[MS-IKEE]:
Internet Key Exchange Protocol Extensions

Revision Summary

Date	Revision History	Revision Class	Comments
06/08/2007	2.0	Major	Updated and revised the technical content.
07/10/2007	2.0.1	Editorial	Revised and edited the technical content.
08/17/2007	3.0	Major	Revised content based on Trustee feedback.
09/21/2007	3.0.1	Editorial	Revised and edited the technical content.
10/26/2007	3.0.2	Editorial	Revised and edited the technical content.

Date	Revision History	Revision Class	Comments
01/25/2008	4.0	Major	Updated and revised the technical content.

Table of Contents

1 Introduction

This document specifies extensions to the Internet Key Exchange (IKE) Protocol version 1, as specified in [RFC2407], [RFC2408], [RFC2409], and [RFC3947]. These extensions provide additional capabilities to **IKE**, including interoperation between different revisions on the Network Address Translation Traversal (NAT-T) specification, fragmentation of large IKE version 1 messages, authentication using **cryptographically generated addresses (CGAs)**, fast failover when communicating with a **cluster** of hosts, easier interoperation with non-Internet Protocol security (IPsec)-capable peers, and acknowledgment of **security association (SA)** deletion messages.

1.1 Glossary

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

Authentication Header (AH)
Authenticated IP (AuthIP)
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)
Keying Material
Main Mode (MM)
Main Mode Security Association (MM SA)
Maximum Transmission Unit (MTU)
Negotiation
Negotiation Discovery
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 normally 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 using a virtual IP address. For more information, see [MSFT-WLBS].

Internet Security Association and Key Management Protocol (ISAKMP) Payload: A modular building block for constructing **ISAKMP** messages. An **ISAKMP payload** is used to transfer information such as **security association (SA)** data, or key generation and authentication data. The presence and order of **ISAKMP payloads** in a packet is defined by and dependent on the type of **exchange** specified in the **ISAKMP** header of the **ISAKMP** message, as specified in [RFC2408] section 4.1.

Main Mode Security Association Database (MMSAD): A database containing the operational state for each **main mode security association (MM SA)**. See section 3.2.1 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://www3.ietf.org/proceedings/02mar/I-D/draft-ietf-ipsec-isakmp-gss-auth-07.txt

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

[IANAIPSEC] Internet Assigned Numbers Authority, "Attribute Assigned Numbers", 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", July 2006.

[MS-DTYP] Microsoft Corporation, "Windows Data Types", March 2007.

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

[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

[RFC3505] Eastlake, D., "Electronic Commerce Modeling Language (ECML): Version 2 Requirements", RFC 3505, March 2003, http://www.ietf.org/rfc/rfc3505.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

1.2.2 Informative References

[DRAFT-NATT] Kivinen, T., Huttunen, A., Swander, B., and Volpe, V., "Negotiation of NAT-Traversal in the IKE", June 2002, http://tools.ietf.org/draft/draft-ietf-ipsec-nat-t-ike/draft-ietf-ipsec-nat-t-ike-02.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-ERREF] Microsoft Corporation, "Windows Error Codes", January 2007.

[MSFT-WLBS] Microsoft Corporation, "MS Windows NT Load Balancing Service (WLBS)", January 1999, http://www.microsoft.com/technet/archive/winntas/downloads/wlbs22.mspx

[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

[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., DiBurro, L., and Stenberg, M., "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

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

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005, http://www.ietf.org/rfc/rfc4309.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

[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 Protocol Overview (Synopsis)

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

The IKE Protocol version 1 is specified in [RFC2409], but it is intimately tied to [RFC2407] and [RFC2408]. In addition, IKE is by far the most commonly implemented protocol making use of [RFC2407] and [RFC2408]. This is further evidenced by the fact that version 2 of the IKE protocol is specified by a single RFC [RFC4306]. For these reasons, it is a commonly accepted industry practice to use the term IKE to collectively refer to [RFC2407], [RFC2408], [RFC2409], and more recently, [RFC3947].

In the remainder of this document, the term IKE collectively applies to [RFC2407], [RFC2408], [RFC2409], and [RFC3947]. Where applicable, the appropriate section of each individual 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. The extensions implemented by Windows are detailed in <1> .

1.3.1 Network Address Translation Traversal (NAT-T)

In the original **IPsec** specifications, the interposition of 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 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.

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 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 only negotiates the interpretation of the NAT-T IKE parameter numbers.

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

1. The host signals which revision(s) 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 3.3.

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. This fragmentation typically happens with IKE messages that contain **certificate chains**. Fragmented UDP packets are commonly blocked by firewalls and routers in an attempt to avoid fragmentation-based attacks. 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 supporting IKE fragmentation advertise this capability through a "FRAGMENTATION" vendor ID payload, see section 1.7, Capability Negotiation. If both peers support fragmentation, a fragmentation timer is started whenever a message is sent. If the timer ever expires, it is assumed that the message associated with the timer never reached its destination because it was too large to traverse the intervening network. In this case, the message is split into a number of small fragments, and all of these small fragments are sent.

For the destination host to 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 3.4.

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 a "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 3.5.

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 the IKE session may be reallocated to a different node within the cluster. The server uses a "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 in the event of a failover.

For more information on clusters based on virtual IP addresses, see [MSFT-WLBS]. For specifications, see sections 3.6 and 3.7.

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 clear text 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 clear text. 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 clear text, 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 clear text 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 clear-text packet should be accepted.

For details, see section 3.8.

1.3.6 Reliable Delete

This extension enables a peer to reliably confirm the deletion of a security association 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 IKEDelete 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 3.9.

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 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 3.10.

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] and also applies to IKE.<2>

1.3.9 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]
NAT-T	(1)	(2) (3)		(7)
IKE Fragmentation		(3)	(8)	
CGA Authentication	(4) (5)	(3)	(9)	

IKE extension	Extends [RFC2407]	Extends [RFC2408]	Extends [RFC2409]	Extends [RFC3947]
Fast Failover		(3)	(10)	
Negotiation Discovery		(3) (6)	(10)	
Reliable Delete			(11)	
Denial of Service Protection		(6)	(12)	

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.

1.4 Relationship to Other Protocols

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

1.5 Prerequisites/Preconditions

The following sections describe the prerequisites and preconditions for using the Internet Key Exchange 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 MUST validate 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 MUST 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 routers 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.

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, as specified in [RFC2408] section 3.1. IKEv1 uses protocol version 1.0.
- **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

The following sections specify how Internet Key Exchange (IKE) Protocol Extensions messages are transported and message syntax.

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 an NAT has been detected, as specified in [RFC3947] section 3.2, but MAY be run over a different port otherwise.<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 one-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 on an alternative set of payload type values, see [DRAFT-NATT].<7>

The following table describes the NAT Discovery (NAT-D) payload type:

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

The following table describes the supported NAT Original Address (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 on 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

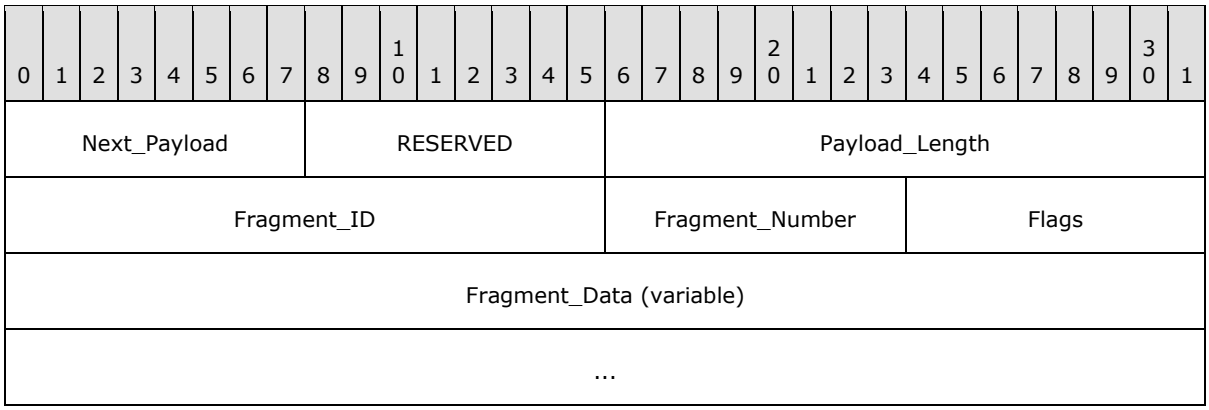
An IKE message fragment contains:

- 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 with this payload type.

