ETSI TS 103 523-5 V1.1.1 (2021-12)

CYBER;
Middlebox Security Protocol;
Part 5: Enterprise Network Security

Reference
DTS/CYBER-0027-5

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 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.
No recommendation as to products and services or vendors is made or should be implied.
No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.
In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.
The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2021.
All rights reserved.

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are 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 Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, 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.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

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

The present document is part 5 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, 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. The IPsec protocol standards [1], [i.2] and [i.3] can be managed in a way that enables this enterprise visibility.

The present document is one of a series of MSP implementation profiles that, to achieve the required visibility, puts the enterprise operators and users in control of the access to their data. It sets forth an "Enterprise Network Security" profile for use in enterprise networks and data centres that meets the security provisions defined in the Middlebox Security Protocol (MSP) Framework [i.1].

Introduction

The present document specifies the MSP profile for Enterprise Network Security, which is an implementation variant of Internet Key Exchange Protocol Version 2 (IKEv2) [1]. Hereafter this profile is referred to as "Enterprise Network Security" or ENS for brevity. This profile maintains interoperability with IPsec peers that are unaware of this profile; however, when a certificate is used, an extension provides notice that this profile is being used.

There are operational circumstances where passive decryption of IPsec-encrypted packets by authorized entities is a requirement. This decryption can be performed in real-time, or the packets can be captured, stored, and then decrypted later.

Situations requiring passive decryption of IPsec-encrypted packets generally occur in environments where both the communicating peers, and by inference the data being exchanged, are under the control of the same organization. IPsec encryption is often stipulated by internal or external security policies. Authorized access to the plaintext packet data is required for operational reasons, including:

- application health monitoring and troubleshooting;
- intrusion detection;
- detection of malware activity, including lateral movement, command and control, and data exfiltration traffic;
- detection of advanced Distributed Denial Of Service (DDOS) attacks; and
- compliance audits.

One possible approach to decrypting IPsec-encrypted packets passively is to export the keying material generated from ephemeral Diffie-Hellman key pairs for each Security Association (SA) to one or more middleboxes. However, this approach has significant limitations: two in particular are as follows. Firstly, it is very difficult to ensure that the exported SA keying material will arrive at the middlebox in sufficient time to allow decryption in real-time. Secondly, the keying material needs to be correlated with every stored packet session in anticipation of post-capture decryption. For these reasons and more, this approach does not scale to the needs of a data centre.

The ENS profile therefore uses longer-lived Diffie-Hellman public/private key pairs for one IKEv2 party. These Diffie-Hellman keys are used by that party to establish many SAs. This enables Diffie-Hellman public/private key pairs used to generate SA keying material to be distributed to real-time decryption middleboxes prior to the arrival of network packets. It also greatly reduces the amount of keying material to be stored and correlated with packet storage systems. Enterprises can implement automated key rotation to frequently change the Diffie-Hellman public/private key pairs.

IKEv2 supports three forms of authentication: certificates and digital signatures, pre-shared keys, and the Extensible Authentication Protocol (EAP) [i.6]. In each case, the ENS profile requires notice to be provided by the party using longer-lived keys that IKEv2-protected packets can be decrypted and inspected by one or more middleboxes. When certificates [3] are used, the ENS profile includes the capability for such notice to be included in certificates, which indicates to the other IPsec peer that the corresponding static Diffie-Hellman private key is being used to enable visibility. This visibility information describes 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 Diffie-Hellman private key.

Annex A describes an exception to the certificate visibility requirement. In these situations, notice of packet inspection will be provided in another way. One use case is where pre-shared keys or Extensible Authentication Protocol are used for authentication and there is no certificate. A second use case of this variant is when the IPsec peers are wholly within a private enterprise network and the operator of the peers has already been notified by alternative means, such as a condition of access to the network, that IKEv2-protected packets can be decrypted and inspected.

The ENS profile is compatible with any IPsec peer that implements IKEv2 for the establishment of SAs. Another variant is described in Annex B - an "Enterprise Network Security aware IPsec peer" - whereby the IPsec endpoint provides additional visibility and control over the use of the ENS profile for the IPsec operator.

1 Scope

The present document specifies a protocol implementation profile to enable secure communication between IPsec-protected network endpoints while enabling network operations. The Enterprise Network Security profile depends on two protocols in the IPsec family of protocols. First, Internet Key Exchange Protocol Version 2 (IKEv2) [1] is used to establish Security Associations (SAs). In this profile, when certificates are used to provide authentication in IKEv2, those certificates include an extension to provide notice that this profile is being used. Second, the IP Encapsulating Security Payload (ESP) [i.2] is used to encrypt packets.

This profile describes two deployment scenarios. In the first one, one of the IPsec peers is inside the enterprise and the other one is outside the enterprise. In the other scenario, both IPsec peers are inside the enterprise. This profile describes the Diffie-Hellman key exchange, and it specifies the certificate extension that provides visibility information to indicate that the ENS profile is being used.

The actions the IPsec peers take upon receiving the visibility information in the certificate extension and structure of the policy included in the visibility information are not normatively defined; however, capabilities for an optional "Enterprise Network Security aware IPsec peer" are defined. The means by which the IPsec endpoints obtain the longer-lived Diffie-Hellman public/private key pairs is specified, and some examples are provided.

A variant of the ENS profile is also provided to enable visibility in circumstances where the operator of an IPsec peer has been informed by other means that packets can be decrypted and inspected.

The present document also includes the security guarantees made by the ENS profile, based on the security guarantees of the IPsec family of protocols.

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 7296: "Internet Key Exchange Protocol Version 2 (IKEv2)".

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

[3] Recommendation ITU-T X.509 (10/2019) | ISO/IEC 9594-8: "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks".

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

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

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

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

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

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

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

[11] IETF RFC 8551: "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification".

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

[13] IETF RFC 8410: "Algorithm Identifiers for Ed25519, Ed448, X25519, and X448 for Use in the Internet X.509 Public Key Infrastructure".

[14] IANA: "Internet Key Exchange Version 2 (IKEv2) Parameters".

NOTE: Available at https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml.

[15] IETF RFC 6989: "Additional Diffie-Hellman Tests for the Internet Key Exchange Protocol Version 2 (IKEv2)".

[16] IETF RFC 4055: "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile".

[17] IETF RFC 3339: "Date and Time on the Internet: Timestamps".

[18] IETF RFC 1123: "Requirements for Internet Hosts -- Application and Support".

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: MSP Framework and Template Requirements".

[i.2] IETF RFC 4303: "IP Encapsulating Security Payload (ESP)".

[i.3] IETF RFC 4301: "Security Architecture for the Internet 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] IETF RFC 3748: "Extensible Authentication Protocol (EAP)".

[i.7] IETF RFC 1983: "Internet Users' Glossary".

[i.8] IETF RFC 5996: "Internet Key Exchange Protocol Version 2 (IKEv2)".

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

Child SA: SA established by IKEv2 for use with the Encapsulating Security Payload (ESP) protocol or Authentication Header (AH) protocol

Diffie-Hellman (DH) elements: longer-lived Diffie-Hellman public/private key pair contained in the Asymmetric Key Package

Internet Key Exchange Security Association (IKE SA): shared secret information that can be used to efficiently establish Child SAs

initiator: IKEv2 endpoint that begins negotiation of a Security Association

responder: peer IKEv2 endpoint to the initiator

Security Association (SA): secret information shared by IPsec peers

Signature (SIG) elements: private signing key and a certificate with the corresponding public key contained in the Asymmetric Key Package

3.2 Symbols

Void.

