetric Key Package shall contain one or more OneAsymmetricKey elements. Such an element will be one of either:

a) a static Diffie-Hellman key pair, hereafter referred to as Type A elements; or

b) a private signing key and a certificate, hereafter referred to as Type B elements.

First the case is defined where elements are static Diffie-Hellman key pairs, and so the Asymmetric Key Package shall contain fields and attributes pertaining to these key pairs, defined below. Though certificates are not sent in the same OneAsymmetricKey element as a static key pair, each Asymmetric Key Package may contain one or more Type B elements (server certificates and corresponding private signing keys). Where such Type B elements are sent, all certificates in the Asymmetric Key Package shall be bound to all of the static Diffie-Hellman key pairs in the Asymmetric Key Package. The use of multiple certificates is intended for the situation where it is necessary to provide certificates with different signature algorithms.

With reference to clause 2 of IETF RFC 5958 [3], the Type A OneAsymmetricKey element used to store each key pair in the Asymmetric Key Package shall have the following fields set as follows:

1) Version shall be set to version 2 (integer value of 1).

2) privateKeyAlgorithm shall be set to the key pair algorithm identifier (see below).

3) privateKey shall be set to the Diffie-Hellman private key encoded as an octet string.

4) publicKey shall be set to the Diffie-Hellman public key encoded as a bit string.

5) Attributes shall include a validity period for the key pair using the attribute defined in clause 15 of IETF RFC 7906 [4].

The algorithm identifier used in the privateKeyAlgorithm fields shall be as defined in clause 2.3.3 of IETF RFC 3279 [5] for Finite Field Diffie-Hellman keys, using IETF RFC 5480 [6] to replace the information in IETF RFC 3279 [5] that relates to Elliptic Curve Diffie-Hellman algorithm identifiers. Clause 3 of IETF RFC 5915 [7] provides information for encoding the ECPrivateKey field in Elliptic Curve algorithm identifiers.

EXAMPLE 1: Finite Field Diffie-Hellman (defined in clause 2.3.3 of IETF RFC 3279 [5]):

object identifier: { 1 2 840 10046 2 1 }

parameter encoding: DomainParameters

private key encoding: INTEGER

public key encoding: INTEGER

EXAMPLE 2: Elliptic Curve Diffie-Hellman (defined as the "restricted" algorithm in clause 2.1.2 of IETF RFC 5480 [6]):

object identifier: { 1 3 132 1 12 }

parameter encoding: ECParameters

private key encoding: ECPrivateKey

public key encoding: ECPoint

When the package contains one or more server certificates and corresponding private signing keys, the certificates shall include a `subjectAltName` containing `VisibilityInformation`, as defined in clause 4.3.3, for each static Diffie-Hellman key pair in the Asymmetric Key Package. The OneAsymmetricKey elements of Type B in the Asymmetric Key Package shall be the last elements in the package. The Type B OneAsymmetricKey elements shall have the following fields set as follows:

1) Version shall be set to version 1 (integer value of 0).
2) PrivateKeyAlgorithm shall be set to the same value that appears in the algorithm field of the signatureAlgorithm field of the enclosed X.509 Certificate, as defined in clause 4.1.1.2 of IETF RFC 5280 [8].
3) PrivateKey shall be set as defined in clause 2 of IETF RFC 5958 [3].
4) PublicKey shall not be present.
5) Attributes shall include a userCertificate attribute, as defined in clause 8 of IETF RFC 7906 [4], containing the server certificate.

## 4.3.7 Protecting the key package

The Cryptographic Message Syntax (CMS) defined in IETF RFC 5652 [i.4] and IETF RFC 5083 [i.5] may be used to provide authentication, integrity and/or confidentiality to the Asymmetric Key Package transported between the key manager and the key consumer.

## 4.3.8 Transferring keys

### 4.3.8.1 Protocol overview

Enterprise Transport Security Asymmetric Key Packages shall be transferred between the key manager and key consumer using HTTPS [10]. The HTTP messages containing key packages shall comprise a suitable HTTP header followed by the Asymmetric Key Package encoded using Distinguished Encoding Rules (DER) in binary format. The HTTP Content-Type header shall be set to application/pkcs8 for plaintext packages and set to application/cms if the package is encapsulated using CMS.

### 4.3.8.2 Transfer initiated by the key manager

Key consumers may support key transfers initiated by the key manager via an HTTPS key installation service, whereby the key manager initiates an HTTPS connection to the key consumer, which acts as an HTTP server.