The following illustration describes the Fragment Payload Packet.



Next_Payload: 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: This field MUST be set to zero. The responder MUST ignore this field on receipt. This behavior is identical to IKE.

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

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

Fragment_Number: 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. Since 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: The flag field MUST have the following value.

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

All other flag fields MUST be set to zero on the initiator and ignored on the responder.

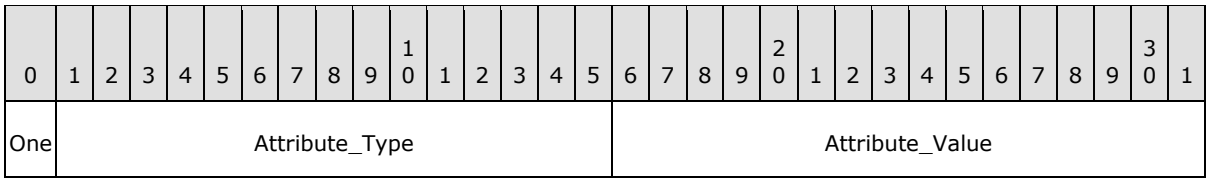
Fragment_Data: The Fragment Data field MUST contain the fragment. The size of the Fragment Data field MUST be computed by subtracting the size of the Fragment Payload header (8 bytes) from the value of the **Payload Length** field.

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.



One: This field MUST be set to 1.

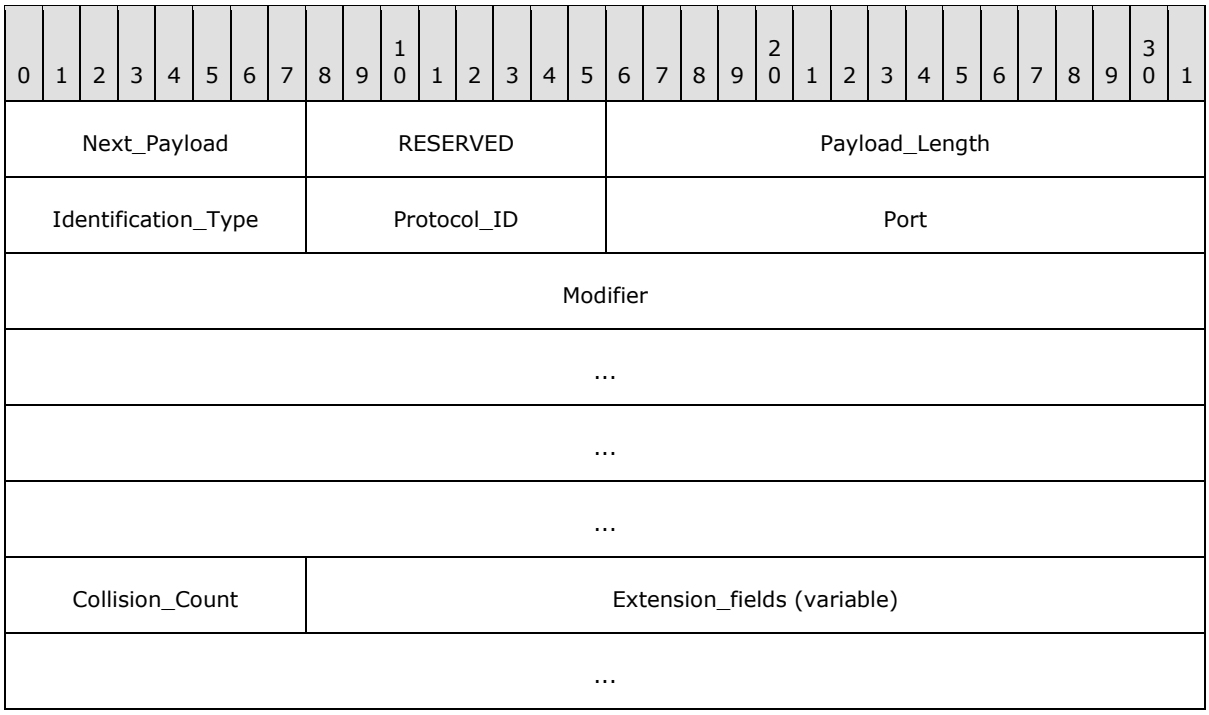
Attribute_Type: For the AUTH_CGA authentication method, this field MUST be set to the value 0x0003. This value corresponds to the authentication method, as specified in [RFC2409] Appendix A.

Attribute_Value: 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.



Next_Payload: 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: This field MUST be set to zero. The responder MUST ignore this field on receipt. This behavior is identical to IKE.

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

Identification_Type: 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: This field MUST be set to zero. The responder MUST ignore this field on receipt. This is identical to IKE.

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

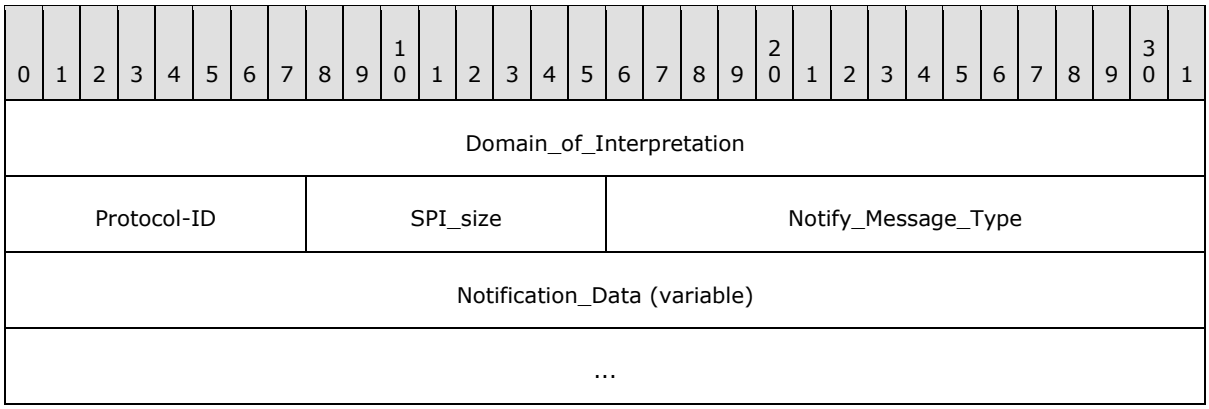
Modifier: This field MUST be as specified in [RFC3972] section 3.

Collision_Count: This field MUST be as specified in [RFC3972] section 3.

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

2.2.6 Notify Payload Packet

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



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

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

SPI_size: This field MUST be as specified in [RFC2408] section 3.14.

Notify_Message_Type: 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 used to request the peer to delete its state for the corresponding security association (SA).
0x9C44	NOTIFY_DOS_COOKIE (check) This notify message type is used by the denial of service protection extension.
0x9C45	EXCHANGE_INFO This notify message type is used by the negotiation discovery extension.