3.3 Abbreviations

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

ASN	Abstract Syntax Notation
CERT	Certificate
CMS	Cryptographic Message Syntax
DDOS	Distributed Denial Of Service
DER	Distinguished Encoding Rules
DH	Diffie-Hellman
EAP	Extensible Authentication Protocol
ENS	Enterprise Network Security
ESP	Encapsulating Security Payload
HTTP	HyperText Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IKE	Internet Key Exchange
IKEv2	Internet Key Exchange Protocol Version 2
IP	Internet Protocol
IPsec	IP security
KE	Key Exchange
MSP	Middlebox Security Protocol
SA	Security Association
SIG	Signature
TLS	Transport Layer Security
UTC	Universal Time Coordinated

4 Enterprise Network Security for the Middlebox Security Protocol (MSP) framework

4.1 Profile characteristics

In the ENS profile defined in the present document, one endpoint uses a longer-lived Diffie-Hellman key and so that endpoint unilaterally enables traffic visibility. The ENS profile defined in the present document also grants access to the entire data stream associated with the Security Associations (SAs) that are established under the longer-lived Diffie-Hellman public/private key pair. The ENS profile shall only be used within an organization's network boundaries and/or on network connections between partner organizations, and not used for internet connections available to the general public.

4.2 Deployment of Enterprise Network Security (informative)

4.2.1 Introduction

IPsec offers two modes of use: transport mode and tunnel mode as defined in IETF RFC 4301 [i.3]. In transport mode, ESP provides protection primarily for next layer protocols, and a transport mode SA is typically employed between a pair of hosts to provide end-to-end security services. In tunnel mode, ESP is applied to tunnelled IP packets, and a tunnel mode SA is typically employed between two security gateways or between a host and a security gateway.

EXAMPLE: An encrypting firewall is a type of a security gateway.

4.2.2 Transport Mode Security Associations

Figure 4.1 depicts the deployment of ENS in a configuration where all IPsec SAs use transport mode. As a result, one set of end-to-end SAs are established between Protected Endpoint 1 and Protected Endpoint 2. The SAs are established under the longer-lived Diffie-Hellman public/private key pair, and this enables the middlebox to decrypt the packets that are encrypted under those SAs.

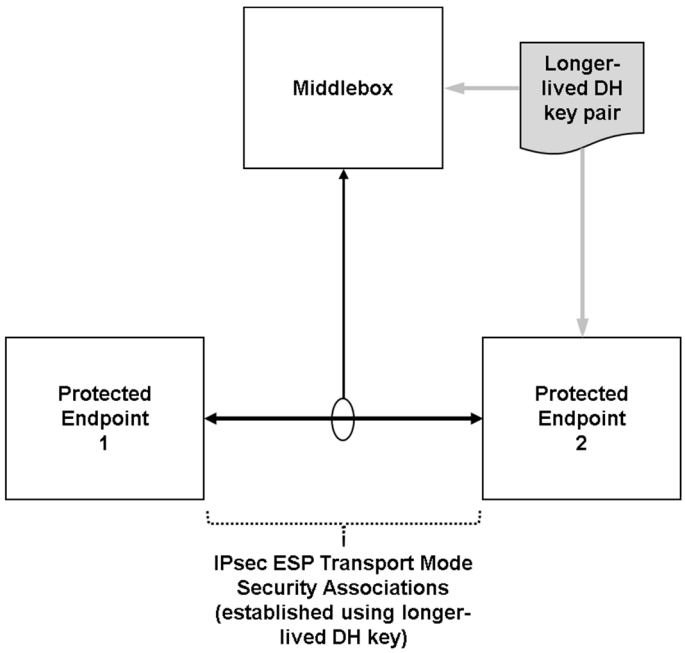

Figure 4.1: ENS with IPsec Transport Mode

The middlebox receives a copy of the longer-lived Diffie-Hellman public/private key pair used by Protected Endpoint 2 to establish the IKEv2 SAs. It also receives a passive copy of packets sent between the two protected endpoints. This enables the middlebox to decrypt the packets in real-time to perform its functions, such as real-time intrusion detection and malware detection. The middlebox can also store the packets for later decryption and back-in-time analysis.

4.2.3 Tunnel Mode Security Associations

Figure 4.2 depicts the deployment of ENS with two tunnel endpoints (enterprise security gateways) protecting communications between two subnets using IKEv2 in tunnel mode. In this deployment environment, all IPsec SAs use tunnel mode.

As in the case of Transport Mode, the middlebox receives a passive copy of the packets between the two tunnel endpoints along with a copy of the longer-lived Diffie-Hellman public/private key pair used by Tunnel Endpoint 2 to establish the IKEv2 SAs. This enables the middlebox to decrypt the packets in real-time or to store the packets for decryption and analysis at a later time.

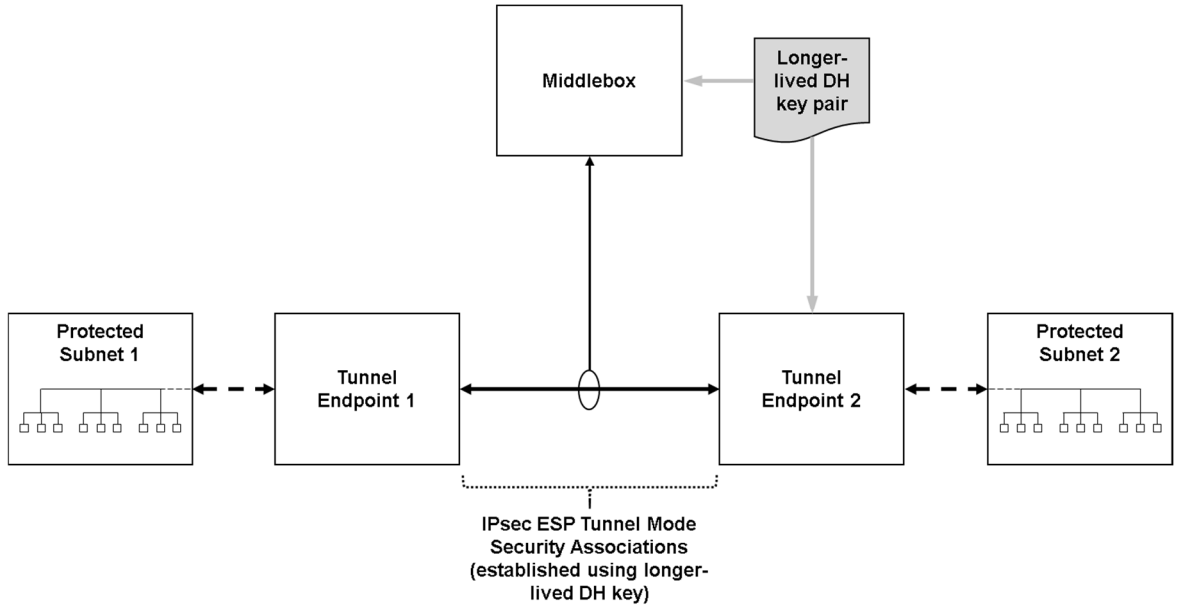

Figure 4.2: ENS with IPsec Tunnel Mode

4.3 Enterprise Network Security

4.3.1 IKEv2 Diffie-Hellman Key Exchange (informative)

IKEv2 [1] is a component of the IPsec family of protocols that performs mutual authentication and establishes Security Associations (SAs). IKEv2 establishes an IKE SA that includes shared secret information that can be used to efficiently establish SAs for the Encapsulating Security Payload (ESP) [i.2]. All IKEv2 communications consist of pairs of messages: a request and a response.

The first exchange of an IKEv2 session, IKE_SA_INIT, negotiates security parameters for the IKE SA, sends nonces, and sends Diffie-Hellman public keys.

