[MS-IKEE]: Internet Key Exchange Protocol Extensions

Revision Summary

Date	Revision History	Revision Class	Comments
10/22/2006	0.01		MCPP Milestone 1 Initial Availability
01/19/2007	1.0		MCPP Milestone 1
03/02/2007	1.1		Monthly release
04/03/2007	1.2		Monthly release
05/11/2007	1.3		Monthly release
06/01/2007	1.3.1	Editorial	Revised and edited the technical content.
07/03/2007	2.0	Major	Updated and revised the technical content.
07/20/2007	2.0.1	Editorial	Revised and edited the technical content.
08/10/2007	3.0	Major	Updated and revised the technical content.
09/28/2007	3.0.1	Editorial	Revised and edited the technical content.
10/23/2007	3.0.2	Editorial	Revised and edited the technical content.
11/30/2007	3.0.3	Editorial	Revised and edited the technical content.
01/25/2008	4.0	Major	Updated and revised the technical content.
03/14/2008	4.0.1	Editorial	Revised and edited the technical content.
05/16/2008	4.0.2	Editorial	Revised and edited the technical content.
06/20/2008	5.0	Major	Updated and revised the technical content.
07/25/2008	6.0	Major	Updated and revised the technical content.
08/29/2008	6.1	Minor	Updated the technical content.
10/24/2008	6.2	Minor	Updated the technical content.
12/05/2008	7.0	Major	Updated and revised the technical content.
01/16/2009	8.0	Major	Updated and revised the technical content.
02/27/2009	9.0	Major	Updated and revised the technical content.
04/10/2009	10.0	Major	Updated and revised the technical content.
05/22/2009	11.0	Major	Updated and revised the technical content.
07/02/2009	12.0	Major	Updated and revised the technical content.
08/14/2009	12.1	Minor	Updated the technical content.

Date	Revision History	Revision Class	Comments
09/25/2009	12.2	Minor	Updated the technical content.
11/06/2009	13.0	Major	Updated and revised the technical content.
12/18/2009	13.1	Minor	Updated the technical content.
01/29/2010	14.0	Major	Updated and revised the technical content.
03/12/2010	15.0	Major	Updated and revised the technical content.
04/23/2010	16.0	Major	Updated and revised the technical content.
06/04/2010	17.0	Major	Updated and revised the technical content.
07/16/2010	18.0	Major	Significantly changed the technical content.
08/27/2010	18.0	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	18.0	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	18.0	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	18.1	Minor	Clarified the meaning of the technical content.
02/11/2011	18.1	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1 Introduction

This document specifies extensions to the Internet Key Exchange (IKE) Protocol versions 1 and 2, as specified in [RFC2407], [RFC2408], [RFC2409], [RFC3947], and [RFC4306]. These extensions provide additional capabilities to **IKE**, including interoperation between different revisions of the **network address translation** traversal (NAT-Traversal or NAT-T) specification, fragmentation of large IKE version 1 messages, authentication by using **cryptographically generated addresses (CGAs)**, fast failover when communicating with a **cluster** of hosts, easier interoperation with non-Internet Protocol security (IPsec)–capable peers, acknowledgment of **security association (SA)** deletion messages, denial of service protection, IKE security association correlation (IKEv2), and IKE server internal addresses configuration attributes (IKEv2).

1.1 Glossary

The following terms are defined in [MS-GLOS]:

Authenticated IP (AuthIP)
authentication header (AH)
certificate
certificate authority (CA)
cryptographic hash function
cryptographically generated address (CGA)
digital signature
domain of interpretation (DOI)
Encapsulating Security Payload (ESP)
exchange
exchange type
flow
Generic Security Services (GSS)
initiator
Internet Key Exchange (IKE)
Internet Protocol security (IPsec)
Internet Security Association and Key Management Protocol (ISAKMP)
ISAKMP payload
keying material
main mode (MM)
main mode security association (MM SA)
maximum transmission unit (MTU)
negotiation
negotiation discovery
network address translation (NAT)
nonce
phase
quick mode (QM)
quick mode security association (QM SA)
responder
root certificate
security association (SA)
security association database (SAD)
security policy database (SPD)
self-signed certificate
transport mode

tunnel mode
vendor ID payload

The following terms are specific to this document:

certificate chain: A sequence of **certificates** where each **certificate** in the chain is signed by the subsequent **certificate**. The last **certificate** in the chain is typically a **self-signed certificate**.

cluster: A group of hosts that can be accessed as though they are a single host. A **cluster** is generally accessed by using a virtual IP address. For more information, see [MSFT-WLBS].

main mode security association database (MMSAD): A database that contains the operational state for each **main mode security association (MM SA)**. See section 3.1.1 for details.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[ECP] Fu, D. and Solinas, J., "ECP Groups For IKE and IKEv2", September 2005, http://tools.ietf.org/id/draft-ietf-ipsec-ike-ecp-groups-02.txt

[GSS] Piper, D., and Swander, B., "A GSS-API Authentication Method for IKE", Internet Draft, July 2001, http://tools.ietf.org/html/draft-ietf-ipsec-isakmp-gss-auth-07

If you have any trouble finding [GSS], please check here.

[IANAIPSEC] Internet Assigned Numbers Authority, "Internet Key Exchange (IKE) Attributes", November 2006, http://www.iana.org/assignments/ipsec-registry

[IANAISAKMP] Internet Assigned Numbers Authority, "'Magic Numbers' for ISAKMP Protocol", October 2006, http://www.iana.org/assignments/isakmp-registry

[MS-AIPS] Microsoft Corporation, "Authenticated Internet Protocol Specification", January 2007.

[MS-ERREF] Microsoft Corporation, "Windows Error Codes", January 2007.

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, http://www.ietf.org/rfc/rfc768.txt

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

[RFC2401] Kent, S., and Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, November 1998, http://www.ietf.org/rfc/rfc2401.txt

[RFC2403] Madson, C., and Glenn, R., "The Use of HMAC-MD5-96 Within ESP and AH", RFC 2403, November 1998, http://www.ietf.org/rfc/rfc2403.txt

[RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation for ISAKMP", RFC 2407, November 1998, http://www.ietf.org/rfc/rfc2407.txt

[RFC2408] Maughan, D., Schertler, M., Schneider, M., and Turner, J., "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998, http://www.ietf.org/rfc/rfc2408.txt

[RFC2409] Harkins, D., and Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, November 1998, http://www.ietf.org/rfc/rfc2409.txt

[RFC2451] Pereira, R., and Adams, R., "The ESP CBC-Mode Cipher Algorithms", RFC 2451, November 1998, http://www.ietf.org/rfc/rfc2451.txt

[RFC3447] Jonsson, J., and Kaliski, B., "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003, http://www.ietf.org/rfc/rfc3447.txt

[RFC3526] Kivinen, T., and Kojo, M., "More Modular Exponential (MODP) Diffie-Hellman Groups for Internet Key Exchange (IKE)", RFC 3526, May 2003, http://www.ietf.org/rfc/rfc3526.txt

[RFC3947] Kivinen, T., Swander, B., Huttunen, A., and Volpe, V., "Negotiation of NAT-Traversal in the IKE", RFC 3947, January 2005, http://www.ietf.org/rfc/rfc3947.txt

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005, http://www.ietf.org/rfc/rfc3972.txt

[RFC4301] Kent, S., and Seo, K., "Security Architecture for the Internet Protocol", RFC 4301, December 2005, http://www.ietf.org/rfc/rfc4301.txt

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005, http://www.ietf.org/rfc/rfc4306.txt

[RFC4555] P. Eronen, Ed., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006, http://www.ietf.org/rfc/rfc4555.txt

1.2.2 Informative References

[DRAFT-NATT] Microsoft Corporation, "Negotiation of NAT-Traversal in the IKE", June 2002, http://tools.ietf.org/id/draft-ietf-ipsec-nat-t-ike-03.txt

[FIPS140] National Institute of Standards and Technology, "Federal Information Processing Standards Publication 140-2: Security Requirements for Cryptographic Modules", December 2002, http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MSFT-WLBS] Microsoft Corporation, "MS Windows NT Load Balancing Service (WLBS)", January 1999, http://www.microsoft.com/technet/archive/winntas/deploy/depovg/wlbsdepl.mspx?mfr=true

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981, http://www.ietf.org/rfc/rfc791.txt

[RFC2404] Madson, C., and Glenn, R., "The Use of HMAC-SHA-1-96 Within ESP and AH", RFC 2404, November 1998, http://www.ietf.org/rfc/rfc2404.txt

[RFC2405] Madson, C., and Doraswamy, N., "The ESP DES-CBC Cipher Algorithm With Explicit IV", RFC 2405, November 1998, http://www.ietf.org/rfc/rfc2405.txt

[RFC2410] Glenn, R., and Kent, S., "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998, http://www.ietf.org/rfc/rfc2410.txt

[RFC3602] Frankel, S., Glenn, R., and Kelly, S., "The AES-CBC Cipher Algorithm and Its Use with IPsec", RFC 3602, September 2003, http://www.ietf.org/rfc/rfc3602.txt

[RFC3715] Aboba, B., and Dixon, W., "IPsec-Network Address Translation (NAT) Compatibility Requirements", RFC 3715, March 2004, http://www.ietf.org/rfc/rfc3715.txt

[RFC3948] Huttunen, A., Swander, B., Volpe, V., et al., "UDP Encapsulation of IPsec ESP Packets", RFC 3948, January 2005, http://www.ietf.org/rfc/rfc3948.txt

[RFC4106] Viega, J., and McGrew, D., "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, June 2005, http://www.ietf.org/rfc/rfc4106.txt

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, December 2005, http://www.ietf.org/rfc/rfc4302.txt

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005, http://www.ietf.org/rfc/rfc4303.txt

[RFC4543] McGrew, D., and Viega, J., "The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH", RFC 4543, May 2006, http://www.ietf.org/rfc/rfc4543.txt

[RFC4621] Kivinen, T., and Tschofenig, H., "Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol", RFC 4621, August 2006, http://www.ietf.org/rfc/rfc4621.txt

[SHA256] National Institute of Standards and Technology, "FIPS 180-2, Secure Hash Standard (SHS)", August 2002, http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf

1.3 Overview

The Internet Key Exchange (IKE) Protocol version 1 is used to negotiate security associations (SAs), as specified in [RFC2409], for the purpose of keying **authentication header (AH)** and **Encapsulating Security Payload (ESP)** packet transformations. For more information, see [RFC4302] and [RFC4303], respectively. For the general security architecture of **IPsec**, see [RFC4301].

The IKE Protocol version 1 is specified in [RFC2409] and is closely tied to [RFC2407] and [RFC2408]. In addition, IKE is clearly the most commonly implemented protocol that uses [RFC2407] and [RFC2408]. Also, version 2 of the IKE protocol is specified by a single Request for Comments [RFC4306]. For these reasons, industry practice supports use of the term IKE to collectively refer to [RFC2407], [RFC2408], [RFC2409], and more recently, [RFC4306].

In the remainder of this document, the term IKE collectively applies to [RFC2407], [RFC2408], [RFC2409], and [RFC4306]. Where applicable, the appropriate section of each RFC is referenced in the document.

This document specifies the extensions to IKE. Each of these IKE extensions is independent and can be implemented in isolation. There is no sequencing between the individual extensions. An implementation of this protocol can support any combination of these IKE extensions.<1>

1.3.1 Network Address Translation Traversal (NAT-T)

In the original IPsec specifications, the interposition of network address translation (NAT) devices between IPsec peers prevents correct IPsec operation. For more information about the incompatibilities, see [RFC3715] section 2.

Two specifications have been defined to address these incompatibilities. For more information about the User Datagram Protocol (UDP) encapsulation of ESP packets, see [RFC3948]. UDP-encapsulated ESP packets are correctly translated by NAT devices. [RFC3947] specifies an IKE extension to detect the presence of NAT devices between two IPsec peers and to negotiate the use of a UDP-encapsulated ESP.

Network address translation traversal (NAT-T) **negotiation** for IKE was first published as an Internet draft before becoming [RFC3947]. In [DRAFT-NATT], the IKE parameter numbers for NAT-T negotiation are chosen from the appropriate private use ranges, as specified in [IANAISAKMP]. In specification [RFC3947], different IKE parameter numbers were assigned by the Internet Assigned Numbers Authority (IANA). As a result, a [DRAFT-NATT]-compliant implementation is incompatible with an [RFC3947]-compliant implementation. For more information, see [DRAFT-NATT].

The NAT-T extension specified in this document enables IKE implementations supporting NAT-T to negotiate the use of either the [DRAFT-NATT] or the [RFC3947] parameters. This specification does not extend the NAT-T protocol itself. It negotiates only the interpretation of the NAT-T IKE parameter numbers. Also, this document specifies the support of NAT-T IKE for IPsec transport mode only.

The extension negotiates the use of the [DRAFT-NATT] or [RFC3947] parameters as follows:

1. The host signals which revisions of the specification it supports (that is, [DRAFT-NATT], [RFC3947], or both) by sending **vendor ID payloads** ("RFC 3947" or "draft-ietf-ipsec-nat-t-ike-02\n") with its first IKE message. See section 1.7, Capability Negotiation.

2. On receipt of the first IKE message from the peer, the host looks up the vendor ID payloads to determine which revision of the NAT-T protocol to use. If both revisions are supported by both hosts, preference is given to [RFC3947] over [DRAFT-NATT].

For details, see section 3.2.

1.3.2 IKE Fragmentation

IKE uses UDP as a transport. IKE messages can be sufficiently large; so the underlying IP layer may fragment them, as described in [RFC791] section 2.3. This fragmentation typically happens with IKE messages that contain **certificate chains**. To avoid fragmentation-based attacks, fragmented UDP packets are commonly blocked by firewalls and routers. Blocking the fragmented UDP packets can lead to IKE failures that are especially difficult to diagnose. The IKE fragmentation extension that is specified in this document avoids fragmentation at the IP level by fragmenting IKE packets into smaller UDP packets that the underlying IP layer is guaranteed not to fragment.

Hosts that support IKE fragmentation advertise this capability through a "FRAGMENTATION" vendor ID payload; for more information, see section 1.7. If both peers support fragmentation, a fragmentation timer is started whenever a message is sent. If the timer expires, it is assumed that the message that is associated with the timer did not reach its destination because it was too large to traverse the intervening network. In this case, the message is split into several small fragments, and all these small fragments are sent.

So that the destination host can correctly reassemble the fragmented message, each fragment carries a fragment ID that is unique to the original message and a fragment number that is unique to the particular fragment. Fragment numbers range from 1 to N, where N is the number of fragments for a message.

Upon receipt of a fragment, the receiving host verifies whether it has already received other fragments for that fragment ID. If not, the receiving host starts a reassembly timer. It then verifies whether it has received all N fragments for the message, where the Nth fragment is indicated by a particular bit in the fragment. If the fragment reassembly timer expires before all fragments are correctly received, the receiving host must discard all fragments.

For details, see section 3.3.

1.3.3 Authentication Using a Cryptographically Generated Address

This extension specifies a new authentication method for IKE based on cryptographically generated addresses (CGAs), as specified in [RFC3972]. A CGA is an IPv6 address for which the interface identifier (that is, the low-order 64 bits) is generated by computing a **cryptographic hash function** of a public key.

Hosts that support CGA authentication advertise their capability through an "IKE CGA version 1" vendor ID payload. CGA authentication is negotiated as a regular IKE authentication method; see section 1.7, Capability Negotiation. The CGA verification that occurs during this authentication ensures that the remote peer has access to the private key that was used to generate the CGA. This CGA verification uses the corresponding public key and a parameters structure that contains information originally used to generate the CGA. The public key and parameters structure must, therefore, be sent to the host that verifies the CGA. The public key is transmitted within an IKE **certificate** payload, and the parameters structure is transmitted by using a new CGA identification payload as part of the IKE **main mode (MM)** negotiation. Successful validation of the CGA completes the IKE main mode negotiation.

For details, see section 3.4.

1.3.4 Fast Failover

This extension reduces the time required for a client to restore an IPsec security association (SA) to the virtual IP address for a cluster of hosts after a failure on one of the hosts that is sharing the virtual IP address.

The client uses a "Vid-Initial-Contact" vendor ID payload (see section 1.7, Capability Negotiation) to signal to the cluster that it does not have any **main mode security association (MM SA)** or **quick mode security association (QM SA)** established with the cluster so that the IKE session may be reallocated to a different node within the cluster. The server uses an "NLBS_PRESENT" vendor ID payload (see section 1.7, Capability Negotiation) to indicate to the client that the client should use a shorter **quick mode (QM)** idle timer. In this way, a new QM SA is renegotiated faster if a failover occurs.

For more information about clusters based on virtual IP addresses, see [MSFT-WLBS]. For specifications, see sections 3.5 and 3.6.

1.3.5 Negotiation Discovery

The Internet Key Exchange Protocol Extensions enables a client to determine whether a remote peer supports IPsec-protected communications.

Negotiation discovery introduces new IPsec policy options. In the case of outbound traffic, if the traffic matches a negotiation discovery policy, the host sends the packet in Cleartext and starts an IKE negotiation in parallel. If the remote peer is not IPsec-capable, the IKE negotiation eventually times out, and the connection stays in Cleartext. If the peer is IPsec-capable and the IKE negotiation eventually succeeds, the connection starts using the negotiated SA. To enforce that a once-secured **flow** can never downgrade back to Cleartext, this extension maintains a per-flow state table that is looked up for every packet.

In the case of inbound traffic, negotiation discovery supports a policy-specified boundary mode in which the host can accept both Cleartext and secured connections to allow inbound traffic from non-IPsec-capable hosts in addition to secure connections from IPsec-capable hosts. The flow state table determines if an incoming Cleartext packet should be accepted.

For details, see section 3.7.

1.3.6 Reliable Delete

This extension enables a peer to reliably confirm the deletion of a security association that is established with another peer. The original IKE specification does not require the acknowledgment of Delete payloads.

This capability is advertised through additional **ISAKMP payloads**. The standard IKE Delete message is sent with an additional **ISAKMP** Nonce payload (as specified in [RFC2408] section 3.13) appended. The host starts a retransmission timer when sending the Delete message. On receipt of the Delete message, the host constructs an acknowledgment message that contains an ISAKMP Nonce payload, an ISAKMP Delete payload, and the Message ID from the received Delete message in the ISAKMP header. On receipt of the acknowledgment message, the host verifies that the Message ID matches the Message ID that was sent with the Delete message. On expiration of the retransmission timer, the Delete message is retransmitted.

For details, see section 3.8.

1.3.7 Denial of Service Protection

A **responder** that implements the IKE protocol must create states for all correctly formed initial requests, even if the **initiator** is flooding the responder with packets from multiple incorrect IP addresses. The vulnerability to denial-of-service (DOS) attacks is mitigated if responders do not create any state until the peer can prove that it exists at a routable address.