Notification_Data: This field's contents are dependent on the notify message type field. The following list describes the field's contents for various notify message types. The field's contents MUST correspond to the Notify Message type as follows:

- NOTIFY_STATUS (4 Bytes): MUST be a status code indicating success or 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:

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

Value	Meaning
	This flag is used by the negotiation discovery extension.

3 Protocol Details

The following sections specify protocol details, including abstract data models and message processing rules that are common and are specific to NAT-T, IKE fragmentation, CGAs, the fast failover client, the fast failover server, negotiation discovery, reliable delete and Denial of Service (DOS) protection.

3.1 Basic 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]; and "Negotiation of NAT-Traversal in the IKE", as specified in [RFC3947].

3.1.1 Abstract Data Model

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

3.1.2 Timers

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

3.1.3 Initialization

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

3.1.4 Higher-Layer Triggered Events

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

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) which 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 MAY send diagnostic notifications unprotected by a SA.

[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, certificate payloads MUST be in the ID exchange.

[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 specified by this document does not set a timer in every case.<10>

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

[RFC3947]: Message processing MUST be as specified in [RFC3947], with the following exception.

[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 NATOA if the host is behind a NAT.

3.1.6 Timer Events

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

3.1.7 Other Local Events

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

3.2 Common Details

3.2.1 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 behavior is consistent with that described in this document.

The following are main data elements that any implementation requires:

- **Main Mode Security Association Database (MMSAD)**: A database containing 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 necessary for managing the IKE MM SA, without extensions.
  - All states necessary for management of other IKE extensions for the SA, as specified in this section and in 3.7.1.
  - For the NAT-T extension:

        - Selected Revision: A flag which 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].
    - For the IKE Fragmentation extension:
        - Fragmentation Supported: A flag which MUST be set if the peer supports receiving fragmented messages.
        - Fragmentation Active: A flag which MUST be set if the IKE messages MUST be fragmented.
        - Fragment Queue: A queue holding the fragments that correspond to incomplete IKE messages, indexed by 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 these values, see section 2.2.4.
    - For the CGA Authentication extension:
        - CGA_CAPABLE: A flag indicating if Authentication Type 0xFDED MUST be interpreted as the AUTH_CGA authentication method.
    - For the negotiation discovery extension, the following information MUST be maintained:
        - Negotiation Discovery Supported: A flag which MUST be set if the peer supports negotiation discovery.
    - For the fast failover extension, the following information MUST be maintained:
        - Fast Failover: A flag indicating that the "NLBS_PRESENT" vendor ID has been received from the peer for this MM SA. For details, see section 3.7.4.1.

    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 Authentication 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 referred to in this specification contains rules describing if and how IKE should negotiate SAs with a remote peer, as specified in [RFC4301].
    - For the CGA Authentication extension, the following information MUST be maintained:
        - Policy information specifying if and how CGA authentication SHOULD be used.

    The PAD MUST be looked up using tuples 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. As specified in [RFC4301], the SPD 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 using tuples comprised of flow information (that is, source and destination IP addresses, port numbers, and protocol) for the packet.

For the Negotiation Discovery extension 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 which MUST be set if clear text is accepted for this rule.
- A policy flag which MUST be set if encryption is guaranteed for this traffic.

- 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.

  For each QM SA:

  - For the Negotiation Discovery extension, the following information MUST be maintained:
    - Boundary flag: A flag which MUST be set if the QM SA matches an inbound negotiation discovery rule on the remote host.
    - Guaranteed Encryption flag: A flag which 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: A table containing the following information for each flow (indexed by the flow source and destination addresses, port numbers, and protocol):

  For the Negotiation Discovery extension:

  - Secure flag: A flag which MUST be set if one or more packets for this flow have been sent over a QM SA.
  - Guaranteed Encryption flag: A flag which MUST be set if encryption is guaranteed for this flow.
  - Acquire flag: A flag which 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 clear text.

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

- For the IKE Fragmentation extension the following information MUST be maintained:
  - Fragment ID counter: A 16-bit counter. A Fragment ID counter MAY be implemented per MM SA or as a global counter.<11>

- For the Denial of Service Protection extension, the following information MUST be maintained:
  - Denial of Service Protection is active.

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

3.3 NAT Traversal Details

Using the notation specified in [RFC2409] section 3.2, the generalized form of an IKE phase 1 exchange using NAT-T is the following, as specified in [RFC3947] section 3.2:

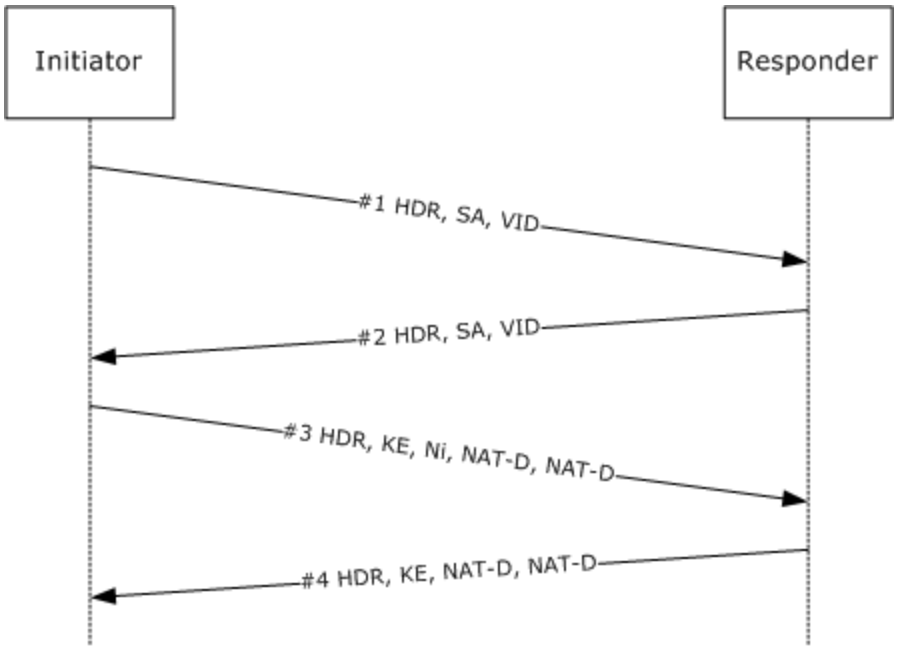


Figure 1: IKE phase 1 exchange using NAT-T

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 specified in [DRAFT-NATT], rather than those specified in [RFC3947]. Both revisions include negotiation of a choice of revision supported by both peers.<12> For more information, see [DRAFT-NATT].

3.3.1 Abstract Data Model

See section 3.2.1.

3.3.2 Timers

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

3.3.3 Initialization

None.

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), 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>

3.3.5 Message Processing Events and Sequencing Rules

3.3.5.1 Receiving Message #1

On receipt of message #1, an NAT-T-supporting host MUST check for the presence of the NAT-T vendor ID payloads that are specified in section 3.3.4.1. 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 3.3.4.

3.3.5.2 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 3.3.5.1.