The second exchange, IKE_AUTH, transmits identities, proves knowledge of the secrets corresponding to the two identities, and sets up an SA for the first ESP SA, which is referred to as a Child SA. The messages in this exchange can include a certificate or certificate chain. The certificate provides evidence that the key used to compute a digital signature belongs to the name in the ID payload. When certificates are not used, either pre-shared keys or the Extensible Authentication Protocol (EAP) [i.6] are used for authentication.

Subsequent CREATE_CHILD_SA exchanges can be used to establish additional Child SAs for use by ESP. In addition, new keys for the IKE SA and existing ESP SAs can be established with the CREATE_CHILD_SA exchange.

For reference, a brief description of the normal IKEv2 exchange is provided below. Complete details can be found in the IKEv2 specification [1]. In many scenarios, only steps 1 through 7 are used. However, complete visibility requires insight into the creation of multiple ESP SAs, rekeying the IKE SA, and rekeying existing ESP SAs. For this reason, steps 8 through 16 are also provided:

1) The initiator sends the IKE_SA_INIT request, which includes the initiator's Diffie-Hellman public key (KEi) and the initiator's nonce (Ni).

2) The responder sends the IKE_SA_INIT response, which completes the Diffie-Hellman exchange by including the responder's Diffie-Hellman public key (KEr) and the responder's nonce (Nr). When certificates are used for authentication, a list of trust anchors is provided (CERTREQ).

3) The initiator and responder each use their own Diffie-Hellman private key, the Diffie-Hellman public key received from the peer, and the exchanged nonces to generate a shared secret, called SKEYSEED. The details of the computation are described in section 2.14 of the IKEv2 specification [1].

4) The initiator and responder then use the SKEYSEED, along with other information from the IKE_SA_INIT exchanges to generate the cryptographic keys for the IKE SA, which includes an encryption key (SK_e) for each direction, and an authentication key (SK_a) for each direction. In addition, another secret quantity is computed (SK_d) for the derivation of further keying material for Child SAs as well as for the rekeying of the IKE SA.

5) The initiator sends the IKE_AUTH request, which asserts an identity (IDi) and authenticates that identity. When certificates are used for authentication, the initiator's certificate is sent (CERT), a list of trust anchors is provided (CERTREQ), and the identity is authenticated by a digital signature and certificate.

6) The responder sends the IKE_AUTH response, which asserts an identity (IDr) and completes negotiation of a Child SA for use with ESP. When certificates are used for authentication, the responder's certificate is sent (CERT), and the identity is authenticated by a digital signature and certificate.

7) The initiator and responder then use SK_d and the exchanged nonces to compute the keying material, called KEYMAT, for the SA used for ESP.

8) If additional Child SAs are needed, the initiator sends the CREATE_CHILD_SA request, which includes a nonce (Ni) and optionally a Diffie-Hellman public key (KEi).

9) The responder replies with the CREATE_CHILD_SA response, which includes a nonce (Nr) and a Diffie-Hellman public key (KEr) if KEi was included in the request.

10) The initiator and responder then use SK_d, the exchanged nonces, and optionally their own Diffie-Hellman private key along with the Diffie-Hellman public key received from the peer to compute the keying material, called KEYMAT, for the SA used for ESP.

11) If there is a desire to rekey the IKE SA, the initiator sends the CREATE_CHILD_SA request, which includes a nonce (Ni) and a Diffie-Hellman public key (KEi).

12) The responder replies with the CREATE_CHILD_SA response, which includes a nonce (Nr) and a Diffie-Hellman public key (KEr).

13) The initiator and responder then use SK_d, the exchanged nonces, and their own Diffie-Hellman private key along with the Diffie-Hellman public key received from the peer to compute a new SKEYSEED and the keying material for the IKE SA.

14) If there is a desire to rekey an existing Child SA, the initiator sends the CREATE_CHILD_SA request, which includes a nonce (Ni) and optionally a Diffie-Hellman public key (KEi).

15) The responder replies with the CREATE_CHILD_SA response, which includes a nonce (Nr) and a Diffie-Hellman public key (KEr) if KEi was included in the request.

16) The initiator and responder then use SK_d, the exchanged nonces, and optionally their own Diffie-Hellman private key along with the Diffie-Hellman public key received from the peer to compute the KEYMAT for the SA used for ESP.

4.3.2 Diffie-Hellman key exchange with visibility

The ENS profile shall use exactly the same messages and procedures as defined in IETF RFC 7296 [1] to establish the initial IKE SA, to rekey the IKE SA, to establish ESP SAs, and to rekey ESP SAs. However, the party located in the enterprise network or data centre (whether initiator or responder), shall do all of the following:

1) use longer-lived Diffie-Hellman public/private key pairs instead of ephemeral keys;

2) when certificates are used for authentication, use a certificate that includes an extension that contains visibility information as defined in clause 4.3.3 to indicate that this profile is being used.

   (When pre-shared keys or the Extensible Authentication Protocol (EAP) [i.6] are used for authentication, notice that packets may be inspected is provided in another way as specified in Annex A).

NOTE 1: Neither the longer-lived Diffie-Hellman public key nor the certificate extension containing visibility information affect the operation of IKEv2, so an IPsec peer that follows the Enterprise Network Security profile is fully interoperable with other IPsec peers.

IPsec peers that follow the Enterprise Network Security profile shall be provisioned with longer-lived Diffie-Hellman public/private key pairs.

NOTE 2: Supported Diffie-Hellman groups can be elliptic curves or finite fields, defined in the IANA Internet Key Exchange Version 2 (IKEv2) parameters registry [14].

The longer-lived Diffie-Hellman public/private key pairs may be shared with middleboxes that are authorized to decrypt packets from the IPsec peers.

The Diffie-Hellman public/private keys shall be either:

1) individually managed on the IPsec peer, and rotated according to organizational policy, described in clause 4.3.4.2; or

2) downloaded from a central key manager and updated according to organizational policy, described in clause 4.3.4.3.

4.3.3 Visibility information

The ENS profile shall support authentication based on a certificate, as defined in Recommendation ITU-T X.509 [3]. The certificate shall include an extension that contains visibility information to indicate that the ENS profile is being used, unless such notice is provided by some other means as described in Annex A.

1) The visibility information in the certificate should be bound to the longer-lived Diffie-Hellman public/private key pairs used in the IKEv2 protocol exchanges by including the fingerprint of the longer-lived Diffie-Hellman public key.

2) The longer-lived Diffie-Hellman public key used for key exchange shall not be the same as the certificate subject public key in the certificate, which is used for the digital signature for IKEv2 authentication.

3) The visibility information in the certificate shall identify, either generally or specifically, the controlling or authorizing entities or roles or domains, or any combination of these, of any middleboxes that can access the longer-lived Diffie-Hellman public/private key pairs described in clause 4.3.2 of the present document.

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

```
VisibilityInformation ::= SEQUENCE {
   keyBindings          KeyBindings OPTIONAL,
   maxLifetime          INTEGER (0..4294967295) OPTIONAL, -- 2**32-1
   accessBy             UTF8String}

KeyBindings ::= SEQUENCE {
   digestAlgorithm[0]   AlgorithmIdentifier DEFAULT sha256Identifier,
   fingerprintSize[1]   INTEGER(10..64) DEFAULT 10,
   fingerprints         SEQUENCE (SIZE(1..MAX)) OF Fingerprint}

Fingerprint ::= OCTET STRING (SIZE(10..64))
```

where:

- The `keyBindings` may be present in one of the `VisibilityInformation` structures present and shall be included in all others. If present, `keyBindings` contains the fingerprints of longer-lived Diffie-Hellman public keys that are bound to the certificate and subject to the other constraints expressed by this instance of `VisibilityInformation`:

  - The `digestAlgorithm` identifies the digest algorithm used to compute the fingerprints. The default digest algorithm is SHA-256 [2], defined by `sha256Identifier` [16].

