[MS-AIPS]:
Authenticated Internet Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
03/14/2007	1.0		Version 1.0 release
04/10/2007	1.1		Version 1.1 release
06/08/2007	2.0	Major	Updated and revised the technical content.
07/10/2007	3.0	Major	Updated and revised the technical content.
08/17/2007	4.0	Major	Revised packet names.

Date	Revision History	Revision Class	Comments
09/21/2007	5.0	Major	Updated and revised the technical content.
10/26/2007	6.0	Major	Updated and revised the technical content.
01/25/2008	7.0	Major	Updated and revised the technical content.

Table of Contents

1 Introduction

The Authenticated Internet Protocol is a protocol derived from the **Internet Key Exchange (IKE)** Protocol as specified in [RFC2409]. This protocol supports a more generalized authentication **exchange** than IKE, in addition to optimizations in key exchange and policy discoverability.

1.1 Glossary

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

Authentication Header (AH)
Authentication Mode
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)
Mutual Authentication
Negotiation
Negotiation Discovery
One-way Authentication
Perfect Forward Secrecy
Phase
Quick Mode (QM)
Quick Mode Security Association (QM SA)
Responder
Security Association (SA)
Security Association Database (SAD)
Security Policy Database (SPD)
Security Principal
Security Principal Name (SPN)
Security Support Provider Interface
Transport Mode
Tunnel Mode

The following terms are specific to this document:

Authenticated Internet Protocol (AuthIP): A protocol derived from the **Internet Key Exchange (IKE)** Protocol as specified in [RFC2409].

Anonymous Authentication: An **authentication mode** in which neither party verifies the identity of the other party.

Extended Mode (EM): An optional phase of **Authenticated Internet Protocol (AuthIP)** **negotiation** during which the peers perform a second round of authentication. This **phase** does not exist in the **Internet Key Exchange (IKE)** protocol.

Main Mode Security Association Database (MMSAD): A database containing operational state for each **Main Mode (MM) security association (SA)**. For more information, see 3.1.1.

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.

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

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

[MS-IKEE] Microsoft Corporation, "Internet Key Exchange Protocol Extensions", July 2006.

[MS-SECO] Microsoft Corporation, "Windows Security Overview", December 2006.

[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

[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

[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

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

[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

[SP800-56A] Barker, E., Johnson, D., and Smid, M., "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", March 2006, http://csrc.nist.gov/groups/ST/toolkit/documents/SP800-56Arev1_3-8-07.pdf

1.2.2 Informative References

[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

[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

[FIPS180] Federal Information Processing Standards Publication, "Secure Hash Standard", FIPS PUB 180-1, April 1995, http://www.itl.nist.gov/fipspubs/fip180-1.htm

[MS-NLMP] Microsoft Corporation, "NT LAN Manager (NTLM) Authentication Protocol Specification", July 2006.

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions", July 2006.

[MSWFPSDK] Microsoft Corporation, "Windows Filtering Platform", http://msdn2.microsoft.com/en-us/library/aa366510.aspx

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996, http://www.ietf.org/rfc/rfc1964.txt

[RFC2104] Krawczyk, H., Bellare, M., and Canetti, R., "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997, http://www.ietf.org/rfc/rfc2104.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

[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

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

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000, http://www.ietf.org/rfc/rfc2743.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

[RFC3546] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and Wright, T., "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003, http://www.ietf.org/rfc/rfc3546.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

[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

[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

[RFC4120] Neuman, C., Yu, T., Hartman, S., and Raeburn, K., "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005, http://www.ietf.org/rfc/rfc4120.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

[RFC4305] Eastlake III, D., "Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4305, December 2005, http://www.ietf.org/rfc/rfc4305.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

1.3 Protocol Overview (Synopsis)

The Authenticated Internet Protocol is a keying protocol designed as a set of extensions to Internet Key Exchange (IKE) v1 as specified in [RFC2409] or IKE v2. For more information on IKE v2, see [RFC4306] for the purpose of keying **Authentication Header (AH)** [RFC4302] and **Encapsulating Security Payload (ESP)** [RFC4303] **security associations (SAs)**. The Authenticated Internet Protocol extends IKE in the areas outlined as follows:

- Two rounds of authentication: The Authenticated Internet Protocol adds an **extended mode (EM)** exchange to IKE during which a second set of credentials (for example, user credentials) can be negotiated. IKE v1 allows only a single round of authentication. IKE v2 can support multiple rounds of authentication by using Extensible Authentication Protocol (EAP) as specified in [RFC4306] section 3.16.

- Per **flow** credentials: The Authenticated Internet Protocol allows credentials to be negotiated on a per flow basis. IKE v1 and IKE v2 allow only a single set of credentials for all flows between two peers.
- Authentication method retry: All the mutually acceptable authentication methods are tried in sequence until one succeeds or all fail. In IKE v1, the first authentication failure causes the entire **negotiation** to fail. IKE v2 has the same behavior for the non-EAP methods. Authentication retry for EAP is implementation dependent.
- One-way trust: The Authenticated Internet Protocol allows one-way trust between the peers. IKE v1 always requires **mutual authentication**. IKE v2 can support **one-way authentication** by using EAP methods that support one-way authentication.
- Optimized exchanges: The **quick mode (QM)** exchange can overlap with the end of the **main mode (MM)** exchange. In the optimal configuration, two roundtrips are sufficient to establish quick mode security associations (SAs). IKE v1 requires three roundtrips for the same negotiation. IKE v2 requires only two roundtrips in the optimal configuration.
- NAT traversal: The Authenticated Internet Protocol uses [RFC3947] when operating over Internet Protocol (IP) v4 and when one or both peers are behind a Network Address Translation (NAT).

In both the Authenticated Internet Protocol and IKE, each negotiation is composed of a series of one or more exchanges. An exchange consists of one request from the **initiator**, followed by one response from the **responder**. The initiator starts all exchanges. The roles do not reverse during a negotiation. The initiator and responder go through successive exchanges until the protocol successfully terminates (and **quick mode security associations (QM SAs)** are created) or fails (and all corresponding states are deleted). During **phase** 1 as specified in IKE of the protocol, a **main mode security association (MM SA)** is established. The MM SA is used to encrypt further Authenticated Internet Protocol and IKE traffic. Multiple negotiations can occur over the lifetime of an MM SA.

In both the Authenticated Internet Protocol and IKE, the initiator and responder roles are determined dynamically based on higher-layer events and local policy. After an MM SA has been established between peers, either peer may function as the initiator or the responder in subsequent negotiations over the lifetime of the MM SA.

An Authenticated Internet Protocol implementation may support Internet Key Exchange Protocol Extensions specified in [MS-IKEE].<1>

1.4 Relationship to Other Protocols

The Authenticated Internet Protocol operates on top of the UDP protocol as specified in [RFC768]. This protocol extends Internet Key Exchange (IKE) v1 and retains most IKE v1 structure and message formats. Authenticated Internet Protocol semantics can be used instead of or in cooperation with IKE v1 and/or IKE v2 between peers when keying authentication header (AH) and Encapsulating Security Payload (ESP) security associations (SAs).

For authentication purposes, the Authenticated Internet Protocol depends on **Generic Security Services (GSS)**-API as specified in [MS-SECO], [RFC2743], and [GSS]. Authentication methods used may introduce dependencies on other protocols such as Kerberos, Transport Layer Security (TLS) and [MS-NLMP]. For more information, see [RFC4120], [RFC1964], [RFC3546] and [MS-NLMP].

The GSS-API implementation details are specified in [MS-SECO].

1.5 Prerequisites/Preconditions

The Authenticated Internet Protocol requires that both the initiator and the responder have user-defined type (UDP) connectivity. The protocol also assumes that the initiator knows the responder's IP address through a manual configuration or through the policy lookup as specified in [RFC4301] section 4.4, in the case of **tunnel mode**.

Authenticated Internet Protocol authentication is performed by exchanging Generic Security Services GSS-API messages. GSS-API support MUST be available on both the initiator and responder.

As with Internet Key Exchange (IKE) v1, successful establishment of a quick mode security association (QM SA) by using the Authenticated Internet Protocol requires that the initiator and the responder have at least one common authentication method and a common set of cryptographic parameters for the main mode security association (MM SA) and QM SA.

1.6 Applicability Statement

The Authenticated Internet Protocol extends Internet Key Exchange (IKE) v1 and can be enabled for negotiating any **Internet Protocol Security (IPsec) transport mode** or tunnel mode security associations (SAs).

Per-flow credentials can be used to implement fine-grained access control on the responder, for example per connection/per user. This is not possible in IKE v1 or IKE v2.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- Protocol Versions: The protocol version is part of the **Internet Security Association and Key Management Protocol (ISAKMP)** header as specified in [RFC2408] section 3.1. Internet Key Exchange (IKE) v1 and Authenticated Internet Protocol both use version 1.0.
- Exchange Modes and Authentication:

  The Authenticated Internet Protocol supports the following varieties of **anonymous**, one-way, and mutual authentication, which depend on the exchange mode (main mode MM or extended mode (EM)) in use.

Exchange Mode	Authentication Modes
Main mode (MM)	Anonymous, one-way, and mutual
Extended mode (EM)	One-way and mutual

  EM does not support an anonymous authentication option because an anonymous authentication in EM would be the same as EM authentication not performing at all. In MM, anonymous authentication mode is accompanied by Diffie-Hellman exchange. The vendor can implement any authentication algorithm.<2>
- Cryptographic Parameters:

  Cryptographic parameters are negotiated in different phases of the protocol (MM, and QM). The algorithm and parameter numbers are specified in [IANAIPSEC] and [IANAISAKMP]. The vendor can implement any cryptographic algorithms.<3>

- Capability Negotiation: Similar to IKE, the Authenticated Internet Protocol can advertise specific capabilities through vendor ID payloads as specified in [RFC2408] section 3.16.<4>

  Vendors can create unique vendor IDs for their specific implementation.<5>

- IKE v1 Co-Existence

  When a peer supports both IKE v1 and the Authenticated Internet Protocol, it MUST determine which protocol to use in negotiating SAs that are based on the capabilities of its remote peer. To determine whether a remote peer supports the Authenticated Internet Protocol when initiating a negotiation, the initiator SHOULD send the following two back-to-back messages to the peer:

  - Message 1: The first message of an MM Authenticated Internet Protocol exchange, as defined in section 3.1.8.2.1.
  - Message 2: The first message sent by an IKE v1 initiator, as defined in [RFC2409] section 5. This message MUST contain the "MS-MamieExists" IKE vendor ID payload.

  If a responder supports the Authenticated Internet Protocol and it receives message 1, it MUST use this message to carry out the negotiation by responding as specified in section 3.1.8.2.1. When message 2 arrives at the responder after message 1, the responder MUST silently discard it.

  If a responder supports the Authenticated Internet Protocol and it receives message 2, it MUST conclude that the initiator supports the Authenticated Internet Protocol and it MUST silently discard message 2 and wait for message 1. This condition occurs if message 2 is received before message 1 (for example, the messages are reordered in the network) or if message 1 is lost in transmission.

  If a responder does not support the Authenticated Internet Protocol, it MUST silently discard message 1, as specified in [RFC2408] section 5.2, because it contains an unknown **exchange type**. If a responder does not support the Authenticated Internet Protocol, it MUST respond to message 2, which is recognizable as an IKE v1 message.

  The initiator MUST continue retransmitting both messages 1 and 2 until it receives a response, or it times out. When the initiator receives a response, it MUST examine the exchange type in the IKE header of the response to determine whether the responder supports the Authenticated Internet Protocol. If the response indicates that the responder does not support the Authenticated Internet Protocol, the initiator MUST carry out the negotiation by using IKE. Otherwise, if the response indicates that the responder supports the Authenticated Internet Protocol, the initiator MUST carry out the negotiation by using the Authenticated Internet Protocol as specified in section 3.1.

  After the IKE Protocol or Authenticated Internet Protocol has been determined for a given MM SA, the peers MUST use the same protocol for the lifetime of the SA.

1.8 Vendor-Extensible Fields

The Authenticated Internet Protocol inherits the extensibility features of Internet Security Association and Key Management Protocol (ISAKMP): private exchange types, private payload types, vendor ID payloads, and encryption/authentication algorithms. These extensibility mechanisms are specified in [RFC2408] section 3. This protocol does not add any new extensibility mechanisms to ISAKMP or Internet Key Exchange (IKE) v1. Vendors MAY extend the Authenticated Internet Protocol by using the mechanisms provided by ISAKMP and IKE v1.

The Authenticated Internet Protocol uses GSS-API exchanges for authentication, and supports the extensibility mechanisms as specified in [RFC2743].

1.9 Standards Assignments

The Authenticated Internet Protocol uses the assigned numbers from Internet Assigned Numbers Authority (IANA), "IPsec Registry" (IANAIPSEC), and "ISAKMP Registry" (IANAISAKMP). All the new values used in the Authenticated Internet Protocol are in the private range as specified in [IANAIPSEC] and [IANAISAKMP]. Standard assignments have not been received for this protocol.

2 Messages

The following sections specify how Authenticated Internet Protocol messages are encapsulated on the wire, and specify common data types.

2.1 Transport

Authenticated Internet Protocol messages are transported over Internet Security Association and Key Management Protocol (ISAKMP) as specified in [RFC2408], which uses UDP port 500 by default. The Authenticated Internet Protocol runs over ports 500 and 4500 if a NAT has been detected, but MAY be run over a different port. For more information see [RFC3947], section 3.2.<6>

Protocol payload formats are specified in [RFC2408], with the additions defined in section 3 of that document. Authenticated Internet Protocol messages have a header with a fixed format, followed by a variable number of payloads. The protocol uses the ISAKMP header format as specified in [RFC2408], section 3.1, with the changes defined in this section.

All fields are sent and encoded in Network Order unless otherwise specified.

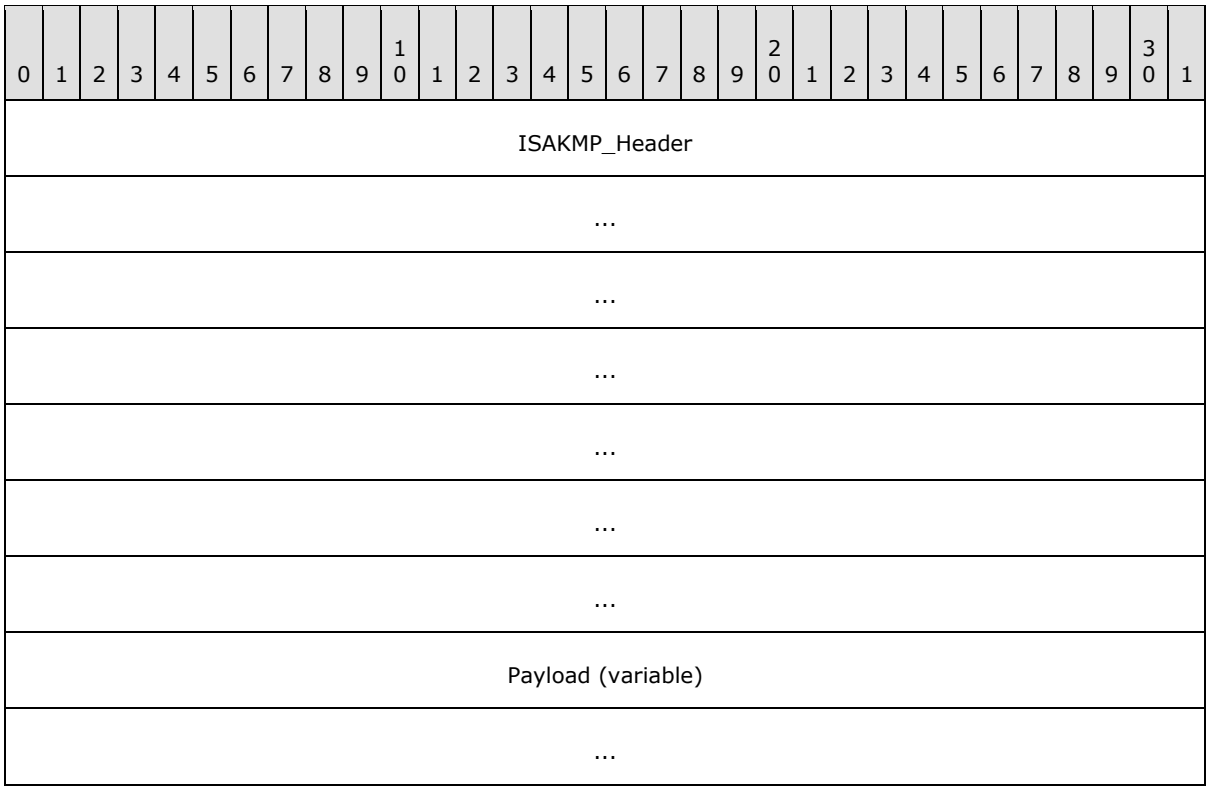
2.2 Message Syntax

2.2.1 ISAKMP Header Format Packet

The Authenticated Internet Protocol messages are Internet Security Association and Key Management Protocol (ISAKMP) messages, as specified in [RFC2408] section 3. Except where otherwise specified, each Authenticated Internet Protocol message consists of an ISAKMP header and a single Crypto payload that encapsulates a sequence of Authenticated Internet Protocol payloads. The payloads encapsulated by the crypto payload are encrypted once the MM session keys are available. Before then, they are in clear text. This protocol defines additional values for the exchange type field of an ISAKMP message.