This extension enables a responder to delay creating state until it has verified the following:

1. That the source of a message is not a spoofed IP address.

2. When a threshold of incoming requests has been reached.

For details, see section 3.9.

1.3.8 IKE/AuthIP Co-Existence

This extension allows two peers that are both IKEv1 and authenticated IP (AuthIP)-capable to negotiate the use of **AuthIP** over IKEv1. This extension is specified in [MS-AIPS] section 1.7 and also applies to IKE.<2>

1.3.9 IKE SA Correlation (IKEv2)

This extension allows two different IKEv2 IKE_SA to be correlated together. Assume that an IKE_SA has been established. This is called SAoriginal. At a later time, to ensure that the client credentials are still valid, but without tearing down the existing SA, a new IKE_SA (called SAcurrent) can be built to embed a new payload in this exchange that securely correlates this SA with the original SA.

1.3.10 IKE Server Internal Addresses Configuration Attributes (IKEv2)

This extension allows the IKEv2 client endpoint of an IPsec remote access client (IRAC), as specified in [RFC4306] section 2.19, to determine the internal IPv4 and IPv6 addresses of the IPsec remote access server (IRAS), as also specified in [RFC4306] section 2.19.

1.3.11 Extension to RFC Cross Reference

The following table summarizes how each IKE extension extends each of the applicable RFCs.

IKE extension	Extends [RFC2407]	Extends [RFC2408]	Extends [RFC2409]	Extends [RFC3947]	Extends [RFC4306]	IKE version
NAT-T transport mode only	(1)	(2) (3)		(7)		IKEv1
IKE fragmentation		(3)	(8)			IKEv1
CGA authentication	(4) (5)	(3)	(9)			IKEv1
Fast failover		(3)	(10)			IKEv1
Negotiation discovery		(3) (6)	(10)			IKEv1
Reliable delete			(11)			IKEv1
Denial of Service protection		(6)	(12)			IKEv1
IKE SA Correlation					(13)	IKEv2
Configuration Attribute					(14)	IKEv2

1. Adjunction of an encapsulation mode in the private range. Encapsulation mode is specified in [RFC2407] section 4.5.

2. Adjunction of a vendor ID. Vendor ID is as specified in [RFC2408] section 3.16.

3. Adjunction of Payload Types in the private range. Payload Types are specified in [RFC2408] section 3.1.

4. Adjunction of an authentication method within an ISAKMP SA payload, as specified in [RFC2407] section 4.6.1.

5. Adjunction of an identification type for an ISAKMP Identification payload from the private Identification Type range, as specified in [RFC2407] section 4.6.2.

6. Adjunction of a notify message type from the private range. The notify message types are specified in [RFC2408] section 3.14.1.

7. Negotiation of the interpretation of Payload Types and Encapsulation Modes.

8. Fragmentation and reassembly. Packet construction and decoding for IKE are specified in [RFC2409] section 5.

9. Extends the IKE phase 1 **exchange** using certificates. For more information, see [RFC2409] section 5.1.

10. Extends the IKE phase 1 exchange. For more information, see [RFC2409] section 5. Extends the QM SAs negotiation. For more information, see [RFC2409] section 5.5.

11. Extends the Notify exchange. For more information, see [RFC2409] section 5.7.

12. Extends the IKE phase 1 exchange. For more information, see [RFC2409] section 5.1.

13. This extension allows two different IKEv2 IKE_SA to be correlated together for the purpose of ensuring that the client credentials are still valid but without tearing down the existing SA. When validation is required, a new IKE_SA (called SAcurrent) can be built to embed a new payload in this exchange that securely correlates this SA with the original SA.

14. This extension allows the IKEv2 client endpoint of an IPsec remote access client (IRAC), as specified in [RFC4306], to determine the internal IPv4 and IPv6 addresses of the IPsec remote access server (IRAS), also as specified in [RFC4306].

1.4 Relationship to Other Protocols

IKE is used for the authentication and keying of IPsec SAs, as specified in [RFC4301] section 3. IKE relies on UDP as a transport, as specified in [RFC768].

1.5 Prerequisites/Preconditions

The following sections describe the prerequisites and preconditions for using the Internet Key Exchange (IKE) Protocol Extensions:

- General Prerequisites/Preconditions (section 1.5.1)
- CGA Authentication Prerequisites/Preconditions (section 1.5.2)

1.5.1 General Prerequisites/Preconditions

IKE assumes that both the initiator and the responder have an IP address and have UDP connectivity. IKE also assumes that the initiator knows the responder's IP address (for example, through manual configuration or through a policy lookup in the case of **tunnel mode**).

Successful establishment of a QM SA using IKEv1 requires that the initiator and the responder have at least one common authentication method and a common set of cryptographic parameters for the MM and the QM SAs. For authentication using certificates, each peer validates the remote peer certificate chain to a locally trusted **root certificate**, as specified in [RFC2409] section 5.1. For pre-shared key authentication, both peers are required to share the same pre-shared secret, as specified in [RFC2409] section 5.4.

1.5.2 CGA Authentication Prerequisites/Preconditions

For CGA authentication, as specified in [RFC3972] section 1, the peers must possess a CGA and the associated **self-signed certificate**.

1.6 Applicability Statement

- NAT-T applies when NAT devices between the IPsec peers can otherwise prevent the establishment of IPsec SAs.
- IKE fragmentation applies when intermediary devices in the path between the IPsec peers can drop fragmented UDP datagrams, that can prevent the establishment of IPsec SAs.
- Authentication using CGA applies when the IPsec peers do not share a common credential distribution infrastructure. CGA authentication allows such peers to verify that the remote peer has access to the public-private key pair used to generate the CGA. CGA authentication only applies to IPv6 addresses.
- Fast failover applies when IPsec clients connect to a cluster of hosts using IPsec, and it is necessary to minimize the amount of time required for a client to failover from one host in the cluster to another.
- Negotiation discovery applies when hosts communicate with both IPsec-aware and non-IPsec-aware devices, and it is necessary to minimize the amount of time required to detect IPsec-awareness on each peer.
- Reliable delete applies when a peer needs to reliably confirm the deletion of a security association (SA) established with another peer.
- IKEv2 SA Correlation applies when two different IKEv2 SAs need to be correlated.
- IKEv2 Server Internal Addresses Configuration Attributes apply when the client endpoint of an IPsec remote access client needs to determine the internal IPv4 and IPv6 addresses of the IPsec remote access server.

1.7 Versioning and Capability Negotiation

This section covers versioning issues in the following areas:

- **Protocol Versions:** The protocol version is part of the ISAKMP header. IKEv1 uses protocol version 1.0, as specified in [RFC2408] section 3.1. IKEv2 uses protocol version 2.0, as specified in [RFC4306] section 3.1.
- **Security and Authentication Methods:** IKE supports multiple authentication and encryption algorithms for both the MM SAs and QM SAs, as specified in [RFC2408] section 5.6. IKE supports the negotiation of the authentication method, the Diffie-Hellman group, and the hashing and authentication algorithm using [RFC2409], [GSS], or [RFC3972].<3>
- **Cryptographic Parameters:** Cryptographic parameters are negotiated in different **phases** of the protocol (that is, initial exchange, MM, and QM, as specified in [RFC2409] section 5). Details about algorithm and parameter numbers are specified in [IANAIPSEC] and [IANAISAKMP].<4>
- **Capability Negotiation:** IKE can advertise specific capabilities through vendor ID payloads, as specified in [RFC2408] section 3.16.<5>

1.8 Vendor-Extensible Fields

The IKE extensions specified in this document do not introduce any new vendor-extensible fields. These extensions inherit the extensibility features of ISAKMP (as specified in [RFC2408]) and IKE (as specified in [RFC2409]).

1.9 Standards Assignments

No standards assignments have been received for the IKE extensions described in this document. All values used in these extensions are in private ranges, as specified in [IANAIPSEC] and [IANAISAKMP].

2 Messages

2.1 Transport

IKE messages MUST be transported over ISAKMP, as specified in [RFC2408], which uses UDP port 500 by default. IKE MUST run over ports 500 and 4500 if a NAT has been detected, as specified in [RFC3947] section 3.2; otherwise, it MAY be run over a different port.<6>

All fields are sent and encoded in network order unless otherwise specified.

2.2 Message Syntax

2.2.1 NAT-T Payload Types

Each ISAKMP message consists of a header and a variable number of payloads, each identified by a 1-octet payload type value in its Payload Type field, as specified in [RFC2408] section 3.1. NAT-T adds two new payload types: NAT Discovery (NAT-D) and NAT Original Address (NAT-OA). The payload type values for these payload types are specified in [RFC3947]. For more information about an alternative set of payload type values, see [DRAFT-NATT].<7>

The following table describes the NAT-D payload type.

NAT Discovery (NAT-D) payload type value	Revision
0x82	[DRAFT-NATT]
0x14	[RFC3947]

The following table describes the supported NAT-OA payload type.

Supported NAT Original Address (NAT-OA) payload type	Revision
0x83	[DRAFT-NATT]
0x15	[RFC3947]

2.2.2 NAT-T UDP Encapsulation Modes

The Encapsulation Mode field is located in the SA payload, as specified in [RFC2407] section 4.5. Specification [RFC3947] introduces new encapsulation mode values for this field. For more information about an alternative set of these values, see [DRAFT-NATT].<8>

The following table describes the UDP-Encapsulated-Tunnel values.

UDP-Encapsulated-Tunnel	Revision
0xF003	[DRAFT-NATT]
0x0003	[RFC3947]

The following table describes the UDP-Encapsulated-Transport values.

UDP-Encapsulated-Transport	Revision
0xF004	[DRAFT-NATT]
0x0004	[RFC3947]

2.2.3 IKE Message Fragment

An IKE message fragment contains:

- An ISAKMP header, as specified in [RFC2408] section 3.1.
- A single, non-encrypted, Fragment payload.

2.2.3.1 Fragment Payload Packet

The Fragment Payload is an ISAKMP payload, as specified in [RFC2408] section 3.1. The Payload Type value for a Fragment payload is 0x84 from the private payload type range, as specified in [RFC2408] section 3.1. A Fragment payload MUST be preceded by an ISAKMP header that has this payload type.

The following illustration describes the Fragment Payload packet.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="8">Next_Payload</td><td colspan="8">RESERVED</td><td colspan="16">Payload_Length</td></tr>
<tr><td colspan="16">Fragment_ID</td><td colspan="8">Fragment_Number</td><td colspan="8">Flags</td></tr>
<tr><td colspan="32">Fragment_Data (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Next_Payload (1 byte): Identifier for the payload type, which MUST specify the next payload in the message. For a Fragment payload, this field MUST be set to 0.

RESERVED (1 byte): This field MUST be set to zero. The responder MUST ignore this field on receipt. This behavior is identical to IKE.

Payload_Length (2 bytes): This field MUST be the length, in bytes, of the payload, including the generic payload header. This is identical to IKE.

Fragment_ID (2 bytes): The **Fragment ID** field is 2 bytes that MUST specify the same value for every fragment that is generated from a particular IKE message.

Fragment_Number (1 byte): The **Fragment Number** field MUST indicate the order in which the fragments are sent. The first fragment MUST have a fragment number of 1, and each subsequent fragment MUST have a fragment number that is one greater than that of the previous fragment. Because the maximum size of an IKE message is limited to 64 KB by UDP and fragments are aligned on the minimum MTU for IPv4 and IPv6, the fragment number cannot wrap.

Flags (1 byte): The flag field MUST have the following value.

Value	Meaning
LAST_FRAGMENT 0x01	This flag indicates the last fragment in the message.

All other bits of the Flags field MUST be set to zero on the initiator and ignored on the responder. For more details on flag semantics, see section 3.1.

Fragment_Data (variable): The **Fragment Data** field MUST contain the fragment. The size of the **Fragment Data** field MUST be computed by subtracting the size of the Fragment Payload header (8 bytes) from the value of the **Payload Length** field.

2.2.4 AUTH_CGA Authentication Method Packet

AUTH_CGA is an authentication method within an ISAKMP SA payload, as specified in [RFC2407] section 4.6.1. The format of the SA payload is the following, as specified in [RFC2408] section 3.4.

- A number of Proposal payloads, as specified in [RFC2408] section 3.5.
- Within each Proposal payload, there is a number of Transform payloads, as specified in [RFC2408] section 3.6.
- Within each Proposal payload, there is a number of Data Attributes payloads, as specified in [RFC2408] section 3.3. In a Data Attribute payload, an authentication method is indicated by the value 0x0003 in the Attribute Type field of the Data Attribute payload, as specified in [RFC2409] Appendix A. The particular authentication method is determined by the value of the Attribute Value field, as specified in [RFC2409] Appendix A.

The Data Attribute payload for the AUTH_CGA Authentication method has the format seen in the following AUTH_CGA packet.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
One	Attribute_Type															Attribute_Value															

One (1 bit): This field MUST be set to 1.

Attribute_Type (15 bits): For the AUTH_CGA authentication method, this field MUST be set to the value 0x0003. This value corresponds to the authentication method, as specified in [RFC2409] Appendix A.

Attribute_Value (2 bytes): For the AUTH_CGA authentication method, this field MUST be set to the value 0xFDED in network order. This value is from the private authentication method range, as specified in [RFC2409] Appendix A.

2.2.5 ID_IPV6_CGA Identification Type Packet

ID_IPV6_CGA is an identification type for an ISAKMP Identification payload, as specified in [RFC2407] section 4.6.2. The ID_IPV6_CGA Identification Type is 0xFA from the private Identification Type range, as specified in [IANAISAKMP].

The format of the Identification payload for an ID_IPV6_CGA identification type is seen in the following packet.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="8">Next_Payload</td><td colspan="8">RESERVED</td><td colspan="16">Payload_Length</td></tr>
<tr><td colspan="8">Identification_Type</td><td colspan="8">Protocol_ID</td><td colspan="16">Port</td></tr>
<tr><td colspan="32">Modifier</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="8">Collision_Count</td><td colspan="24">Extension_fields (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Next_Payload (1 byte): This field is the identifier for the payload type of the next payload in the message. This field MUST be identical to the corresponding IKE field.

RESERVED (1 byte): This field MUST be set to zero. The responder MUST ignore this field on receipt. This behavior is identical to IKE.

Payload_Length (2 bytes): This field MUST be the length in bytes of the payload, including the Generic Payload header. This is identical to IKE.

Identification_Type (1 byte): This field is the value describing how the fields after the Port field should be interpreted. The ID_IPV6_CGA identification type MUST be 0xFA, from the private Identification Type range, as specified in [IANAISAKMP].

Protocol_ID (1 byte): This field MUST be set to zero. The responder MUST ignore this field on receipt. This is identical to IKE.

Port (2 bytes): This field MUST be set to zero. The responder MUST ignore this field on receipt. This is identical to IKE.

Modifier (16 bytes): This field MUST be as specified in [RFC3972] section 3.

Collision_Count (1 byte): This field MUST be as specified in [RFC3972] section 3.

Extension_fields (variable): This field MUST be as specified in [RFC3972] section 3.

2.2.6 Notify Payload Packet

The Notify Payload packet is specified in [RFC2408] section 3.14. The format is as follows.

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>20</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>30</th><th>1</th></tr>
<tr><td colspan="8">Next_Payload</td><td colspan="8">RESERVED</td><td colspan="16">Payload_Length</td></tr>
<tr><td colspan="32">Domain_of_Interpretation</td></tr>
<tr><td colspan="8">Protocol-ID</td><td colspan="8">SPI_size</td><td colspan="16">Notify_Message_Type</td></tr>
<tr><td colspan="32">Security_Parameter_Index (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Notification_Data (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Next_Payload (1 byte): This field MUST be as specified in [RFC2408] section 3.14.

RESERVED (1 byte): This field MUST be as specified in [RFC2408] section 3.14.

Payload_Length (2 bytes): This field MUST be as specified in [RFC2408] section 3.14.

Domain_of_Interpretation (4 bytes): The domain of interpretation (DOI) field MUST be set to 1 (IPSEC_DOI) as specified in [RFC2408] section A.2.

Protocol-ID (1 byte): This field MUST be as specified in [RFC2408] section 3.14.

SPI_size (1 byte): This field MUST be as specified in [RFC2408] section 3.14. The **SPI_size** is updated to a value of 8 when the Message ID is appended to the notification data as described in this section under **Notification_Data**.

Notify_Message_Type (2 bytes): This MUST identify the type of notification being sent with this message, in network byte order. The notify message types MUST be one of the following values, which are from the private range, as specified in [RFC2408] section 3.14.1.

Value	Meaning
0x9C43	NOTIFY_STATUS (check) This notify message type is a status code indicating the failure to establish a security association (SA) with a peer.
0x9C44	NOTIFY_DOS_COOKIE (check) This notify message type is used by the DOS protection extension.
0x9C45	EXCHANGE_INFO This notify message type is used by the negotiation discovery extension.

Security_Parameter_Index (variable): This is the Security Parameter Index (SPI) of size SPI_size. This field MUST be as specified in [RFC2408] section 3.14.

Notification_Data (variable): The content of this field depends on the **Notify_Message_Type** field. The following list describes field content for various notify message types. If the peer has previously sent the Vendor Id "MS NT5 ISAKMPOAKLEY" as specified in the footnote regarding Capability Negotiation in section 1.7, and the notify corresponds to the quick mode exchange, then the Message ID (in network order) of the quick mode is appended as the first 4 bytes of the notification data. In particular, the NOTIFY_DOS_COOKIE will never have the Message ID in the notification data because that is always a main mode operation. The EXCHANGE_INFO notify will always have the Message ID appended if the peer sends the above vendor ID. The NOTIFY_STATUS will only have the Message ID appended if the failure is a quick mode failure.

Field content MUST correspond to the **Notify_Message_Type** as follows:

- NOTIFY_STATUS (4 Bytes): MUST be a status code indicating failure. The values transmitted as status codes are implementation-specific.<9>

- NOTIFY_DOS_COOKIE (8 Bytes): MUST be the responder cookie value.

- EXCHANGE_INFO (4 Bytes): The flag values MUST be one of the following values.

Value	Meaning
0x00000001	IKE_EXCHANGE_INFO_ND_BOUNDARY This flag is used by the negotiation discovery extension.
0x00000002	IKE_EXCHANGE_INFO_GUARANTEE_ENCRYPTION This flag is used by the negotiation discovery extension.

2.2.7 Notify Payload (IKEv2) Packet

The Notify Payload packet is specified in [RFC4306] section 3.10. The format is as follows.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>2
0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>3
0</td><td>1</td></tr>
<tr><td colspan="8">Protocol-ID</td><td colspan="8">SPI_size</td><td colspan="16">Notify_Message_Type</td></tr>
<tr><td colspan="32">SPI</td></tr>
<tr><td colspan="32">Notification_Data (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Protocol-ID (1 byte): This field MUST be as specified in [RFC4306] section 3.10.