3.3.5.3 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.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

3.4 IKE Fragmentation Details

Using the notation as specified in [RFC2409] section 3.2, the generalized form of an IKE phase 1 exchange is the following (for more information, see [RFC4309] section 5):

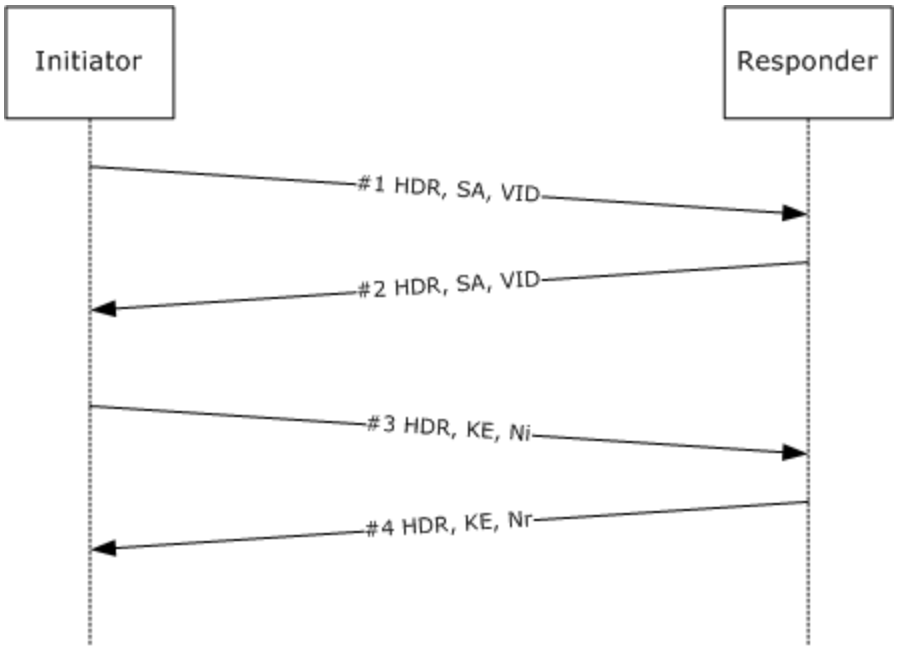


Figure 2: IKE phase 1 exchange

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

3.4.1 Abstract Data Model

See section 3.2.1.

3.4.2 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.<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.<16>

3.4.3 Initialization

The Fragment ID counter MUST be set to zero.

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 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.

3.4.5 Message Processing Events and Sequencing Rules

3.4.5.1 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.

3.4.5.2 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.

3.4.5.3 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 then MUST check if all Fragment payloads for this Fragment ID have been received (that is, if Fragment payloads with 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.

Then, the host MUST stop the fragment reassembly timer and process the reassembled IKE message as a normal 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 3.4.6.1. If the Fragmentation Active flag is not set, the host MUST start the fragmentation timer for the message it is about to send.<18>

3.4.6 Timer Events

3.4.6.1 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; so the size of each fragment (after adding IP, UDP, and ISAKMP headers) is 576 bytes for IPv4 and 1,280 bytes for IPv6.
- For each fragment, a message MUST be constructed in the following way:
  - The ISAKMP header of the original IKE message with the Next Payload field set to the Fragment payload and the Encrypted flag cleared (as specified in [RFC2408] section 3.1).
  - A Fragment payload header with 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 on Fragment number n.

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

3.4.6.2 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.

3.4.7 Other Local Events

None.

3.5 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 the following (for more information, see [RFC4309] section 5.1):

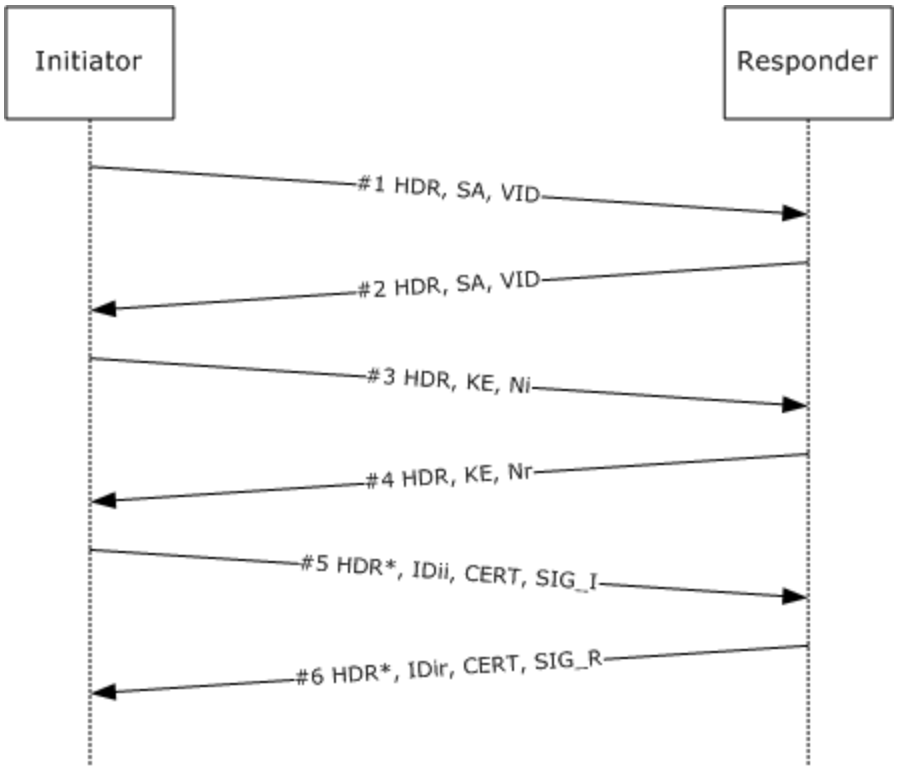


Figure 3: IKE phase 1 exchange using certificates

The CGA Authentication Protocol extension uses the same exchanges. The description in this section uses the message numbers from the protocol sequence diagram.

3.5.1 Abstract Data Model

See section 3.2.1.

3.5.2 Timers

None.

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, 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.

3.5.5 Message Processing Events and Sequencing Rules

3.5.5.1 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, 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.

3.5.5.2 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. The host then MUST construct message #3, as specified in [RFC2409] section 5.1.

3.5.5.3 Receiving Message #3

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

3.5.5.4 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 CERT payload MUST contain the self-signed certificate that corresponds to the CGA.

3.5.5.5 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 3.5.5.4.

3.5.5.6 Receiving Message #6

On receipt of message #6, the host MUST validate the message using the procedure specified for validating message #5 in section 3.5.5.5.

3.5.6 Timer Events

None.

3.5.7 Other Local Events

None.

3.6 Fast Failover Client Details

Using the notation as specified in [RFC2409] section 3.2, the generalized form of an IKE phase 1 exchange is the following (for more information, see [RFC4309] section 5):

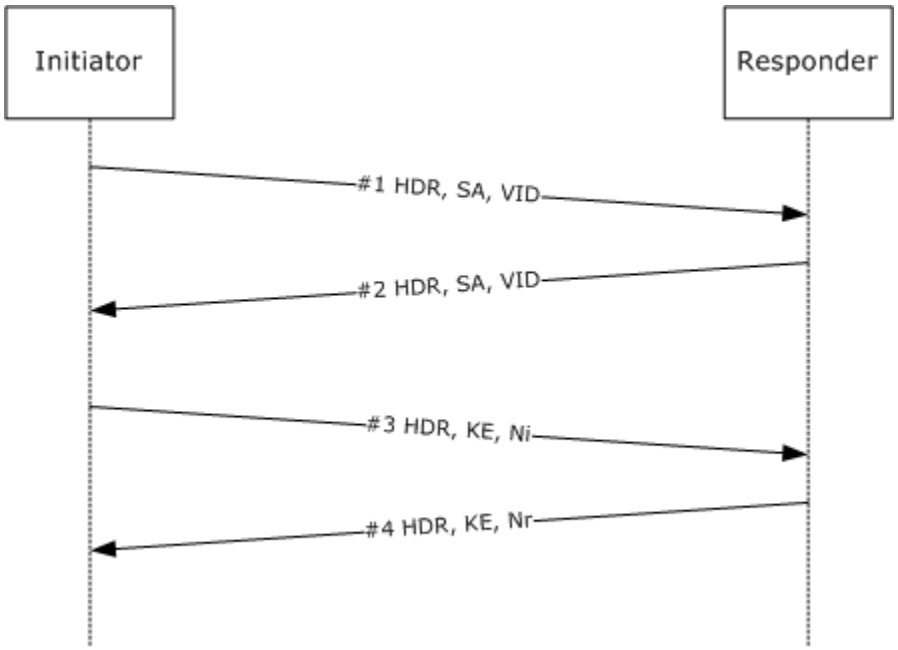


Figure 4: IKE phase 1 exchange

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

3.6.1 Abstract Data Model

See section 3.2.1.