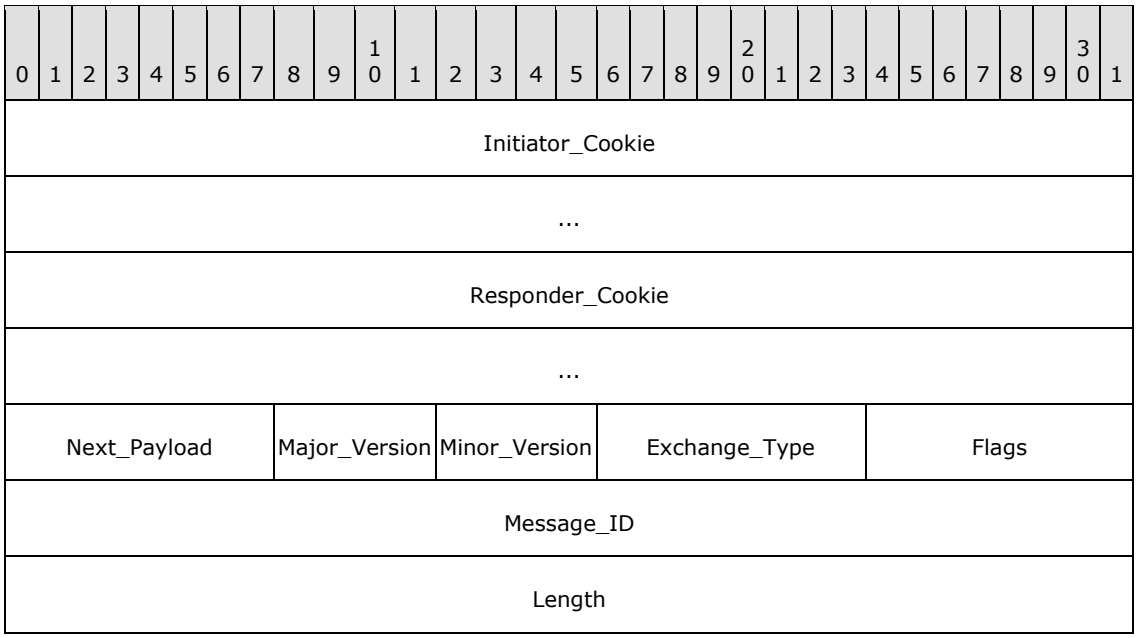
The ISAKMP message packet is used in the establishment, negotiation, modification, and deletion of security associations (SAs).

The following is the format of an ISAKMP message:



ISAKMP_Header: Contains the information required by the protocol to maintain state, process payloads, and possibly prevent denial-of-service or replay attacks. This is the standard ISAKMP header; Microsoft adds no extensions. For more information about the ISAKMP header, see [RFC2408].

This field contains the following subfields:



Initiator_Cookie: Cookie of entity that initiated security association (SA) establishment, SA notification, or SA deletion. This is identical to those specified in [RFC2408] section 3.1.

Responder_Cookie: Cookie of entity that is responding to an SA establishment request, SA notification, or SA deletion, in network order. This is identical to those specified in [RFC2408] section 3.1.

Next_Payload: Indicates the payload type of the first payload in the message. This is identical to those specified in [RFC2408] section 3.1.

ISAKMP Payload Type	Value
None	0
SecurityAssociation	1
Proposal	2
Transform	3
KeyExchange	4
Identification	5
Certificate	6
CertificateRequest	7
Hash	8
Signature	9
Nonce	10

ISAKMP Payload Type	Value
Notification	11
Delete	12
VendorID	13
Reserved	14 — 127
PrivateUse	128 — 255

Major_Version: Indicates the major version of the ISAKMP protocol in use. Implementations based on this specification MUST set the Major Version to 1. This is identical to those specified in [RFC2408] section 3.1.

Minor_Version: Indicates the minor version of the ISAKMP protocol in use. Implementations based on this specification MUST set the minor version to 0. An implementation SHOULD silently ignore packets with a minor version number larger than its own, given that the major version numbers are identical. This is identical to those specified in [RFC2408] section 3.1.<7>

Exchange_Type: The Authenticated Internet Protocol exchange types are in the private use range, as specified in [RFC2408] Section 3.1. The Authenticated Internet Protocol main mode (MM), quick mode (QM), and notify exchanges correspond respectively to the Internet Key Exchange (IKE) v1 identity protection and the QM (as specified in [RFC2408] section 4.5), and the notify exchanges, as specified in [RFC2408] section 4.8.

This protocol defines the following exchange types:

Value	Meaning
0xF3 243	MM exchange type
0xF4 244	QM exchange type
0xF5 245	EM exchange type
0xF6 246	Notify exchange type

Flags: Identical to those specified in [RFC2408] section 3.1.

Message_ID: The unique message identifier used to demultiplex messages from concurrent QM negotiations. This value is generated by the initiator of the QM negotiation. This is identical to those specified in [RFC2408] section 3.1.

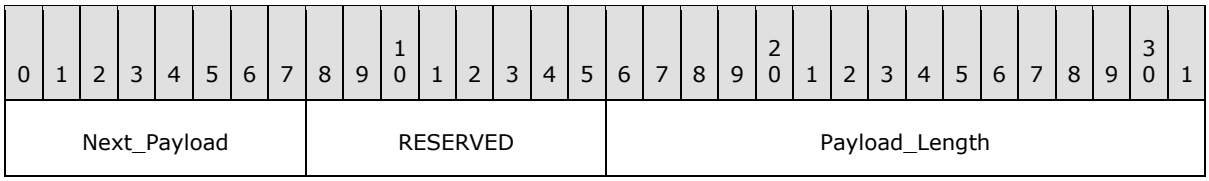
Length: The length of total message (header + payloads) in bytes. This is identical to those specified in [RFC2408] section 3.1.

Payload: An ISAKMP payload is used to transfer information such as SA data, or key generation and authentication data.

2.2.2 Generic Payload Header Packet

Each payload in Authenticated Internet Protocol messages starts with the generic payload header, as specified in [RFC2408] section 3.2.

The Generic Payload Header packet:



Next_Payload: Indicates the payload type of the next payload in the message. If the current payload is the last in the message, then this field MUST be 0.

RESERVED: Unused MUST be set to 0. The responder MUST ignore this field.

Payload_Length: The length in bytes of this payload, including the generic payload header.

2.2.3 Payload Types

The Authenticated Internet Protocol reuses the following payload types from [RFC2408] section 3.1:

RFC 2408 Payload Types	Value
NONE	0x00
Security Association (SA)	0x01
Proposal (P)	0x02
Transform (T)	0x03
Key Exchange (KE)	0x04
Identification (ID)	0x05
Hash (HASH)	0x08
Nonce (NONCE)	0x0A
Notification (N)	0x0B
Vendor ID	0x0D

The Certificate, Certificate Request, Signature, and Delete payload types are not used by the Authenticated Internet Protocol. Certificate-based authentication is performed in AuthIP through GSS-API, as specified in [MS-SECO].

This protocol also defines the following additional payload types. The payload types are allocated from the Private Use range as defined in [RFC2408] section 3.1.

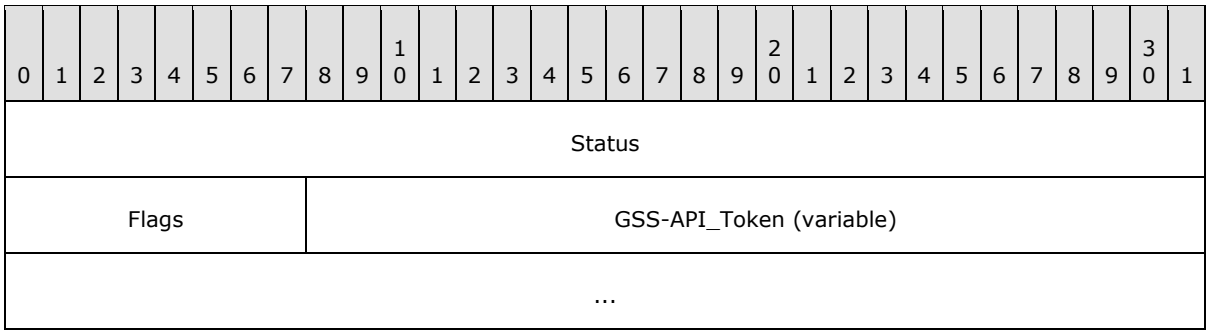
AuthIP Payload Types	Value
Security Support Provider Interface Token (GSS-API) as defined in [GSS]	0x81

AuthIP Payload Types	Value
Crypto Payload (CRYPTO)	0x85
GSS-API Endpoint Name (GSS_ID)	0x86
Authentication (Auth)	0x87

2.2.3.1 GSS-API Payload (Payload Type 0x81) Packet

This is the standard GSS-API payload (as specified in [GSS] section 3.3.3) without the vendor-encoding field and with additional status and flag fields.

The GSS-API Payload 0x81 packet:



Status: On failure, a 4-byte error code in network order is returned by GSS-API. The error is not reported to the application. Error logging can be implemented in a variety of ways.<8> **Note** 0x00000000 on success.

Flags: All other flag fields MUST be set to 0 on the initiator and ignored on the responder. The GSS_IMPERSONATION_ACTIVE, GSS_EXPLICIT_CREDENTIALS, and GSS_NEW_GSS_EXCHANGE flags are valid only on messages sent by an initiator. The GSS_RESPONDER_AUTH_COMPLETE flag is only valid on messages sent by a responder.

The possible flag values are the following:

Value	Meaning
0x01	GSS_NEW_GSS_EXCHANGE This flag indicates the start of a new authentication exchange.
0x02	GSS_IMPERSONATION_ACTIVE This flag indicates that the GSS-API **security principal name (SPN)** for this exchange should be interpreted as a user SPN.
0x04	GSS_RETRY_CURRENT_AUTHENTICATION This flag is set by the initiator or the responder to indicate the retry of the current authentication exchange with different credentials.
0x08	GSS_EXPLICIT_CREDENTIALS This flag indicates that explicit credentials are being used for this GSS-API exchange.
0x10	GSS_RESPONDER_AUTH_COMPLETE

Value	Meaning
	This flag is set by the responder to indicate that the authentication has completed successfully .

GSS-API_Token: As specified in [GSS] section 3.3.3. For anonymous authentication, the GSS-API data field has a zero length (for example, it is omitted).

2.2.3.2 Crypto Payload (Payload Type 0x85) Packet

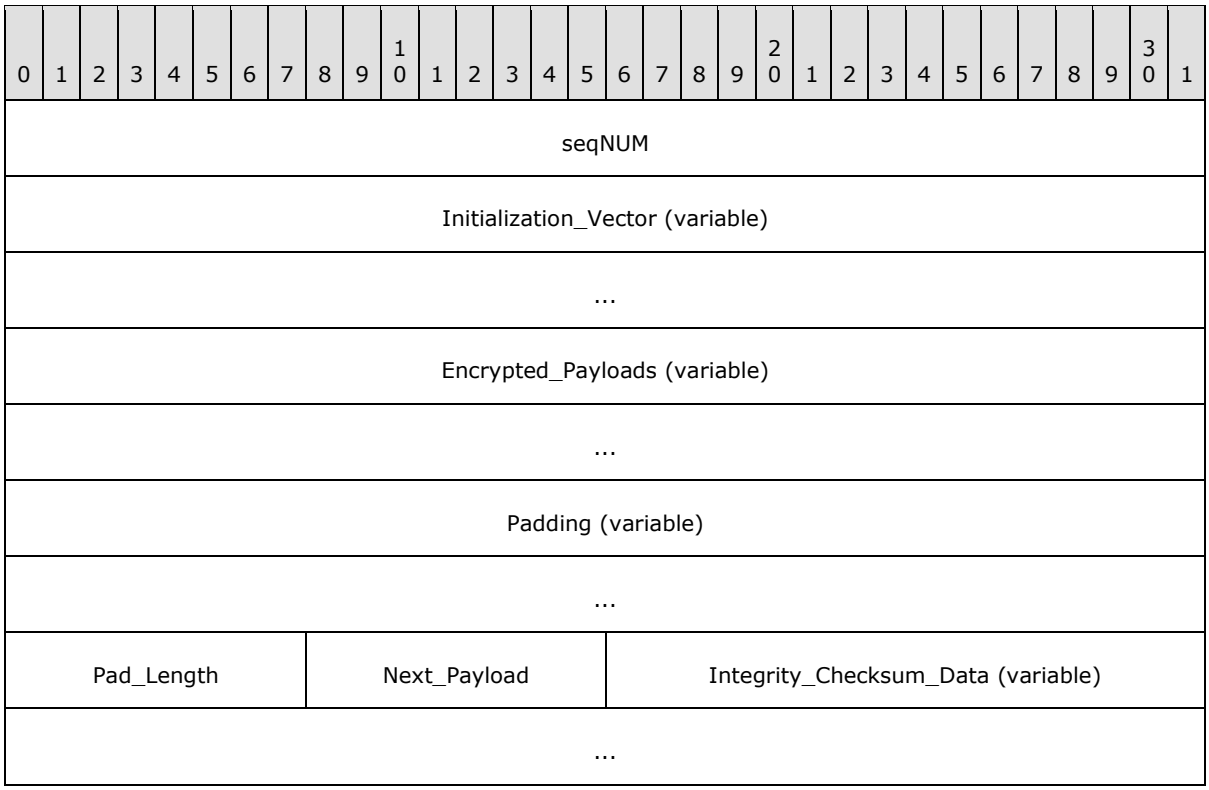
The crypto payload is used to encrypt other payloads. On the wire, Authenticated Internet Protocol messages MUST contain one and only one crypto payload, which MUST immediately follow the Internet Security Association and Key Management Protocol (ISAKMP) header. The only exception is the message containing the notify payload with a NOTIFY_DOS_COOKIE notify message type, as specified in [MS-IKEE] section 2.2.6). This message MUST NOT contain the crypto payload and MUST contain only a Notify payload.

The format of the crypto payload differs based on whether the encryption flag is set in the flags field of the ISAKMP header, as specified in [RFC2408] section 3.1.

If the encryption flag is set, all other payloads MUST be embedded within the crypto payload. If the encryption flag is not set, the crypto payload MUST be inserted in front of the all other payloads.

2.2.3.2.1 Crypto Payload 0x85 Encryption Flag Set

The Crypto Payload type 0x85 with the encryption flag set format:



seqNUM: The **seqNum** field is a sequence number for the next expected packet number. Sequence number generation is described in section 3.