SPI_size (1 byte): This field MUST be as specified in [RFC4306] section 3.10.

Notify_Message_Type (2 bytes): This MUST identify the type of notification being sent with this message, in network byte order. The notify message types MUST be one of the following values, which are from the private error range, as specified in [RFC4306] section 3.10.1.

Value	Meaning
0x3039	Notify status. This notify message type is used to tell the peer of a private failure reason.

SPI (4 bytes): The Security Parameter Index (SPI) field MUST be as specified in [RFC4306] section 3.10.

Notification_Data (variable): The content of this field depends on the **Notify_Message_Type** field. The following list describes field content for various notify message types. Field content MUST correspond to the notify message type as follows:

- NOTIFY_STATUS (4 bytes): MUST be a status code indicating failure. The values transmitted as status codes are implementation specific.<10>

2.2.8 Configuration Attribute (IKEv2) Packet

The Configuration Attribute packet is specified in [RFC4306] section 3.15.1. The format is as follows.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td>R</td><td colspan="15">Attribute Type</td><td colspan="16">Length</td></tr>
<tr><td colspan="32">Value (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

R (1 bit): This reserved field MUST be as specified in [RFC4306] section 3.15.1.

Attribute Type (15 bits): This field MUST be as specified in [RFC4306] section 3.15.1.

Length (2 bytes): The length of the data in the value field.

Value (variable): The internal IPv4 or IPv6 address of the server.

Two additional **Attribute Types** from the private-use range are defined as follows.

Attribute type	Length (bytes)	Value
INTERNAL_IP4_SERVER 0x5BA0	4	The internal IPv4 address of the server.
INTERNAL_IP6_SERVER 0x5BA1	16	The internal IPv6 address of the server.

2.2.9 Correlation Payload (IKEv2) Packet

The Correlation Payload (IKEv2) packet format is as follows. There are two IKE_SAs here, SAcurrent and SAoriginal. This payload is sent under the protection of SACurrent. The Payload Type value for a Correlation payload is 0xc8 from the private payload type range, as specified in [RFC4306] section 3.2.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Next_Payload								RESERVED								Payload_Length															
IKE_SA_Initiator_SPI																															
...																															
IKE_SA_Responder_SPI																															
...																															
Correlation_Hash (variable)																															
...																															

Next_Payload (1 byte): This field MUST be as specified in [RFC2408] section 3.2.

RESERVED (1 byte): This field MUST be as specified in [RFC2408] section 3.2.

Payload_Length (2 bytes): This field MUST be as specified in [RFC2408] section 3.2.

IKE_SA_Initiator_SPI (8 bytes): This MUST be set to the initiator's spi from the IKE_SA being correlated, SAoriginal. This value is taken from the IKEv2 header of the prior IKE_SA, as specified in [RFC4306] section 3.1.

IKE_SA_Responder_SPI (8 bytes): This MUST be set to the responder's spi from the IKE_SA being correlated, SAoriginal. This value is taken from the IKEv2 header of the prior IKE_SA, as specified in [RFC4306] section 3.1.

Correlation_Hash (variable): This computes a keyed hash using the SAcurrent's negotiated PRF function. The key used is the SK_ai on the initiator and the SK_ar for the responder from SAoriginal. See [RFC4306] section 2.14. The correlation hash is as follows.

```
prf(SK_a(i or r),
SAcurrent.InitiatorSpi|SAcurrent.ResponderSpi|SAoriginal.InitiatorSpi|SAoriginal.re
sponderSpi)
```

3 Protocol Details

The following sections specify protocol details, including abstract data models and message processing rules, that are common and that are specific to NAT-T, IKE fragmentation, CGAs, the fast-failover client, the fast-failover server, negotiation discovery, reliable delete, IKE SA correlation (IKEv2), IKE Server Internal Addresses Configuration Attributes (IKEv2), and DOS protection.

3.1 Common Details

This section documents deviations from "The Internet IP Security Domain of Interpretation for ISAKMP", as specified in [RFC2407]; "Internet Security Association and Key Management Protocol (ISAKMP)", as specified in [RFC2408]; "The Internet Key Exchange (IKE)", as specified in [RFC2409]; "Internet Key Exchange (IKEv2) Protocol", as specified in [RFC4306]; and "Negotiation of NAT-Traversal in the IKE", as specified in [RFC3947]. These deviations affect each of these RFC standards as described in the table in section 1.3.11.

The flags bit semantics used by this document are as follows: for a flag, its "value" signifies a mask which, when its bitwise logical AND with the flags field is computed, yields either a zero value (all zero bits) if the flag is unset (set to FALSE), and a nonzero value otherwise. For example, a flag mask/value of 0x01 signifies that the bitwise logical AND of a single-byte flag field with 0x01 is zero if and only if the flag is set to FALSE. Assuming no other flag masks/values for this field, then, both 0x00 and 0x01 are valid values for this single-byte flag field: the former corresponding to the flag being unset, and the latter to the flag being set.

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol in addition to what is specified in [RFC2407], [RFC2408], [RFC2409], [RFC3947], and [RFC4301] for IKEv1, or [RFC4306] for IKEv2. The described organization is provided to explain how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with the behavior described in this document.

The following main data elements are required by any implementation:

- **Main mode security association database (MMSAD)**: A database that contains the operational state for each MM SA. The entry for each MM SA contains the following data elements.

  For each IKE MM SA, the following information MUST be maintained:

  - All states that are necessary for managing a standard IKE MM SA as defined in [RFC2409] appendix A for IKEv1 and [RFC4306] section 3.3.2 for IKEv2.

  - All states that are necessary for management of other IKE extensions for the SA, as specified in this section and in sections 3.2.1, 3.3.1, 3.4.1, 3.5.1, 3.6.1, 3.7.1, 3.8.1 for IKEv1 only, and 3.10.1 for IKEv2 only.

  The MMSAD MUST be indexed by the local and peer IP addresses and the initiator and responder cookies found in the ISAKMP header, as specified in [RFC2408].

- Peer authorization database (PAD): The PAD and its management operations are specified in [RFC4301] section 4.4.3. This specification does not extend that definition. The PAD that is referred to in this specification contains rules that describe if and how IKE should negotiate SAs with a remote peer, as specified in [RFC4301].

All states that are necessary for the management of IKE extensions are described in section 3.4.1 for IKEv1 only.

The PAD MUST be looked up by using tuples that are composed of local and remote IP addresses.

- Security policy database (SPD): The SPD and its management operations are specified in [RFC4301] section 4.4.1. The SPD that is referred to in this specification contains rules that describe if and how IPsec protection is applied to inbound or outbound IP traffic. The SPD MUST be looked up by using tuples that are composed of flow information (that is, source and destination IP addresses, port numbers, and protocol) for the packet.

  All states that are necessary for management of IKE extensions are described in section 3.7.1 for IKEv1 only.

- Security association database (SAD): The SAD contains the parameters of each QM SA. The SAD and its management operations are specified in [RFC4301] section 4.4.2.

  All states that are necessary for management of IKE extensions are described in section 3.7.1 for IKEv1 only.

- Flow state table: A table that contains the following information for each flow (indexed by the flow source and destination addresses, port numbers, and protocol).

  Entries in the Flow table MUST be updated upon sending and receiving packets.

  All states that are necessary for management of IKE extensions are described in section 3.7.1 for IKEv1 only.

  - Other states: Additional states are defined in section 3.9.1 and section 3.11.1.

Note The preceding conceptual data can be implemented by using a variety of techniques. Any data structure that stores the preceding conceptual data may be used in the implementation.

3.1.2 Timers

None beyond what is specified in [RFC2407], [RFC2408], [RFC2409], [RFC3947], or [RFC4306].

3.1.3 Initialization

None beyond what is specified in [RFC2407], [RFC2408], [RFC2409], [RFC3947], or [RFC4306] .

3.1.4 Higher-Layer Triggered Events

None except what is specified in [RFC2407], [RFC2408], [RFC2409], [RFC3947], or [RFC4306].

3.1.5 Message Processing Events and Sequencing Rules

[RFC2407]: Message processing MUST be as specified in [RFC2407] with the following exceptions:

- [RFC2407] section 4.5.2: "If conflicting attributes are detected, an ATTRIBUTES-NOT-SUPPORTED Notification Payload SHOULD be returned and the security association setup MUST be aborted."

  The IKE variant specified by this document MUST NOT terminate the SA setup when it encounters an unknown attribute.

- [RFC2407] section 4.5.3: "If an implementation receives a defined IPSEC DOI attribute (or attribute value) that it does not support, an ATTRIBUTES-NOT-SUPPORTED SHOULD be sent and the security association setup MUST be aborted, unless the attribute value is in the reserved range."

  The IKE variant specified by this document MUST NOT terminate the SA setup when it encounters an unknown attribute.

- [RFC2407] section 4.5.3: "Notification Status Messages MUST be sent under the protection of an ISAKMP SA, either as a payload in the last main mode exchange; in a separate informational exchange after main mode or aggressive mode processing is complete; or as a payload in any quick mode exchange."

  The IKE variant specified by this document SHOULD send notifications unprotected by an SA, without the hash payload, as specified in [RFC2409] section 5.7, if the notify occurs during the first two round trips of main mode. If the notify occurs in the last round trip of main mode, then this notify SHOULD be protected by the SA.<11>

[RFC2408]: Message processing MUST be as specified in [RFC2408] with the following exceptions:

- [RFC2408] section 3.9: "The certificate payload MUST be accepted at any point during an exchange."

  The IKE variant specified by this document MUST NOT accept certificate payloads at any time; a certificate payload MUST be in a message that contains an ID payload.

- [RFC2408] section 5.1: "When transmitting an ISAKMP message, the transmitting entity (initiator or responder) MUST do the following: 1. Set a timer and initialize a retry counter."

  The IKE variant timer specified by this document does not set a retransmission timer in the following cases:

  - The responder never sets a retransmission timer.
  - A notify message is sent to a peer.
  - A delete message is sent to a peer that does not support reliable deletes, that is, a peer that has not sent the Microsoft Implementation Vendor ID.

[RFC2409]: Message processing MUST be as specified in [RFC2409].

[RFC3947]: Message processing MUST be as specified in [RFC3947] with the following exceptions:

- [RFC3947] section 5.2: "In the case of **transport mode**, both ends MUST send both original initiator and responder addresses to the other end" and "The initiator MUST send the payloads if it proposes any UDP-Encapsulated-Transport mode, and the responder MUST send the payload only if it selected UDP-Encapsulated-Transport mode."

  The IKE variant specified by this document MUST send the NAT-OA if the host is behind a NAT.

- [RFC4306] section 2.7: "This hierarchical structure was designed to efficiently encode proposals for cryptographic suites when the number of supported suites is large because multiple values are acceptable for multiple transforms. The responder MUST choose a single suite, which MAY be any subset of the SA proposal following the rules below:"

  The responder MUST consult its SPD and loop through the SPD entries, comparing each SPD entry in turn with all the proposal suites from the peer. If a match is found from the list of proposal suites, the responder MUST accept that proposal suite. This MUST repeat until a match is found, or policy comparison, and the negotiation fails.

- [RFC4306] section 3.12: "Writers of Internet-Drafts who wish to extend this protocol MUST define a Vendor ID payload to announce the ability to implement the extension in the Internet-Draft."

The IKE variant specified by this document does not define a Vendor ID to announce the implementation of CFG attributes described in section 3.11.

3.1.6 Timer Events

None beyond what is specified in [RFC2407], [RFC2408] , [RFC2409], [RFC3947], or [RFC4306].

3.1.7 Other Local Events

None beyond what is specified in [RFC2407], [RFC2408], [RFC2409], [RFC3947], or [RFC4306].

3.2 NAT Traversal Details

Using the notation specified in [RFC2409] section 3.2, the generalized form of an IKE phase 1 exchange that uses NAT-T is as shown in the following figure and as specified in [RFC3947] section 3.2.

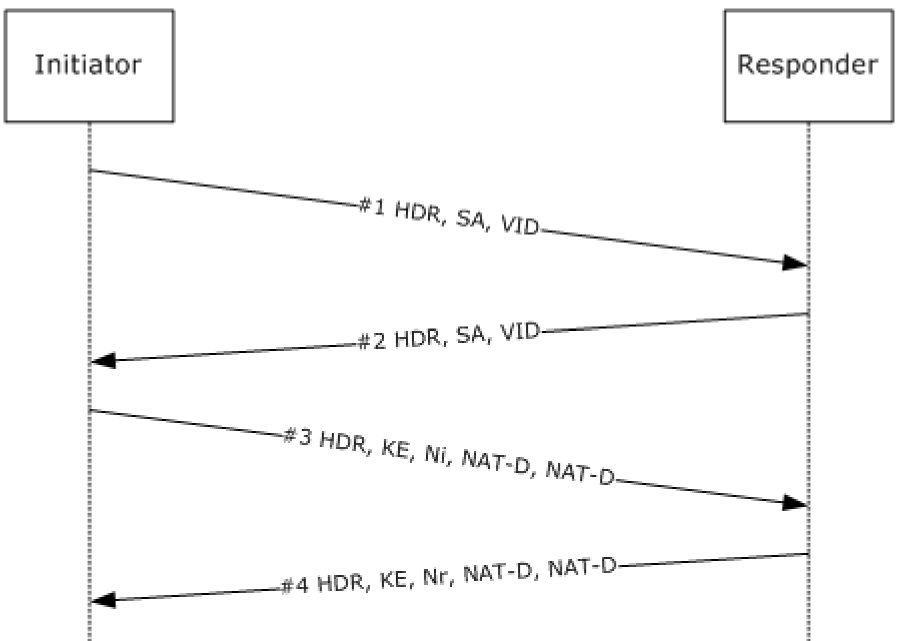

Figure 1: IKE phase 1 exchange using NAT-T

The description in this section uses the message numbers from the protocol sequence diagram.

The IKE NAT Traversal Protocol extension exists in two revisions. The [RFC3947] revision is specified in [RFC3947]. The [DRAFT-NATT] revision is identical to the [RFC3947] revision, except that the values used for the types defined in sections 2.2.1 and 2.2.2 are those that are specified in [DRAFT-NATT], instead of those that are specified in [RFC3947]. Both revisions include the negotiation of a choice of revision supported by both peers.<12> For more information, see [DRAFT-NATT].

3.2.1 Abstract Data Model

When this extension is implemented, the following additional state should be maintained. This is an extension to IKE Protocol version 1 as specified in [RFC2409].

Main mode security association database (MMSAD): The entry for each MM SA contains the following specific data element for NAT-T:

- Selected Revision: A flag that MUST specify what revision of the NAT-T protocol extension (as specified in [RFC3947]) has been selected for this MM SA. For more information, see [DRAFT-NATT].

3.2.2 Timers

The NAT-T keep-alive timer (per MM SA) is as specified in [RFC3948] section 4.<13>

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

3.2.4.1 Start of an IKE MM SA Negotiation

As part of the construction of message #1 for a new MM SA negotiation (as specified in [RFC2409] section 5), a NAT-T supporting host MUST include with its first IKE message extra vendor ID payloads (as specified in [RFC2408] section 3.16) to advertise its NAT-T revision support (as specified in [RFC3947] section 3.1). If the host supports only [DRAFT-NATT], it MUST include only the vendor ID "draft-ietf-ipsec-nat-t-ike-02\n" within message #1. If it supports only [RFC3947], it MUST include only the vendor ID "RFC 3947" within message #1. If it supports both [DRAFT-NATT] and [RFC3947], it MUST include both vendor IDs "draft-ietf-ipsec-nat-t-ike-02\n" and "RFC 3947" within message #1.<14>

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Receiving Message #1

On receipt of message #1, a NAT-T supporting host MUST check for the presence of the NAT-T vendor ID payloads that are specified in section 3.2.4.1. If NAT-T vendor ID payloads are present in the message, the host MUST set the Selected Revision for the corresponding MMSAD entry according to the following rules:

- If both hosts support [RFC3947] and [DRAFT-NATT], the host MUST set the Selected Revision to [RFC3947]. For more information, see [DRAFT-NATT].
- If both hosts share only one common revision, the host MUST set the Selected Revision to the common revision.
- If the hosts do not share a common revision, the host MUST ignore the payload.

Then, the host MUST construct message #2 (as specified in [RFC2409] section 5) and add vendor ID payloads that advertise its NAT-T capabilities, setting the values of those payloads exactly as it would if it were constructing IKE message #1. For details, see section 3.2.4.

3.2.5.2 Receiving Message #2

On receipt of message #2, the host MUST check for the presence of NAT-T vendor ID payloads and set the Selected Revision as specified in section 3.2.5.1.

3.2.5.3 Receiving Other Messages

As specified in [RFC3947] section 5.2, NAT-OA payloads can be sent within the first two QM messages. On receipt of the first or second QM message, the host MUST use the Selected Revision flag of the SA's corresponding entry in the MMSAD to interpret the payload type, as defined in section 2.2.1.

A UDP Encapsulation type can be negotiated through the SA payload, as specified in [RFC3947] section 5.1. On receipt of an IKE message that may contain an SA payload, the host MUST use the Selected Revision flag of the SA's corresponding entry in the MMSAD to interpret the Encapsulation Type, as defined in section 2.2.2.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 IKE Fragmentation Details

Using the notation as specified in [RFC2409] section 3.2, the generalized form of an IKE phase 1 exchange that is authenticated with signatures is as shown in the following figure, as a fragmentation example. For more information, see [RFC2409] section 5.