When key transfers initiated by the key manager via an HTTPS key installation service are supported by the key consumer, the key consumer shall support receiving a key package via an HTTP PUT request to a request-target, given here in origin-form, of `/ets/keys`. When key transfers initiated by the key manager via an HTTPS key installation service are not supported by the key consumer, how the key consumer receives key packages is out of scope of the present document.

It is the responsibility of the key consumer to determine that the key manager is authorized to provide keys, and it is the responsibility of the key manager to determine that the key consumer is authorized to receive these keys.

### 4.3.8.3 Transfer initiated by the key consumer

Key managers may support key transfers initiated by the key consumer via an HTTPS key retrieval service, whereby the key consumer initiates an HTTPS connection to the key manager, which acts as an HTTP server.

When key transfers initiated by the key consumer via an HTTPS key retrieval service are not supported by the key manager, how the key consumer retrieves the key package is out of scope of the present document.

When key transfers initiated by the key consumer via an HTTPS key retrieval service are supported by the key manager, the key manager shall both support retrieval of a key package via all of the following HTTP GET requests (request-targets shown in origin-form):

1) `GET /.well-known/ets/keys?fingerprints=[fingerprints]`, where:

   a) `fingerprints` shall be present and its value, `[fingerprints]`, shall be either empty or shall be a comma-separated list of the hexadecimal string representation where each entry in the list is the static Diffie-Hellman public key fingerprint, as defined in clause 4.3.3, for which the corresponding public/private key pairs are being requested.

   b) The key manager shall return a key package that contains the corresponding public/private key pair for each fingerprint for which it has a record. In the unlikely case that the key manager has more than one public/private key pair corresponding to a given fingerprint, it shall return all of them in the key package. If `[fingerprints]` is empty, the actions of the implementation are out of scope of the present document.

   c) The key manager shall return an appropriate HTTP error code if there is not at least one matching public/private key pair [12].

EXAMPLE 1: The key consumer requests two keys, one with the fingerprint 00010203040506070809 and the other with the fingerprint 09080706050403020100. The key consumer supports responses in either format. The HTTP GET request would be:

```
GET /.well-
known/ets/keys?fingerprints=00010203040506070809,09080706050403020
100
Accept: application/pkcs8, application/cms
```

2) `GET /.well-known/ets/keys?groups=[groups]&certs=[sigalgs]&context=contextstr`, where:

   a) `groups` shall be non-empty and its value, `[groups]`, shall be a comma-separated list where each entry in the list is a NamedGroup value defined in clause 4.2.7 in IETF RFC 8446 [2], represented in hexadecimal notation, for which an associated static Diffie-Hellman key pair is being requested.

   b) `certs` may be included. If `certs` is included, its value, `[sigalgs]`, shall be a comma-separated list where each entry is a colon-separated pair of SignatureScheme values defined in clause B.3.1.3 in IETF RFC 8446 [2], in hexadecimal notation. The first value in the pair shall indicate the requested algorithm for the certificate issuer to use to sign the certificate. The second value in the pair shall indicate the requested algorithm to be used to generate the certificate subject's signing key pair. If `certs` is included, then for each entry in the list, the key consumer shall request one additional server certificate using that scheme, which is bound to all returned key pairs. If `certs` is not included, then no certificates are being requested, and so none shall be provided by the key manager.

   c) `context` may be included. If `context` is included, its value, `contextstr`, is a free string that the key manager shall use to determine what key pair and certificate contents to return. The structure of `contextstr` is not specified in the present document.

   d) The key manager shall return a key package containing a static Diffie-Hellman key pair for each group listed in `[groups]` that the key manager supports. For each static Diffie-Hellman key pair in the key package, the key manager shall also return a corresponding server certificate for each given signature algorithm pair listed in `[sigalgs]` that it supports.

e) If no group in `[groups]` is supported by the key manager, the key manager shall return an appropriate HTTP error code as defined in clause 6 of IETF RFC 7231 [12]. If the key manager is unable to use contextstr, the key manager may return an appropriate HTTP error code, as defined in clause 6 of IETF RFC 7231 [12], or it may handle the error itself in a way outside the scope of the present document.