Initialization_Vector: The length of the Initialization Vector (IV) field MUST be equal to the block size used by the encryption algorithm. Encryption begins immediately after the initialization vector, and encrypts up to the beginning of the integrity checksum data.

Encrypted_Payloads: A variable-length sequence of encrypted Authenticated Internet Protocol payloads, each starting with the generic payload header. The entire payload sequence, including the generic payload headers, MUST be encrypted.

Padding: 0 to 255 bytes of padding as required by the encryption algorithm.

Pad_Length: The length of the preceding padding.

The pad length field is located after the variable-length padding, hence the payload MUST be decoded starting from the end. Encapsulating Security Payload (ESP) uses an identical technique for pad length encoding, as specified in [RFC4303] section 2.

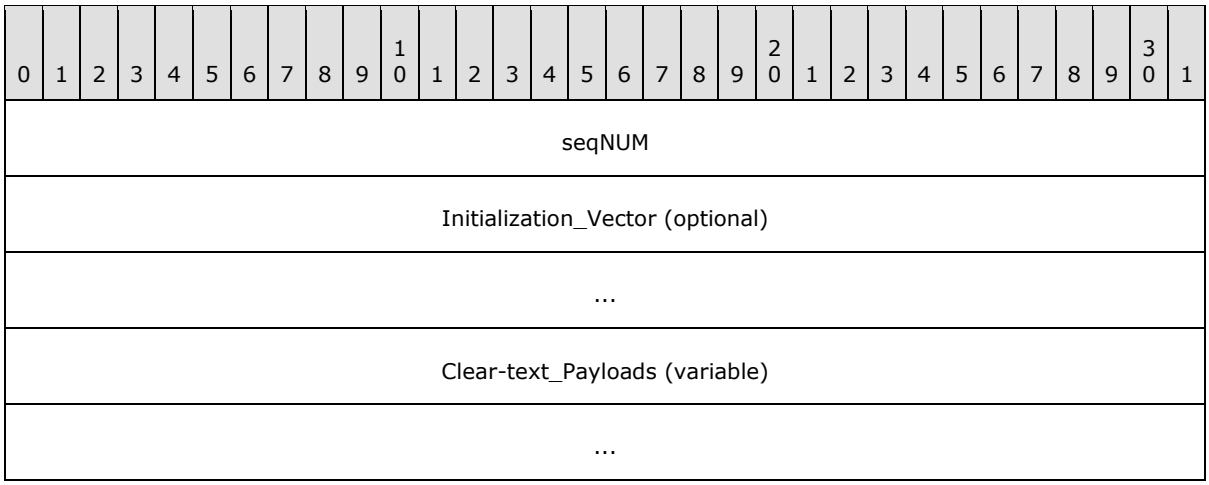
Next_Payload: The payload type of the first payload in the Encrypted Payload sequence carried by this Crypto Payload.

Integrity_Checksum_Data: As required by the negotiated authentication algorithm ([RFC4303] section 2.8). The integrity checksum data covers the ISAKMP header to the beginning of the integrity checksum data.

2.2.3.2.2 Crypto Payload 0x85 Encryption Flag Not Set

If the encryption flag is not set in the flags field of the Internet Security Association and Key Management Protocol (ISAKMP) header, the format is the following:

The Crypto Payload 0x85 Encryption Flag Not Set format:



seqNUM: The **seqNum** field is a sequence number for the next expected packet number. Sequence number generation is specified in section 3.1.4.

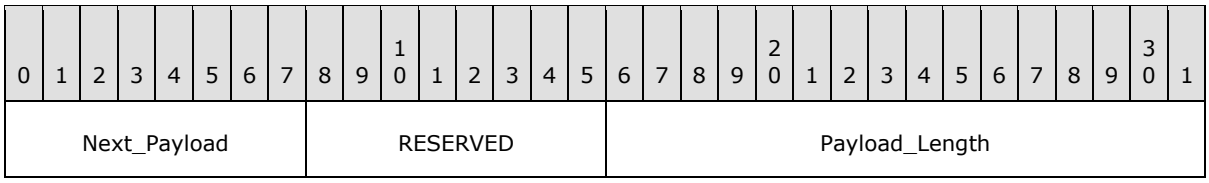
Initialization_Vector: The **Initialization Vector** field is an optional field which MAY be present<9>. The presence of the Initialization Vector is indicated by the length of the Crypto payload (greater than 8 bytes if the Initialization Vector is present, 8 bytes otherwise).

Clear-text_Payloads: A variable-length sequence of unencrypted Authenticated Internet Protocol payloads, each starting with the generic payload header.

If the encryption flag is not set in the flags field of the ISAKMP header, the next payload field of the generic payload header of the crypto payload MUST be set to the payload type of the first payload in the clear-text payload sequence carried by the crypto payload.

2.2.3.2.3 Format of the Generic Payload Header for the Crypto Payload

The Format of the Generic Payload Header for the Crypto Payload is the following:



Next_Payload: 0 if the encrypted flag is set, next payload otherwise

RESERVED: MUST be set to 0. The responder MUST ignore this field.

Payload_Length: The value of Payload Length is computed as:

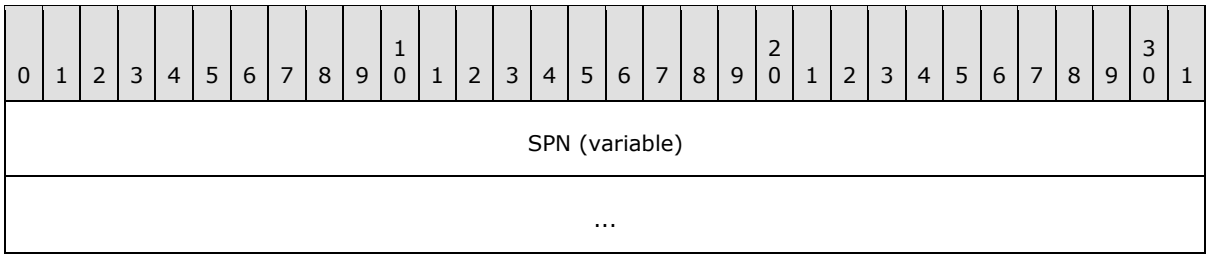
4 (the header) + 4 (the sequence number) + the length of the IV in bytes

2.2.3.3 GSS_ID 0x86 Payload Packet

In Internet Key Exchange (IKE) v1, the security principal name (SPN) information is passed between peers as an attribute of the security association (SA), as specified in [RFC2409] section 3.4. In the Authenticated Internet Protocol, a separate payload is used to convey the SPN.

The Generic Security Services (GSS) GSS_ID payload fields are defined as follows:

The GSS_ID 0x86 Payload packet:

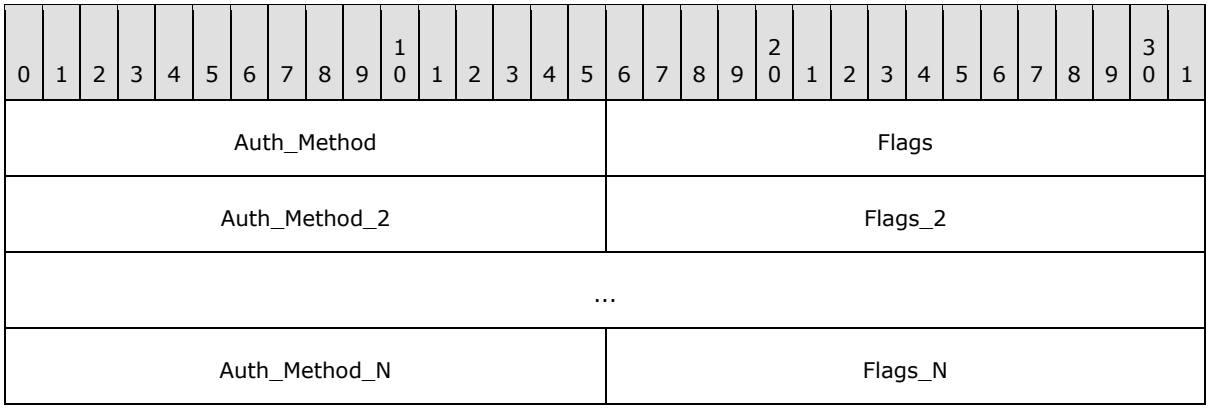


SPN: GSS-API representation of an SPN, as specified in [GSS] section 3.

2.2.3.4 Auth Payload (Payload Type 0x87) Packet

The authenticated payload is used to convey a list of authentication methods to the responder. The number of authentication methods MUST be computed from the payload size in the generic header.

The Auth Payload 0x87 Payload packet:



Auth_Method: Indicates the proposed authentication method. The allowed values are:

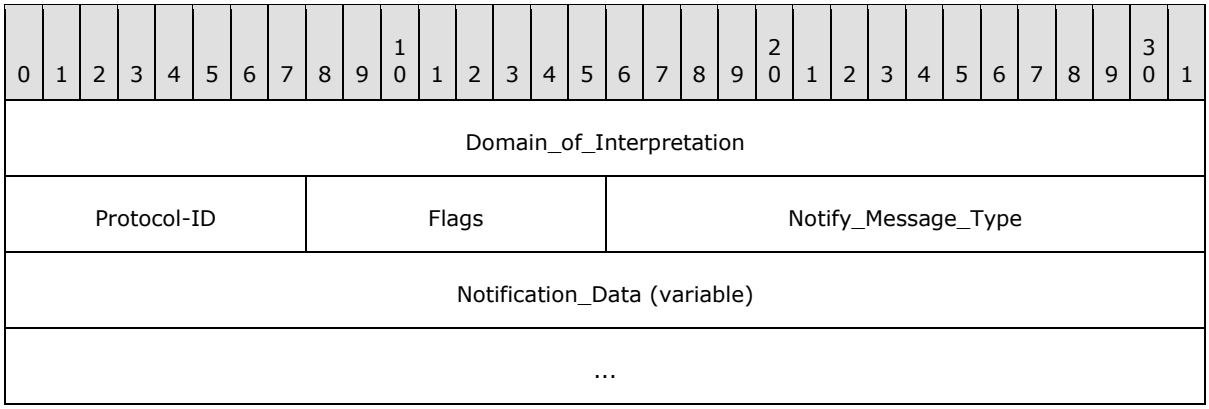
Value	Meaning
0x0002	Kerberos
0x0003	anonymous
0x0004	TLS
0x0005	[MS-NLMP]

Flags: This is currently unused. MUST be set to 0 by the initiator and ignored by the responder.

2.2.3.5 Notify Payload (Payload Type 0x0B) Packet

The notify payload is similar to the Internet Key Exchange (IKE) v1 notification payload, as specified in [RFC2408] section 3.14. However, the field referred to as **spiSize** in IKE v1 is referred to as flags in the Authenticated Internet Protocol and the SPI field is absent. The receiver MUST interpret the **spiSize** field as a flags field if the exchange type for the message is an Authenticated Internet Protocol private exchange type, as specified in section 2.2.1.

The Notify Payload 0x0B packet:



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 indicates the exchange type to which this notification applies. This MUST be one of the following values.

Value	Meaning
0x01	MM/EM notification
0x02	QM notification

Flags: This flag indicates a reliable notify message. Reliable notify messages MUST be acknowledged by the peer. If this flag is not set, the responder MUST NOT acknowledge the notify message. See section 3.1.8.1.1 for additional information. The only allowed flag field is:

Value	Meaning
0x01	RELIABLE_NOTIFY_FLAG Reliable notification

Notify_Message_Type: This identifies the type of notification being sent with this message, in network byte order. The notify message types are from the private range, as specified in [RFC2408] section 3.14.1.

Value	Meaning
0x9C45	EXCHANGE_INFO This notify message type is used by the **negotiation discovery** Internet Key Exchange (IKE) protocol extension, as specified in [MS-IKEE].

Value	Meaning
0x9C54	NOTIFY_STATUS This notify message type is used to request the peer to delete its state for the corresponding security association (SA).
0x9C55	NOTIFY_DOS_COOKIE This notify message type is used by the responder under denial of service protection mode, as specified in [MS-IKEE] section 3.10.
0x9C56	NOTIFY_ACK This notify message type is used to acknowledge a reliable notify message.
0x9C57	NOTIFY_QM_SYNCHRONIZE This notify message type is used to signal the end of the quick mode (QM) phase.
0x9C58	NOTIFY_ACQUIRE This notify message type is used to instruct the peer to negotiate a new main mode security association (MM SA).

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:

- NOTIFY_QM_SYNCHRONIZE (0 Bytes): No notification data.
- NOTIFY_STATUS (4 Bytes): Status code in network byte order, indicating success or failure. The values transmitted as status codes are implementation-specific.<10>
- NOTIFY_ACK (4 Bytes): Sequence number in network byte order, containing the sequence number of the crypto payload that carried the notify payload being acknowledged.
- NOTIFY_ACQUIRE: (See section 2.2.3.6)

  A notify payload of type NOTIFY_ACQUIRE MUST be followed by two Phase 2 identification payloads, as specified in [RFC2407], section 4.6.2. Each of these Phase 2 identification payloads MUST have the generic payload header, and MUST immediately follow the notify payload. The first identification payload MUST be the initiator identification payload. The second payload MUST be the peer identification payload. If any of these conditions is not met, the responder MUST silently discard the packet.
- NOTIFY_DOS_COOKIE (8 Bytes): The responder cookie value.

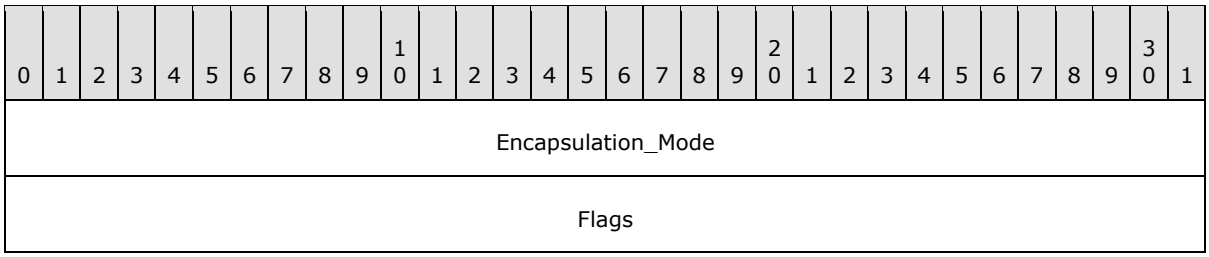
  This notify message type MUST be used in a message containing a single notify payload after the ISAKMP header. The message MUST NOT contain the crypto payload.
- EXCHANGE_INFO (4 Bytes): Flags in network byte order. The flag values are:

Value	Meaning
0x00000001	IKE_EXCHANGE_INFO_ND_BOUNDARY This flag is used by the negotiation discovery IKE protocol extension, as specified in [MS-IKEE].
0x00000002	IKE_EXCHANGE_INFO_GUARANTEE_ENCRYPTION This flag is used by the negotiation discovery IKE protocol extension, as specified in

Value	Meaning
	[MS-IKEE].

2.2.3.6 Notify Payload (Payload Type 0x0B) Notify Acquire Packet

The Notify Acquire packet:



Encapsulation_Mode: This is an encapsulation mode in network order, as specified in [IANAISAKMP].

Flags: Flags in network order:

Value	Meaning
0x0000000	IPSEC_ACQUIRE_FLAG_GUARANTEE_ENCRYPTION This flag is used by the negotiation discovery Internet Key Exchange (IKE) protocol extension, as specified in [MS-IKEE].

3 Protocol Details

The following sections specify details of the Authenticated Internet Protocol, including abstract data models, timer events, and message processing rules.