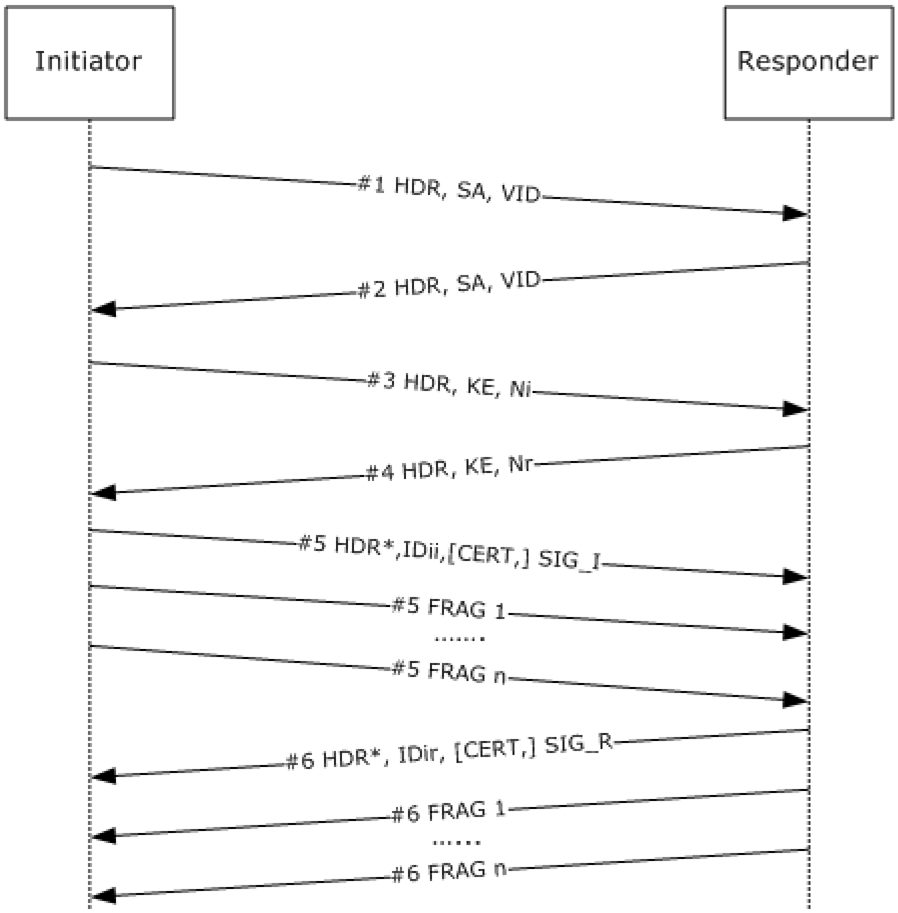

Figure 2: IKE phase 1 exchange

The description in this section uses the message numbers from the protocol sequence diagram.

3.3.1 Abstract Data Model

When this extension is implemented, the following additional state should be maintained. This is an extension to IKE Protocol version 1 as specified in [RFC2409].

Main mode security association database (MMSAD): The entry for each MM SA contains the following IKE fragmentation–specific data elements.

- Fragmentation supported: A flag that MUST be set if the peer supports receiving fragmented messages.
- Fragmentation active: A flag that MUST be set if the IKE messages MUST be fragmented.
- Fragmentation determination: The fragmentation need is determined by the firing of the fragmentation timer. See section 3.3.2 and the associated endnotes for more details. After determining that fragmentation is needed, the chosen MTU MUST be the minimum **MTU** for the protocol, which is 576 bytes for IPv4 and 1280 bytes for IPv6.
- Fragment queue: A queue holding the fragments that correspond to incomplete IKE messages, indexed by the Fragment ID. Each entry in the queue MUST contain:

- The Fragment ID.
- The Fragment Number.
- A Flag that indicates if this fragment is the last one (that is, the Last Fragment bit is set in the Fragment payload).
- The Fragment Data.

For definitions of the previous values, see section 2.2.3.1.

Flow state table: The following information MUST be maintained.

- Fragment ID counter: MUST be maintained and MUST be a 16 bit number. A Fragment ID counter SHOULD be implemented as a global counter.

3.3.2 Timers

IKE fragmentation uses the following timers:

- Fragmentation timer (for each IKE message): This timer MUST trigger fragmentation. The fragmentation timer MUST be started after sending each IKE message. The expiration of the fragmentation timer MUST indicate that the message should be fragmented the next time it is retransmitted. There MUST be one fragmentation timer per MM SA. The fragmentation timer must fire within the retransmission duration of the IKE negotiation and MUST be between 1 and 5 seconds. <15>
- Fragment reassembly timer (for each Fragment ID value): This timer MUST trigger the discarding of all the fragments received for this message. The fragment reassembly timer MUST be started when a fragment payload is received and the timer has not been started for the corresponding Fragment ID value. When the fragmentation reassembly timer fires, the delay MUST NOT exceed 90 seconds.<16>

3.3.3 Initialization

The Fragment ID counter MUST be set to zero.

3.3.4 Higher-Layer Triggered Events

3.3.4.1 Start of an IKE MM SA Negotiation

As part of the construction of message #1 for a new MM SA negotiation (as specified in [RFC2409] section 5), an IKE fragmentation-supporting host MUST include a "FRAGMENTATION" vendor ID payload (that is, a vendor ID payload that is generated by using the Vendor ID string "FRAGMENTATION", as specified in [RFC2408] section 3.16) to advertise its fragmentation capability.

3.3.5 Message Processing Events and Sequencing Rules

3.3.5.1 Receiving Message #1

On receipt of message #1, the host MUST check for the presence of a "FRAGMENTATION" vendor ID payload. If a "FRAGMENTATION" vendor ID payload is present in the message, the host MUST set the Fragmentation supported flag for the corresponding MMSAD entry.

Then, the host MUST construct message #2 (as specified in [RFC2409] section 5) and add the "FRAGMENTATION" vendor ID payload to advertise its fragmentation capability.

3.3.5.2 Receiving Message #2

On receipt of message #2, the host MUST check for the presence of a "FRAGMENTATION" vendor ID payload and set the Fragmentation supported flag, as specified in section 3.3.5.1.

3.3.5.3 Receiving Other IKE Messages

On receipt of an IKE message, the host MUST check if the message contains a Fragment payload. If a Fragment payload is present, this payload MUST be the only payload in the message. If not, the host MUST silently discard the message.

On receipt of a Fragment payload, the host MUST:

- Retrieve the Fragment ID from the Fragment payload.
- Start a fragmentation reassembly timer for this Fragment ID if no fragments are currently queued for this Fragment ID.
- If the queue for this Fragment ID already contains a fragment with the same Fragment number, the host MUST silently discard the message. If not, the host MUST queue the Fragment payload's fields in the corresponding entry of the MMSAD, indexed by the Fragment ID.

In addition, the host SHOULD set the Fragmentation active flag in the corresponding MMSAD entry.<17>

The host MUST then check whether all Fragment payloads for this Fragment ID have been received (that is, whether Fragment payloads that have a Fragment number from 1 to n have been received, and fragment n has the Last Fragment flag set).

The host MUST silently discard all Fragment payloads for this Fragment ID if any of the following error conditions occur:

- More than one Fragment payload has the Last Fragment flag set.
- A Fragment payload has been received with a Fragment number greater than the Fragment number of the fragment with the Last Fragment flag set.

If all Fragment payloads for a Fragment ID have been received, the host MUST construct the reassembled message by concatenating the following:

- The ISAKMP header from the first fragment.
- Fragment payloads (without the Fragment payload header) in the order of their Fragment number.

The host MUST then stop the fragment reassembly timer and process the reassembled IKE message as a typical message.

If the received message is a response to a previously sent message, the host MUST clear the fragmentation timer for the previously sent message.

If the processing of the IKE message results in the host sending a message, and the Fragmentation active flag is set for the corresponding MM SA, the host SHOULD fragment this message following the steps specified in section 3.3.6.1. If the Fragmentation active flag is not set, the host MUST start the fragmentation timer for the message it is about to send.<18>

3.3.6 Timer Events

3.3.6.1 Expiration of Fragmentation Timer

When the fragmentation timer expires, the host SHOULD start fragmenting the message that caused the timer to start. Note that the host does not need to buffer every message for fragmentation purposes because the IKE protocol has provisions for regenerating lost messages.<19>

The fragments MUST be constructed as follows:

- The Fragment ID counter is incremented.
- The IKE message is split into "n" fragments that are numbered 1 to n; the size of each fragment (after adding IP, UDP, and ISAKMP headers) is 576 bytes for IPv4 and 1,280 bytes for IPv6; however, the last fragment, which contains the remainder of the message, may be smaller.
- IKE does not adjust packet size based on router MTU advertisement; it continues to send packets for IPv4 (576 bytes) and IPv6 (1,280 byes). Therefore, IP-level fragmentation is possible in this case.
- For each fragment, a message MUST be constructed as follows:
  - The ISAKMP header of the original IKE message has the Next Payload field set to the Fragment payload and the Encrypted flag cleared (as specified in [RFC2408] section 3.1).
  - The Fragment payload header has the Fragment ID set to the current value of the Fragment ID counter, the Fragment number set to the current Fragment number, and the Last Fragment flag set to Fragment number n.

The fragments MUST be sent back-to-back to the peer.

The only messages that IKE fragments are those that contain the Identification payload, as specified in [RFC2408] section 3.8.

3.3.6.2 Expiration of the Fragment Reassembly Timer

When the fragment reassembly timer expires, the host MUST silently discard all the fragments currently queued under the Fragment ID of the Fragment payload whose receipt caused the timer to start.

3.3.7 Other Local Events

None.

3.4 CGA Authentication Details

Using the notation as specified in [RFC2409] section 3.2, the generalized form of an IKE phase 1 exchange using certificates is as shown in the following figure. For more information, see [RFC2409] section 5.1.

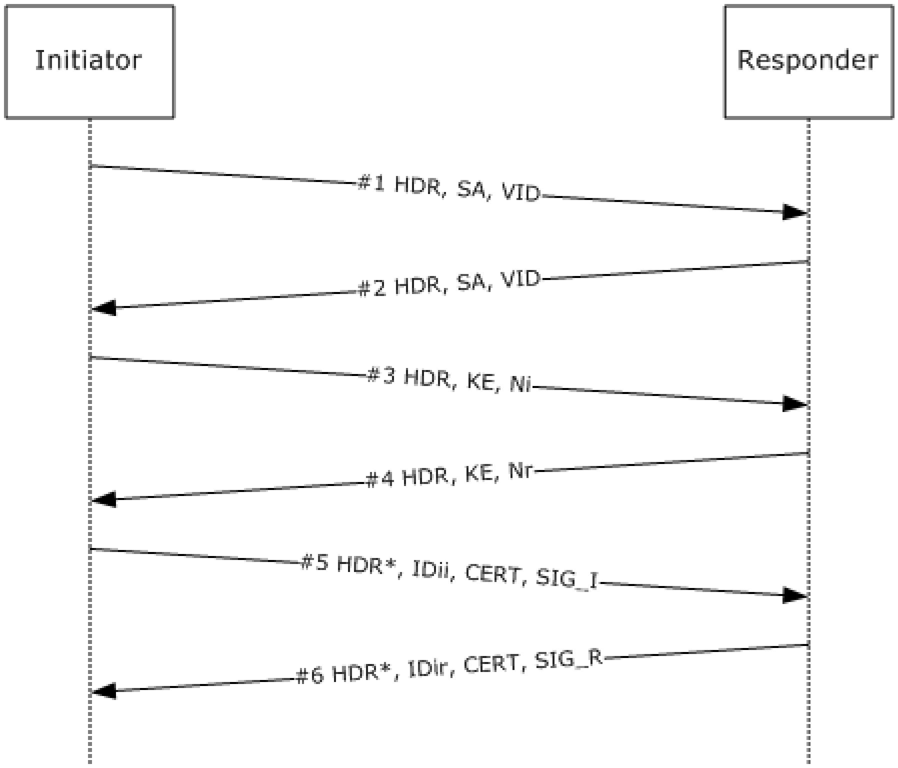

Figure 3: IKE phase 1 exchange using certificates

The CGA Authentication Protocol extension uses the same exchanges as an IKE phase 1 certificate exchange. The description in this section uses the message numbers from the protocol sequence diagram above.

The ID_IPV6_CGA identification type packet (section 2.2.5) does not contain the subnet. The subnet is determined by using the following algorithm.

1. Compare the first 4 bytes of the CGA address to a well-known prefix—0x3f, 0xfe, 0x83, 0x1e—to get the prefix length. If the values match, the prefix length is equal to 88 bits; otherwise, the prefix length is 64 bits.

2. Using the prefix length, the subnet is determined by taking the leftmost number of bits equal to the prefix length from the CGA address in the packet from the peer.

3.4.1 Abstract Data Model

When this extension is implemented, the following additional state should be maintained. This is an extension to IKE Protocol version 1 as specified in [RFC2409].

Main mode security association database (MMSAD): The entry for each MM SA contains the following CGA authentication–specific data elements:

- CGA_CAPABLE: A flag that indicates if the authentication type 0xFDED MUST be interpreted as the AUTH_CGA authentication method.

Peer authorization database (PAD): The following information MUST be maintained:

- A new valid value AUTH_CGA that identifies the CGA authentication method, added to the locally-configurable list of acceptable authentication methods.
- A new CGA ID data structure to hold the following parameters:
  - Modifier: size: 16 octets, type: unsigned integer. See [RFC3972] section 3.
  - Subnet Prefix: size: 8 octets, type: IPv6 subnet. See [RFC3972] section 3.
  - Collision Count: size: 1 octet, type: unsigned integer. See [RFC3972] section 3.
  - Public Key: size: variable, type: cryptographic key. See [RFC3972] section 3.
- A self-signed certificate (type X.509) compatible with the IKE exchange. See [RFC2409] section 5.1.

This data structure is used during:

- Generation of a CGA and its associated self-signed certificate (see section 3.4.3).
- Construction of an identity payload (see section 3.4.5.4).
- Verification of its association with a public key (see section 3.4.5.5).

3.4.2 Timers

None.

3.4.3 Initialization

Each host configured to use CGA authentication MUST generate an RSA public/private key pair (see [RFC3447] section 3 and [RFC3972] section 3). The host MUST then generate a self-signed X.509 certificate that uses this key pair and is compatible with IKE (see [RFC2409] section 5.1).

The CGA itself MUST be created as described in [RFC3972] section 4. This IP address is used to send and receive the IKE packets described in section 3.4.5.

3.4.4 Higher-Layer Triggered Events

3.4.4.1 Start of an IKE MM SA Negotiation

As part of the construction of message #1, a CGA authentication-supporting host MUST include an "IKE CGA version 1" vendor ID payload (that is, a vendor ID payload generated by using the vendor ID string "IKE CGA version 1", as specified in [RFC2408] section 3.16) to advertise its CGA authentication capability.

If the PAD requires CGA authentication, the host MUST include the AUTH_CGA Authentication method in its SA payload, as specified in section 2.2.4.

The host MUST use its CGA to communicate with the peer for this negotiation.

3.4.5 Message Processing Events and Sequencing Rules

3.4.5.1 Receiving Message #1

On receipt of message #1, a CGA authentication-supporting host MUST check for the presence of the "IKE CGA version 1" vendor ID payload. If an "IKE CGA version 1" vendor ID payload is present in message #1, the host MUST set the CGA_CAPABLE flag for the corresponding MMSAD entry.

The host MUST then look up its PAD to select one of the transforms that the peer proposes, as specified in [RFC2408] section 5.4.

If the host selects the proposed AUTH_CGA authentication method defined in section 3.4.1, the host MUST construct message #2, as specified in [RFC2409] section 5.1, and add an "IKE CGA version 1" vendor ID payload to advertise its CGA authentication capability.

The host MUST also use its CGA to communicate with the peer for this negotiation.

3.4.5.2 Receiving Message #2

On receipt of message #2, the host MUST check whether the proposal that the peer selected contains the AUTH_CGA authentication method defined in section 3.4.1. The host then MUST construct message #3, as specified in [RFC2409] section 5.1.

3.4.5.3 Receiving Message #3

Processing MUST be identical to that specified in [RFC2409] section 5.1.

3.4.5.4 Receiving Message #4

Processing MUST be identical to that specified in [RFC2409] section 5.1.

The host MUST then construct message #5, as specified in [RFC2409] section 5.1, with the following differences:

- The Identity payload MUST have the Identification type ID_IPV6_CGA and contain the identification data that corresponds to the host CGA (for details, see section 2.2.5). The ID IPV6 CGA fields are read from the CGA ID (see section 3.4.1).
- The CERT payload MUST contain the self-signed certificate that corresponds to the CGA.

3.4.5.5 Receiving Message #5

On receipt of message #5, the host MUST validate the message in the following ways:

- Use the SIG_I payload to verify the signature, as specified in [RFC2409] section 5.1. A successful verification proves that the peer has access to the private key that corresponds to the self-signed certificate passed in the CERT payload of message #5.
- Retrieve the CGA parameter structure (that is, Modifier, Collision Count, and Extension Fields) from the ID_IPV6_CGA Identity payload (for details, see section 2.2.4).
- Verify that the public key contained in the self-signed certificate and the parameter structure were used to generate the peer CGA, as specified in [RFC3972] section 5.

If an error is encountered during payload processing, or the CGA cannot be validated, the host MUST fail the negotiation, as specified in [RFC2408] section 5.

Then, the host MUST construct message #6 by using the procedure for constructing message #5, as specified in section 3.4.5.4.

3.4.5.6 Receiving Message #6

On receipt of message #6, the host MUST validate the message using the procedure specified for validating message #5 in section 3.4.5.5.

3.4.6 Timer Events

None.

3.4.7 Other Local Events

None.

3.5 Fast Failover Client Details

Using the notation as specified in [RFC2409] section 3.2, the generalized form of an IKE phase 1 exchange is as shown in the following figure. For more information, see [RFC2409] section 5.

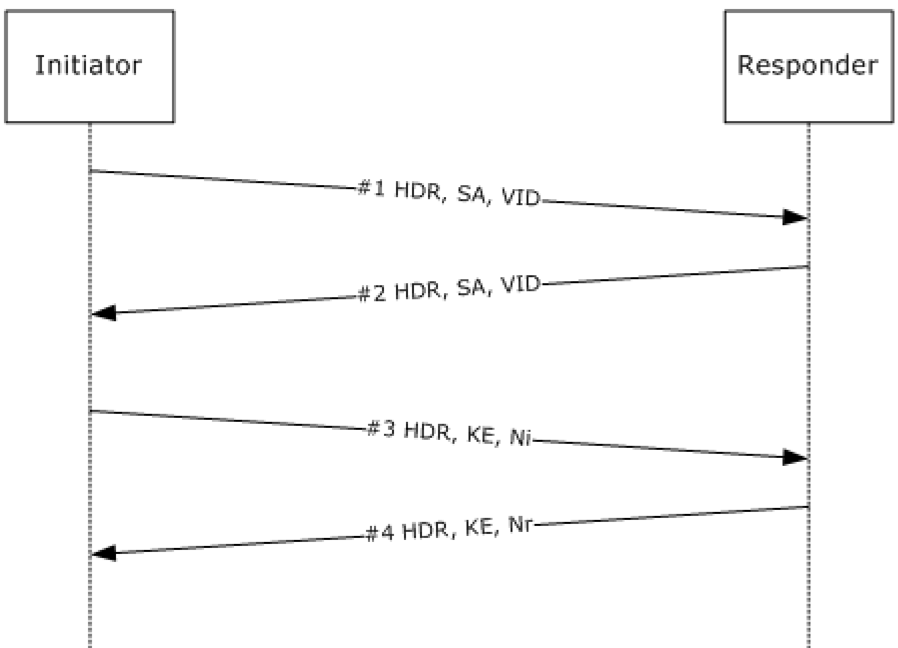

Figure 4: IKE phase 1 exchange

The description in this section uses the message numbers from the protocol sequence diagram.