3.6.2 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 3.6.7.1). This timer MUST be set when the QM SA has been negotiated.<20>

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 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 "IKE CGA version 1", 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 new connection attempts cause the retransmission of SYN packets.<21>

3.6.5 Message Processing Events and Sequencing Rules

3.6.5.1 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 3.7.4.1). 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.

3.6.5.2 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 3.7.4.1). 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.

3.6.6 Timer Events

3.6.6.1 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.

3.6.7 Other Local Events

3.6.7.1 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.<22>

3.7 Fast Failover Server Details

The description in this section uses the message numbers from the protocol sequence diagram in section 3.6.

3.7.1 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.

The data elements any implementation requires include:

- 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 3.2.1 and 3.7.1.
  - Initial Contact: A flag indicating if the "Vid-Initial-Contact" Vendor ID payload (see section 3.6.4.1) 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 above conceptual data can be implemented using a variety of techniques. An implementation is at liberty to implement such data in any way it pleases.

3.7.2 Timers

None.

3.7.3 Initialization

None.

3.7.4 Higher-Layer Triggered Events

3.7.4.1 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).

3.7.5 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 "Vid-Initial-Contact" vendor ID (as specified in section 3.6.4.1). 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.<23>

3.7.5.2 Receiving Message #2

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

3.7.6 Timer Events

None.

3.7.7 Other Local Events

None.

3.8 Negotiation Discovery Details

Using the notation as specified in [RFC2409] section 3.2, the generalized form of an IKE phase 1 (the MM) exchange is the following (for more information, see [RFC4309] section 5):

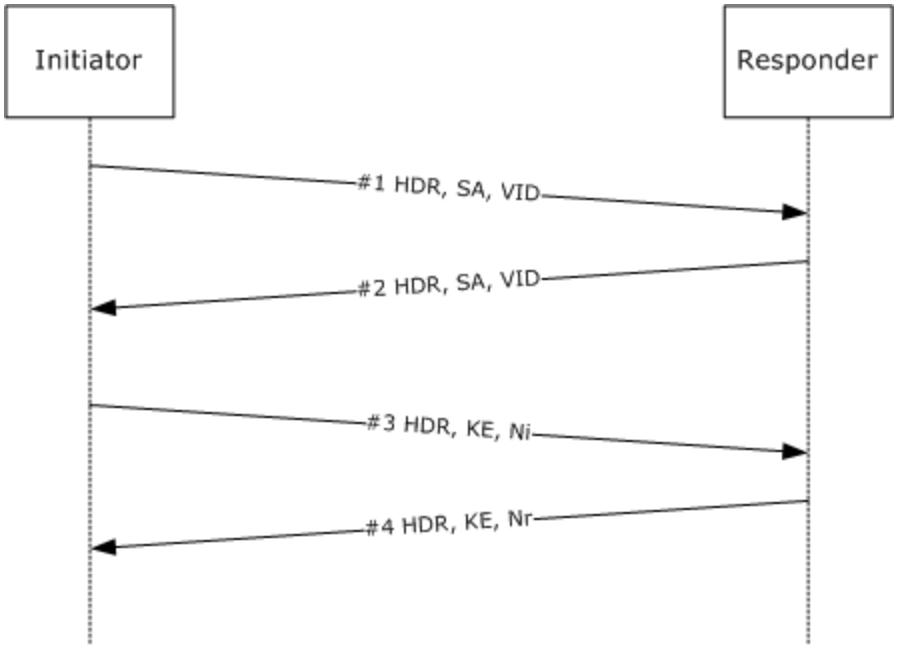


Figure 5: IKE phase 1 (MM) exchange

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 (the QM) exchange is the following (for more information, see [RFC4309] section 5.5):

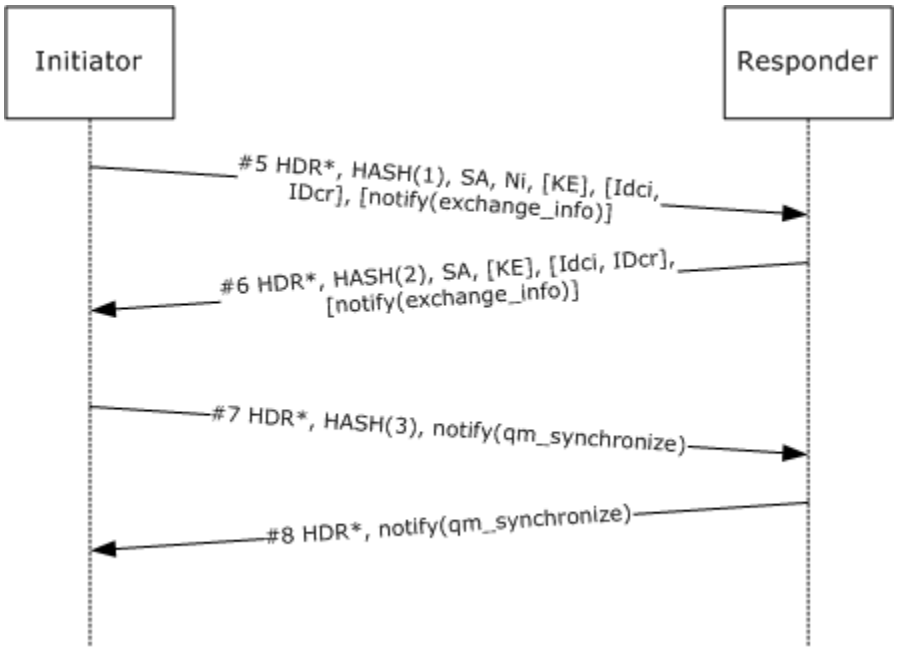


Figure 6: IKE phase 2 (QM) exchange

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

3.8.1 Abstract Data Model

See section 3.2.1.

3.8.2 Timers

None.

3.8.3 Initialization

None.

3.8.4 Higher-Layer Triggered Events

3.8.4.1 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:

  Domain of interpretation (DOI) (4 bytes): 0x00000001 (IPSEC_DOI), as specified in [RFC2408] section A.2.

  Protocol ID (1 byte): 0x02 (QM notification).

  Flag (1 byte): 0x00.

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

  Notification Data (4 bytes): 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.

  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 with the Boundary flag set, the host MUST send the packet in clear text. For more information, see [RFC3948].
  - Otherwise, the IPsec implementation MUST send the packet encapsulated by using the matching QM SA, and 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].

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.

3.8.4.2 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 clear text, one of the following occurs.
  - 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 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 plain text, 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.

3.8.5 Message Processing Events and Sequencing Rules

3.8.5.1 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 3.8.4.1). 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.

3.8.5.2 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 3.8.4.1) 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.

3.8.5.3 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 #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 for which the QM SA is negotiated has the Boundary flag set, the host MUST add a notification payload with the following fields:

  Domain of Interpretation (DOI) (4 bytes): 0x00000001 (IPSEC_DOI), as specified in [RFC2408] section A.2.

  Protocol ID (1 byte): 0x02 (QM notification).

  Flag (1 byte): 0x00.

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

  Notification Data (4 bytes): 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.