3.1 Common Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of 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 main data elements required by any implementation are:

- Main Mode Security Association (MM SA) Database : A database containing the operational state for each MM SA. The entry for each MM SA contains the following data elements:
  - Current role (initiator or responder).
  - Current state (the state is the message number within a negotiation).
  - Per-negotiation timers.
  - Sequence numbers for main mode (MM), quick mode (QM), extended mode (EM), and notification exchange types. The sequence numbers start at zero and are incremented for each exchange. The sequence number is used as the **SeqNUM** field of the Crypto payload.
  - Cryptographic parameters for the MM SA.
  - QM SAs during this negotiation.

  The MM SA is indexed by the local and peer IP address and the initiator and responder cookies found in the Internet Security Association and Key Management Protocol (ISAKMP) header, like in Internet Key Exchange (IKE) v1.
- **Security Policy Database (SPD)**: The SPD and its management operations are specified in [RFC4301] section 4.4.1. This specification does not extend that definition. The SPD referred to in this specification contains rules describing whether and how Internet Protocol Security (IPsec) protection is applied to inbound or outbound IP traffic. The SPD is looked up by using tuples containing flow information for the packet.
- 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 whether and how the Authenticated Internet Protocol should negotiate SAs with a remote peer. The PAD is looked up using tuples composed of local and remote IP addresses.
- **Security Association Database (SAD)**: The SAD contains the parameters of each QM SAs. This specification does not extend that definition. The SAD and its management operations are specified in [RFC4301] section 4.4.2.

3.1.2 State Machines

3.1.2.1 Exchange State Machine

The state diagrams for a single exchange within a negotiation for the initiator and the responder are as follows:

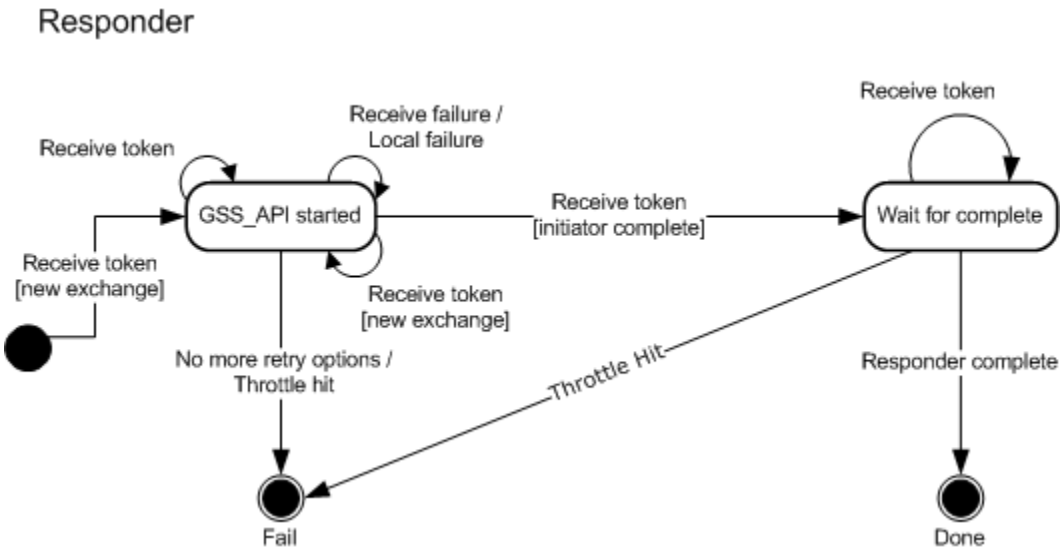


Figure 1: Single exchange state

The initiator MUST start a retransmission timer when sending the first packet of an exchange. If the timer expires before a response is received, the initiator SHOULD retransmit the message and restart the timer. After a predefined number of retransmissions, the exchange SHOULD be aborted.<11>

3.1.2.2 Authentication Retry State Machine

The logic for authentication retry is triggered within a negotiation when the current authentication method fails and there are remaining authentication methods (or remaining authentication parameters for the current methods) that can be tried before failing the negotiation. Authenticated Internet Protocol implementations MUST support authentication retry.

The state diagrams for authentication retry for the initiator and responder are the following:

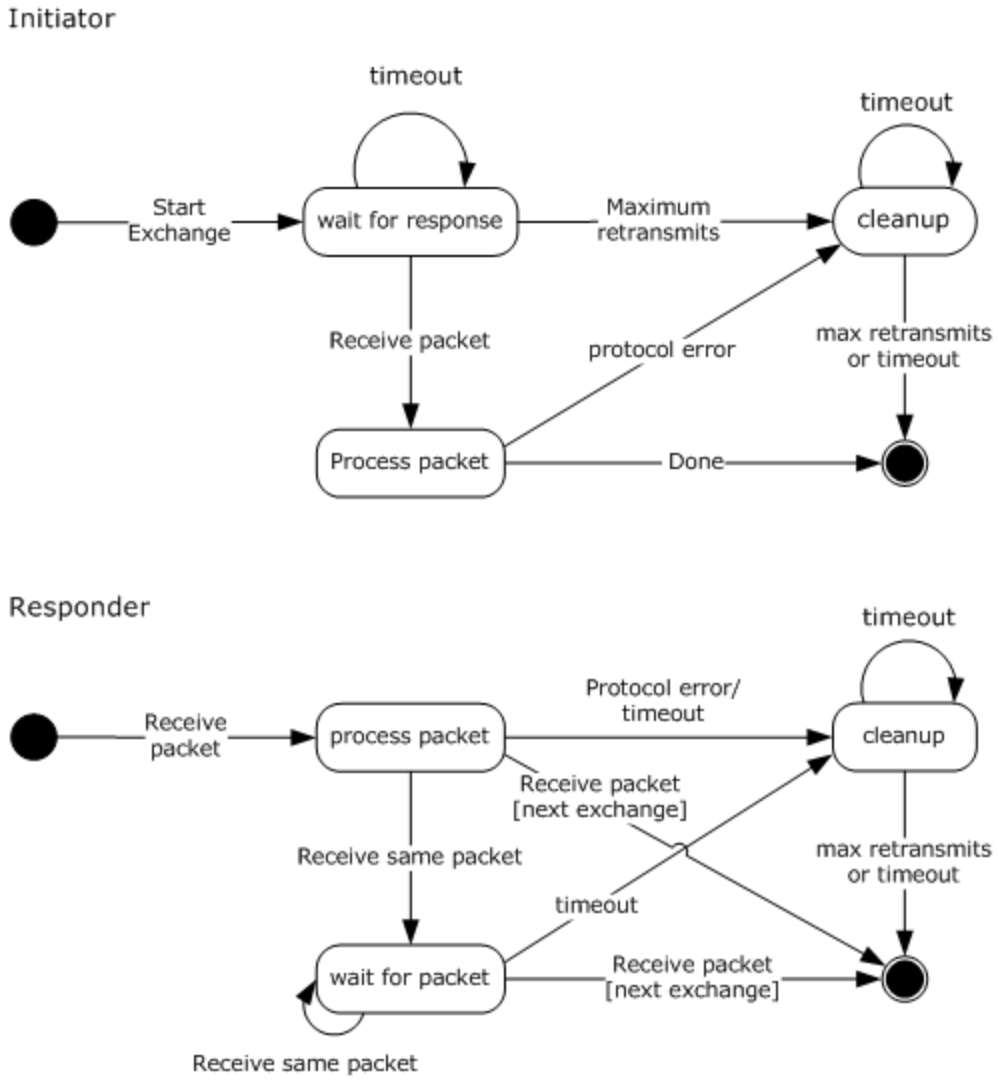


Figure 2: Authentication retry state (initiator and responder)

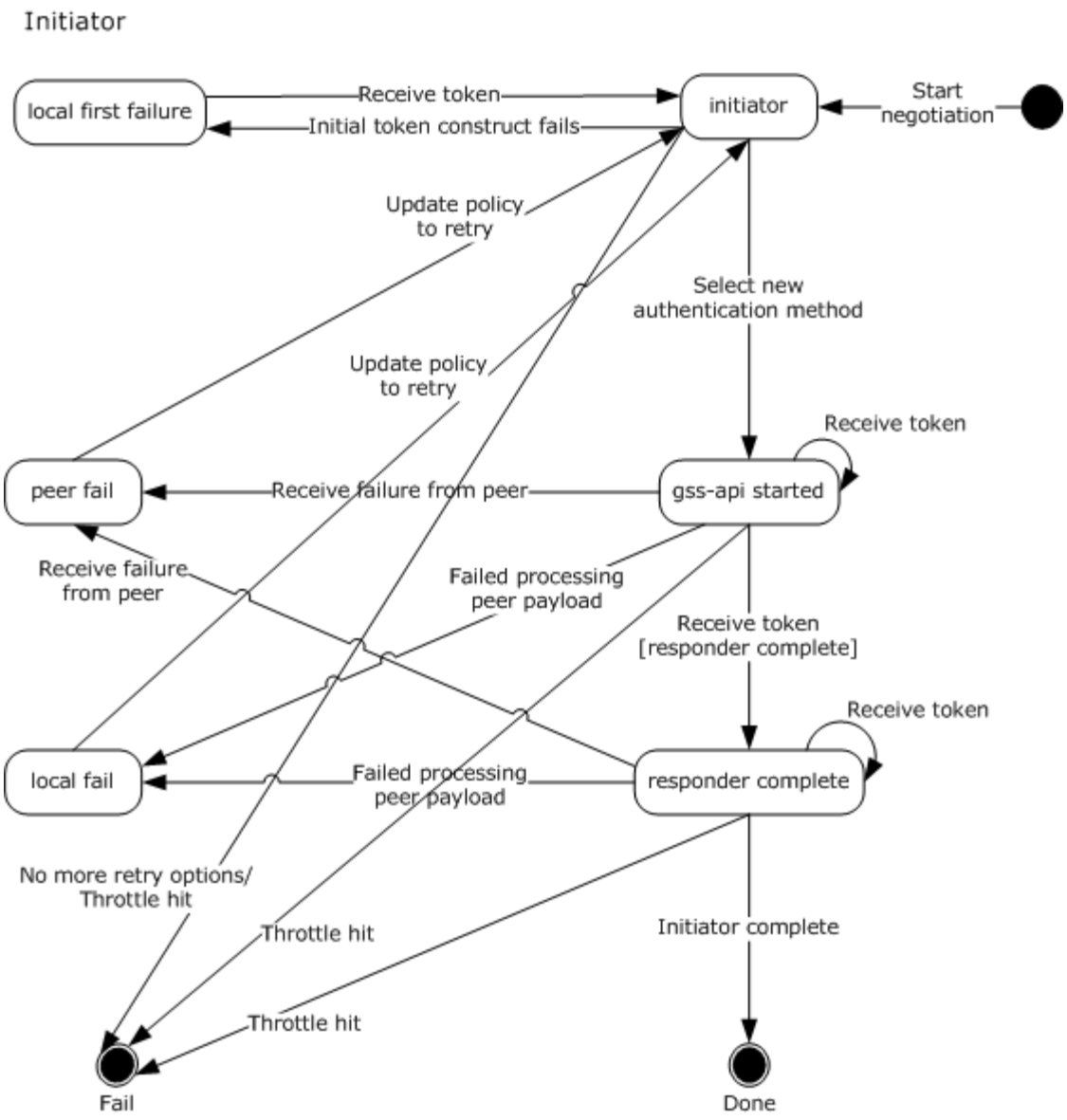


Figure 3: Authentication retry state (negotiation)

TLS, as specified in [RFC3546], performs certificate-based authentication. The local certificate to use for the exchange is a parameter to TLS. When negotiating the TLS authentication method, each Authenticated Internet Protocol peer MUST retry all its applicable credentials (for example, certificates) before failing the TLS negotiation and moving to the next authentication method.

The rules for setting the GSS-API payload flags are the following:

- The GSS-API flags (GSS_RETRY_CURRENT_AUTHENTICATION and GSS_NEW_GSS_EXCHANGE, as specified in section 2.2.3.1) MUST be set by the initiator to signal that TLS negotiation will be retried.

- The Generic Security Services GSS-API flag GSS_NEW_GSS_EXCHANGE MUST be set by the initiator when TLS negotiation has failed, to advance to the next authentication method.
- The **Status** field in the GSS-API payload MUST be set to a nonzero value by the responder if it determines that TLS will not succeed. When this field is set, the initiator MUST advance to the next authentication method.
- The **Status** field in the GSS-API payload MUST be set to a non-zero value and the GSS_RETRY_CURRENT_AUTHENTICATION flag MUST be set by the responder to indicate that the initiator MUST retry TLS with a different credential, if available. When the initiator retries TLS with a different credential, it MUST set the Status field using the non-zero value received from the responder in the GSS-API payload.

3.1.3 Notation

This specification uses the same notation, as specified in [RFC2409] section 3.2, with a few additions noted below as "additional notation":

HDR: An Internet Security Association and Key Management Protocol (ISAKMP) header whose exchange type is the method. When written as HDR*, it indicates payload encryption.

Security Association (SA): An SA negotiation payload that embeds one or more Proposal and Transform payloads. See [RFC2409] section 3.4.

CKY-I and CKY-R: The Initiator's cookie and the Responder's cookie, respectively, from the ISAKMP header. Used together with initiator and responder IP addresses as a unique key for looking up each session.

g^xy: The Diffie-Hellman shared secret.

g(qm)^xy (additional notation): The Diffie-Hellman secret in the Quick Mode negotiation phase.

KE: The Key Exchange payload that contains the public information exchanged in a Diffie-Hellman exchange.

Nx: The Nonce payload. x can be: i or r for the ISAKMP initiator and responder respectively. x(qm) is the quick-mode nonce.

IDx: The Identity payload for "x". x can be: "i" or "r" for the ISAKMP initiator and responder, respectively. The Identity payload format for the Internet DOI is specified in [RFC2407] section 4.6.2.

hash(msg) (additional notation): The negotiated hash algorithm for the main mode security association (MM SA). Msg is the quantity to hash.

sha1 (additional notation): The SHA-1 (as specified in [FIPS180]) hash function. [SHA256] is used until the hash function has been negotiated.

prf(key, msg): The keyed Pseudo-Random Function (PRF) used to generate a deterministic output that appears pseudo-random. PRF is used both for key derivations and for authentication (for example, as a keyed MAC). PRF is implemented as an HMAC of the negotiated hash algorithm, as defined in [RFC2104].

SKEYID: A string derived from a secret known only to the active players in the exchange.

SKEYID_e: The **keying material** used by the ISAKMP SA to protect its messages.

SKEYID_a: The keying material used by the ISAKMP SA to authenticate its messages.

SKEYID_d: The keying material used to derive keys for non-ISAKMP SAs.

SKEYID_em: The keying material used by **AuthIP** to authenticate extended mode negotiation messages.

Notify (additional notation): The Notify payload.

Authx (additional notation): The authentication proof for x, where x is either the first or second authentication and has the value 1 to 4. It is embedded in a Hash payload.

VID (additional notation): The vendor ID payload.

A-KDF(Z, OtherInfo, keydatalen) (additional notation): Invoke the Key Derivation Function (KDF), as specified in [SP800-56A]. The Key Derivation Function uses the negotiated main mode hash function as its hash function.

Crypto-ID (additional notation): A 16 bits number corresponding to the negotiated main mode encryption algorithm. The table mapping the algorithm identifiers to the corresponding numbers is defined in [RFC2409] Appendix A.

GSS-APIsecret, GSS-APIsecret_em (additional notation): The session key obtained from a successful GSS-API exchange in main mode (MM) and extended mode (EM), respectively, as specified in [GSS].

hashLength, cryptLength (additional notation): The length in bytes of the key for the negotiated MM SA authentication and encryption algorithms.<12>

ipsechashLength, ipseccryptLength (additional notation): The length in bytes of the key for the QM SA authentication and encryption algorithms.

#n: Numbers the packet with a series of exchanges for reference. This information is not on the wire.

[x]: Indicates that x is optional.