3.5.1 Abstract Data Model

When this extension is implemented, the following additional state should be maintained. This is an extension to IKE Protocol version 1 as specified in [RFC2409].

Main mode security association database (MMSAD): The entry for each MM SA contains the following fast-failover client-specific data elements:

- Fast Failover: A flag that indicates that the "NLBS_PRESENT" vendor ID was received from the peer for this MM SA. For more details, see section 3.6.4.1.

3.5.2 Timers

QM SA idle timer (for each QM SA): This timer controls the inactivity time before the QM SA can be deleted (as specified in section 3.5.7.1). This timer MUST be set when the QM SA has been negotiated. The QM SA idle timer is 1 minute if the peer has sent an "NLBS_PRESENT" Vendor ID payload during the negotiation of the MM SA under which this QM SA was negotiated (as specified in section 3.6.4.1). Otherwise, the QM SA idle timer is 5 minutes.

3.5.3 Initialization

None.

3.5.4 Higher-Layer Triggered Events

3.5.4.1 Start of an IKE MM SA Negotiation

As part of the construction of message #1 for a new MM SA negotiation (as specified in [RFC2409] section 5), a fast failover-supporting host MUST include a "Vid-Initial-Contact" vendor ID payload (that is, a vendor ID payload that is generated using the vendor ID string "Vid-Initial-Contact", as specified in [RFC2408] section 3.16) if the host does not have any active MM SAs to the peer. This is determined by looking up the MMSAD using the peer IP address.

In addition, the host MAY also add the "Vid-Initial-Contact" vendor ID payload to message #1 if it has no open TCP connections to the peer and if new connection attempts cause the retransmission of SYN packets.<20>

3.5.5 Message Processing Events and Sequencing Rules

3.5.5.1 Receiving Message #1

On receipt of message #1, a fast failover-supporting host MUST check for the presence of the "NLBS_PRESENT" vendor ID (as specified in section 3.6.4.1). If the "NLBS_PRESENT" vendor ID payload is present in the message, the host MUST set the Fast Failover flag for the corresponding MMSAD entry.

If no errors are found, the host MUST construct message #2 in response. The host MUST add the "Vid-Initial-Contact" Vendor ID payload to message #2 under the conditions that are specified in section 3.5.4.1. Otherwise, the host MUST silently ignore the packet.

3.5.5.2 Receiving Message #2

On receipt of message #2, the host MUST check for the presence of the "NLBS_PRESENT" vendor ID (for details, see section 3.6.4.1). If the "NLBS_PRESENT" vendor ID payload is present in the message, the host MUST set the Fast Failover flag for the corresponding MMSAD entry.

3.5.6 Timer Events

3.5.6.1 Expiration of the QM SA Idle Timer

Upon expiration of the QM SA idle timer, the host MUST delete all states for the corresponding QM SA in the SAD.

3.5.7 Other Local Events

3.5.7.1 Successful Negotiation of a QM SA

QM SAs MUST be negotiated as specified in [RFC2409] section 5.5. Upon successful negotiation of a QM SA, the host MAY set the QM SA idle timer to a lower value than the default value if the Fast Failover flag is set on the corresponding MM SA.<21>

3.6 Fast Failover Server Details

The description in this section uses the message numbers from the protocol sequence diagram in section 3.5.

3.6.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to explain how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behaviors are consistent with what is described in this document. This is an extension to IKE Protocol version 1 as specified in [RFC2409].

The data elements any implementation requires include the following:

- Main mode security association database (MMSAD):

  For each MM SA (as specified in [RFC2409]), the following information MUST be maintained:

  - All IKE states necessary for managing an IKE MM SA, without extensions.
  - All states necessary for managing other IKE extensions for the SA, as specified in sections 3.1.1 and 3.6.1.
  - Initial Contact: A flag indicating if the "Vid-Initial-Contact" Vendor ID payload (see section 3.5.4.1) has been received for the MM SA.

  The MMSAD MUST be indexed by the local and peer IP addresses and the initiator and responder cookies found in the ISAKMP header (as specified in [RFC2408]).

Note The preceding conceptual data can be implemented by using a variety of techniques. An implementation is at liberty to implement such data in any way it pleases.

3.6.2 Timers

None.

3.6.3 Initialization

None.

3.6.4 Higher-Layer Triggered Events

3.6.4.1 Start of an IKE MM SA Negotiation

As part of the construction of message #1, a fast failover-supporting host MUST include an "NLBS_PRESENT" vendor ID payload (that is, a vendor ID payload generated by using the vendor ID string "NLBS_PRESENT", as specified in [RFC2408] section 3.16).

3.6.5 Message Processing Events and Sequencing Rules

3.6.5.1 Receiving Message #1

On receipt of message #1, the host MUST check for the presence of the "Vid-Initial-Contact" vendor ID (as specified in section 3.5.4.1). If the "Vid-Initial-Contact" vendor ID payload is present in the message, the host MUST set the Initial Contact flag for the corresponding MMSAD entry.

If the host is part of a cluster, it MAY use this information to rebalance the MM SA to a different host within the cluster.<22>

3.6.5.2 Receiving Message #2

Message #2 has the same processing as message #1.

3.6.6 Timer Events

None.

3.6.7 Other Local Events

None.

3.7 Negotiation Discovery Details

Using the notation as specified in [RFC2409] section 3.2, the generalized form of an IKE phase 1 (MM) exchange is as shown in the following figure. For more information, see [RFC2409] section 5.

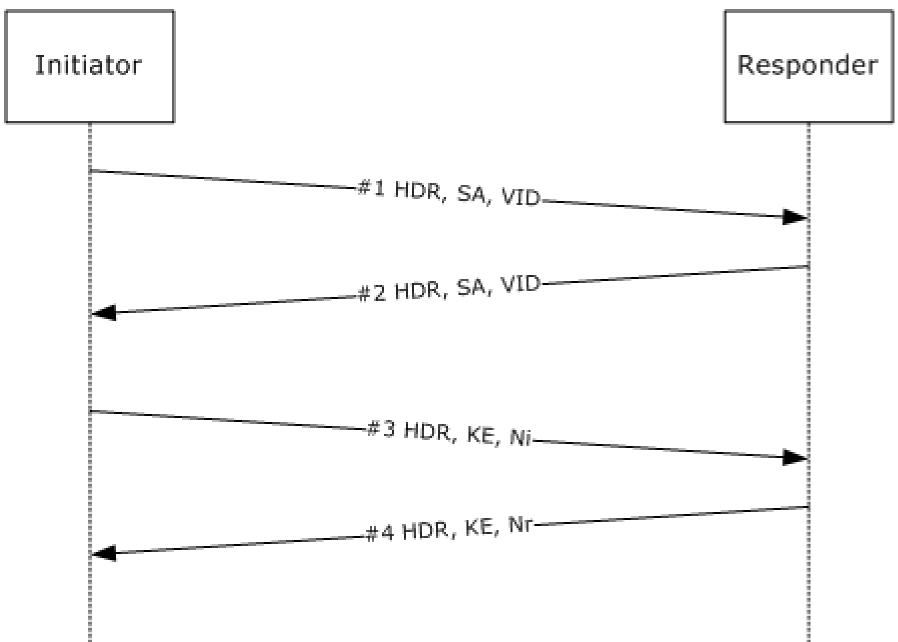

Figure 5: IKE phase 1 (MM) exchange

The description in this section uses the MM message numbers from the protocol sequence diagram.

Using the notation as specified in [RFC2409] section 3.2, the generalized form of an IKE phase 2 (QM) exchange is as shown in the following figure. For more information, see [RFC2409] section 5.5.

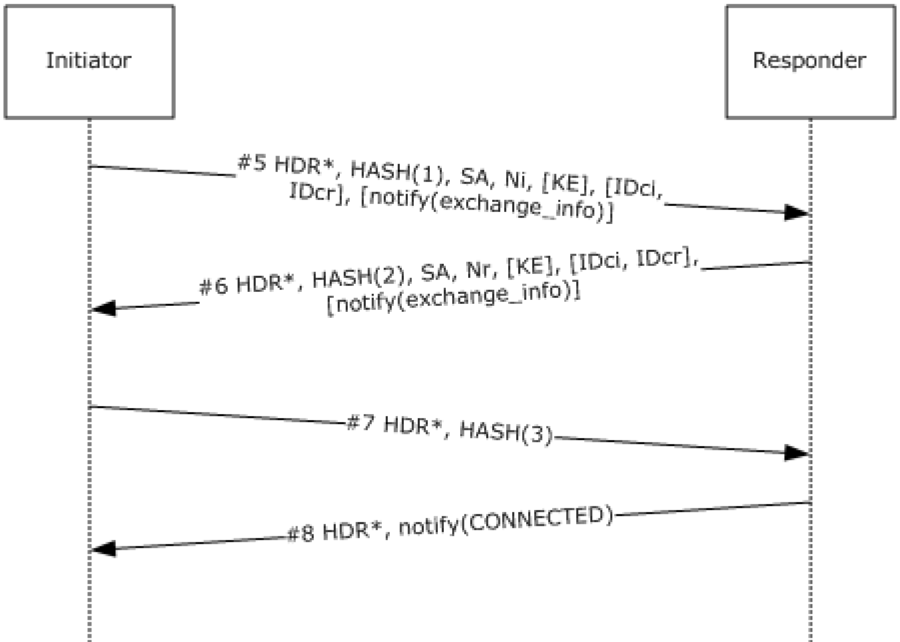

Figure 6: IKE phase 2 (QM) exchange

The description in this section uses the QM message numbers from the protocol sequence diagram.

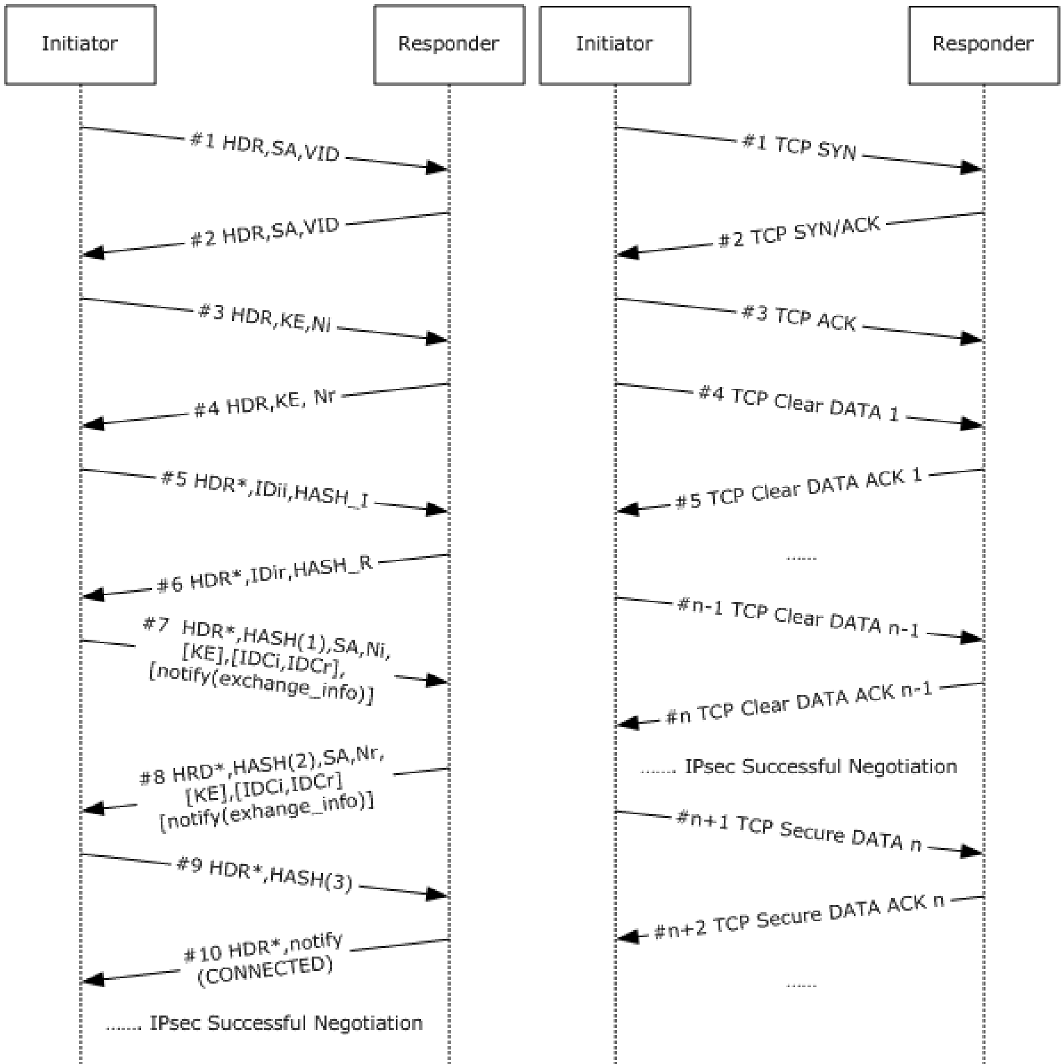

Figure 7: Negotiation discovery of a TCP connection between two IPsec-capable peers.

The TCP packet exchanges happen in parallel with the IKE exchanges that are described in figures 5 and 6. The preceding figure (figure 7) illustrates one of many ways in which the packets might interleave. When the IKE exchange completes (figure 6) the successful IPsec negotiation, the TCP connection is secured.

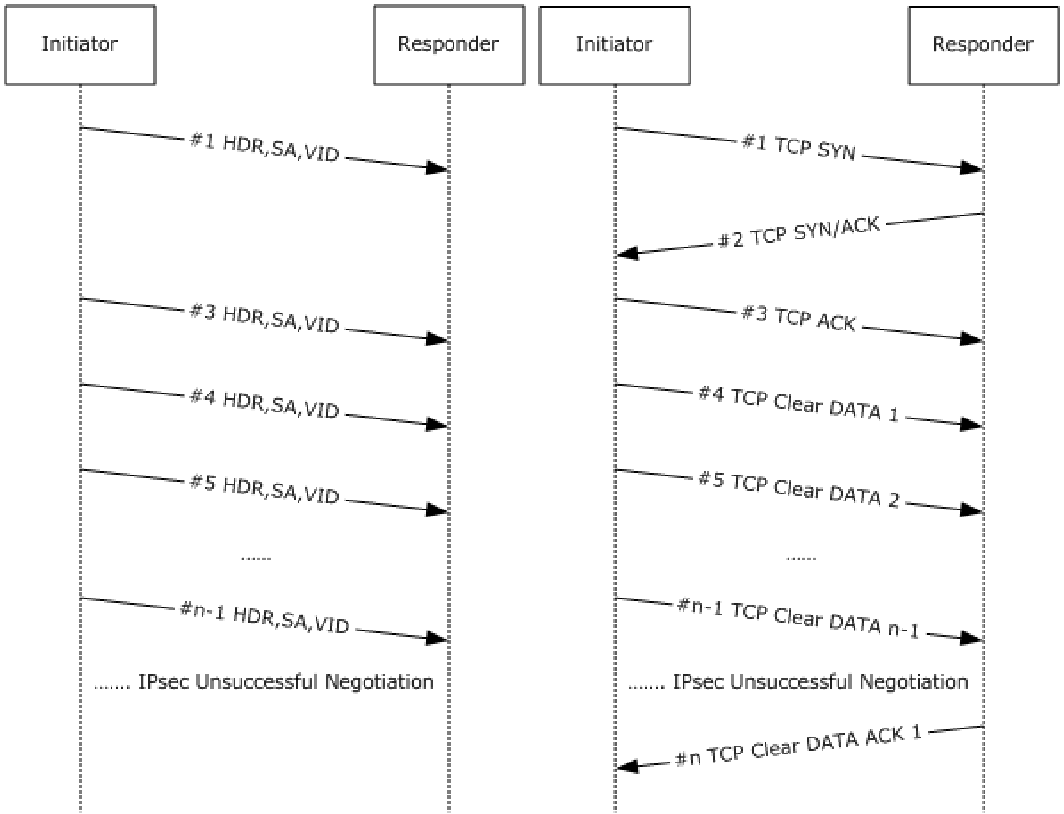

Figure 8: Negotiation discovery of a TCP connection between an IPsec-capable peer and a non-IPsec-capable peer.

The TCP packet exchanges happen in parallel with the IKE exchanges that are described in figures 5 and 6. The preceding figure (figure 8) illustrates one of many ways in which the packets might interleave. The responder does not respond to the IKE negotiation (an unsuccessful IPsec negotiation), and the TCP connection continues in the clear.

If the responder responds to the IKE negotiation, IKE fails because the responder does not have, by definition, a valid credential (it is non-IPsec-capable). However, the IKE failure does not affect the TCP stream, and the TCP connection continues in the clear.

3.7.1 Abstract Data Model

When this extension is implemented, the following additional states should be maintained. This is an extension to IKE Protocol version 1 as specified in [RFC2409].

Main mode security association database (MMSAD): The entry for each MM SA contains the following specific data element for negotiation discovery:

- Negotiation Discovery Supported: A flag that MUST be set if the peer supports negotiation discovery.

Security policy database (SPD): The following information MUST be maintained:

- A policy flag indicating that negotiation discovery MUST be applied to inbound and/or outbound traffic.

- A Boundary policy flag for negotiation discovery inbound rules that MUST be set if plaintext is accepted for this rule.
- A policy flag that MUST be set if encryption is guaranteed for this traffic.

Security association database (SAD): The following information MUST be maintained:

- Boundary flag: A flag that MUST be set if the QM SA matches an inbound negotiation discovery rule on the remote host.
- Guaranteed Encryption flag: A flag that MUST be set if the QM SA is an encryption SA and can be used for flows that have the Guaranteed Encryption flag set.

Flow state table: The following information MUST be maintained:

- Secure flag: A flag that MUST be set if one or more packets for this flow have been sent over a QM SA.
- Guaranteed Encryption flag: A flag that MUST be set if encryption is guaranteed for this flow.
- Acquire flag: A flag that MUST be set if a QM SA negotiation has already been triggered for this flow. This flag prevents triggering of an Acquire for each packet over a connection that stays in plaintext.

3.7.2 Timers

None.

3.7.3 Initialization

None.

3.7.4 Higher-Layer Triggered Events

3.7.4.1 Outbound Packet

An outbound packet MUST be matched against the SPD to determine if and how it needs to be protected, as specified in [RFC4301] section 5.