  - The `fingerprintSize` indicates the size of the fingerprint in octets and shall be no greater than the size of the digest algorithm output. If the digest algorithm produces a value that is larger than this value, then the digest is truncated so that only the high-order bits are used.

  - Each `Fingerprint` in `fingerprints` shall be 10 octets (80 bits) or the size indicated by `fingerprintSize,` and shall be set to the (possibly truncated) SHA-256 digest or the digest identified by `digestAlgorithm`, and shall be computed on the longer-lived Diffie-Hellman public key for use in the IKEv2 exchanges. Specifically, the input to the digest shall be the contents of the Key Exchange Data field of the KE payload defined in section 3.4 of IETF RFC 7296 [1]. For the default digest algorithm with the default fingerprint size of 10 octets (80 bits), the digest shall be represented as the vector of 32-bit words (H_0,H_1,…,H_7) as defined in FIPS 180-4 [2], and then truncated to $H_0||H_1||(H_2>>16)$, which is the high-order 10 octets (80 bits) of the vector in big-endian format. The `fingerprints` SEQUENCE shall contain one or multiple `Fingerprint` entries.

- The `maxLifetime` may be present. If included, `maxLifetime` is the maximum time period a longer-lived Diffie-Hellman public key for IKEv2 exchanges shall be used with the certificate, in units of seconds.

- The `accessBy` 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 can be granted access to the longer-lived Diffie-Hellman private key. The `accessBy` field shall be accurate. A structure for the description is not defined in the present document.

`maxLifetime` and `accessBy` provide visibility to the operator of the IPsec peer about the key rotation frequency and the administrative element that is making use of this facility.

The `VisibilityInformation` structure shall be sent as an `AttributesSyntax` field entry of the `associatedInformation` certificate extension as defined in clause 9.3.2.4 of Recommendation ITU-T X.509 [3]. When the `associatedInformation` extension is being used for ENS `VisibilityInformation`, the extension shall be flagged as non-critical.

NOTE 1: Flagging the extension as non-critical is to aid interoperability and deployment. If the extension was flagged as critical, it would prevent an ENS unaware peer from accepting the connection, because the peer would not be able to process the extension.

The attribute type shall be set to the Object Identifier 0.4.0.3523.5.1 {itu-t(0) identified-organization(4) etsi(0) msp(3523) ens(5) visibility(1)} and the attribute value shall be set to the DER encoding of `VisibilityInformation`. This is described by the following ASN.1:

```
associatedInformation EXTENSION ::= {
      SYNTAX         AttributesSyntax
      IDENTIFIED BY  id-ce-associatedInformation }

visibilityInformation ATTRIBUTE ::= {
      WITH SYNTAX     VisibilityInformation
      SINGLE VALUE    FALSE
      ID              id-msp-ENS-visibility }

id-msp-ENS-visibility OBJECT IDENTIFIER ::= {itu-t(0)
   identified-organization(4) etsi(0) msp(3523) ens(5) visibility(1)}
```

A `Fingerprint`, in conjunction with the certificate's validity period, binds the certificate to a particular longer-lived Diffie-Hellman public/private key pair for the entire validity time period. `maxLifetime` limits the amount of time any particular longer-lived Diffie-Hellman public/private key pair shall be re-used for multiple IKEv2 key exchanges with the ENS peer using that certificate.

A longer-lived Diffie-Hellman public/private key pair shall not be used to establish an SA unless at the time of SA establishment the ENS peer has:

1) a valid certificate to which the key pair is bound, where the matching `VisibilityInformation` structure either contains no `maxLifetime` field or whose `maxLifetime` has not yet expired; or

2) a valid certificate to which the key pair is not bound which contains a `VisibilityInformation` structure that includes no fingerprints and either contains no `maxLifetime` field or whose `maxLifetime` has not yet expired; or

3) a valid certificate which contains no `VisibilityInformation` (per Annex A).

There shall be zero or one `visibilityInformation` attribute per `associatedInformation` extension.

The certificate issuer has several options for including the `visibilityInformation` attribute in the `associatedInformation` extension:

1) The certificate issuer may include a single `VisibilityInformation` structure with no fingerprints. In this case, the optional `keyBindings` field is absent. This serves as an indicator that ENS is being used, but certificate issuance is decoupled from the Diffie-Hellman public/private key pairs used with ENS. Where the `keyBindings` field is absent, the `maxLifetime` field should be used.

NOTE 2: It can be impractical to define fingerprints in cases where generation of longer-lived Diffie-Hellman public/private key pairs is decoupled from certificate generation. In such cases, longer-lived Diffie-Hellman public/private key pairs can be changed more frequently than the certificate. The `maxLifetime` field defines the maximum interval of time for which any particular longer-lived Diffie-Hellman public/private key pair can be used, and makes it available to the IPsec peer.

Figure 4.3: VisibilityInformation Structure with No Fingerprint

2) The certificate issuer may bind a single longer-lived Diffie-Hellman public/private key pair to a certificate by including a single fingerprint in the `keyBindings` field of a single `VisibilityInformation` structure in the `visibilityInformation` attribute. This binds the certificate to the indicated longer-lived Diffie-Hellman public/private key pair and requires issuing a new certificate when the longer-lived Diffie-Hellman public/private key pair is rotated.

Figure 4.4: VisibilityInformation Structure with a Single Fingerprint

3) The certificate issuer may bind multiple longer-lived Diffie-Hellman public/private key pairs to a single certificate by including multiple fingerprints in the `keyBindings` field of a single `VisibilityInformation` structure, one `Fingerprint` for each key pair. This binds the certificate to the indicated set of longer-lived Diffie-Hellman public/private key pairs and allows for key rotation within this set of longer-lived Diffie-Hellman public/private key pairs without issuing a new certificate.

NOTE 3: In this case, the `maxLifetime` field defines the maximum interval of time that any one of the set of bound longer-lived Diffie-Hellman public/private key pairs can be used for IKEv2 exchanges, and makes this time interval available to the IPsec peer.

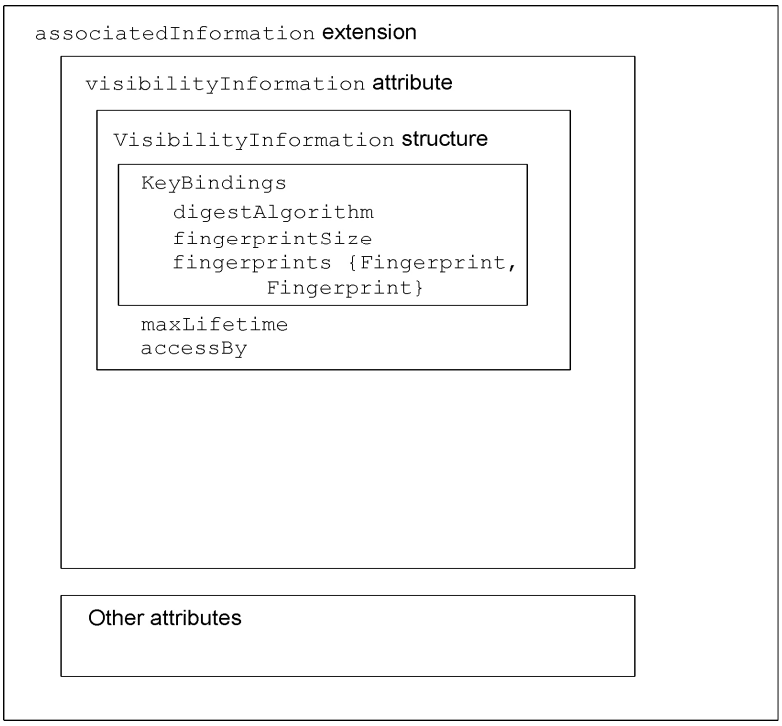

Figure 4.5: VisibilityInformation Structure with Multiple Fingerprints

4) The certificate issuer may include multiple `VisibilityInformation` structures in the `visibilityInformation` attribute. This allows different combinations of `accessBy` and the optional `maxLifetime` field to be bound to different sets of fingerprints, including binding to no fingerprint.