3.1.4 AuthIP Key Material Generation

A number of Authenticated Internet Protocol exchanges involve generating keying material. Each AuthIP peer MUST generate keying material as follows:

Following the steps in [SP800-56A] concatenation Key Derivation Function (KDF). The output of the [SP800-56A] KDF is the bit string DerivedKeyingMaterial of length ByteLength bytes.

```
SET AlgorithmID to Crypto-ID
Set PartyUInfo to CKY-I
Set PartyVInfo to CKY-R
COMPUTE SuppPubInfo-SKEYID as Ni Concatenated with Nr
COMPUTE SuppPubInfo-SKEYID_d as Ni Concatenated with Nr
  Concatenated with 0
COMPUTE SuppPubInfo-SKEYID_a as Ni Concatenated with Nr
  Concatenated with 1
COMPUTE SuppPubInfo-SKEYID_e as Ni Concatenated with Nr
  Concatenated with 2
Set SuppPrivInfo-SKEYID to GSS-APIsecret
Set SuppPrivInfo-SKEYID_em to GSS-APIsecret_em
```

```
Set SuppPrivInfo-SKEYID_d to SKEYID
COMPUTE SuppPrivInfo-SKEYID_a as SKEYID_d Concatenated with SKEYID
COMPUTE SuppPrivInfo-SKEYID_e as SKEYID_a Concatenated with SKEYID
If DH is used then
set Z to g^xy
Else
set Z to a zero-length byte string
END IF
COMPUTE OtherInfo (pubInfo,privInfo) as AlgorithmID Concatenated
  with PartyUInfo Concatenated with PartyVInfo
  Concatenated with pubInfo Concatenated with privInfo
COMPUTE SKEYID as A-KDF
  (Z, OtherInfo (SuppPubInfo-SKEYID, SuppPrivInfo-SKEYID),
   hashLength)
COMPUTE SKEYID_d as A-KDF (Z, OtherInfo (SuppPubInfo-SKEYID_d,
  SuppPrivInfo-SKEYID_d), hashLength)
COMPUTE SKEYID_a as A-KDF (Z, OtherInfo (SuppPubInfo-SKEYID_a,
  SuppPrivInfo- SKEYID_a), hashLength)
COMPUTE SKEYID_e as A-KDF (Z, OtherInfo (SuppPubInfo-SKEYID_e,
  SuppPrivInfo-SKEYID_e), max(hashLength, cryptLength))
COMPUTE SKEYID_em as A-KDF (Z, OtherInfo (SuppPubInfo-SKEYID,
  SuppPrivInfo-SKEYID_em), hashLength)
COMPUTE SuppPubInfo-IpsecEncryptKey as MessageId Concatenated with
  SPI Concatenated with Ni(qm) Concatenated with Nr(qm)
  Concatenated with SKEYID_d
if EM exchanges THEN
COMPUTE SuppPrivInfo-IpsecEncryptKey as Ni(mm)
  Concatenated with Nr(mm) Concatenated with SKEYID_em
ELSE
Set SuppPrivInfo-IpsecEncryptKey to zero-length byte string
END IF
COMPUTE IPSecEncryptKey as A-KDF (Z(qm),
  OtherInfo(SuppPubInfo-IpsecEncryptKey,
  SuppPrivInfo-IpsecEncryptKey), ipsechashLength+ipseccryptLength)
COMPUTE Auth1 as prf(SKEYID,
  hash(#4 Concatenated with hash(#3 Concatenated with
  sha256(#2 Concatenated with sha256(#1)))) Concatenated with 1)
COMPUTE Auth2 as prf(SKEYID, hash(#4 Concatenated with
  hash(#3 Concatenated with sha256(#2 Concatenated with sha256(#1))))
  Concatenated with 2)
```

For the first two payloads, the cumulative hash for Auth1 MUST start with hard coded sha256 (as specified in [SHA256]) since a hash method has not yet been negotiated between the peers. Starting with payload #3, the peers MUST use the negotiated hash algorithm.

Extended mode (EM) also uses the following hashes:

```
COMPUTE Auth3 as prf(SKEYID_em, hash(#12 Concatenated with
  hash(#11 Concatenated with hash(#10 Concatenated with
  hash(#9 Concatenated with hash(#8 Concatenated with hash(#7))))))
  Concatenated with Auth1)
COMPUTE Auth4 as prf(SKEYID_em, hash(#12 Concatenated with
  hash(#11 Concatenated with hash(#10 Concatenated with
  hash(#9 Concatenated with hash(#8 Concatenated with hash(#7))))))
  Concatenated with Auth2)
```

The length of the hash chains for computing Auth3 and Auth4 depends on the number of messages in the exchange. Each peer MUST hash all the Internet Key Exchange (IKE) payloads as plaintext, for example, unencrypted.

The quick mode security association (QM SA) authentication key MUST be set to the first part of IPsecEncryptKey (up to ipsechashLength bytes). The QM encryption key MUST be set the remainder of IPsecEncryptKey.

Some encryption and authentication algorithms have specific key format requirements. For these algorithms, SKEYID_e, SKEYID_a, SKEYID_d, SKEYID_em, and IPsecEncryptKey MUST be properly formatted before they can be used. DES-CBC [RFC2405] and TripleDES-CBC [RFC2451] have such key formatting requirements.

For DES-CBC and TripleDES-CBC, the key MUST be computed in the following way:

1. Truncate the key to 8 bytes (DES-CBC) or 24 bytes (TripleDES-CBC)
2. For each byte, if the number of bits set to one in the byte is even adjust the least significant bit so that the number of bits set to one is odd.

3.1.5 Timers

The following timers are used by the Authenticated Internet Protocol:

- Negotiation retransmission timer (for each main mode security association (MM SA)): Triggers a message retransmission by the initiator.
- Notify retransmission timer (for each MM SA): Triggers a Notification payload retransmission.
- Responder time out timer (for each MM SA): Controls how long the responder should wait for a message from the initiator.
- MM SA lifetime (for each MM SA): The MM SA lifetime is negotiated between the peers.
- NAT-T keep-alive timer (for each MM SA): As specified in [RFC3948] section 2.3).
- Quick mode (QM) rekey timer (for each MM SA): This timer is used during QM rekey.
- Quick mode security association (QM SA) lifetime (for each QM SA): The lifetime for QM SAs is specified in terms of bytes and/or seconds. Upon expiration of either the bytes or the seconds lifetime, the Internet Protocol Security (IPsec) implementation MUST start a new QM exchange. The Authenticated Internet Protocol then acts as the initiator for the rekey.

See <13> for more information on these timers.

3.1.6 Initialization

Upon initialization, Authenticated Internet Protocol creates listening endpoints on UDP ports 500 and 4500.

3.1.7 Higher-Layer Triggered Events

Inbound or Outbound Packet: As specified in [RFC4301] section 5, incoming or outgoing IP traffic is matched against the SPD in order to determine whether and how it needs to be protected. If the traffic needs to be protected and no quick mode security association (QM SA) matches the packet (such as the lifetime in seconds or in bytes has expired), the Internet Protocol security (IPsec) implementation MUST trigger the Authenticated Internet Protocol to negotiate the corresponding QM

SAs. The protocol then acts as the initiator for this negotiation and sends message #1 of the first main mode (MM) exchange.

Explicit Negotiation Request: The Authenticated Internet Protocol negotiation may be explicitly triggered by the user or administrator prior to packets being sent or received. The protocol then acts as the initiator for this negotiation and sends message #1 of the first MM exchange.

3.1.8 Message Processing Events and Sequencing Rules

Authenticated Internet Protocol negotiations are tracked according to the same rules specified for Internet Key Exchange (IKE) negotiations, as specified in [RFC2409]. Each Authenticated Internet Protocol negotiation is uniquely identified by the pair of the initiator cookie and responder cookie (plus UDP ports and IP addresses of each peer). The cookies are transmitted in the ISAKMP header, which begins every message.

Within an Authenticated Internet Protocol negotiation, quick mode (QM) exchanges are de-multiplexed using the **message ID** field of the IKE header. The **message ID** field MUST be zero during main mode (MM) and MUST be one during extended mode (EM).

All Authenticated Internet Protocol exchanges follow the retransmission state transitions presented in section 3.1.

3.1.8.1 Common Message Processing Logic

3.1.8.1.1 Receiving a Reliable Notify Message

When an Authenticated Internet Protocol peer receives a Notify message with RELIABLE_NOTIFY_FLAG set (as specified in section 2.2.3.5), it MUST send an acknowledgment consisting of a Notify message of type NOTIFY_ACK. The data of the Notify message is initialized to contain the sequence number from the Crypto header of the message being acknowledged.

3.1.8.1.2 Queuing Multiple Outstanding Notify Messages

Each Authenticated Internet Protocol peer MUST allow only one outstanding message at a time. The initiator MUST NOT send a message until it has received a response to all previous messages.

Notify messages are generated over the lifetime of a main mode security association (MM SA) when QM SAs time out. The messages generated by a QM SA expiration are embedded within a Notify exchange (exchange type 246). The peer sending the Notify message acts as the initiator for this exchange. If the initiator of the Notify exchange generates one or multiple Notify messages while waiting for a response to a previously sent message, each message MUST be queued until all preceding Notify messages are acknowledged.

This can happens in the following situation:

1. A QM SA timing out or traffic being sent over a QM SA with expired byte lifetime. This triggers a NOTIFY_STATUS message.

2. A QM SA timing out or traffic being sent over a QM SA with expired byte lifetimes and meeting the conditions described in section 3.1.9.6.1. This triggers a NOTIFY_ACQUIRE message.

3.1.8.2 Main Mode Exchanges

3.1.8.2.1 First Exchange

The generalized form of the first exchange is shown in the following illustration:

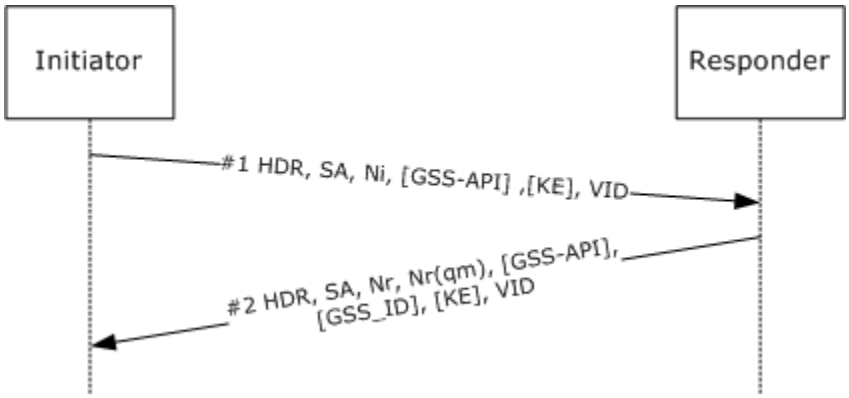


Figure 4: First exchange between initiator and responder

The responder MUST verify that message #1 is constructed as follows:

- HDR: The Internet Security Association and Key Management Protocol (ISAKMP) header is identical to the first Internet Key Exchange (IKE) phase 1 initiator packet (as specified in [RFC2409] section 5.1), except that the exchange type is 243 (main mode (MM) exchange type). The Encrypted flag SHOULD not be set.<14>

  1. The responder SHOULD also verify that the message ID field MUST be zero in message #1.<15>

  2. The Responder MUST also set the message ID value to zero in message #2(in MM Exchange).

- The remaining payloads MUST be after a Crypto payload. If an error has occurred at this stage or before, the responder MUST silently discard the message.

- Security association (SA): The SA payload is determined by looking up the PAD and the SPD. The SA payload is constructed in an identical way as the IKE SA payload. If a Diffie-Hellman exchange is required by the Authentication method (for example, the Authentication method is Anonymous or NT LAN Manager (NTLM) authentication protocol), the Diffie-Hellman group MUST be the same for all proposals in the SA payload.

- Auth payload (auth) : The Auth payload is constructed by looking up the PAD and the SPD. It contains the authentication methods valid for this negotiation.

- Ni: The initiator Nonce payload. This is identical to IKE.

Nonces SHOULD be cryptographically strong random numbers.<16>

- GSS-API: The Generic Security Services GSS-API payload is optional. If the initiator already knows the peer Security Principal Name (SPN) (for example, by locating it in the SAD), the initiator MUST use the peer SPN to construct the GSS-API payload as specified in [GSS].

- [KE]: The KE payload is optional. The KE payload is identical to IKE. The KE payload MUST be included if the Authentication method is Anonymous or NT LAN Manager (NTLM) authentication protocol (see section 3.1.8.2.2). Otherwise it MUST NOT be included.

- VID: A sequence of vendor ID payloads indicating the capabilities supported. This is identical to IKE.

If a receiver encounters any errors in the processing of a message, it MUST handle them by returning a Notify message with Notify type NOTIFY_STATUS and an error code, as described by the "protocol error transition" on the state machine from section 3.1. This behavior applies to every message received, as well as to messages received out of order. This is identical to IKE. The only exception is the DoS protection mode, as described in [MS-IKEE] section 3.10.

Upon receipt, the responder MUST look up its SAD and SPD to determine if one of the proposals in the SA payload is acceptable. The responder MUST choose one proposal or none.<17> This is identical to IKE.

If the GSS-API payload was provided in message #1, the responder MUST generate the response GSS-API payload as defined in [GSS]. If the GSS_API payload was not provided in message #1, the responder MUST send its security principal name (SPN) in a GSS_ID payload.

If the responder encounters no errors in processing message #1, it MUST construct message #2 in response, as follows:

- HDR: The ISAKMP header is identical to the first IKE phase 1 responder packet, except that the exchange type is 243 (MM Exchange Type). The Encrypted flag MUST not be set.

  1. The responder SHOULD also verify that the message ID field MUST be zero in message #1.<18>

  2. The Responder MUST also set the message ID value to zero in message #2(in MM Exchange).

- The remaining payloads MUST be after a Crypto payload.

- SA: The SA payload MUST contain the accepted proposal. This is identical to IKE.

- Auth: The responder MUST determine which set of the proposed authentication methods is acceptable, and MUST respond with at least one method of that set.To allow for Auth retry, the responder SHOULD respond with all valid authentication methods.<19> When responding with multiple authentication methods the responder MUST put the methods in its Auth payload in the same order as they were in the initiator's payload.

- Nr, Nr(qm): The responder Nonce and the QM responder Nonce payloads. The nonce value generation is the same as IKE.

- GSS-API: The GSS-API response payload if a GSS-API payload was present in message #1.

- GSS_ID: The responder SPN, if a GSS-API payload was not present in message #1.

- KE: The KE payload if requested by the initiator. The KE payload is identical to IKE.

- VID: A sequence of vendor ID payloads indicating the capabilities supported. This is identical to IKE.

3.1.8.2.2 First Exchange with Diffie-Hellman

A Diffie-Hellman exchange is required if the authentication method does not provide a shared secret that can be used for key generation. The anonymous and NT LAN Manager (NTLM) authentication protocol methods require a Diffie-Hellman exchange. The initiator MUST choose the Diffie-Hellman group and send a key exchange payload in message #1. The Diffie-Hellman parameters chosen by the initiator are indicated in the security association (SA) payload.

Upon receipt of the message, the responder MUST look up its SAD and security principal SPD to determine if the Diffie-Hellman parameters are acceptable. This is identical to Internet Key Exchange (IKE). If the Diffie-Hellman parameters are not acceptable, the responder MUST fail the negotiation and send a Notify message with a Notify type of NOTIFY_STATUS.