- If the packet matches a negotiation discovery rule in the SPD, and no QM SA matches the packet, one of the following MUST occur:
  - If the Secure flag is not set for the corresponding flow:

    The IPsec implementation MUST send the packet and MUST trigger IKE to negotiate the corresponding QM SA if the Acquire flag is not set on the corresponding flow. Otherwise, the IPsec implementation MUST send the packet and MUST NOT trigger IKE. The first QM negotiation message is message #5. Message #5 MUST be constructed as follows:

    - The header and payloads MUST be constructed as specified in [RFC2409] section 5.5.
    - If the SPD rule matching the traffic has the Boundary flag set, or if the Guarantee Encryption flag is set for the flow, the host MUST include a notification payload with the following fields and values:

      Notify Message Type (2 bytes): 0x9C45 (EXCHANGE_INFO).

The Notification Data field is interpreted as a flags field.

- Flag 0x00000001 (IKE_EXCHANGE_INFO_ND_BOUNDARY) MUST be set if the corresponding rule in the SPD has the Boundary flag set.
- Flag 0x00000002 (IKE_EXCHANGE_INFO_GUARANTEE_ENCRYPTION) MUST be set if the Guarantee Encryption flag is set on the corresponding flow.
- This notification payload MUST be constructed as specified in section 2.2.6.

The host MUST then set the Acquire flag on the corresponding flow.

- If the Secure flag is set for the corresponding flow:

  The IPsec implementation MUST NOT send the packet (it MAY queue or silently discard the packet) and MUST trigger IKE to negotiate the corresponding QM SA. Message #5 MUST be constructed as previously specified.

If a QM SA needs to be negotiated, and no corresponding MM SA exists (as determined by using the outbound packet destination IP address to look up the MMSAD), an MM SA MUST be negotiated. The host MUST construct and send packet #1 as specified in [RFC2409] section 5. The host MUST include in it an "MS-Negotiation Discovery Capable" vendor ID payload (a vendor ID payload generated by using the vendor ID string "MS-Negotiation Discovery Capable", as specified in [RFC2408] section 3.16).

- If the packet matches a negotiation discovery rule in the SPD, and a QM SA matches the packet, the following MUST occur:

  If the matching QM SA and the corresponding flow do not have the same value for the Guaranteed Encryption flag, the host MUST trigger IKE to negotiate the corresponding QM SA, as previously described in the case where there is no matching QM SA for the packet.

  Otherwise, one of the following MUST occur:

  - If the matching QM SA is a UDP-ESP SA ([RFC3947] section 5) with the Boundary flag (defined in section 3.7.1) set, the host MUST send the packet in Cleartext.
  - Otherwise, the IPsec implementation MUST send the packet encapsulated by using the matching QM SA, and it MUST set the Secure flag for this flow.

- If the packet does not match a negotiation discovery rule, packet processing MUST be performed as specified in [RFC4301] section 5.

If the packet matches a Guaranteed Encryption rule in the SPD, the host MUST set the Guaranteed Encryption flag on the corresponding flow. This rule MUST apply regardless of whether a matching QM SA is found or not.

3.7.4.2 Inbound Packet

An inbound packet is matched against the SPD after IPsec decapsulation to determine if and how it needs to be treated, as specified in [RFC4301] section 5. The following rules MUST be applied to the packet:

- If the packet is in Cleartext:
  - If the packet is the first packet for a new flow (for example, an inbound TCP SYN packet):

If the packet matches an inbound negotiation discovery rule in the SPD, the host MUST accept the packet. Otherwise, the host MUST silently discard the packet.

- If the packet belongs to an already existing flow:

  If the Secure flag is not set on the flow, the host MUST accept the packet. Otherwise, the host MUST silently discard the packet.

- If the packet was encapsulated using ESP or authentication header (AH):

  The host MUST set the Secure flag on the flow and process the packet as specified in [RFC4301] section 5.

Regardless of whether the packet is in plaintext, if there is an SA that matches the packet, and its Guaranteed Encryption flag is set, the host MUST set the Guaranteed Encryption flag on the corresponding flow.

3.7.5 Message Processing Events and Sequencing Rules

3.7.5.1 Receiving Message #1

On receipt of message #1, the host MUST check for the presence of the "MS-Negotiation Discovery Capable" vendor ID payload (as specified in section 3.7.4.1). If the "MS-Negotiation Discovery Capable" vendor ID payload is present in the message, the host MUST set the Negotiation Discovery Supported flag for the corresponding MMSAD entry.

Then, the host MUST construct message #2, as specified in [RFC2409] section 5, and add the "MS-Negotiation Discovery Capable" vendor ID payload to advertise its negotiation discovery capability.

3.7.5.2 Receiving Message #2

On receipt of message #2, the host MUST check for the presence of the "MS-Negotiation Discovery Capable" vendor ID payload (for details, see section 3.7.4.1) and set the Negotiation Discovery Supported flag for the corresponding MMSAD entry.

Messages #3 and #4 MUST be constructed and processed as specified in [RFC2409] section 5.

3.7.5.3 Receiving Message #5

On receipt of message #5, the host MUST check for the presence of flags within a notification payload of type EXCHANGE_INFO.

- IKE_EXCHANGE_INFO_ND_BOUNDARY: If this flag is set, the host MUST set the Boundary flag for the corresponding QM SA.
- IKE_EXCHANGE_INFO_GUARANTEE_ENCRYPTION: If this flag is set, the host MUST set the Guaranteed Encryption flag for the corresponding QM SA.

Message #6 MUST be constructed in response as follows:

The IPsec implementation MUST send the packet and MUST trigger IKE to negotiate the corresponding QM SA. The first QM negotiation message is message #5. Message #6 MUST be constructed as follows:

- The header and payloads MUST be constructed as specified in [RFC2409] section 5.5.

- If the SPD rule matching the traffic for which the QM SA is negotiated has the Boundary flag set, the host MUST add a notification payload with the following fields:

  Notify Message Type (2 bytes): 0x9C45 (EXCHANGE_INFO).

  The Notification Data field is interpreted as a flags field.

  Flag 0x00000001 (IKE_EXCHANGE_INFO_ND_BOUNDARY) MUST be set if the corresponding rule in the SPD has the Boundary flag set.

  This notification payload MUST be constructed as specified in section 2.2.6.

3.7.5.4 Receiving Message #6

On receipt of message #6, the host MUST check for the presence of flags within a notification payload of type EXCHANGE_INFO:

- IKE_EXCHANGE_INFO_ND_BOUNDARY: If this flag is set, the host MUST set the Boundary flag for the QM SA. For more details see section 2.2.6.

Messages #7 and #8 are constructed and processed as specified in [RFC2408] section 3.1.

3.7.6 Timer Events

None.

3.7.7 Other Local Events

None.

3.8 Reliable Delete Details

Using the notation as specified in [RFC2408] section 4.1.1, the generalized form of an IKE Delete exchange using the Reliable Delete extension is as shown in the following figure. For more information, see [RFC2409] section 5.

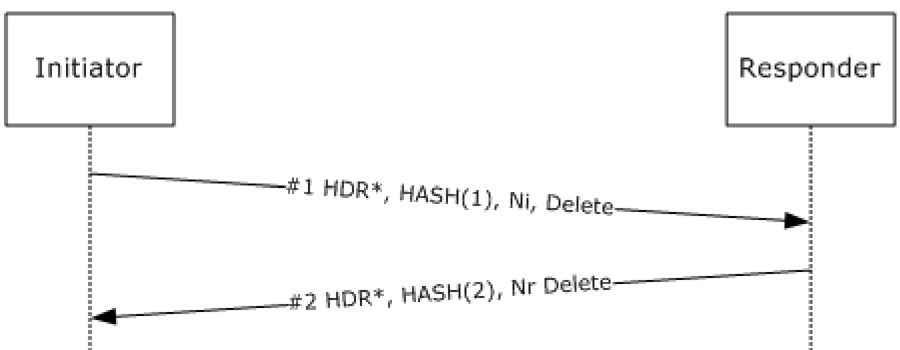

Figure 9: IKE Delete exchange

The description in this section uses the message numbers from the protocol sequence diagram.

3.8.1 Abstract Data Model

When this extension is implemented, the following additional state should be maintained. This is an extension to IKE Protocol version 1 as specified in [RFC2409].

Flow state table: The following information MUST be maintained:

- Ni payload: The exact Ni payload that is sent with the delete message#1 should be preserved as part of the IKE MM SA state in order to validate the acknowledgment response. The Ni payload is a Nonce payload and MUST be constructed as specified in [RFC2408] section 3.13.

3.8.2 Timers

The delete retransmission timer (for each MM and QM SA): This triggers a Delete payload retransmission. The start and duration of the timer MUST be as specified in sections 3.8.4.1, 3.8.6.1, and 3.8.7.1.

3.8.3 Initialization

None.

3.8.4 Higher-Layer Triggered Events

3.8.4.1 SA Deletion/Invalidation

The higher layer application may cause SAs to be deleted by changing the underlying security policy, or by triggering a local state cleanup (see section 3.8.7). In such cases, the host SHOULD delete the SAs, as specified in [RFC2408] section 5.15.

After a delete has been triggered, a delete notify MUST be sent immediately, but the MM SA MUST NOT be deleted until QM delete processing has been completed. Moreover, the QM SAs associated with the MM SA MUST NOT be deleted until deletion is triggered by other protocol events, as specified in [RFC2409] section 5.5. These protocol events are QM lifetime expiry as specified in [RFC2409] Section 5.5, policy changes (see section 3.8.7) or the peer sending a QM delete (See section 3.8.5). Once all the QM SAs associated with the MM SA have been deleted the MM SA MUST be deleted.

The host MUST then construct message #1 as follows:

- Message #1 MUST consist only of an ISAKMP header, a Hash payload, a Nonce payload, and a Delete payload, as specified in [RFC2408] section 3.15.<23>
- The ISAKMP header MUST be constructed as specified in [RFC2409] section 5.7.
- The Hash payload MUST be constructed in the following manner:

```
HASH(1) = prf(SKEYID_a, M-ID | Ni | Delete)
```

  as specified in [RFC2409] section 5.7.

- The Ni payload is a Nonce payload and MUST be constructed as specified in [RFC2408] section 3.13.
- The Delete payload MUST be constructed as specified in [RFC2408] section 3.15.

If the "MS NT5 ISAKMPOAKLEY" vendor ID payload (see section 1.7) has been received from the peer for the corresponding MM SA, the host MUST then start the delete retransmission timer and set it to expire in 1 second. Otherwise, the host MUST NOT start the delete retransmission timer.

3.8.5 Message Processing Events and Sequencing Rules

3.8.5.1 Receiving Message #1

On receipt of message #1, the host MUST validate the message, as specified in [RFC2408] section 5. If message #1 is correctly validated, the host MUST delete the corresponding SA and MUST construct message #2 in response.

- The message MUST consist only of an ISAKMP header as specified in [RFC2408] section 3.1, a Hash payload as specified in [RFC2408] section 3.11, a Delete payload as specified in [RFC2408] section 3.15, and a Nonce payload structured as specified in [RFC2408] section 3.13.
- The ISAKMP header MUST be constructed as specified in as specified in [RFC2408] section 3.1. The Message ID field MUST be copied from message #1.
- The Hash payload MUST be constructed in the following manner:

```
HASH(2) = prf(SKEYID_a, Ni | M-ID | Nr | Delete)
```

  Once computed as above, this hash value MUST be sent on the wire format specified in section 3.11 of [RFC2408].

- The Ni payload is the Nonce payload without a generic payload header.
- The Delete payload MUST be copied from message #1.
- The Nr payload is a Nonce payload and MUST be constructed as specified in [RFC2408] section 3.13.

Otherwise, the host MUST silently discard message #1.

3.8.5.2 Receiving Message #2

On receipt of message #2, the host MUST validate the message as follows:

- Validate the ISAKMP header, as specified in [RFC2408] section 5.2.
- Verify that the message ID in the ISAKMP payload is identical to the message ID from message #1.

If this verification succeeds, the host MUST stop the delete retransmission timer. Otherwise, the host MUST silently discard message #2.

3.8.6 Timer Events

3.8.6.1 Expiration of the Delete Retransmission Timer

When this timer expires, the initiator MUST retransmit message #1, as specified in section 3.8.4.1, and it SHOULD reset the timer to double the previous duration unless a total of four retransmissions has already occurred. If four retransmissions have occurred, the host MUST remove the corresponding MM SA or QM SA from the MMSAD or the SAD without retransmitting message #1 or resetting the timer.<24>

When each timer expires, if a message #2 has not been received and verified for that SA, as specified in section 3.8.5.2, it SHOULD retransmit the notification message for that SA without resetting the timer.

3.8.7 Other Local Events

An administrator can trigger local SA state deletion via a local-only interface to delete all active SAs.

The abstract interface for security policy configuration changes is specified in [RFC4301] section 4.4.1. The administrator MUST be able to specify a new local security policy as defined in [RFC4301] section 4.4.1. Any MM SAs established with a policy invalidated by the new policy are deleted as specified in section 3.8.4.1.

3.8.7.1 Shutdown

IKE protocol shutdown: IKE MUST send Delete notification messages for all SAs, as specified in section 3.8.4.1, and then SHOULD set the delete retransmission timer to 1 second for each SA.<25>

3.8.7.2 MM SA Exhaustion

Establishment of a successful QM SA may exhaust the limits for the number of QM SAs allowed for a given MM. This QM limit is a local policy setting in the PAD.<26>In this case, the host MUST NOT explicitly delete the SA. Instead, the SA MUST be invalidated, and not used for establishing any new QM SAs.

3.9 Denial of Service Protection Details

IKE goes into DOS protection under the condition described in section 3.9.7.

Using the notation, as specified in [RFC2408] section 4.1.1, the generalized form of an IKE exchange using the DOS Protection extension is as shown in the following figure. For more information, see [RFC2409] section 5.

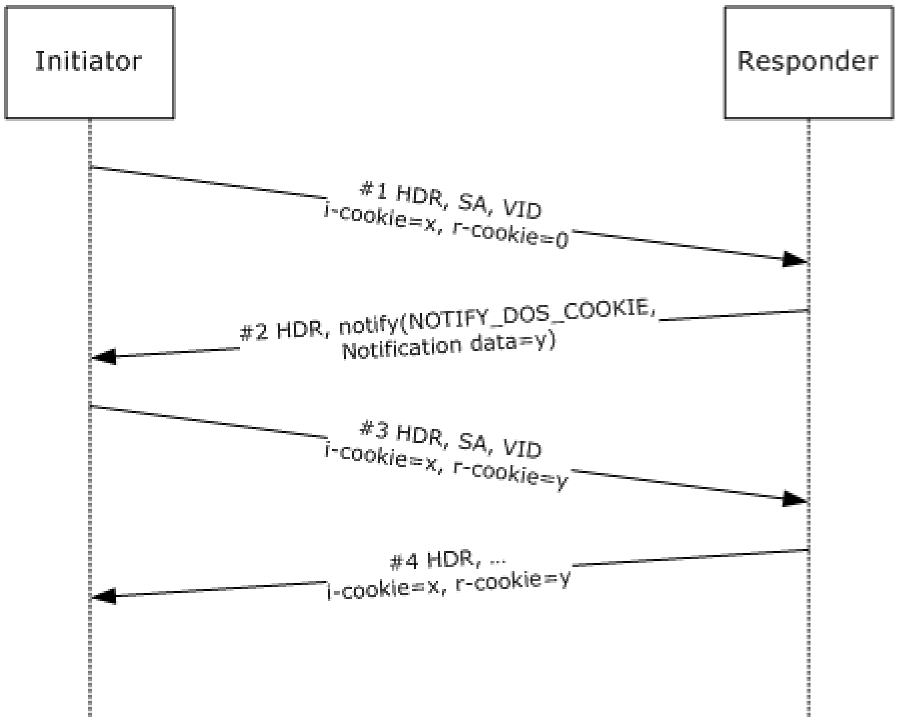

Figure 10: IKE using the DOS Protection extension

The description in this section uses the message numbers from the protocol sequence diagram.

3.9.1 Abstract Data Model

When this extension is implemented, the following additional state should be maintained. This is an extension to IKE Protocol version 1 as specified in [RFC2409].

Flow state table: The following information MUST be maintained:

- A flag indicating that DOS protection is active.

DOS Protection mode state: responder MUST maintain the following state to implement Denial of Service Protection mode.

- A cookie field consisting of random data.
- A cookie timeout period, initialized to 150 secs.

This state is used by the cookie generation algorithm that is described in section 3.9.5.1.

3.9.2 Timers

None.

3.9.3 Initialization

None.

3.9.4 Higher-Layer Triggered Events

None.

3.9.5 Message Processing Events and Sequencing Rules

3.9.5.1 Receiving Message #1

On receipt of message #1, the host MUST validate the message, as specified in [RFC2408] section 5. If message #1 is correctly validated, the host MUST construct message #2 in response, as follows:

- The message MUST consist of only an ISAKMP header and a Notify payload structure, as specified in [RFC2408] section 3.14.
- The ISAKMP header MUST be constructed as specified in [RFC2409] section 5.7. The message ID field is unique to this exchange, as specified in [RFC2409] section 5.7.
- The notify message type MUST be set to NOTIFY_DOS_COOKIE, and the notification data MUST contain an 8-byte cookie value. The cookie generation mechanism is implementation-dependent but SHOULD be stateless to provide good DOS protection.<27>

The host MUST then silently discard message #1, even if the message is correctly validated.

3.9.5.2 Receiving Message #2

On receipt of message #2, the host MUST validate the message, as specified in [RFC2408] section 5. In addition, the host MUST:

- Verify that the message contains a single Notify payload, that the notify message type is set to NOTIFY_DOS_COOKIE, and that the notification data contains an 8-byte cookie value. No checks on the actual value are performed at this stage.

If this verification succeeds, the host MUST construct message #3 as follows:

- Message #3 is the same as message #1, except that the Responder Cookie field of the ISAKMP header ([RFC2408] section 3.1) is the cookie from the notify NOTIFY_DOS_COOKIE payload in message #2.

Otherwise the host MUST process message #2 as a normal ISAKMP message.

3.9.5.3 Receiving Message #3

On receipt of message #3, the host MUST validate the message, as specified in [RFC2408] section 5. In addition, the host MUST:

- Verify that the **Responder Cookie** field in the ISAKMP header is not zero.
- Verify that the **Responder Cookie** field in the ISAKMP header is the same as the cookie sent in the Notify payload of message #2. The actual verification mechanism is implementation-dependent.<28>

If this verification succeeds, the host MUST process message #3 as a normal ISAKMP message. Otherwise, the host MUST process message #3 in the same way as message #1.

Subsequent messages received for this SA on the host in DOS Protection mode MUST be processed the same as message #3.

Subsequent messages received for SAs for which no state exists in the SAD MUST be processed in the same way as message #1.

3.9.6 Timer Events

None.