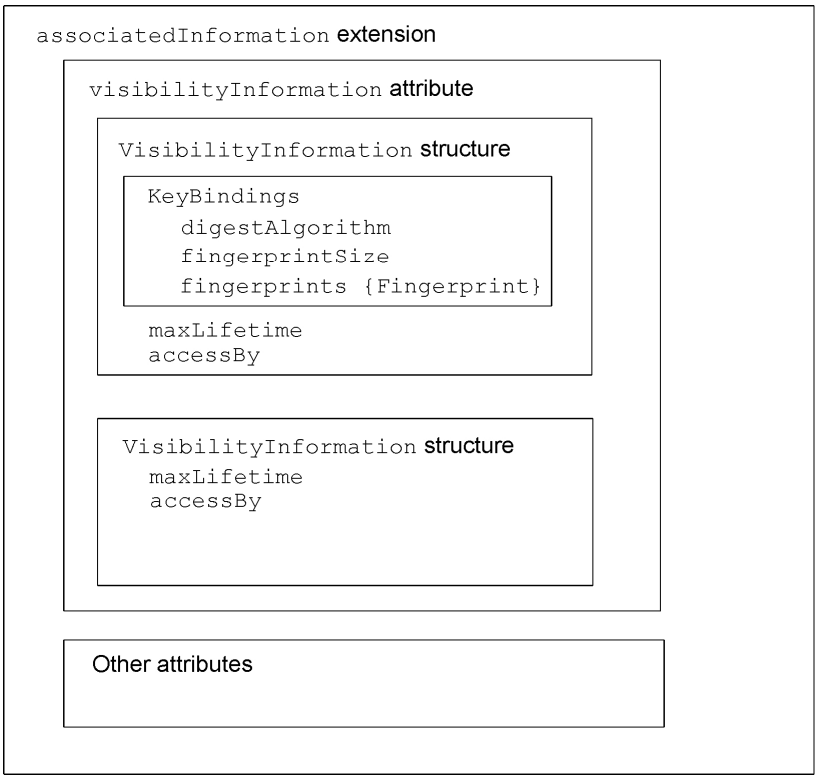

Figure 4.6: VisibilityInformation attribute with Multiple VisibilityInformation structures

When an IPsec peer presents a certificate with one or more `VisibilityInformation` structures, the public key fingerprint shall match at most one `VisibilityInformation` structure, and the processing rules for the Diffie-Hellman public key in the IKEv2 key exchange are as follows:

1) If the public key fingerprint matches one `VisibilityInformation` structure, then the `accessBy` and `maxLifetime` fields of this structure shall apply.

2) If the public key fingerprint does not match any `VisibilityInformation` structures, the `accessBy` and `maxLifetime` fields of the `VisibilityInformation` structure with no `KeyBindings` shall apply.

3) If the public key fingerprint does not match any structure and there is no structure without `KeyBindings`, this is a violation and the connection should be refused.

An IPsec peer receiving the visibility information should follow the actions for an ENS-aware peer that can be found in Annex B, which is optional.

An optional variant of ENS, where visibility information is not sent, is defined in Annex A. This variant may be used when visibility information is not suitable. This variant shall not be used unless the operator of the IPsec peer has been informed by some means that the packets can be inspected. The means by which the operator is informed is out of scope of the present document.

EXAMPLE: A client could be informed of the visibility information 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 via a configuration option; otherwise, the IPsec peer shall not establish Enterprise Network Security using certificates without visibility information.

4.3.4 Longer-lived Diffie-Hellman public/private key pair management

4.3.4.1 General

The present document does not mandate how the longer-lived Diffie-Hellman public/private key pairs defined in clause 4.3.2 are shared between IPsec peers, middleboxes, and other key consumers. However, an ENS implementation may use the mechanisms specified in clauses 4.3.4.2 to 4.3.4.3.4. The security of the ENS IPsec session depends on the protection of the longer-lived Diffie-Hellman private key through its lifecycle.

4.3.4.2 Individually managed keys

This clause describes one of the simplest means of management. When the individually managed keys mechanism is used, the longer-lived Diffie-Hellman public/private key pair shall be manually or automatically installed in both the IPsec peer and permitted middleboxes.

4.3.4.3 Centrally managed keys

4.3.4.3.1 Introduction

Figure 4.7 shows the architecture for using a central key manager to deploy the Enterprise Network Security profile longer-lived Diffie-Hellman keys to a key consumer, which include the IPsec peer and passive decryption middleboxes.

When this centrally managed keys mechanism is used, the key manager shall generate the longer-lived Diffie-Hellman public/private key pairs. The key manager may actively push keys to the key consumer or the key consumer may pull keys from the key manager. The Asymmetric Key Package is specified in clause 4.3.4.3.2, the protection of the key package is specified in clause 4.3.4.3.3, and the transfer mechanism is specified in clause 4.3.4.3.4.

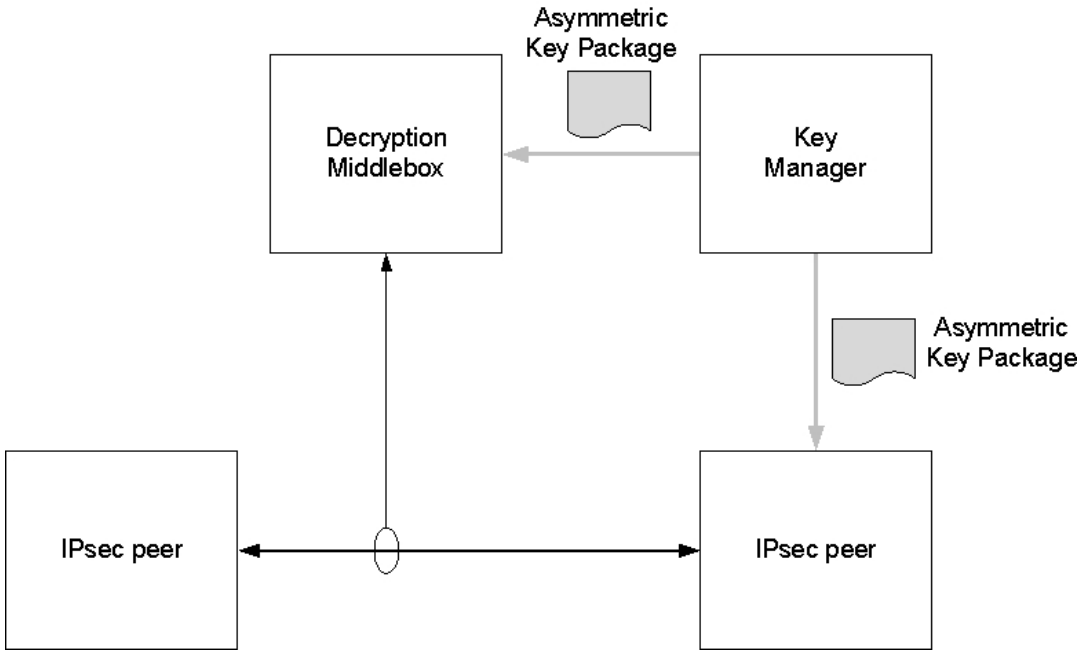

Figure 4.7: Centralized management of longer-lived Diffie-Hellman keys

4.3.4.3.2 Asymmetric key package

When Enterprise Network Security profile longer-lived Diffie-Hellman public/private key pairs are sent from the key manager to a key consumer using this centrally managed keys mechanism, they shall be packaged using the Asymmetric Key Package defined in IETF RFC 5958 [4]. Each Asymmetric Key Package shall contain one or more `OneAsymmetricKey` elements. Such an element shall be either:

a) longer-lived Diffie-Hellman public/private key pairs, hereafter referred to as DH elements; or

b) a private signing key and a certificate with the corresponding public key, hereafter referred to as SIG elements.