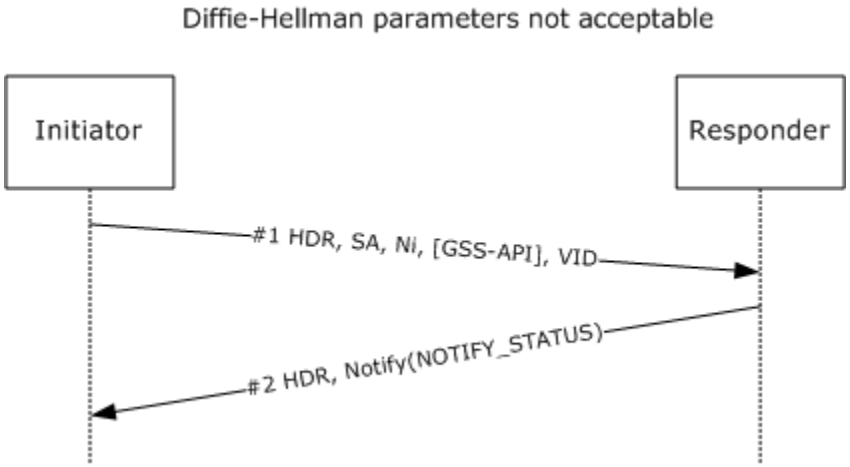


Figure 5: First exchange with Diffie-Hellman (failure)

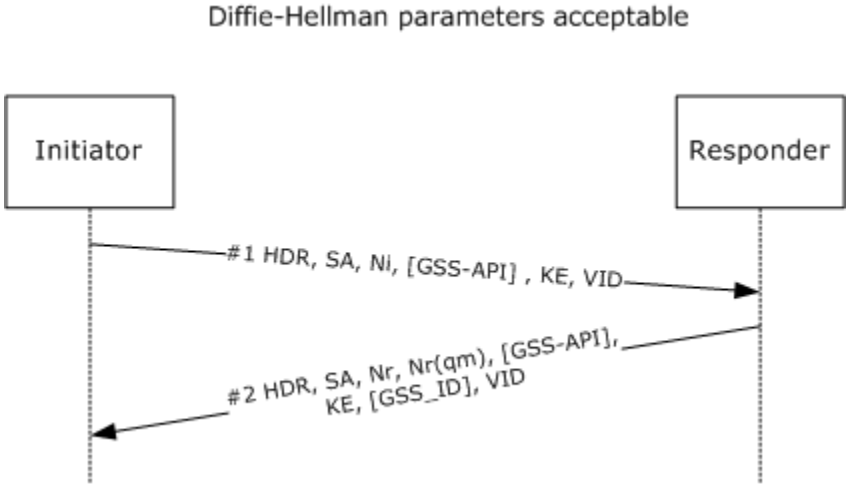


Figure 6: First exchange with Diffie-Hellman (success)

The format of the first exchange varies depending on whether the initiator knows the responder's SPN or not.

3.1.8.2.3 First Exchange with Unknown Peer SPN

The responder's SPN is unknown, for example, when the initiator only knows the responder's IP address. Another example is a case where the SPN is hostA.somecompany.com and the initiator only knows the responder as hostA. Knowing just hostA (a flat name) is sufficient to resolve the name to an IP address which can allow the initiator to initiate but is not sufficient to start a GSSAPI authentication within AuthIP.

If the peer security principal name (SPN) is unknown, the first exchange is as follows:

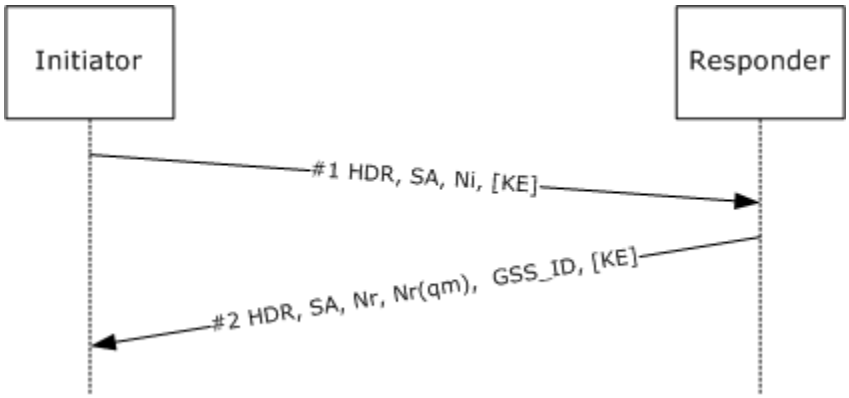


Figure 7: First exchange with unknown SPN

The initiator needs to know the peer SPN to be able to construct its first GSS-API payload. If the peer SPN is unknown, the initiator MUST NOT send the GSS-API payload in message #1, and the responder MUST send its SPN wrapped in a GSS_ID payload in message #2. The initiator can then construct its first GSS-API payload.

3.1.8.2.4 First Exchange with Known Peer SPN

If the peer security principal name (SPN) is known, the first exchange is as follows:

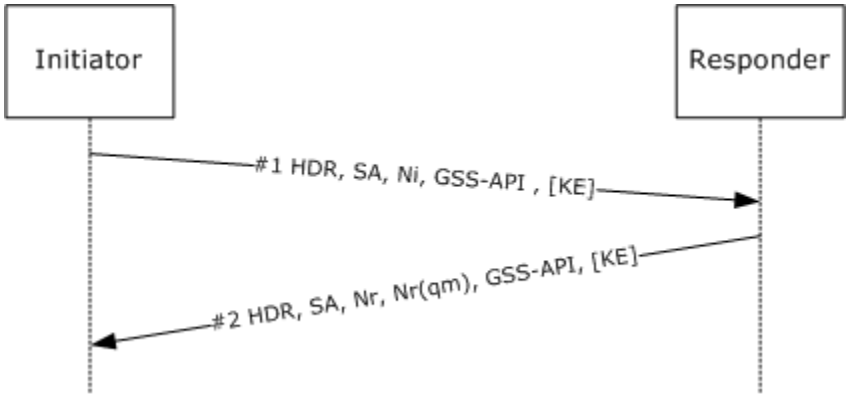


Figure 8: First exchange with known SPN

If the peer SPN is known, the initiator MUST include one or more authentication method in its Authentication (Auth) payload for the GSS-API exchange in message #1. The GSS-API payload MUST correspond to the first authentication method in the Authentication (Auth) payload. If the responder chooses the same first authentication method, it MUST process the GSS-API payload, it

MUST respond with the corresponding GSS-API payload, and it MUST not send its SPN. To allow for Auth retry, the responder SHOULD respond with all valid Auth methods as described in section 3.1.8.2.1.

If the responder does not choose the same first authentication method, the responder MUST ignore the GSS-API. The initiator MUST then send a new GSS-API payload for the negotiated GSS-API type on the next exchange (message #3).

If the responder encounters an error in processing the GSS-API payload from message #1 but other authentication methods or credentials may succeed it MUST send a GSS-API payload indicating the error to trigger the Authentication Retry (see section 3.1.2.2).

Peer SPN is known, initiator with incorrect first GSS-API, second GSS-API is correct:

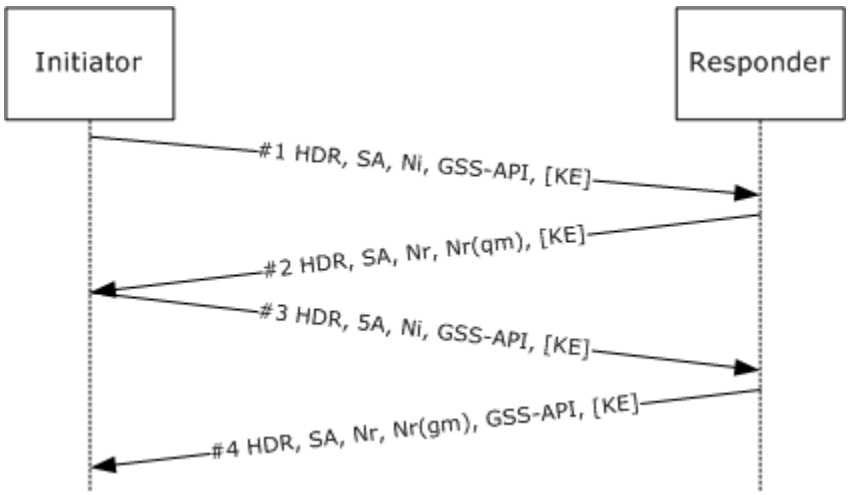


Figure 9: Peer SPN known, first GSS-API incorrect, second GSS-API correct

The initiator creates a main mode security association (MM SA) in its **main mode security association database (MMSAD)** before constructing message #1. The responder creates an MM SA in its MMSAD before processing message #2.

After the first exchange has completed without error, the initiator starts a quick mode (QM) exchange if the GSS-API exchange completed with message #2. Otherwise, the initiator starts a GSS-API exchange. If the exchange fails, the responder MUST delete its state for this MM SA from the MMSAD (see section 3.1) and the responder MUST send a Notify message with a Notify type of NOTIFY_STATUS. Upon receipt of this message, the peer MUST delete its state for this MM SA.

3.1.8.2.5 GSS-API Exchanges

The GSS-API exchanges are specified in [GSS]. The generalized form of the GSS-API exchanges is the following:

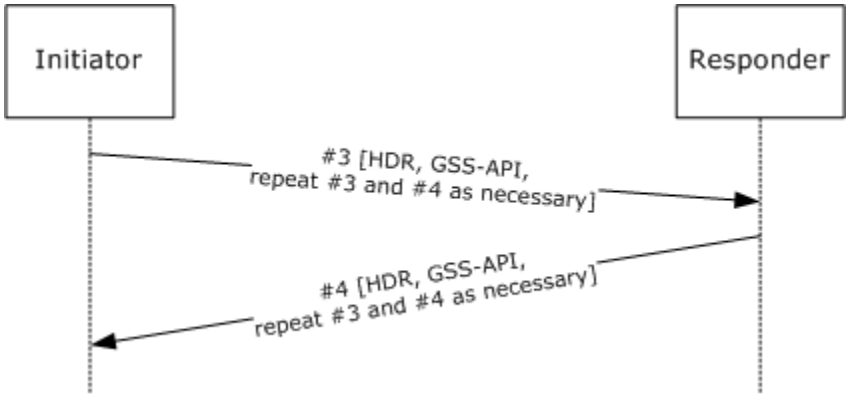


Figure 10: Generalized GSS-API exchange

The initiator MUST start the GSS handshake using the first GSS-API type returned in the responder's Auth payload. If the GSS handshake fails, initiator and responder try the next GSS-API type in order from the responder's Auth payload. See section 3.1.2.2 for more details.

The responder MUST verify that message #3 is constructed as follows:

- HDR: The Internet Security Association and Key Management Protocol (ISAKMP) header MUST use exchange type 243 (MM Exchange Type). The Encrypted flag SHOULD NOT be set.<20>
- The remaining payloads MUST be after a Crypto payload. If an error has occurred at this stage or before, the responder MUST silently discard the message.
- GSS-API: As defined in [GSS]. If the initiator is using explicit credentials, it MUST set the GSS_EXPLICIT_CREDENTIALS in the **Flags** field of the GSS-API payload at the beginning (see section 2.2.3.1).

If the responder encounters no errors in processing message #3, it MUST construct message #4 in response as follows:

- HDR: The ISAKMP header MUST use exchange type 243 (main mode (MM) exchange type). The Encrypted flag MUST NOT be set.
- The remaining payloads MUST be after a Crypto payload.
- GSS-API: As defined in [GSS].

The initiator and responder MUST implement the authentication retry state machine (described in section 3.1) to determine if further GSS-API exchanges are required.

If the GSS-API exchange completes successfully, both the initiator and the responder compute the SKEYID_a and SKEYID_e keys for the main mode security association (MM SA) by using the keying material generation algorithm described in section 3.1.

If the current exchange fails and no more authentication methods are available, the initiator MUST fail the negotiation and send a Notify message with a Notify type of NOTIFY_STATUS.

If the GSS-API authentication completes successfully, the initiator MUST start the quick mode (QM) exchange or the extended mode (EM) exchange if required by the SAD. If all exchanges fail and authentication retry is no longer possible, all responder state for this MM SA is deleted from the main mode security association database (MMSAD) (see section 3.1) and the responder MUST send a Notify message with a Notify type of NOTIFY_STATUS. Upon receipt of this message, the initiator

MUST delete its state for its corresponding MM SA. If other authentication methods are available, the responder MUST reply with an empty GSS-API payload with a non-zero error code in the Status field.

3.1.8.3 Quick Mode Exchanges

The generalized form of quick mode exchanges is as follows:

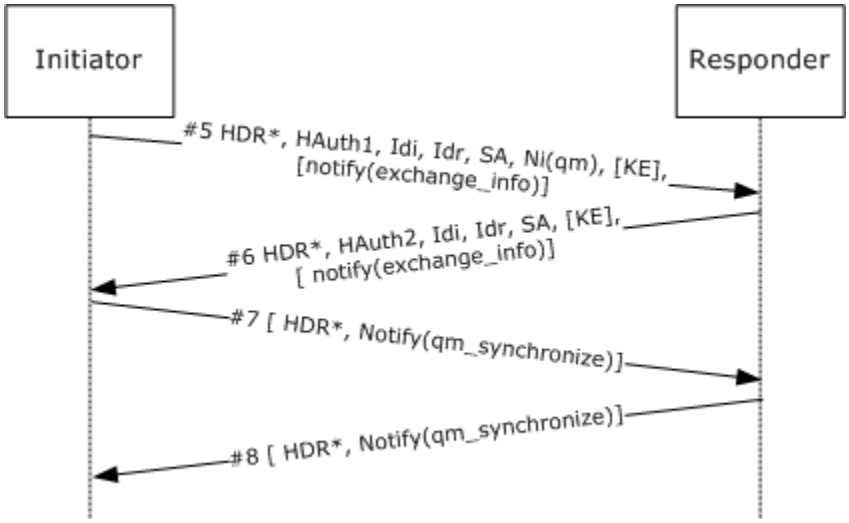


Figure 11: Quick mode exchanges

The Authenticated Internet Protocol supports two types of quick mode (QM) exchanges: normal (two exchanges) and fast (one exchange). The initiator MAY use fast QM if its SPD lookup returns a single QM proposal and quick mode **perfect forward secrecy** is not required by the SPD. The responder MUST implement fast QM. The Initiator SHOULD implement fast QM.<21>

The first QM exchange (normal or fast) MUST be embedded in the MM exchange.

The responder MUST verify that message #5 is constructed as follows:

- HDR: The Internet Security Association and Key Management Protocol (ISAKMP) header is identical to the first Internet Key Exchange (IKE) phase 2 initiator packet (as specified in [RFC2409] section 5.5), except that the exchange type is 243 (MM exchange type) for the first quick mode exchange under this main mode or 244 (QM exchange type) for any subsequent QM exchange. The Encrypted flag MUST be set.
- The remaining payloads MUST be encapsulated in a Crypto payload. If an error has occurred at this stage or before, the responder MUST silently discard the message. The crypto payload MUST be constructed in the following way:
  - The **seqNum** field is set to the current value of the main mode security association database (MMSAD) sequence number for the current exchange type, as specified in section 3.1.4.
  - The **Initialization Vector** and **Encrypted** payload fields are computed according to [RFC4303] section 2.3.
  - The **Padding** field is computed according to [RFC4303] section 2.4.

    - The **Integrity Checksum Data** field is computed according to [RFC4303] section 2.8, and covers the ISAKMP Header to the beginning of the **Integrity Checksum Data**. This is computed using the HMAC of the hash algorithm negotiated in main mode. The default mechanism specified in [RFC4303] MUST be used to generate the pad contents.

- HAuth1: The ISAKMP Hash payload containing Auth1. Auth1 is computed as defined in section 3.1.

- IDi, IDr, SA, Ni(qm), KE: These payloads are identical to the corresponding IKE payloads. The security association (SA) IDi, and IDr payloads are constructed by looking up the SPD. The key exchange (KE) payload is sent if quick mode (QM) perfect forward security (PFS) is required by the SPD.

- Notify(exchange_info): The EXCHANGE_INFO Notify payload MAY be used to indicate to the peer that either guaranteed encryption is active, or that the sender uses negotiation discovery. These two policy options apply to both IKE and Authenticated Internet Protocol and are documented in [MS-IKEE].

Upon receipt of message #5, the responder MUST:

- Decrypt the Crypto payload.

- Compute Auth1.