3.9.7 Other Local Events

DOS Protection threshold: If the number of negotiations for which only one message has been received from any initiator is above a predefined threshold, IKE MUST go into DOS Protection mode (see section 3.1 for details). The threshold may be implemented in a number of ways.<29>

3.10 IKE SA Correlation (IKEV2) Details

See [RFC4306] section 1.2. If SA Correlation is used, during the IKE_SA exchange the Correlation payload MUST be inserted immediately prior to the SA payload.

On initiator:

HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] NOTIFY, AUTH, CORRELATION, SAi2, TSi, TSr}

This is similar to the behavior for the Extensible Authentication Protocol (EAP) exchange, as defined in [RFC4306] section 2.16.

NOTIFY is related to the Mobility and Multihoming Protocol (MOBIKE). See [RFC4555] section 4 for information about the Notify message type. See [RFC4306] section 3.10 for the general Notify header format.

The correlation exchange MUST use the same authentication as the original exchange. If the original exchange did EAP authentication, then the correlation exchange MUST use EAP authentication. Similarly, if the original exchange used certificate authentication (and not EAP authentication), then the correlation exchange MUST use certificate authentication, and MUST NOT use EAP authentication.

3.10.1 Abstract Data Model

When this extension is implemented, the following additional state should be maintained. This is an extension to IKE Protocol version 2 as specified in [RFC4306].

Main mode security association database (MMSAD): The entry for each MM SA contains the following specific data elements for IKE SA Correlation.

For IKE_SA correlation (IKEv2), the following information MUST be maintained:

- The index of the entry in the MMSAD for the other SA to which this SA has been correlated, if it exists (see section 3.10.5.1).

3.10.2 Timers

None.

3.10.3 Initialization

None.

3.10.4 Higher-Layer Triggered Events

None.

3.10.5 Message Processing Events and Sequencing Rules

The following figures show the standard and EAP exchange sequences, as specified in [RFC4306] sections 1.2 and 2.16, respectively.

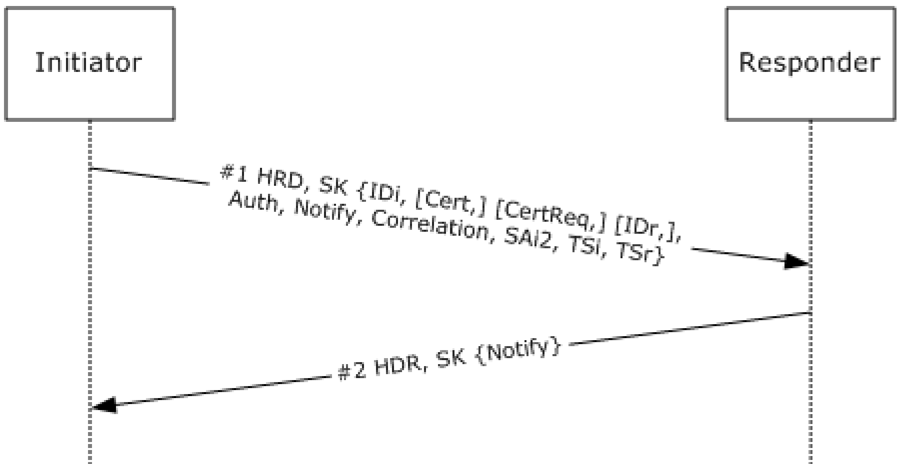

Figure 11: Standard IKEv2 exchange

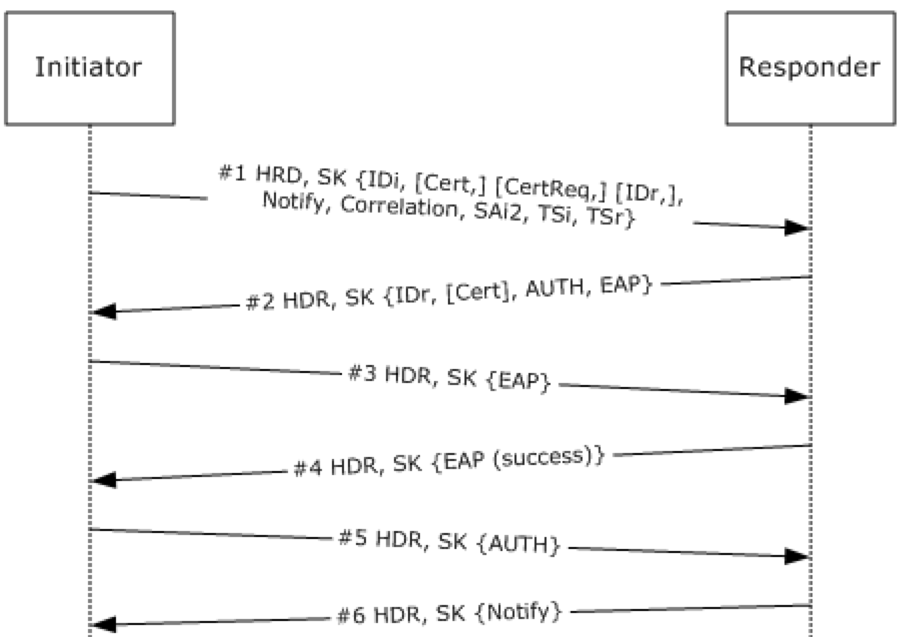

Figure 12: IKEv2 EAP exchange

3.10.5.1 Receiving Message #1

The responder processes all payloads prior to the correlation payload as per [RFC4306], [RFC4555], and [RFC4621]. Note that message #1 corresponds to the third packet in the IKEv2 exchange. See [RFC4306] section 1.2.

When the host receives the correlation payload, it MUST validate its generic header as specified in [RFC4306] section 3.2. Additionally, the host MUST:

1. See whether an existing IKE_SA in its SADB table matches the initiator and responder SPIs from the correlation payload.
2. If there is an existing SA, the host MUST validate the correlation hash by computing its own value given its local SA state, and comparing it with the value of the correlation hash in the payload. If they are equal, the host flags these SAs as correlated.

Any failures in this exchange MUST NOT affect the state of the correlated IKE_SA.

3.10.5.2 Receiving Subsequent Messages

All subsequent messages in the exchange—except the final message—are processed as usual. At the end of the exchange, when the responder has successfully finished processing the final message, the responder tears down this exchange and sends back an IKEV2 error notify via the notification mechanism in [RFC4306] section 1.4.

For the standard exchange, there are no subsequent messages. For the EAP exchange, the subsequent messages 2–5 are constructed and processed identically to [RFC4306].

3.10.5.3 Receiving the Error Notify

The error notify MUST be processed as specified in [RFC4306] section 1.4 and MUST delete the SA as specified in [RFC4306] section 3.10.1.

The initiator, who is receiving the error notify, SHOULD process the extended error information as defined in 2.2.7.

3.10.6 Timer Events

None.

3.10.7 Other Local Events

None.

3.11 IKE Server Internal Addresses Configuration Attributes (IKEv2) Details

See [RFC4306] section 2.19. During the IKE_AUTH exchange, the IPsec remote access client (IRAC) MUST request the IPsec remote access server (IRAS)–controlled address.<30>

On initiator:

HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,] AUTH, CP(CFG_REQUEST),SAi2, TSi, TSr}

The server (IRAS) replies with:

HDR, SK {IDr, [CERT,] AUTH, CP(CFG_REPLY), SAr2, TSi, TSr}

3.11.1 Abstract Data Model

When this extension is implemented,<31> the following additional state should be maintained. This is an extension to IKE Protocol version 2 as specified in [RFC4306].

Flow state table: The following information MUST be maintained:

- The internal IPv4 address of the server.
- The internal IPv6 address of the server.

The initiator SHOULD request this attribute for each IP version it supports.

3.11.2 Timers

None.

3.11.3 Initialization

None.

3.11.4 Higher-Layer Triggered Events

None.

3.11.5 Message Processing Events and Sequencing Rules

The following figure shows the exchange sequence for IKEv2 Non-EAP embedded quick mode negotiation with Configuration payloads.

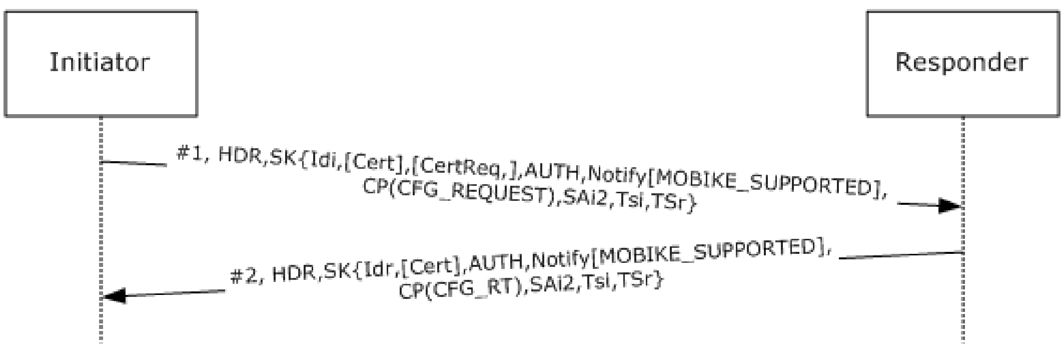

Figure 13: IKEv2 Non-EAP embedded quick mode negotiation with Configuration payload exchange

The following figure shows the Configuration payload exchange sequence with EAP, as specified in [RFC4306] section 3.15.

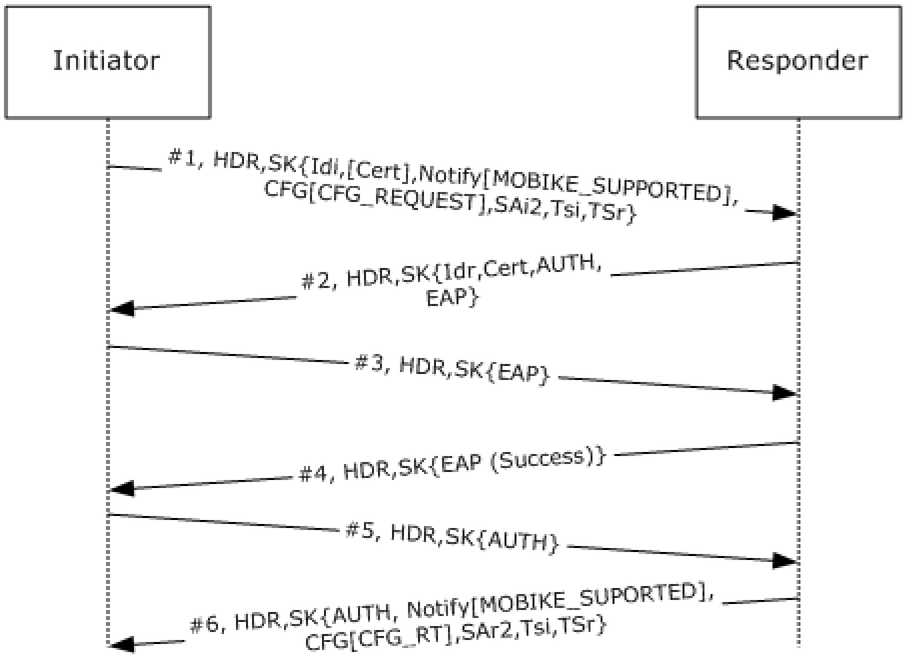

Figure 14: IKEv2 Configuration payload exchange with EAP

3.11.5.1 Receiving Message #1

When the host receives the CFG_REQUEST (as specified in [RFC4306] section 3.15) for the INTERNAL_IP4_SERVER or INTERNAL_IP6_SERVER attribute, it MUST validate the message as also specified in [RFC4306] section 3.15. Additionally, the host MAY<32>:

- See whether the server has an internal IPv4 address or an internal IPv6 address.
- If either or both are present, add these attributes in CFG_REPLY.

Any failures in this exchange MUST NOT affect the state of the IKE_SA.

3.11.5.2 Receiving Message #2

When the host receives the CFG_REPLY (as specified in [RFC4306] section 3.15) for the INTERNAL_IP4_SERVER or INTERNAL_IP6_SERVER attribute, it MUST validate the message as also specified in [RFC4306] section 3. Additionally, the host MAY:<33>

- See whether the server has sent an internal IPv4 address or an internal IPv6 address.
- If either or both are present, store these values in its local data structures and use these addresses to send packets to the internal address of IRAS.

Any failures in this exchange MUST NOT affect the state of the IKE_SA.

3.11.6 Timer Events

None.

3.11.7 Other Local Events

None.

4 Protocol Examples

4.1 Negotiation Discovery Examples

The following protocol sequence diagram depicts communication between a client with a negotiation discovery policy and a server with negotiation discovery in boundary mode.

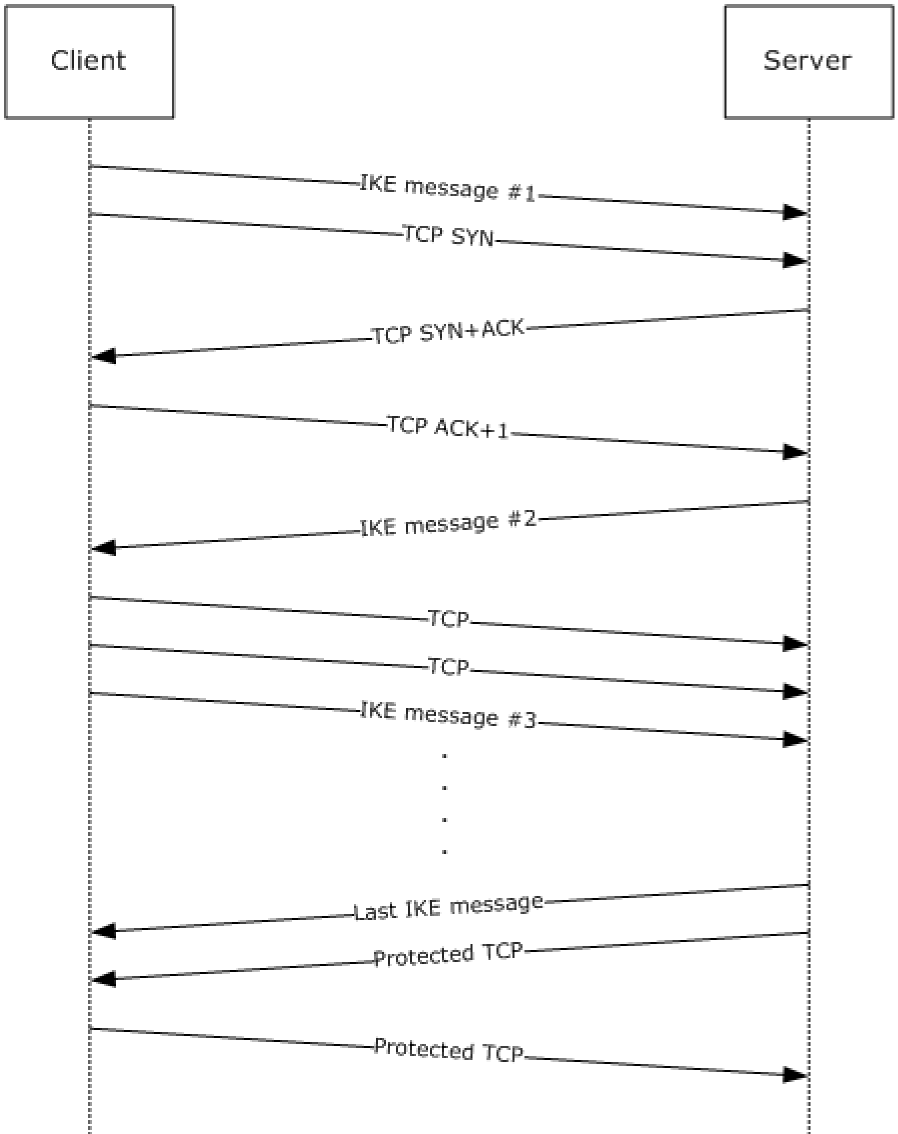

Figure 15: Negotiation discovery between client and server

In this example, the client initiates a TCP connection to the server. At the same time that it sends the TCP SYN packet, the client initiates the IKE to the server. TCP traffic flows in the clear until the IKE negotiation completes with IKE message #6. Then, the traffic for this connection is protected.

In the second example, the server requires all inbound traffic to be protected.

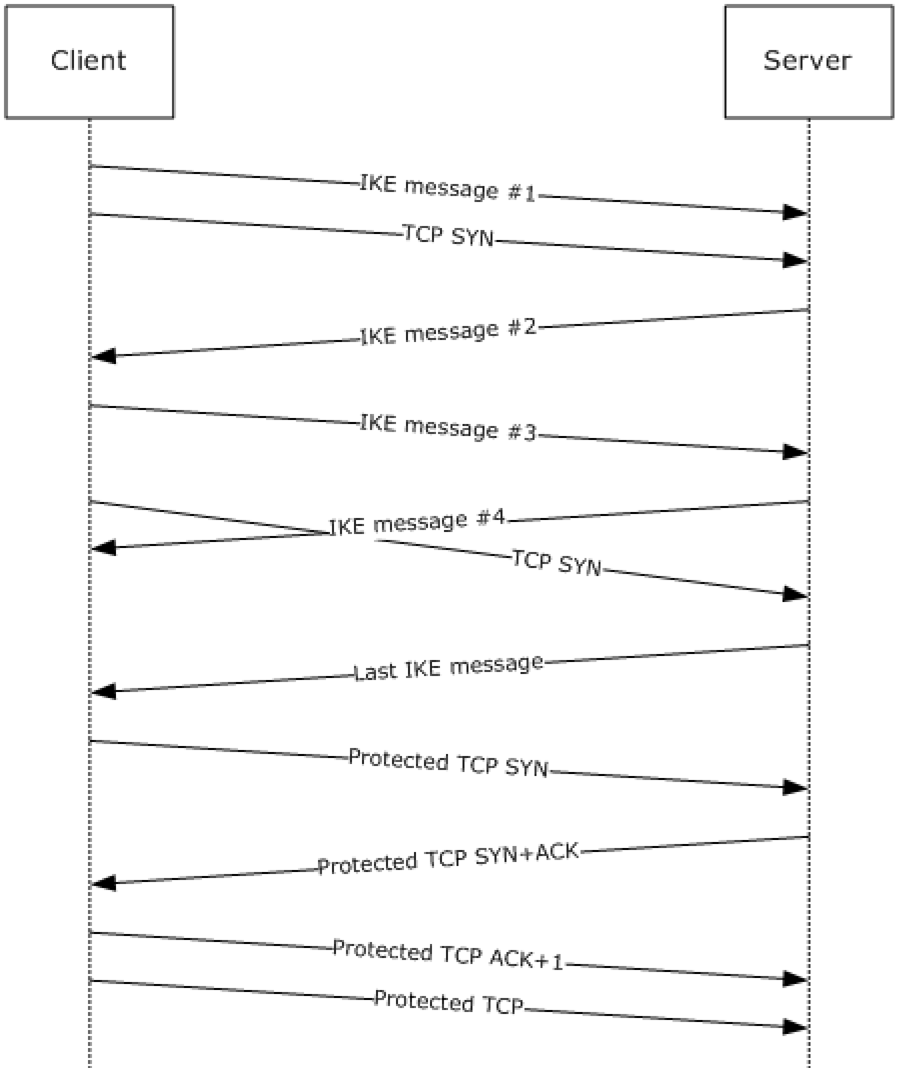

Figure 16: Negotiation discovery between client and server, all inbound traffic protected

In this example, the client initiates a TCP connection to the server. At the same time that it sends the TCP SYN packet, the client initiates the IKE to the server. The Cleartext TCP SYN packets are dropped by the server and retransmitted by the client until the IKE negotiation completes with IKE message #6. The server then accepts the protected traffic.