Although certificates are not sent in the same `OneAsymmetricKey` element as longer-lived Diffie-Hellman keys, each Asymmetric Key Package may contain one or more SIG elements in the Asymmetric Key Package that are bound to the longer-lived Diffie-Hellman public keys 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. Within the Asymmetric Key Package, the DH elements shall appear before the SIG elements.

For DH elements, the `OneAsymmetricKey` structure that is used to store the longer-lived Diffie-Hellman public/private key pairs in the Asymmetric Key Package shall be as defined in IETF RFC 5958 [4]. The `OneAsymmetricKey` fields are set as follows:

1) The `version` field shall be set to version 2 (integer value of 1).

2) The `privateKeyAlgorithm` field shall be set to the appropriate Diffie-Hellman algorithm identifier for the Diffie-Hellman group (defined as `PrivateKeyAlgorithmIdentifier` in IETF RFC 5958 [4]), whether the group uses an elliptic curve or a finite field.

3) The `privateKey` field shall be set to the Diffie-Hellman private key, encoded as an `OCTET STRING`.

4) The `publicKey` field shall be present, and the field shall be set to the Diffie-Hellman public key, encoded as a `BIT STRING`.

5) The `attributes` field shall include a validity period for the Diffie-Hellman keys using the `KeyValidityPeriod` attribute defined in section 15 of IETF RFC 7906 [5].

For elliptic curve groups, the `OneAsymmetricKey` fields shall follow the conventions specified in IETF RFC 5480 [6], IETF RFC 5915 [7] and IETF RFC 8410 [13].

EXAMPLE 1:

```
object identifier:   { 1 3 132 1 12 }
parameter encoding:  ECParameters
private key encoding:    ECPrivateKey
public key encoding: ECPoint
```

For finite field groups, the `OneAsymmetricKey` fields shall follow the conventions specified in IETF RFC 3279 [8].

EXAMPLE 2:

```
object identifier:   { 1 2 840 10046 2 1 }
parameter encoding:  DomainParameters
private key encoding:    INTEGER
public key encoding: INTEGER
```

When the package contains one or more certificates and corresponding signature private keys, the certificates shall include `VisibilityInformation` as defined in clause 4.3.3, unless Annex A is being used.

For SIG elements, the element that is used to store the public and private keys in the Asymmetric Key Package shall be as defined in IETF RFC 5958 [4], and the `OneAsymmetricKey` fields are set as follows:

1) The `version` field shall be set to version 1 (integer value of 0).

2) The `privateKeyAlgorithm` field shall be set to the same `AlgorithmIdentifier` that appears in the `algorithm` field of the `SubjectPublicKeyInfo` element of the certificate as defined in Recommendation ITU-T X.509 [3].

3) The `privateKey` field shall be set to the signature private key, encoded as an `OCTET STRING`.

4) The `publicKey` field shall be absent.

5) The `attributes` field shall include the certificate, which includes the signature public key, as a `userCertificate` attribute as defined in section 8 of IETF RFC 7906 [5].

4.3.4.3.3 Protecting the key package

When the centrally managed keys mechanism described in clause 4.3.4.3 is used, Enterprise Network Security Asymmetric Key Packages shall be transferred between the key manager and key consumer using HTTP Over TLS [9], commonly called HTTPS.

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

4.3.4.3.4 Transferring keys

4.3.4.3.4.1 Protocol overview

The HTTP messages containing key packages shall comprise an HTTP header followed by the Asymmetric Key Package, encoded using Distinguished Encoding Rules (DER). The HTTP Content-Type header shall be set to `application/pkcs8` as defined in IETF RFC 5958 [4] for plaintext packages and set to `application/pkcs7-mime` as defined in IETF RFC 8551 [11] if the package is encapsulated using CMS.

4.3.4.3.4.2 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 the key consumer supports an HTTPS key installation service, the key consumer shall support receiving a key package via an HTTP PUT request to a request-target, given here in origin-form, of `/.well-known/enterprise-network-security/new-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 authenticate the key manager and determine that the key manager is authorized to provide keys, and it is the responsibility of the key manager to authenticate the key consumer and determine that the key consumer is authorized to receive these keys.

4.3.4.3.4.3 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 supported by the key manager, the key manager shall support retrieval of a key package using all of the following HTTP GET request formats.

For a key consumer other than an IPsec peer, such as a middlebox doing "always on" decryption, to retrieve keys from the key manager that were issued to IPsec peers, perhaps within a certain time range and/or limited to a specific peer:

```
GET /.well-known/enterprise-network-security/keys?from=[time]&to=[time]&
fqdn=[fqdn]
```

where:

a) `[time]` values shall be UTC times with one second resolution in Internet date/time format `date-time` per section 5.6 of IETF RFC 3339 [17], with no `time-secfrac` present and with a `time-offset` of `Z` or `z`. Per IETF RFC 3339 [17], both the `T` and `Z` should be upper case.

b) `from` may be present. If `from` is present, all longer-lived Diffie-Hellman public/private key pairs issued for use by IPsec peers at or after the specified time shall be returned, except as limited by the `to` and `fqdn` parameters. If `from` is not present, the key manager shall process the request as if it was specified to be the earliest issue time in the key manager's database.

c) `to` may be present. If `to` is present, all longer-lived Diffie-Hellman public/private key pairs issued for use by IPsec peers at or before the specified time shall be returned, except as limited by the `from` and `fqdn` parameters. If `to` is not present, the key manager shall process the request as if it was specified to be the latest issue time in the key manager's database.

d) `fqdn` may be present. If `fqdn` is present, its value, `[fqdn]`shall be a fully qualified domain name as defined in IETF RFC 1983 [i.7] and using the syntax defined in section 2.1 of IETF RFC 1123 [18]. The key manager shall limit the set of longer-lived Diffie-Hellman public/private key pairs it returns to those in its database that were issued to the IPsec peer having the given fully qualified domain name.

e) The key manager shall return an appropriate HTTP error code, as defined in section 6 of IETF RFC 7231 [12], if there is not at least one matching longer-lived Diffie-Hellman public/private key pair.

NOTE: Key consumers using this request with time parameters need to take into account potential clock skew between their local clock and the key manager's clock as well as the possibility that the key manager issues new keys to IPsec peers in the same UTC second specified in one of the request's time parameters, but after the request is processed.