3.8.5.4 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.
- IKE_EXCHANGE_INFO_GUARANTEE_ENCRYPTION: If this flag is set, the host MUST set the Guaranteed Encryption flag for the QM SA.

Messages #7 and #8 are constructed and processed as specified in [RFC2409] section 5.5.

3.8.6 Timer Events

None.

3.8.7 Other Local Events

None.

3.9 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 the following (for more information, see [RFC4309] section 5):

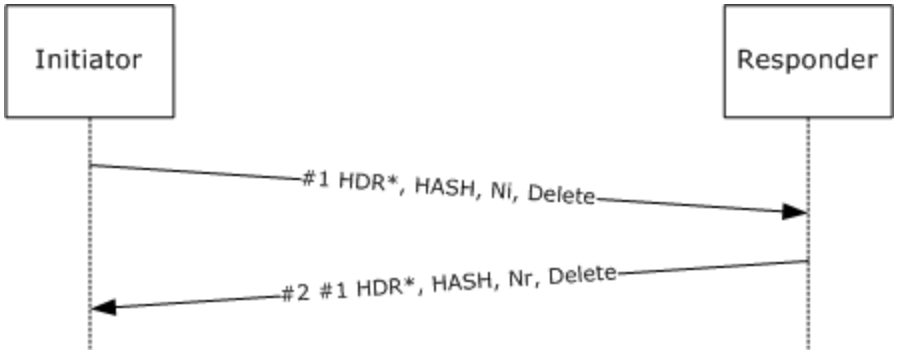


Figure 7: IKE Delete exchange

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

3.9.1 Abstract Data Model

See section 3.2.1.

3.9.2 Timers

Delete the 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 3.9.4.1 and 3.9.6.1.

3.9.3 Initialization

None.

3.9.4 Higher-Layer Triggered Events

3.9.4.1 SA Deletion

When some events, such as communication events, are compromised, the host MAY have to delete SAs, as specified in [RFC2408] section 5.15. The host MUST then construct message #1 as follows:

- Message #1 MUST consist only of an ISAKMP header, a Hash payload, a Delete payload, and a Nonce payload structured as specified in [RFC2408] section 3.15.<24>
- The ISAKMP header MUST be constructed as specified in [RFC2409] section 5.7.
- The Hash payload MUST be constructed as specified in [RFC2409] section 5.7.
- The Delete payload MUST be constructed as specified in [RFC2408] section 3.15.
- The Ni payload is a Nonce payload and MUST be constructed as specified in [RFC2408] section 3.13.

The host MAY then start the delete retransmission timer, setting it to expire in two seconds.<25>

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 delete the corresponding SA and MUST construct message #2 in response.

- The message MUST consist only of an ISAKMP header, a Hash payload, a Delete payload, and a Nonce payload structured as specified in [RFC2408] section 3.15.
- The ISAKMP header MUST be constructed as specified in [RFC2409] section 5.7. The Message ID field MUST be copied from message #1.
- The Hash payload MUST be constructed as specified in [RFC2409] section 5.7.
- 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.

3.9.5.2 Receiving Message #2

On receipt of message #2, the host MUST validate the message in the following way:

- 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.

3.9.6 Timer Events

3.9.6.1 Expiration of the Delete Retransmission Timer

When this timer expires, the initiator MUST retransmit message #1, as specified in section 3.9.4.1, and SHOULD reset the timer to double the previous duration unless a total of four retransmissions have 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.<26>

3.9.7 Other Local Events

3.9.7.1 Shutdown

Internet Key Exchange (IKE) protocol shutdown.<27>

3.10 Denial of Service Protection Details

IKE goes into Denial of Service (DOS) protection under the condition described in section 3.10.7.

Using the notation, as specified in [RFC2408] section 4.1.1, the generalized form of an IKE exchange using the Denial of Service Protection extension is the following (for more information, see [RFC4309] section 5):

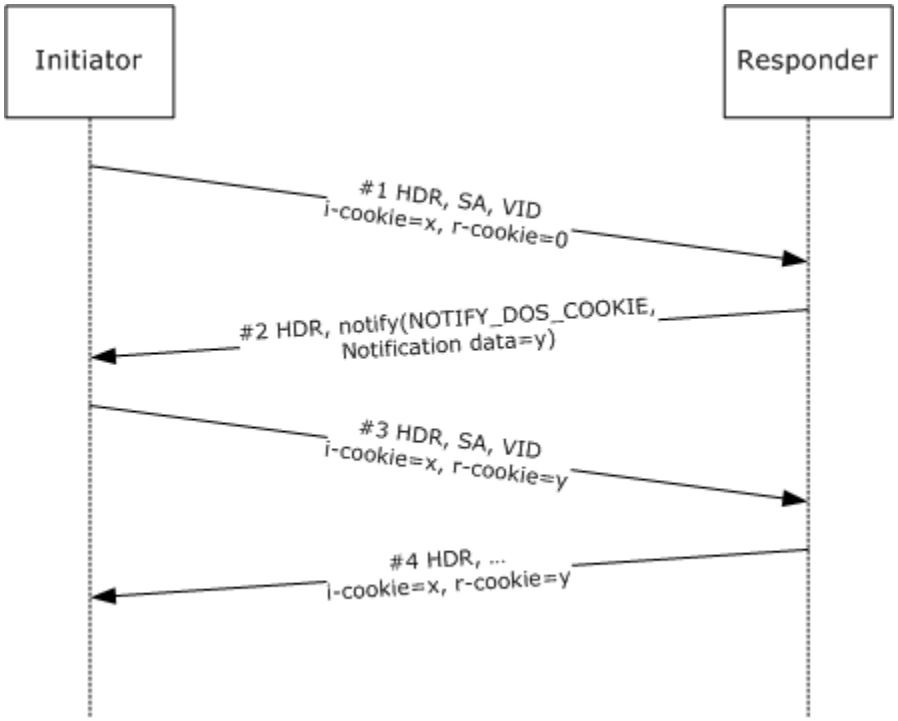


Figure 8: IKE using the DOS Protection extension

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

3.10.1 Abstract Data Model

See section 3.2.1.

An implementation MAY maintain state to implement Denial of Service Protection Mode.<28>

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

3.10.5.1 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:

- The message MUST consist only of a ISAKMP header, and a Notify payload structured as specified in [RFC2408] section 3.14.
- The ISAKMP header MUST be constructed as specified in [RFC2409] section 5.7. The message ID field MUST be copied from message #1.
- 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 in order to provide good denial of service protection.<29>

Then the host MUST silently discard message #1 whether it is correctly validated or not.

3.10.5.2 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 ID in the ISAKMP payload is identical to the message ID from message #1.
- 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 in the following way:

- 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.

3.10.5.3 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.<30>

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

Subsequent messages received for this SA on the host in denial of service 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 the same as message #1.

3.10.6 Timer Events

None.

3.10.7 Other Local Events

Denial of Service 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 Denial of Service Protection mode (see section 3.2 for details). The threshold may be implemented in a number of ways.<31>

4 Protocol Examples

The following section describes a common scenario to illustrate the negotiation discovery function of IKE protocol extensions.

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.