- Verify that the HAuth1 payload contains the computed Auth1 value.

- Consult the SPD to select a proposal from the SA and IDi, IDr payloads and generate the KE using Diffie-Hellman, as in IKE.

If the responder encounters no errors while processing message #5 and at least one of the initiator's proposals in acceptable, it MUST construct message #6 in response as follows:

- HDR: The ISAKMP header is identical to the first IKE phase 2 initiator packet (as specified in [RFC2409] section 5.5), except that the exchange type is 243 (MM exchange type) for the first quick mode exchange under this main mode or 244 (QM exchange type) for any subsequent QM exchange. The Encrypted flag MUST be set.

- The remaining payloads MUST be encapsulated in a Crypto payload. The crypto payload MUST be constructed as described in this section.

- HAuth2: The ISAKMP Hash payload containing Auth2. Auth2 is computed as defined in section 3.1.

- IDi, IDr, SA, KE: These payloads are identical to the corresponding IKE payloads. The SA and IDi, IDr payloads are constructed.

- Notify(exchange_info): The EXCHANGE_INFO Notify payload MAY be used to indicate to the peer that either guaranteed encryption is active, or that the sender uses negotiation discovery. These two policy options apply to both IKE and Authenticated Internet Protocol, and are specified in [MS-IKEE].

If no proposal is acceptable, the responder MUST send a Notification payload indicating that no proposal was chosen. This is similar to IKE (see [RFC2408] section 5.4).

Upon receipt of message #6, the initiator MUST verify Auth2. If the verification fails, the initiator MUST send a NOTIFY_STATUS Notify payload and delete its state.

If fast QM is not selected, the QM exchange MUST be followed by a second exchange (messages #7 and #8), which is identical to IKE v1. In particular, messages #7 and #8 MUST contain only the Notify payload of Notify type NOTIFY_QM_SYNCHRONIZE.

The QM SA keys are computed at this stage by using the key material generation algorithm from section 3.1. The first ipsechashLength bytes of IPSecEncryptKey are the QM SA authentication key and the last ipseccryptLength bytes are the QM SA encryption key. This is identical to IKE.

If no extended mode (EM) exchange is following the QM exchange, the following QM SA addition rules apply to the initiator and responder.

Initiator:

Inbound QM SA added to the SAD prior to sending #5 (or #7 for normal QM).

Outbound QM SA added to the SAD after processing #6 (or #8 for normal QM).

Responder:

Inbound/Outbound QM SAs added to the SAD prior to sending #6 (or #8 for normal QM).

Otherwise, the QM SA addition is deferred until the end of the EM exchange. The responder MUST look up the SAD to determine if an EM exchange is expected.

After the QM exchanges have completed without error, the initiator may start an EM exchange if the PAD requires it. If one of the QM exchanges fails, the responder state for this MM SA is deleted from the MMSAD and the responder MUST send a Notify message with a Notify type of NOTIFY_STATUS. Upon receipt of this message, the initiator MUST delete its state for its corresponding MM SA.

3.1.8.4 Extended Mode Exchanges

An extended mode (EM) exchange MAY be carried out on top of the main mode (MM) to include an additional mutual or one-way authentication phase. Extended mode can be configured in many ways.<22>

The EM exchange MUST happen after the quick mode (QM) negotiation (message #6 from section 3.1.8.3).

The generalized form of the EM exchange is the following:

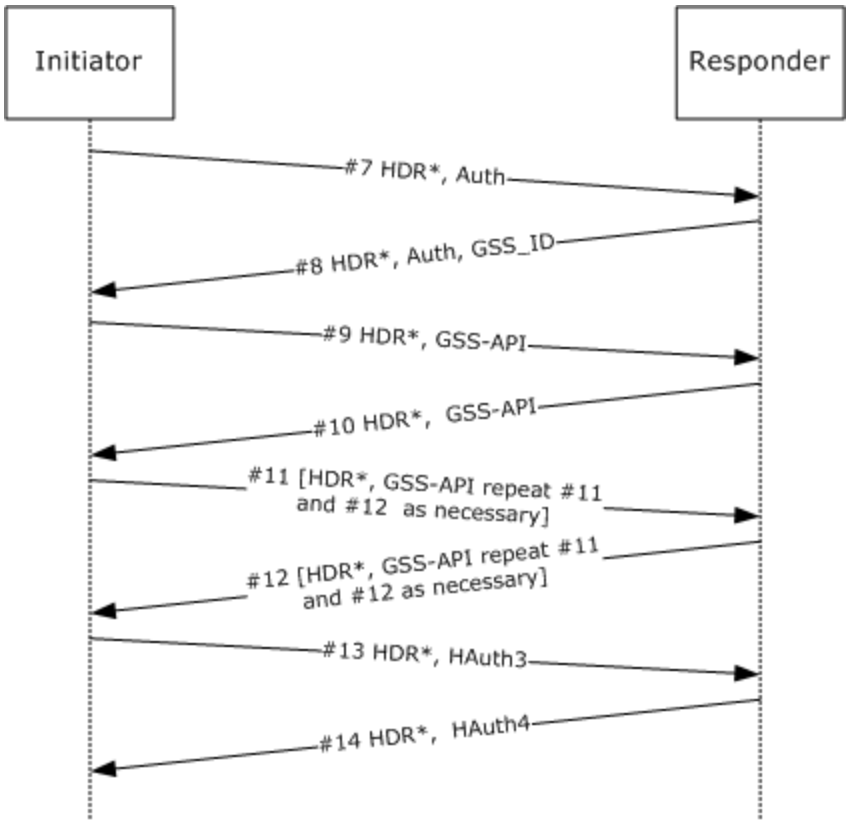


Figure 12: EM exchange

The responder MUST verify that message #7 is constructed as follows:

- HDR: The Internet Security Association and Key Management Protocol (ISAKMP) header is identical to the first Internet Key Exchange (IKE) phase 2 initiator packet (as specified in [RFC2409] section 5.5), except that the exchange type is 245 (EM exchange type). The Encrypted flag MUST be set. The responder SHOULD also verify that the message ID field MUST be one.<23>
- The remaining payloads MUST be encapsulated in a Crypto payload. If an error has occurred at this stage or before, the responder MUST silently discard the message. The crypto payload MUST be constructed as described in section 3.1.8.3.
- Auth: The proposed authentication methods. Processing is identical to the MM exchange.

If the responder encounters no errors while processing message #7, the responder MUST construct message #8 in response as follows:

- HDR: The ISAKMP header is identical to the first IKE phase 2 initiator packet (as specified in [RFC2409] section 5.5), except that the exchange type is 245 (EM exchange type). The Encrypted flag MUST be set. The responder MUST also set the message ID value to zero.
- The remaining payloads MUST be encapsulated in a Crypto payload. The crypto payload MUST be constructed as described in section 3.1.8.3.

- Auth: The chosen authentication method selected by looking up the PAD. Processing is identical to the MM exchange.
- GSS_ID: Identical to the MM exchange.

Upon receipt of message #8, the initiator MUST send the GSS-API payload in message #9. The GSS-API negotiation proceeds identically to section 3.1.8.2.5 with message #10 to #12. The initiator MUST then send message #13.

The responder MUST verify that message #13 is constructed as follows:

- HDR: The ISAKMP header is identical to the first IKE phase 2 initiator packet (as specified in [RFC2409] section 5.5), except that the exchange type is 245 (EM exchange type). The Encrypted flag MUST be set.
- The remaining payloads MUST be encapsulated in a Crypto payload. The crypto payload MUST be constructed as described in section 3.1.8.3. If an error has occured at this stage or before, the responder MUST silently discard the message.
- HAuth3: The ISAKMP Hash payload containing Auth3. Auth3 is computed as defined in section 3.1.

Upon receipt of message #13, the responder MUST:

- Decrypt the Crypto payload.
- Compute Auth3.
- Verify that the HAuth3 payload contains the computed Auth3 value.

If the responder encounters no errors while processing message #13, the responder MUST send message #14 in response.

The responder MUST verify that message #14 is constructed as follows:

- HDR: The ISAKMP header is identical to the first IKE phase 2 initiator packet (as specified in [RFC2409] section 5.5), except that the exchange type is 245 (EM exchange type). The Encrypted flag MUST be set.
- The remaining payloads MUST be encapsulated in a Crypto payload. The crypto payload MUST be constructed as described in section 3.1.8.3. If an error has occured at this stage or before, the responder MUST silently discard the message.
- HAuth4: The ISAKMP Hash payload containing Auth4. Auth4 is computed as defined in section 3.1.

Upon receipt of message #14, the initiator MUST:

- Decrypt the Crypto payload.
- Compute Auth4.
- Verify that the HAuth4 payload contains the computed Auth4 value.

The following QM SA addition rules apply to the initiator and responder.

- Initiator adds inbound QM SA to the SAD immediately before sending payload containing Auth3 (message #13).

- Responder adds both QM SAs to the SAD before sending Auth4 (message #14).
- Initiator adds outbound QM SA to the SAD after verifying Auth4 (message #14).

3.1.9 Timer Events

3.1.9.1 Negotiation Retransmission Timer

The negotiation retransmission timer is started by the initiator when it sends a message. Upon expiration of this timer, the initiator MUST retransmit the message. After a maximum number of retransmissions, the initiator MUST clean up the state for the corresponding main mode security association (MM SA).

3.1.9.2 Notify Retransmission Timer

The notify retransmission timer is started by the initiator or the responder when it sends a Notify payload within a message. Upon expiration of this timer, the initiator MUST retransmit the Notify payload. After a maximum number of retransmissions, the initiator MUST clean up the state for the corresponding main mode security association (MM SA).

3.1.9.3 Responder Time-Out Timer

This timer MUST be started by the responder upon receipt of a message from the initiator if the responder expects a subsequent message from the initiator. Upon expiration of this timer, the responder MUST delete the corresponding main mode security association (MM SA).

3.1.9.4 MM SA Lifetime

The main mode security association (MM SA) lifetime is negotiated between the peers. Upon expiration of this timer, the initiator or responder MUST delete the corresponding MM SA.

3.1.9.5 QM Rekey Timer

This timer starts at the beginning of a quick mode (QM) rekey. Upon expiration, any other quick mode security associations (QM SAs) MUST be deleted.

3.1.9.6 QM SA Lifetime

A quick mode (QM) security association (SA) rekey is triggered by the Internet Protocol security (IPsec) implementation when the QM SA lifetime (bytes or seconds) has expired.

The exchanges are the following:

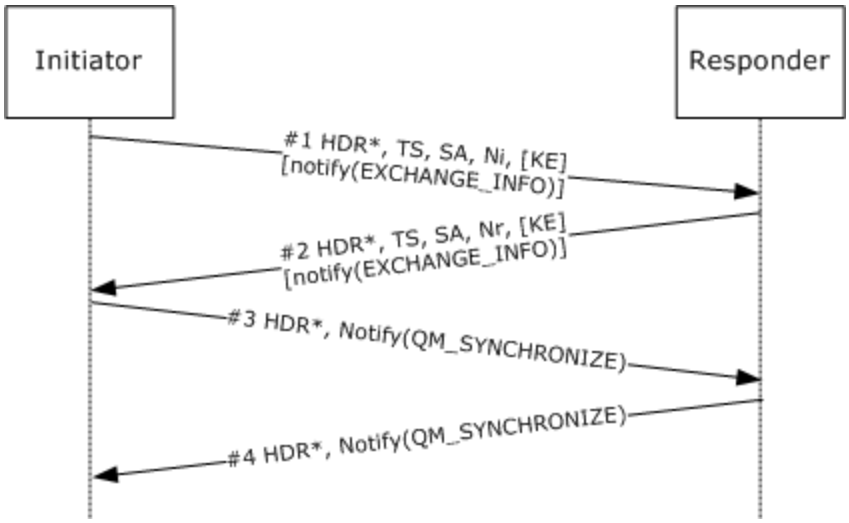


Figure 13: QM SA lifetime exchanges

The responder MUST verify that message #1 is constructed as follows:

- HDR: the Internet Security Association and Key Management Protocol (ISAKMP) header is identical to the first Internet Key Exchange (IKE) phase 2 initiator packet (as specified in [RFC2409] section 5.5), except that the exchange type is 244 (QM exchange type). The Encrypted flag MUST be set.
- The remaining payloads MUST be encapsulated in a Crypto payload. The Crypto payload MUST be constructed as specified in section 3.1.8.3. If an error has occured at this stage or before, the responder MUST silently discard the message.
- IDi, IDr, SA, Ni[qm], KE: these payloads are identical to the corresponding IKE payloads. The SA and IDi, IDr payloads are constructed by looking up the SPD. The KE payload is sent if QM Perfect Forward Secrecy (PFS) is required by the SPD.
- Notify(exchange_info): The EXCHANGE_INFO Notify payload MAY be used to indicate to the peer that either guaranteed encryption is active, or that the sender uses negotiation discovery. These two policy options apply to both IKE and Authenticated Internet Protocol and are documented in [MS-IKEE].

Upon receipt of message #1, the responder MUST:

- Decrypt the Crypto payload.
- Consult the SPD to select a proposal from the SA and IDi, IDr payloads and generate the KE using Diffie-Hellman, as in IKE.

If the responder encounters no errors while processing message #1, the responder MUST construct message #2 in response as follows:

- HDR: The ISAKMP header is identical to the first IKE phase 2 initiator packet (as specified in [RFC2409] section 5.5), except that the exchange type is 244 (QM exchange type). The Encrypted flag MUST be set.
- The remaining payloads MUST be encapsulated in a Crypto payload. The Crypto payload MUST be constructed as specified in section 3.1.8.3.

- IDi, IDr, SA, Nr[qm], KE: These payloads are identical to the corresponding IKE payloads. The SA and IDi, IDr payloads are constructed.
- Notify(exchange_info): The EXCHANGE_INFO Notify payload MAY be used to indicate to the peer that either guaranteed encryption is active, or that the sender uses negotiation discovery. These two policy options apply to both IKE and Authenticated Internet Protocol and are documented in [MS-IKEE].

This QM exchange MUST be followed by a second exchange (messages #3 and #4). This is identical to IKE v1. As in IKE v1, messages #3 and #4 MUST only contain the Notify payload of Notify type NOTIFY_QM_SYNCHRONIZE.

The QM SA keys are computed at this stage using the key material generation algorithm from section 3.1. The first ipsechashLength bytes of IPSecEncryptKey are the QM SA authentication key and the last ipseccryptLength bytes are the QM SA encryption key. This is identical to IKE.

The following quick mode security association (QM SA) addition rules apply to the initiator and responder.

Initiator:

Inbound QM SA added to the SAD prior to sending #3.

Outbound QM SA added to the SAD after processing #4.

Responder:

Inbound/Outbound QM SAs added to the SAD prior to sending #4.

The initiator and responder SHOULD start the rekey timer for the old inbound SAs. When the timer expires, the old inbound SAs MUST be removed from the SAD.<24>

3.1.9.6.1 Quick Mode Rekey Acquire Notification

A Notify payload with a Notify type of NOTIFY_ACQUIRE is sent if ALL of the following conditions are met:

- Quick mode (QM) rekey is requested by the Internet Protocol security (IPsec) stack on the initiator.
- The corresponding main mode security association (MM SA) cannot be used for the QM rekey (for example, the maximum number of QMs for this main mode (MM) has been reached).
- One-way authentication was used for establishing the MM.