EXAMPLE 1: A key consumer requests all keys in the key manager's database:

```
GET /.well-known/enterprise-network-security/keys
Accept: application/pkcs8,application/pkcs7-mime
```

EXAMPLE 2: A key consumer requests all keys issued by the key manager since IETF RFC 5996 [i.8] was published:

```
GET /.well-known/enterprise-network-security/keys?
from=2010-09-14T22:23:58Z
Accept: application/pkcs8,application/pkcs7-mime
```

EXAMPLE 3: A key consumer requests all keys issued by the key manager before IETF RFC 6989 [15] was published:

```
GET /.well-known/enterprise-network-security/keys?
to=2013-07-25T15:24:34Z
Accept: application/pkcs8,application/pkcs7-mime
```

EXAMPLE 4: A key consumer requests all keys issued by the key manager in December 2016 to the IPsec peer ipsec-peer.example.com and only in CMS format:

```
GET /.well-known/enterprise-network-security/keys?
from=2016-12-01T00:00:00Z&to=2016-12-31T23:59:60Z&
fqdn=ipsec-peer.example.com
Accept: application/pkcs7-mime
```

For a key consumer other than an IPsec peer, such as an analytical tool performing after-the-fact decryption, to retrieve one or more specific keys from the key manager:

```
GET /.well-known/enterprise-network-security/keys?digest=[digest]&
fingerprints=[fingerprints]
```

where:

a) `digest` shall be present. The value, `[digest]`, indicates which digest algorithm was used to generate the provided fingerprints. The key manager shall support the following values: `sha256`. The key manager may support other values.

b) `fingerprints` shall be present, and its value, `[fingerprints]`, shall be a comma-separated list of the hexadecimal strings representing the fingerprints of the keying material being requested, where each of the strings in the list is a (possibly truncated) digest, computed using the digest algorithm corresponding to the request, of the longer-lived Diffie-Hellman public key. Digest value minimum size and method of truncation are as defined in clause 4.3.3. If `fingerprints` is not present, or if `fingerprints` is present but with an empty value, the key manager shall return an appropriate HTTP error code as defined in section 6 of IETF RFC 7231 [12].

c) The key manager shall return a key package that contains the corresponding longer-lived Diffie-Hellman public/private key pair for each fingerprint for which it has a record. In the very unlikely case that the key manager has more than one longer-lived Diffie-Hellman public/private key pair corresponding to a requested fingerprint, it shall return all matching longer-lived Diffie-Hellman public/private key pairs in the key package.

d) The key manager shall return an appropriate HTTP error code, as defined in section 6 of IETF RFC 7231 [12], if an unsupported digest algorithm is provided or if there is not at least one matching longer-lived Diffie-Hellman public/private key pair.

EXAMPLE 5: For a key consumer requesting two keys based on 80-bit SHA-256 fingerprints and supporting responses in either format, the HTTP GET request would be:

```
GET /.well-known/enterprise-network-security/keys?digest=sha256&
fingerprints=000102030405060708090,0908070605040302010 0
Accept: application/pkcs8,application/pkcs7-mime
```

For retrieval of new key material by the IPsec peer for use in subsequent IPsec sessions:

```
GET /.well-known/enterprise-network-security/new-keys?
ikegroups=[groups]&certs=[sigalgs]&visibilityInformation=[counts]&
fingerprints=[fingerprints]&digest=[digests]&fingerprintSize=[sizes]&
maxLifetime=[lifetimes]&context=[contextstr]
```

where:

a) `ikegroups` may be included. The value, `[groups]`, shall be a comma-separated list where each entry in the list is a Diffie-Hellman Group Transform ID defined in the IANA Internet Key Exchange Version 2 (IKEv2) Parameters registry [14], represented as a hexadecimal string, for which the associated longer-lived Diffie-Hellman public/private key pairs are being requested. The key manager shall return a key package containing a longer-lived Diffie-Hellman public/private key pair for each group listed in `[groups]`. If `ikegroups` is not included, then the number of keys generated and the group used for each is determined by the key manager.

b) `certs` may be included only if `ikegroups` is included. If `certs` is included, the value, `[sigalgs]`, shall be a comma-separated list where each entry is a colon-separated pair of compatible SignatureScheme values defined in section B.3.1.3 of IETF RFC 8446 [10], represented as a hexadecimal string, where the compatible values are those with names beginning with 'rsa_pkcs1_', 'ecdsa_', or 'dsa_sha1_'. The first value in the pair shall indicate the requested signature algorithm to be used by the certificate issuer to sign the certificate. The second value in the pair shall indicate the subject public key algorithm in the certificate. If `certs` is included, then for each entry in the list, the key consumer shall request one server certificate using that scheme. The key manager shall return a certificate, corresponding to all longer-lived Diffie-Hellman public/private key pair in the key package, for each given signature algorithm pair listed in `[sigalgs]` that it supports.

c) If `certs` is included, `visibilityInformation` shall be included and the value `[counts]` shall be a comma-separated list of integers whose sum is less than or equal to the number of list entries present in the value of the `ikegroups` parameter. Each list entry in the value of `visibilityInformation` indicates a `VisibilityInformation` structure that shall be added to each generated certificate, and the value of each list entry indicates the number of generated keys that will be represented by the corresponding `VisibilityInformation` structure. If Annex A is in use, the value -1 may appear as the sole list entry, in which case no `VisibilityInformation` structures shall be added to the generated certificates. Otherwise, all list entry values shall be non-negative, and the value zero shall not appear more than once. As all keys are required to correspond to one `VisibilityInformation` structure, if the sum of the entries in `visibilityInformation` is less than the number of entries in `ikegroups`, then one of the entries in `visibilityInformation` shall be zero, resulting in a `VisibilityInformation` structure with no key bindings. Generated keys are associated with the indicated `VisibilityInformation` structure(s) in the order of declaration. That is, if the first entry in `visibilityInformation` is `m`, then the first `m` keys indicated in `ikegroups` are associated with the first `VisibilityInformation` structure, and if the second entry in `visibilityInformation` is `n`, then the next `n` keys indicated in `ikegroups` are associated with the second `VisibilityInformation` structure and so on. If `certs` is not included, `visibilityInformation` shall be ignored.

d) If `certs` is included, `fingerprints` may be included and, if present, the value `[fingerprints]` shall be a comma-separated list of `yes` or `no` values, one for each entry in the value of `visibilityInformation`. A value of `yes` indicates that fingerprints will be included in the corresponding `VisibilityInformation` structure, and a value of `no` indicates that fingerprints will not be included in the corresponding `VisibilityInformation` structure. A `fingerprints` list entry corresponding to an entry in `visibilityInformation` whose value is zero or -1 shall be ignored. As only one `VisibiltyInformation` structure without `keyBindings` may be present, if the values specified in the `visibilityInformation` and `fingerprints` parameters would result in multiple `VisibilityInformation` structures without `keyBindings`, the key manager shall return an error. If `fingerprints` is not included, the key manager shall include fingerprints in each `VisibilityInformation` structure corresponding to a positive entry in `visibilityInformation`.

e) If `certs` is included, `digest` may be included and, if present, the value `[digests]` shall be a comma-separated list of digest names, one for each entry in the value of `visibilityInformation`, each indicating which digest algorithm to use to generate the fingerprints included in the corresponding `VisibilityInformation` structure. A `digest` list entry corresponding to an entry in `visibilityInformation` whose value is zero or -1 shall be ignored. The key manager shall support the following digest names: `sha256`. The key manager may support other values. If `digest` is not included, the key manager shall determine via other means which digest algorithm to use to generate the fingerprints for each `VisibilityInformation` structure that includes fingerprints.

f) If `certs` is included `fingerprintSize` may be included and, if present, the value `[sizes]` shall be a comma-separated list of integers in the range of valid sizes for the `fingerprintSize` member of the `KeyBindings` data structure defined in clause 4.3.3, each indicating the number of octets to truncate the fingerprints to in the corresponding `VisibilityInformation` structure using the procedure described in that clause. A `fingerprintSize` list entry corresponding to an entry in `visibilityInformation` whose value is zero or -1 shall be ignored. If `fingerprintSize` is not included, the key manager shall determine via other means the size of the fingerprints included in each `VisibilityInformation` structure that includes fingerprints.

g) If `certs` is included, `maxLifetime` may be included, and if present, the value `[lifetimes]` shall be a comma-separated list of integers in the valid range of the `maxLifetime` member of the `VisibilityInformation` data structure defined in clause 4.3.3, each indicating the maximum lifetime of the keys associated with the corresponding `VisibilityInformation` structure. If `maxLifetime` is not included, the key manager shall determine via other means whether to include a `maxLifetime` member in each `VisibilityInformation` structure, and if included, its value.

h) `context` may be included. If `context` is included, the value, `[contextstr]`, is a free string that the key manager shall use to determine the longer-lived Diffie-Hellman public/private key pairs and certificate contents to return. The structure of `[contextstr]` is not specified.

i) If any group in `[groups]` is not supported by the key manager, or if an unsupported value (that is not otherwise ignored) is provided for any of the `certs`, `visibilityInformation`, `fingerprints`, `digest`, `fingerprintSize`, or `maxLifetime` parameters, the key manager shall return an appropriate HTTP error code as defined in section 6 of IETF RFC 7231 [12]. If the key manager is unable to use the value provided for `context`, the key manager may return an appropriate HTTP error code, which shall be as defined in clause 6 of IETF RFC 7231 [12]. Other ways to handle the error are outside the scope of the present document.

j) If `ikegroups` is included but neither `certs` nor `context` are included, then the key manager will not provide any certificates.