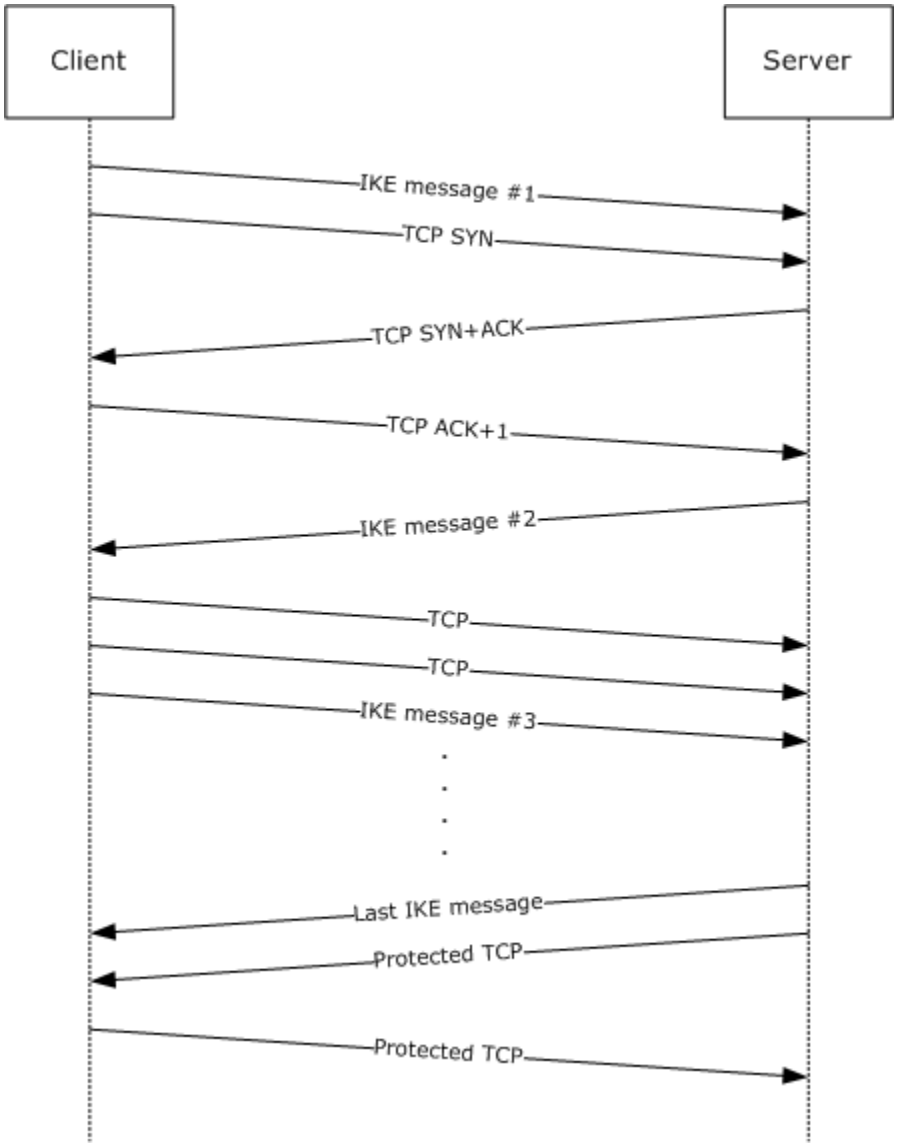


Figure 9: Negotiation discovery between client and server

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 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.

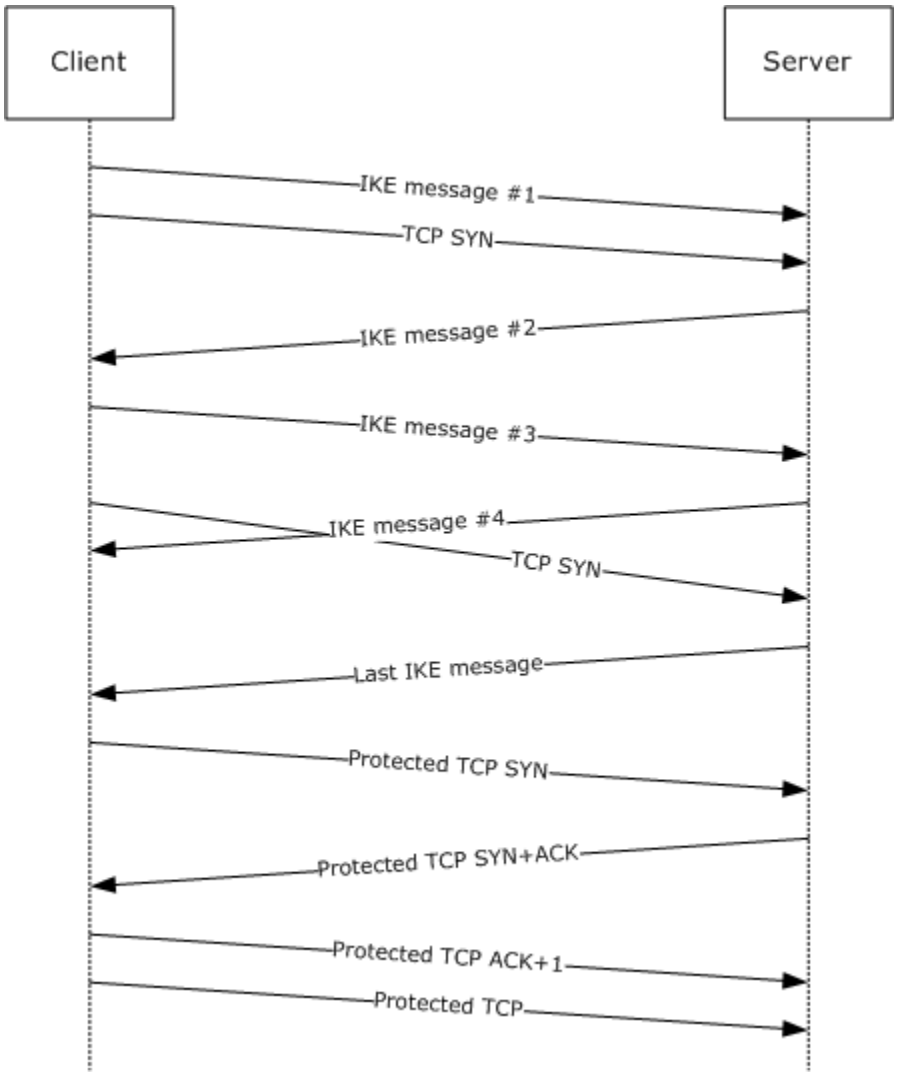


Figure 10: Negotiation discovery between client and server, all inbound traffic protected

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 IKE to the server. The clear text TCP SYN packets are dropped by the server and retransmitted by the client until the IKE negotiation completes with IKE message #6. Then, the server accepts the protected traffic.

5 Security

The following sections specify security considerations for implementers of IKE protocol extensions.

5.1 Security Considerations for Implementers

5.1.1 Negotiation Discovery

Negotiation discovery allows clear text outbound and inbound connections if the peer is not IPsec–capable. Connections that are clear text 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: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows 2000
- Windows XP
- Windows Server 2003
- Windows Vista

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that Windows does not follow the prescription.

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

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	Windows Server 2008
NAT-T		X	X	X	X
IKE Fragmentation		X	X	X	X

IKE extension	Windows XP	Windows XP SP2	Windows Server 2003	Windows Vista	Windows Server 2008
CGA Authentication				X	X
Fast Failover		X	X	X	X
Negotiation Discovery				X	X
Reliable Delete	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, and Windows Server 2008. The following table describes the extensions that are supported by IKE in each release:

IKE extension	Windows XP SP2	Windows Server 2003	Windows Vista	Windows Server 2008
Denial of Service Protection	X	X	X	X
IKE/AuthIP Co-Existence			X	X
Exchange Information Notification Payload			X	X

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

Authentication method	Windows 2000	Windows XP	Windows Server 2003	Windows Vista	Windows Server 2008
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, and Windows Server 2008. The following table lists the parameters that are implemented.