5 Security

5.1 Security Considerations for Implementers

5.1.1 Negotiation Discovery

Negotiation discovery allows Cleartext outbound and inbound connections if the peer is not IPsec–capable. Connections that are Cleartext should be considered when designing the policy.

5.2 Index of Security Parameters

Security parameter	Section
Authentication method	1.7
Encryption/authentication algorithms	1.7
Diffie-Hellman	1.7

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft Windows® 2000 operating system
- Windows® XP operating system
- Windows Server® 2003 operating system
- Windows Vista® operating system
- Windows Server® 2008 operating system
- Windows® 7 operating system
- Windows Server® 2008 R2 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 1.3: IKE extensions by operating system cross-reference. Implemented in Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The following tables describe the extensions that each release supports.

The IKE proposal for Encapsulating Security Payload (ESP) and Authentication Headers (AH) is deprecated in the Windows 7 implementation of IKE *v2* but supported by IKE version 1 in Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2.

IKE extension	Windows NT 4.0 (with additional download)	Windows 2000	Windows 2000 SP4 post-SP4 rollup
NAT-T	X		X
IKE fragmentation			X
CGA authentication			
Fast failover			
Negotiation discovery			
Reliable delete			X

IKE extension	Windows XP	Windows XP SP2	Windows Server 2003	Windows Vista and Windows Server 2008	Windows 7 and Windows Server 2008 R2
NAT-T		X	X	X	X
IKE fragmentation		X	X	X	X
CGA authentication				X	X
Fast failover		X	X	X	X
Negotiation discovery				X	X
Reliable delete	X	X	X	X	X
IKEv2 SA Correlation					X
IKEv2 Configuration Attributes					X
Denial of Service protection	X	X	X	X	X

<2> Section 1.3.8: IKE extensions by operating system cross reference. Implemented in Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The following table describes the extensions that are supported by IKE in each release.

IKE extension	Windows XP SP2	Windows Server 2003	Windows Vista and Windows Server 2008	Windows 7 and Windows Server 2008 R2
IKE/AuthIP co-existence			X	X
Exchange information notification payload	X	X	X	X

<3> Section 1.7: Algorithms implemented by operating system cross reference. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The following table lists the algorithms that are implemented.

Authentication method	Windows 2000	Windows XP	Windows Server 2003	Windows Vista and Windows Server 2008	Windows 7 and Windows Server 2008 R2
Pre-shared key (as specified in [RFC2409])	X	X	X	X	X
RSA signature (as specified in [RFC2409])	X	X	X	X	X
Kerberos using **GSS**-API (as specified in [GSS])	X	X	X	X	X
CGA (as specified in [RFC3972])				X	X

<4> Section 1.7: Cryptographic parameters implemented by operating system cross reference. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The following tables list the parameters that are implemented.

Diffie-Hellman group	Windows 2000	Windows XP	Windows Server 2003	Windows Vista and Windows Server 2008	Windows 7 and Windows Server 2008 R2
Default 768-bit MODP group (as specified in [RFC2409])	X	X	X	X	X
Alternate 1,024-bit MODP group (as specified in [RFC2409])	X	X	X	X	X
2,048-bit MODP group (as specified in [RFC3526])			X	X	X
ECP256 (as specified in [ECP])				X	X
ECP384 (as specified in [ECP])				X	X

Authentication algorithm	Windows 2000	Windows XP	Windows Server 2003	Windows Vista and Windows Server 2008	Windows 7 and Windows Server 2008 R2
NULL (as specified in [RFC2410])	X	X	X	X	X
HMAC-SHA1-96 (for more information, see [RFC2404])	X	X	X	X	X
HMAC-MD5-96 (as specified in [RFC2403])	X	X	X	X	X
AES-MAC (for more information, see [RFC4543])					X
SHA-256 (for more information, see [SHA256])					X

Encryption algorithm	Windows 2000	Windows XP	Windows Server 2003	Windows Vista and Windows Server 2008	Windows 7 and Windows Server 2008 R2
NULL (as specified in [RFC2410])	X	X	X	X	X
DES-CBC (for more information, see [RFC2405])	X	X	X	X	X
3DES-CBC (as specified in [RFC2451])	X	X	X	X	X
AES-CBC with 128, 192, and 256 Bit Keys (for more information, see [RFC3602])				X	X
AES-GCM with 128, 192, and 256 Bit Keys (for more information, see [RFC4106])					X

<5> Section 1.7: Vendor ID payloads. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The Microsoft implementation of IKE supports the following vendor IDs.

The Microsoft implementation vendor ID (the first rows of the second table that follows, where the common name starts with Microsoft implementation) is constructed by appending a 32-bit (4-byte) version number in network order to the 128-bit (16-byte) MD5 hash of the "MS NT5 ISAKMPOAKLEY" string. The version number is the additional 4 bytes that denote the Windows operating system version as detailed in the first table that follows.

Operating system version	4-byte version number
Windows 2000	00 00 00 02
Windows XP	00 00 00 03
Windows Server 2003	00 00 00 04
Windows Vista	00 00 00 05
Windows Server 2008	00 00 00 06
Windows 7	00 00 00 07
Windows Server 2008 R2	00 00 00 08

Common name	String representation	Wire representation (MD5 hash of string)	Version
Microsoft implementation Windows 2000	"MS NT5 ISAKMPOAKLEY" + version number 2	1E 2B 51 69 05 99 1C 7D 7C 96 FC BF B5 87 E4 61 00 00 00 02	Windows 2000
Microsoft implementation Windows XP	"MS NT5 ISAKMPOAKLEY" + version number 3	1E 2B 51 69 05 99 1C 7D 7C 96 FC BF B5 87 E4 61 00 00 00 03	Windows XP
Microsoft implementation Windows Server 2003	"MS NT5 ISAKMPOAKLEY" + version number 4	1E 2B 51 69 05 99 1C 7D 7C 96 FC BF B5 87 E4 61 00 00 00 04	Windows Server 2003
Microsoft implementation Windows Vista	"MS NT5 ISAKMPOAKLEY" + version number 5	1E 2B 51 69 05 99 1C 7D 7C 96 FC BF B5 87 E4 61 00 00 00 05	Windows Vista
Microsoft implementation Windows Server 2008	"MS NT5 ISAKMPOAKLEY" + version number 6	1E 2B 51 69 05 99 1C 7D 7C 96 FC BF B5 87 E4 61 00 00 00 06	Windows Server 2008
Microsoft implementation Windows 7	"MS NT5 ISAKMPOAKLEY" + version number 7	1E 2B 51 69 05 99 1C 7D 7C 96 FC BF B5 87 E4 61 00 00 00 07	Windows 7
Microsoft implementation Windows Server 2008 R2	"MS NT5 ISAKMPOAKLEY" + version number 8	1E 2B 51 69 05 99 1C 7D 7C 96 FC BF B5 87 E4 61 00 00 00 08	Windows Server 2008 R2

Common name	String representation	Wire representation (MD5 hash of string)	Version
Kerberos authentication supported (as specified in [GSS])	"GSSAPI"	62 1B 04 BB 09 88 2A C1 E1 59 35 FE FA 24 AE EE	Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2
NLB/MSCS fast failover supported	"Vid-Initial-Contact"	26 24 4D 38 ED DB 61 B3 17 2A 36 E3 D0 CF B8 19	Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2
NLB/MSCS fast failover supported	"NLBS_PRESENT"	72 87 2B 95 FC DA 2E B7 08 EF E3 22 11 9B 49 71	Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2
Fragmentation avoidance supported	"FRAGMENTATION"	40 48 B7 D5 6E BC E8 85 25 E7 DE 7F 00 D6 C2 D3	Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2
NAT-T supported	"draft-ietf-ipsec-nat-t-ike-02\n"	90 CB 80 91 3E BB 69 6E 08 63 81 B5 EC 42 7B 1F	Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2
NAT-T supported	"RFC 3947"	4A 13 1C 81 07 03 58 45 5C 57 28 F2 0E 95 45 2F	Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2
AuthIP supported	"MS-MamieExists"	21 4C A4 FA FF A7 F3 2D 67 48 E5 30 33 95 AE 83	Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2
CGA supported	"IKE CGA version 1"	E3 A5 96 6A 76 37 9F E7 07 22 82 31 E5 CE 86 52	Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2
Negotiation discovery supported	"MS-Negotiation Discovery Capable"	FB 1D E3 CD F3 41 B7 EA 16 B7 E5 BE 08 55 F1 20	Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2

<6> Section 2.1: IKE transport port assignments. Implemented in Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. These IKE extensions run on UDP ports 500 and 4500 only.

<7> Section 2.2.1: NAT-T payload types. Implemented in Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The NAT-T payload type values that are used by each Windows version are specified in section 3.2.4.1.

<8> Section 2.2.2: NAT-T UDP encapsulation modes. Implemented in Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The UDP encapsulation mode values used by each Windows version are specified in section 3.2.4.1.

<9> Section 2.2.6: Error codes. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. This field may contain any Windows error code value. For more information about these codes, see [MS-ERREF].

<10> Section 2.2.7: Error codes. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. This field may take on any Windows error code value. For more information about these codes, see [MS-ERREF].

<11> Section 3.1.5: Initialization vectors (IV) choice for encrypted notifications sent prior to MM SA establishment:

If the peer sent the MS NT5 ISAKMPOAKLEY notify vendor ID and the 4-byte version number is 0x00000002, 0x00000003, 0x0000004, or 0x00000005, (denoting Windows 2000, Windows XP, Windows Server 2003 and Windows Vista, respectively), the IV used in encrypting the notify is the last cipher block of the last sent packet. Otherwise, the IV will be the last cipher block of the last decrypted packet.

<12> Section 3.2: [RFC3947] and [DRAFT-NATT] implementation. Both revisions are implemented in Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. Windows 2000 SP4, Windows XP SP2, and Windows Server 2003 implement the [DRAFT-NATT] revision. For more information, see [DRAFT-NATT].

<13> Section 3.2.2: NAT-T keep-alive timer: A keep-alive message is sent every 20 seconds. Implemented in Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista SP2, Windows Server 2008 SP1, Windows 7, and Windows Server 2008 R2. Windows Vista, Windows Vista SP1, and Windows Server 2008 do not send keep-alives.

<14> Section 3.2.4.1: [NAT-T IKE] message construction. Implemented in Windows 2000 Server SP4, Windows XP SP2, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2.

NAT-T revision support	Version
[DRAFT-NATT] and [RFC3947]	Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2
[DRAFT-NATT]	Windows 2000 Server SP4, Windows XP SP2, and Windows Server 2003

Windows does not support NAT-T for IPv6 and therefore, does not send the NAT-T vendor IDs for IPv6 negotiations.

<15> Section 3.3.2: Fragmentation timer. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The fragmentation timer is variable.

This timer interval is computed as the sum of the first 2 packet retransmission times. In Windows 2000, Windows XP, and Windows Server 2003, this is started from the IKE exchange (the second round trip in Main Mode). For Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, this is started from the first Main Mode packet of the exchange. This should sum to 3 seconds. However, there is variance in the timer implementation up to ½ second per retransmission. This is an implementation artifact in that the underlying timer implementation is only this accurate. Hence the observed timer will be within the range of 2–4 seconds.

- In Windows 2000, Windows XP, and Windows Server 2003, both the initiator and the responder implement a fragmentation timer.
- In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, only the initiator implements a fragmentation timer.

<16> Section 3.3.2: Fragment reassembly timer: Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The fragment reassembly timer is set to 70 seconds.

<17> Section 3.3.5.3: Fragmentation active flag. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The Fragmentation active flag is set on receipt of a Fragment payload.

<18> Section 3.3.5.3: IKE Message Fragmentation active flag behavior. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. IKE messages are fragmented if the Fragmentation active flag is set, as per the conditions specified in section 3.3.6.1.

<19> Section 3.3.6.1: Expiration of the fragmentation timer. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2.

<20> Section 3.5.4.1: Vendor ID payload. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. In Windows Vista and Windows Server 2008, the host sends the "Vid-Initial-Contact" Vendor ID payload if it has no open TCP connections to the peer and new connection attempts cause the retransmission of SYN packets.

<21> Section 3.5.7.1: QM SA idle timer. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The QM SA idle timer is set to 1 minute if the Fast Failover flag is set on the parent MM SA, and it is set to 5 minutes if the Fast Failover flag is not set.

<22> Section 3.6.5.1: Vendor ID processing. Implemented in Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. This information is used to evaluate whether the MM SA should be allocated to a different host within the cluster. For more information, see [MSFT-WLBS].

<23> Section 3.8.4.1: Nonce. Implemented in Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. Nonces are 32-byte random numbers that are generated from a FIPS-140–compliant random-number generator. For more information, see [FIPS140].

<24> Section 3.8.6.1: Delete Retransmission timer. Implemented in Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The first retransmission occurs after 1 second. The time-out is doubled for each subsequent retransmission up to a maximum of six retransmissions. The maximum retransmission interval is capped at 16 seconds; so if the doubling of the previous interval exceeds 16 seconds, 16 seconds is used. The timer is started only if the remote host is a Windows peer, as identified by the "MS NT5 ISAKMPOAKLEY" vendor ID payload.

<25> Section 3.8.7.1: Shutdown behavior. On shutdown for Windows 2000, Windows XP, and Windows Server 2003, IKE runs as specified in the footnote regarding the delete transmission timer in section 3.8.6.1. Note that the machine may shut down before the maximum number of retransmissions has actually been sent.

<26> Section 3.8.7.2: After a delete has been triggered, Windows immediately sends the delete notify, and delays deleting the MM state internally to handle QM delete processing. Also, Windows does not immediately delete the QM(s) associated with the MM on receiving the MM delete, but waits for them to be deleted as a result of other protocol events.

<27> Section 3.9.5.1: DOS Protection mode cookie generation. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The Windows implementation uses the following algorithm to generate the cookie (*prevTimeSlice* is a Boolean input parameter to the algorithm). iCookie is the Initiator Cookie as defined in [RFC2408] section 3.1.

```
Set Curtime to the 32 bits number of seconds
    elapsed since midnight, January 1, 1970
Set LocalIPaddr to the local IP address in
    network order
Set Localport to the 16 bits local listening UDP
    port (500 or 4500) in network order /* This port is the local port that the packet was
received on. */
Set Peerport to the 16 bits remote port in
    network order
Set PeerIPaddr to the peer IP address in network order
Set cookieKey to a 50-byte random number
Set COOKIE_KEY_TIME to 150 seconds
If LocalIPaddr and PeerIPaddr are IPv4 addresses then
Compute localAddr as 01 00 02 00 concatenated with LocalPort concatenated with LocalIPAddr
    concatenated with 26 bytes of 0
Compute peerAddr as 01 00 02 00 concatenated with peerPort concatenated with peerIPAddr
    concatenated with 26 bytes of 0
end if
If LocalIPaddr and PeerIPaddr are IPv6 addresses then
Compute localAddr as 0x01 0x00 0x02 0x00 concatenated with LocalPort
    concatenated with LocalIPAddr concatenated with 14 bytes of 0
Compute peerAddr as 0x05 0x00 0x17 0x00 concatenated with peerPort
    concatenated with peerIPAddr concatenated with 14 bytes of 0
end if
Compute Curtime as  ((Curtime + COOKIE_KEY_TIME) / COOKIE_KEY_TIME) * COOKIE_KEY_TIME
If prevTimeSlice is true then
Compute Curtime as Curtime - COOKIE_KEY_TIME
End if
Compute tempCookie as SHA1(cookieKey concatenated with iCookie concatenated with peerAddr
    concatenated with localAddr concatenated with curTime)
Compute cookie as the first 8 bytes of tempCookie
```

<28> Section 3.9.5.3: DOS Protection mode cookie validation. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. The Windows implementation checks the validity of the responder cookie field by regenerating the cookie using the algorithm specified in section 3.9.5.1. The algorithm is as follows.

```
Set RCookie to the cookie field from message #2
Set prevTimeslice to FALSE
```

```
Compute cookie as described in <ref2>
If RCookie=cookie then
RCookie is valid
Else
Set prevTimeslice to TRUE
Compute cookie as described in <ref2>
If RCookie=cookie then
RCookie is valid
Else
RCookie is invalid
End if
End if
```

<29> Section 3.9.7: DOS Protection Threshold. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2. In Windows Vista and Windows Server 2008, Windows goes into DOS Protection mode if the number of negotiations for which only one message has been received from any initiator is more than 500. This is detected when the number of MM SAs in the MMSAD (see section 3.1.1) is more than 500, and these SAs have only received one message. For a given IP address, if the number of negotiations for which only one message has been received is above 35, Windows drops new incoming negotiations from this IP address. For this reason, incoming messages have to come from multiple IP addresses in order to trigger the Denial of Service Protection mode. In Windows 2000, Windows XP, or Windows Server 2003, Windows goes into DOS protection mode immediately after setting the registry key and restarting the service.

Windows goes out of DOS Protection mode if the number of MM SAs in the MMSAD for which only one message has been received from any initiator is less than 100.

To enable the Windows DOS Protection mode in Windows Vista, Windows Server 2008, Windows 7, or Windows Server 2008 R2, set the following Windows registry DWORD to 1.

SYSTEM\\CurrentControlSet\\Services\\IKEEXT\\Parameters\EnableDOSProtect (DWORD)

To enable Windows DOS Protection mode in Windows 2000, Windows XP, or Windows Server 2003, set the following Windows registry DWORD to 1.

SYSTEM\\CurrentControlSet\\Services\\PolicyAgent\\Oakley\EnableDOSProtect (DWORD). Stop and restart the PolicyAgent service for this setting to take effect.

<30> Section 3.11: This feature is only supported in Windows 7 and Windows Server 2008 R2.

<31> Section 3.11.1: This feature is only supported in Windows 7 and Windows Server 2008 R2.

<32> Section 3.11.5.1: The Windows 7 and Windows Server 2008 R2 responder adds this attribute.

<33> Section 3.11.5.2: The Windows 7 and Windows Server 2008 R2 initiator processes this attribute.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index

V