EXAMPLE 6: A key consumer requests new key material, leaving the number of keys and the group used for each to be determined by the key manager. It is not possible for the key manager to provide certificates as part of the response in this case. The HTTP GET request would be:

```
GET /.well-known/enterprise-network-security/new-keys
Accept: application/pkcs8
```

EXAMPLE 7: A key consumer requests the generation of two keys, one that uses the 384-bit random ECP group and one that uses Curve 25519. The key consumer also requests two corresponding certificates bound to both keys via a single `VisibilityInformation` structure. One of the certificates is to be signed with `rsa_pkcs1_sha256` and to be associated with an `ecdsa_secp256r1_sha256` subject signing key. The other certificate is to be signed with `ecdsa_secp384r1_sha384` and to be associated with an `ecdsa_secp384r1_sha384` subject signing key. The key consumer leaves the choice of fingerprint inclusion, fingerprint digest algorithm and size, and maximum lifetime up to the key manager, and requests the resulting key package in only the `pkcs8` format. The HTTP GET request would be:

```
GET /.well-known/enterprise-network-security/new-keys?
ikegroups=0x0014,0x001f&certs=0x0401:0x403,0x0503:0x0503&
visibilityInformation=2
Accept: application/pkcs8
```

EXAMPLE 8: A key consumer requests the generation of six keys, four that use the 384-bit random ECP group and two that use Curve 25519. The key consumer also requests one corresponding certificate, bound to the keys via three `VisibilityInformation` structures, two of which each bind a pair of 384-bit random ECP group keys and one of which binds the two Curve 25519 keys. The certificate is to be signed with `ecdsa_secp384r1_sha384` and is to be associated with an `ecdsa_secp384r1_sha384` subject signing key. The key consumer explicitly requests fingerprint inclusion and specifies maximum lifetimes, but leaves the choice of fingerprint digest algorithm and size up to the key manager. The resulting key package is to be provided only in the `pkcs8` format. The HTTP GET request would be:

```
GET /.well-known/enterprise-network-security/new-keys?
ikegroups=0x0014,0x0014,0x0014,0x0014,0x001f,0x001f&
certs=0x0503:0x0503&visibilityInformation=2,2,2&
fingerprints=yes,yes,yes&maxLifetime=600,600,1200
Accept: application/pkcs8
```

EXAMPLE 9: As in EXAMPLE 8, except that the third `VisibilityInformation` structure is to have no key bindings (and thus the two Curve 25519 keys will be considered to correspond to it). The HTTP GET request would be:

```
GET /.well-known/enterprise-network-security/new-keys?
ikegroups=0x0014,0x0014,0x0014,0x0014,0x001f,0x001f&
certs=0x0503:0x0503&visibilityInformation=2,2,0&
fingerprints=yes,yes,no&maxLifetime=600,600,300
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 authenticate the key consumer and determine that the key consumer is authorized to receive these keys.

5 Security

The Enterprise Network Security profile does not provide all the security guarantees provided by IKEv2 [1] and ESP [i.2].

The ENS profile does not provide the same forward secrecy for IPsec Security Associations (SAs). Knowledge of a given longer-lived Diffie-Hellman private key can be used to decrypt all of the SAs with keying material that was established with that longer-lived Diffie-Hellman private key; however, forward secrecy for all of the protected packets begins when all copies of that longer-lived Diffie-Hellman private key have been destroyed.

It is possible that both sides of the IPsec tunnel could decide to use longer-lived Diffie-Hellman private keys. In this scenario, recovery of the longer-lived Diffie-Hellman private key or shared secret (g^ir as per IETF RFC 7296 [1]) from either IPsec peer would allow decryption of all traffic encrypted under the SAs that were established with the recovered private key or shared secret. Because of this possibility, the administrator(s) of the IPsec peers should rotate longer-lived Diffie-Hellman private keys frequently. If both IPsec peers are using their own longer-lived Diffie-Hellman private key, forward secrecy for all of the packets protected under a given SA begins when all copies of both longer-lived Diffie-Hellman private keys used to derive that SA have been destroyed.

The relaxation of forward secrecy is deliberate. With proper management of the longer-lived Diffie-Hellman private key, the ENS profile maintains a high-degree of authentication, integrity and confidentiality while also allowing security breaches to be rapidly detected. The ENS profile also maximizes service availability by enabling rapid detection of services and application problems. It is the responsibility of each organization deploying the ENS profile to evaluate the trade-off between the IPsec security guarantees and the operational visibility provided by this profile.

The variant of this profile described in Annex A meets the same IPsec security guarantees as the full ENS profile, with the same exception regarding forward security. In the Annex A variant, the certificate visibility information from clause 4.3.3 is not sent; however, this is compensated for by the operator of the IPsec peer having been informed by other means that the packets can be inspected.

When an organization uses IPsec to protect the traffic between an external IPsec peer and the enterprise network or data centre, and then uses the Enterprise Network Security profile to protect traffic inside the enterprise network or data centre (as in Figure 4.2), use of this profile shall be explicitly enabled via a configuration option.

When this profile is enabled, the applicable additional key material tests required by IETF RFC 6989 [15] for IKEv2 implementations that reuse Diffie-Hellman keys shall be implemented.

Annex A (normative): Middlebox visibility information variant

This annex is optional. It establishes a variant of the Enterprise Network Security profile that was defined in clause 4, which is the same in all respects except that the ENS peer sends no visibility information in the certificate; thus clause 4.3.3 is not carried out.

This variant of the Enterprise Network Security profile shall not be used unless the operator of the IPsec peer has been informed by some means that the packets can be inspected. The means by which the operator is informed is out of scope of the present document.

EXAMPLE 1: The IPsec peers are wholly within a private enterprise network and the operator of the IPsec peer has already been notified by a contract denoted condition of access to the network that packets can be inspected.

EXAMPLE 2: The operator of the IPsec peer agrees to an acceptable use policy to access a private enterprise network. This acceptable use policy states that packets can be inspected.

Annex B (informative): Requirements for an Enterprise Network Security aware IPsec peer

This annex is optional. An IPsec peer that satisfies the requirements specified in this annex can be designated as "Enterprise Network Security aware".

a) The IPsec peer may provide configuration means to accept all Enterprise Network Security connections or deny all Enterprise Network Security connections as indicated by the certificate extension defined in clause 4.3.3.

b) If the IPsec implementation provides a means of viewing an IPsec peer's certificate, it should correctly parse and display the contents of the certificate extension defined in clause 4.3.3.

c) The IPsec peer may provide a means to establish only Enterprise Network Security profile Security Associations (SAs) that match a whitelist.

d) The IPsec peer may provide a means to refuse to establish any Enterprise Network Security profile SAs that match a blacklist.

e) The IPsec peer may offer a user prompt to establish an SA in accordance with the Enterprise Network Security profile.

History

Document history		
V1.1.1	December 2021	Publication