EXAMPLE 2: The key consumer requests two keys, one that uses secp384r1 and one that uses x25519. The key consumer also requests two corresponding certificates bound to both of these keys. Of these two certificates:

1) One requests to be signed with rsa_pkcs1_sha256 and to be associated with a rsa_pss_pss_sha256 subject signing key.

2) The other is requested to be signed with ecdsa_secp384r1_sha384 and to be associated with an ecdsa_secp384r1_sha384 subject signing key.

The key consumer requests the resulting key package in only the `pkcs8` format. The HTTP GET request would be:

```
GET /.well-known/ets/keys?groups=0x0018,0x001d&certs=0x0401:0x0809,0x0503:0x0503
Accept: application/pkcs8
```

It is the responsibility of the key consumer to determine that the key manager is authorized to provide keys, and it is the responsibility of the key manager to determine that the key consumer is authorized to receive these keys.

# 5 Security

The Enterprise Transport Security profile does not provide all of the security assurances provided by TLS 1.3.

EXAMPLE 1: The Enterprise Transport Security profile does not provide per-session forward secrecy. Knowledge of a given static Diffie-Hellman private key can be used to decrypt all sessions encrypted with that key, and forward secrecy for all of those sessions begins when all copies of that static Diffie-Hellman private key have been destroyed.

This relaxation is deliberate. With proper management of the static Diffie-Hellman private key, the Enterprise Transport Security profile maintains a high-degree of authentication, integrity and confidentiality while also ensuring that:

a) security breaches are rapidly detected;

b) service availability is maximized by ensuring that services and application problems can be rapidly detected and rectified.

It is the responsibility of each entity deploying the Enterprise Transport Security profile to evaluate the trade-off between the security assurances of TLS 1.3 and the operational visibility provided by the Enterprise Transport Security profile.

EXAMPLE 2: An organization uses TLS 1.3 to connect to clients external to the enterprise network or data centre, but uses the Enterprise Transport Security profile for connections within its own data centre and cloud deployments (as in Figure 4.2). An organization can rotate their keys as frequently as they choose.

If the Enterprise Transport Security profile implementation is combined with a TLS 1.3 implementation that otherwise conforms to IETF RFC 8446 [2], use of the Enterprise Transport Security profile mode of operation shall be explicitly enabled.

As well as meeting TLS 1.3 security assurances, with the exemption described in Example 1, the Enterprise Transport Security profile is also mapped against other desired capability properties for MSP in annex C.

The annex A variant meets the same security assurances of TLS 1.3 as the full Enterprise Transport Security profile (i.e. with the exemption described in EXAMPLE 1). MSP profile capabilities relating to client visibility of middleboxes are not met; this is compensated for by the client operator having been informed by other means that the connection can be inspected.

# Annex A (normative): Middlebox visibility information variant

The present annex is optional. It establishes a variant of the Enterprise Transport Security profile that was defined in clause 4. The profile described in the present annex is the same as the profile in clause 4 in all respects, except that the server sends no visibility information and thus clause 4.3.3 is not carried out.

The variant of the Enterprise Transport Security profile described in the present annex may be implemented. If it is implemented, the client operator shall be informed by some means that the connection can be inspected. The means by which the client operator is informed is out of scope for the present document.

EXAMPLE 1: The client and server are wholly within a private enterprise network and the client operator has already been notified by alternative means, such as a condition of access to the network, that connections can be inspected.

EXAMPLE 2: The client operator agrees to an acceptable use policy in order to access a private enterprise network. This acceptable use policy indicates that connections can be inspected.

# Annex B (normative): Requirements for an Enterprise Transport Security aware client

The present annex is optional.

A TLS 1.3 client that satisfies the requirements specified in the present annex can be designated as "Enterprise Transport Security aware":

a) The client shall provide configuration means to accept all Enterprise Transport Security connections or deny all Enterprise Transport Security connections as indicated by the certificate extension defined in clause 4.3.3.

b) If the client is a browser, it shall be possible to control the configuration in requirement (a) above by means of policy provided by a centralized policy controller.

c) If the client provides a means of viewing a server's TLS certificate, it shall correctly parse and display the contents of the certificate extension defined in clause 4.3.3.

d) If the client is a browser, the user of the browser shall be able to see the Enterprise Transport Security policy that is in force, even if the policy is not under their direct control.

e) The client may provide a means to only accept Enterprise Transport Security connections that match a whitelist.

f) The client may provide a means to only deny Enterprise Transport Security connections that match a blacklist.

g) The client may offer a user prompt to accept Enterprise Transport Security connections.