Diffie-Hellman group	Windows 2000	Windows XP	Windows Server 2003	Windows Vista	Windows Server 2008
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	Windows Server 2008
NULL (as specified in [RFC3505])	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	Windows Server 2008
NULL (as specified in [RFC3505])	X	X	X	X	X
DES-CBC (for more information, see [RFC2405])	X	X	X	X	X

Encryption algorithm	Windows 2000	Windows XP	Windows Server 2003	Windows Vista	Windows Server 2008
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, and Windows Server 2008. The Microsoft implementation of IKE supports the following vendor IDs.

The Microsoft implementation vendor ID (the first 5 rows of the below table) 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 denotes the Windows operating system version.

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
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, and Windows Server 2008
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,

Common name	String representation	Wire representation (MD5 hash of string)	Version
			Windows Vista, and Windows Server 2008
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, and Windows Server 2008
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, and Windows Server 2008
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, and Windows Server 2008
NAT-T supported	"RFC 3947"	4A 13 1C 81 07 03 58 45 5C 57 28 F2 0E 95 45 2F	Windows Vista and Windows Server 2008
AuthIP supported	"MS-MamieExists"	21 4C A4 FA FF A7 F3 2D 67 48 E5 30 33 95 AE 83	Windows Vista and Windows Server 2008
CGA supported	"IKE CGA version 1"	E3 A5 96 6A 76 37 9F E7 07 22 82 31 E5 CE 86 52	Windows Vista and Windows Server 2008
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 and Windows Server 2008

<6> Section 2.1: IKE transport port assignments. Implemented in Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. 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, and Windows Server 2008. The NAT-T payload type values used by each Windows version are specified in section 3.3.4.1.

<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, and Windows Server 2008. The UDP encapsulation mode values used by each Windows version are specified in section 3.3.4.1.

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

<10> Section 3.1.5: IKE variant timer. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. The Windows implementation does not set a retransmission timer in the following cases:

- The responder never sets a restransmission timer.
- Sending Notify message to a peer.
- Sending a delete message to a peer that does not support reliable deletes, that is, a peer which has not sent the Microsoft Implementaion Vendor ID.

<11> Section 3.2.1: Fragment ID counter. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. The fragment ID is a monotonically increasing counter value. On Windows 2000, Windows XP, and Windows Server 2003, the counter is maintained for each main mode security association (MM SA). On Windows Vista and Windows Server 2008, the counter is global.

<12> Section 3.3: [RFC3947] implementation. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. Windows Vista and Windows Server 2008 implement both revisions. Windows 2000, Windows XP, and Windows Server 2003 implement the [DRAFT-NATT] revision. For more information, see [DRAFT-NATT].

<13> Section 3.3.2: NAT-T Keep-Alive timer. Implemented in Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. NAT-T keep-alive timer: A keep-alive message is sent every 20 seconds.

<14> Section 3.3.4.1: [NAT-T IKE] message construction. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008.

NAT-T Revision Support	Version
[DRAFT-NATT] and [RFC3947]	Windows Vista and Windows Server 2008
[DRAFT-NATT]	Windows 2000, Windows XP, and Windows Server 2003

<15> Section 3.4.2: Fragmentation timer. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. Fragmentation timer: Four seconds.

- On Windows 2000, Windows XP, and Windows Server 2003, both the initiator and the responder implement a fragmentation timer.
- On Windows Vista and Windows Server 2008, only the initiator implements a fragmentation timer.
- Fragment reassembly timer: 70 seconds.

<16> Section 3.4.2: Fragmentation timer. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. Fragmentation timer: Four seconds.

- On Windows 2000, Windows XP, and Windows Server 2003, both the initiator and the responder implement a fragmentation timer.
- On Windows Vista and Windows Server 2008, only the initiator implements a fragmentation timer.
- Fragment reassembly timer: 70 seconds.

<17> Section 3.4.5.3: Fragmentation Active flag. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. The Fragmentation Active flag is set on receipt of a Fragment payload.

<18> Section 3.4.5.3: IKE Message Fragmentation Active flag behavior. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. IKE messages are fragmented if the Fragmentation Active flag is set, subject to the conditions specified in section 3.4.6.1.

<19> Section 3.4.6.1: Expiration of the fragmentation timer. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. On Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008, IKE fragments only messages that contain the Identification payload, as specified in [RFC2407] section 3.8.

On Windows Vista and Windows Server 2008, AuthIP fragments all messages except QM exchanges.

<20> Section 3.6.2: QM SA idle timer. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. The QM SA idle timer is one minute if the peer has sent a "NLBS_PRESENT" vendor ID payload during the negotiation of the MM SA under which this QM SA was negotiated (as specified in section 3.7.4.1). The QM SA idle timer is five minutes otherwise.

<21> Section 3.6.4.1: Vendor ID payload. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. On Windows Vista and Windows Server 2008, the host sends "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.

<22> Section 3.6.7.1: QM SA idle timer. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. The QM SA idle timer is set to one minute if the Fast Failover flag is set on the parent MM SA, and it is set to five minutes if the Fast Failover flag is not set.

<23> Section 3.7.5.1: Vendor ID processing. Implemented in Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. 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].

<24> Section 3.9.4.1: Nonce. Implemented in Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. Nonces are 32-byte random numbers generated from a FIPS-140 compliant random-number generator (for more information, see [FIPS140]).

<25> Section 3.9.4.1: Retransmission timer triggering. Implemented in Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. The retransmission timer is set only if the host is talking to a Windows peer; that is, the "MS NT5 ISAKMPOAKLEY" vendor ID payload (see section 1.7) has been received from the peer for the corresponding MM SA.

<26> Section 3.9.6.1: Delete Retransmission timer. Implemented in Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. The first retransmission occurs after two seconds. The time-out is doubled for each subsequent retransmission up to a maximum of four retransmissions. In the shutdown phase, at most, one retransmission is performed, as specified in section 3.9.7.1. The timer is started only if the remote host is a Windows peer, as identified by the "MS NT5 ISAKMPOAKLEY" vendor ID payload.

<27> Section 3.9.7.1: Shutdown behavior. Implemented in Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. When the IKE protocol shuts down, IKE sends Delete notification messages for all SAs, as specified in section 3.9.4.1, and then sets the delete retransmission timer to two seconds for each SA. When each timer expires, if a message #2 has not

been received and verified for that SA, as specified in section 3.9.5.2, it retransmits the notification message for that SA without resetting the timer.

<28> Section 3.10.1: Denial of Service Protection Mode state. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. Windows maintains the following state to implement Denial of Service Protection mode:

```
Set cookieKey to 50 bytes of randomly generated data
Set COOKIE_KEY_TIME to 150
```

This state is used by the cookie generation algorithm described in the Windows Behavior Note in section 3.10.5.1

<29> Section 3.10.5.1: Denial of Service Protection Mode cookie generation. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. The Windows implementation uses the following algorithm to generate the cookie (prevTimeSlice is a Boolean input parameter to the algorithm):

```
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
Set Peerport to the 16 bits remote port in
    network order
Set LocalIPaddr to the peer IP address in network order
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 cookie as SHA1(cookieKey concatenated with iCookie concatenated with peerAddr
    concatenated with localAddr concatenated with curTime)
Compute cookie as the first 8 bytes of cookie
```

<30> Section 3.10.5.3: Denial of Service Protection Mode cookie validation. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. The Windows implementation checks the validity of the responder cookie field by regenerating the cookie using the algorithm specified in the Windows Behavior note in section 3.10.5.1. The algorithm is the following:

```
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
```

<31> Section 3.10.7: Denial of Service Protection Threshold. Implemented in Windows 2000, Windows XP, Windows Server 2003, Windows Vista, and Windows Server 2008. Windows goes into Denial of Service Protection mode if the number of negotiations for which only one message has been received from any initiator is above 500 and goes out of Denial of Service Protection mode if the number of negotiations for which only one message has been received from any initiator is below 100.

7 Index

I

L

M

N

O

P

Q

R

S

T

V

W