# Annex C (informative): Mapping MSP desired capabilities to the Enterprise Transport Security profile

The reader is advised that some of the capabilities described here are not necessarily assured by the protocol, but are assured to the extent that entities with access to the Enterprise Transport Security private key are trusted. This trust in these Enterprise Transport Security entities - for the server to provide accurate visibility information, and for all entities to share the Enterprise Transport Security private key only according to that policy - assures some of the capabilities described in present annex. The remaining capabilities are assured by the Enterprise Transport Security profile.

The Enterprise Transport Security profile is defined as a 1-sided, single-context MSP profile. These definitions are in the planned MSP Part 1 [i.1] as well as in the present document, as described in clause 4.1.

The Enterprise Transport Security profile meets the mandatory capabilities for a 1-sided, single-context MSP profile. MSP capabilities, defined for the MSP standards, are split into three groupings:

- **Audit:** capabilities that relate to the ability for a middlebox to be audited using MSP.
- **Access:** capabilities that relate to the access granted to a middlebox that is using MSP.
- **Visibility:** capabilities that relate to the ability for a middlebox to be discovered using MSP.

Firstly, mapping to audit capabilities, the Enterprise Transport Security profile meets the following:

a) Destination endpoint able to detect if an unauthorized change to the data has occurred.

b) Middleboxes with read-only access able to detect if an unauthorized change to the data has occurred.

c) Middleboxes with permission to modify content able to detect if an unauthorized change to the data has occurred.

d) Middleboxes modifying content able to validate that no unauthorized changes have occurred prior to receipt by middleboxes or 3rd parties.

These audit capabilities are satisfied by the Enterprise Transport Security profile, as they are inherited from the security properties of TLS 1.3 (see clause 5 of the present document).

Secondly, the following access capability is met:

e) Client or server, alone, able to grant middlebox access permissions.

This is satisfied, as the Enterprise Transport Security server alone can grant access to a device by sharing its Diffie-Hellman private key with that device.

Thirdly, these visibility capabilities are met:

f) Client able to learn the owner of all middleboxes.

g) Client able to learn the identity and function of all third-party middleboxes or groups of third-party middleboxes (i.e. middleboxes not under ownership of client or server endpoints).

h) Client able to learn the identity and function of all middleboxes or groups of middleboxes.

i) Server able to learn the owner of all middleboxes.

j) Server able to learn the identity and function of all third-party middleboxes or groups of third-party middleboxes.

k) Server able to learn the identity and function of all middleboxes or groups of middleboxes.

l) The client and server able to receive validation of identity of all middleboxes or groups of middleboxes.

m) Client able to learn the long-term identity of the server.

n) The client able to receive validation of the server long-term identity.

o) Server able to learn an identity (which may include "anonymous" or similar) for the client.

p) Server able to reject anonymous clients (if anonymous clients are supported).

The visibility capabilities f), g), h), i), j), k) and l) are satisfied by the inclusion of visibility information in the Enterprise Transport Security server certificate.

The visibility requirements m), n), o) and p) are satisfied by inheritance of the security assurances from TLS 1.3.

The accessDescription field, part of the Visibility Information field defined in clause 4.3.3, identifies, either generally or specifically, the controlling or authorizing entities or roles or domains, or any combination of these, of any middleboxes who may be given access to the private key of the static Diffie-Hellman public key that is identified in the fingerprint field.

EXAMPLE: An accessDescription can restrict access of the private key to the administrative domain that controls all middleboxes of a company.

The certificate issuer and the Enterprise Transport Security server are trusted to enforce the visibility restriction on the middlebox entities that are given access to the private key. Though any dishonest holder of key material in an Enterprise Transport Security connection can give it to any party, this is also the case in TLS 1.3 - and the protection of the key material is assured to the extent that the Enterprise Transport Security endpoint is trusted by the other endpoint in the connection.

The annex A variant of the Enterprise Transport Security profile meets the same MSP profile capabilities except for f), g), h) and l) which are the capabilities relating to client visibility of middleboxes. This is because the client does not receive the visibility information. This is compensated for by the client operator having been informed by other means that the connection can be inspected.

# History

| Document history | | |
|---|---|---|
| V1.1.1 | October 2018 | Publication |
| V1.2.1 | March 2019 | Publication |
| | | |
| | | |
| | | |