If one-way authentication was used for establishing the MM SA (for example, one-way certificate in MM, and NT LAN Manager (NTLM) authentication protocol in EM) the original initiator and responder roles cannot be reversed while keying a new MM, as such a reversal would cause the MM negotiation to fail. The Acquire notification is necessary to avoid reversing the roles in the MM exchange. The exchange is the following:

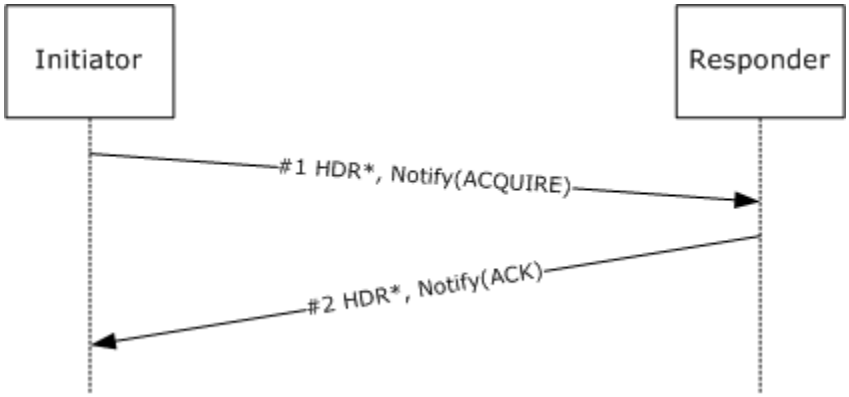


Figure 14: Quick mode rekey acquire notification exchange

Upon receipt, the responder MUST initiate a QM rekey using the two phase 2 Identification payloads and the Encapsulation Mode present in the Acquire notification message (see section 2.2.3.5).

The Internet Security Association and Key Management Protocol (ISAKMP) header in messages #1 and #2 MUST use exchange type 246 (Notify exchange type). The Crypto payload is encrypted by using the MM SA keys. The Notify payload in message #1 has the Reliable notify flag set (see section 2.2.3.5) and MUST be acknowledged.

3.1.10 Other Local Events

The behavior to IP address plug and play notifications can be implemented in many ways.

Responder throttle: The thresholds for the responder throttle can be implemented with different values.

Shutdown: The response to a machine shutdown can be implemented in many ways.

Policy change: The response to Policy Change can be implemented in many ways.

SA deletion: The response to SA deletion can be implemented in many ways.<25>

4 Protocol Examples

The following sections describe several operations as used in common scenarios to illustrate the function of the Authenticated Internet Protocol.

4.1 Main Mode - No Extended Mode

The following details a main mode (MM) exchange without a Diffie-Hellman exchange where the GSS-API secret is used to generate the keying material.

The exchanges are the following:

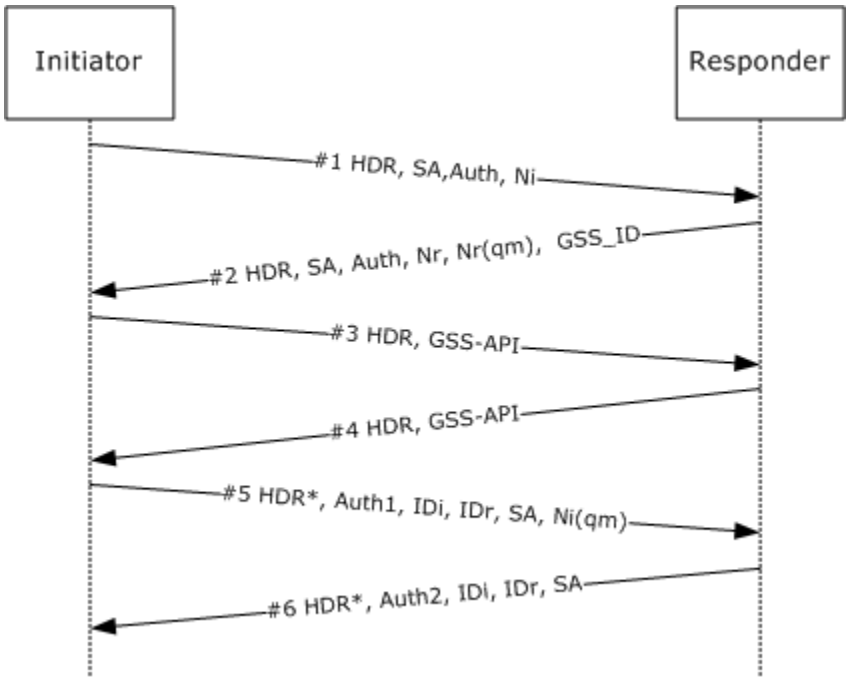


Figure 15: Main mode - no extended mode exchange

Message #1: The initiator sends security association (SA) proposals (Kerberos or TLS), the MM crypto options, and a nonce. Diffie-Hellman is not required, and the initiator does not send a key exchange (KE) payload.

Message #2: The responder looks up the policy based on IP addresses only. It chooses an SA proposal, such as Kerberos with 3DES. The responder also sends its SPN and the MM nonce, and the quick mode (QM) nonce.

Message #3: The initiator calls GSS-API to get a token for the peer SPN. This token is sent unmodified.

Message #4: The responder calls GSS-API on the initiator's token. This generates a new token, which the responder sends back. In this example, only one exchange is needed. The responder's GSS-API exchange is done, and a GSS-API cryptographic context is available.

Message #5: The initiator calls GSS-API on the responder token. This succeeds, and the initiator now has a cryptographic context. All the previous payloads are hashed, and signed with the key generated by the GSS-API exchange. This is the HAuth1 payload. The initiator includes the traffic

selectors (IDi, IDr) for the traffic to secure on the wire, and proposals for encryption/integrity in the QM SA payload, and a nonce.

Message #6: The responder verifies HAuth1, and generates a similar HAuth2. The responder looks up its policy based on the traffic selectors. The responder sends back the traffic selectors and SA proposals it has chosen. The responder adds the QM SAs at this point.

When the initiator receives the last packet, it validates HAuth2, and adds its QM SAs. If the last packet sent by the responder is lost, the initiator retransmits its last packet. Upon receiving it, the responder resends its last Authenticated Internet Protocol packet.

4.2 Kerberos Extended Mode

The following example shows a Kerberos authentication in extended mode (EM):

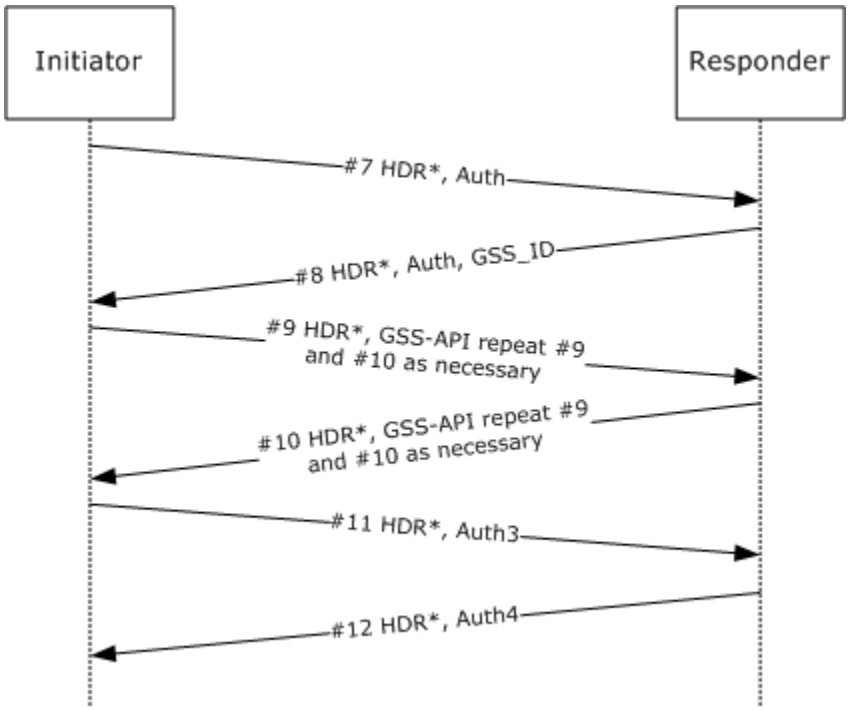


Figure 16: Kerberos extended mode authentication

Message numbering starts on top of the base mode numbering. Message 7 here replaces message 7 above.

The following security association (SA) addition rules apply to the initiator and responder.

Initiator:

- Inbound SA added prior to sending #9.
- Outbound SA added after processing #10.

Responder:

- Inbound/Outbound SAs added prior to sending #10.

4.3 Extended Mode Authentication Retry

In extended mode (EM), failures are handled by retrying the authentication if possible. The following example shows an authentication retry:

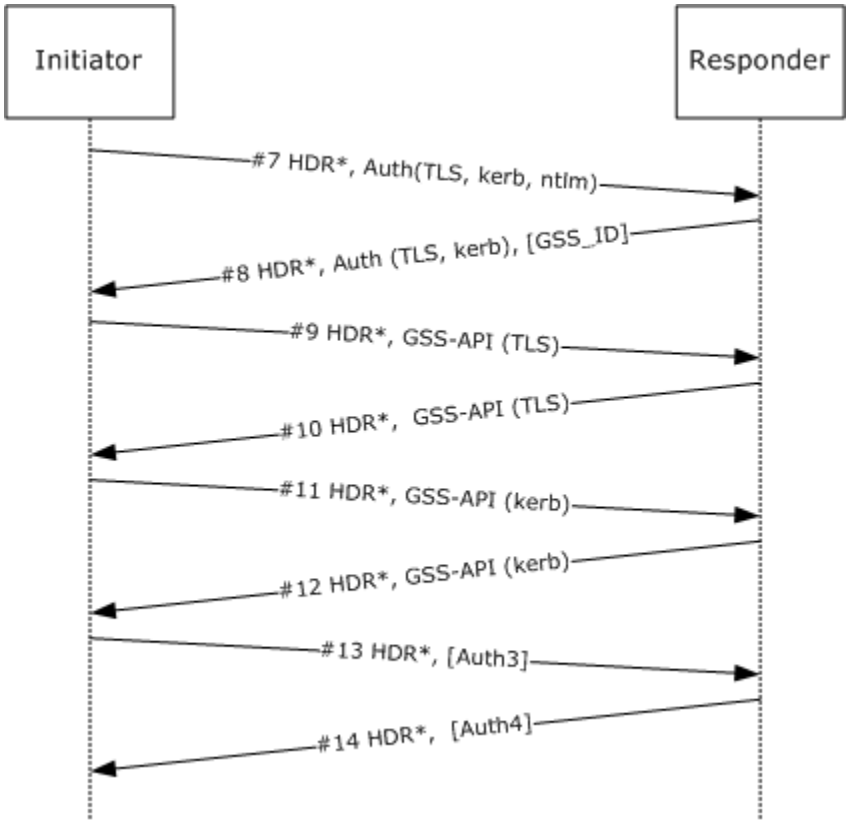


Figure 17: Extended mode authentication retry

In message #7, the initiator proposes TLS, Kerberos, and NT LAN Manager (NTLM) authentication protocol. The responder, in message #8, accepts only TLS and Kerberos. The initiator begins the TLS exchange in message #9. Any failures during TLS in message #9 and message #10 cause empty GSS-API payloads to be sent. When constructing message #11, the initiator attempts the next agreed upon authentication method, which is Kerberos. Whenever the initiator begins a new GSS-API exchange, it sets the GSS_NEW_GSS_EXCHANGE bit in the flags field of the GSS-API payload. Only the initiator can begin a new GSS-API exchange. If the responder fails and wants to initiate a new GSS-API exchange, it sets the status field to a non-zero value corresponding to the error code <26> in the next GSS-API payload. This signals the initiator to retry.

5 Security

The following sections specify security considerations for implementers of the Authenticated Internet Protocol.

5.1 Security Considerations for Implementers

The security considerations listed for Internet Key Exchange (IKE) v1 in the relevant RFCs apply to the Authenticated Internet Protocol. In addition, there are protocol-specific security considerations.

5.1.1 Policy Construction

The Authenticated Internet Protocol supports different authentication methods which do not necessarily offer the same strength. This should be taken into account when designing Authenticated Internet Protocol policies in order to minimize the risk of downgrade attacks.

5.1.2 Credential/Identity Protection

When the Authenticated Internet Protocol performs an authentication, it may be necessary for one party to reveal information about its identity to the other party in order to perform the authentication. The type of information leaked depends on the authentication method used as described below. The choice of authentication methods supported will therefore affect the privacy of the authenticating parties.

Kerberos

In a Kerberos authentication exchange, the initiator never has to leak its identity and only members of the trusted domain are able to access the ticket sent, assuming the integrity of Kerberos. The Kerberos responder needs to provide information to the initiator about its identity so that the Kerberos token can be validated; to do this, the responder puts its Security Principal Name (SPN) in the Principal Name field of the GSS_ID payload.

Certificates

As part of a TLS exchange, the responder may pass a list of trusted roots in the clear so that the initiator can select a certificate that is trusted by the responder.

5.2 Index of Security Parameters

Security Parameter	Section
Authentication method	1.7
Encryption/authentication algorithms	1.7
Diffie-Hellman	1.7
Target principal name	2.2.2
Key derivation function	3.1

6 Appendix A: Windows Behavior

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

- Windows Vista
- Windows Server 2008

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. Implemented in Windows Vista and Windows Server 2008. The following Internet Key Exchange Protocol Extensions (defined in [MS-IKEE]) are supported:

IKE protocol extension [MS-IKEE]
IKE Fragmentation
Fast Failover
Negotiation Discovery
Denial of Service Protection

<2> Section 1.7: Authentication Methods. Implemented in Windows Vista and Windows Server 2008. The following authentication methods are supported:

Authentication Method [MS-RPCE]	**Options**
Kerberos [RFC4120]	Mutual authentication only.
[MS-NLMP]	One-way authentication only (client authentication).
TLS [RFC3546]	One-way or mutual authentication. If one-way, always server authentication.

<3> Section 1.7: Cryptographic parameters. The Authenticated Internet Protocol is implemented in Windows Vista and Windows Server 2008 only. Microsoft implements the following algorithms:

Message Authentication Algorithm	**Operating Systems**
NULL [RFC3505]	Windows Vista, Windows Server 2008
HMAC-SHA1-96 [RFC2404]	Windows Vista, Windows Server 2008
HMAC-MD5-96 [RFC2403]	Windows Vista, Windows Server 2008
AES-GMAC [RFC4543]	Windows Server 2008
SHA-256 [SHA256]	Windows Server 2008

Encryption Algorithm	Operating Systems
NULL [RFC3505]	Windows Vista, Windows Server 2008
DES-CBC [RFC2405]	Windows Vista, Windows Server 2008
TripleDES-CBC [RFC2451]	Windows Vista, Windows Server 2008
AES-CBC with 128, 192, and 256 bits keys [RFC3602]	Windows Vista, Windows Server 2008
AES-GCM with 128, 192, and 256 bits keys [RFC4106]	Windows Server 2008

Diffie-Hellman	Operating Systems
default 768-bit MODP group [RFC2409]	Windows Vista, Windows Server 2008
alternate 1024-bit MODP group [RFC2409]	Windows Vista, Windows Server 2008
2048-bit MODP group [RFC3526]	Windows Vista, Windows Server 2008
ECP_256 [ECP]	Windows Vista, Windows Server 2008
ECP_384 [ECP]	Windows Vista, Windows Server 2008

<4> Section 1.7: Capability Negotiation. Implemented in Windows Vista and Windows Server 2008. The following vendor IDs are supported by the Microsoft implementation of the Authenticated Internet Protocol:

Common Name	String Representation	Wire Representation (MD5 hash of string)
MSFT implementation [MS-IKEE]	"MS NT5 ISAKMPOAKLEY" +version 5	1E 2B 51 69 05 99 1C 7D 7C 96 FC BF B5 87 E4 61 00 00 00 05
Kerberos authentication supported [GSS]	"GSSAPI"	62 1B 04 BB 09 88 2A C1 E1 59 35 FE FA 24 AE EE
NLB/MSCS fast failover supported [MS-IKEE]	"Vid-Initial-Contact"	26 24 4D 38 ED DB 61 B3 17 2A 36 E3 D0 CF B8 19
NLB/MSCS fast failover supported [MS-IKEE]	"NLBS_PRESENT"	72 87 2B 95 FC DA 2E B7 08 EF E3 22 11 9B 49 71
Fragmentation Avoidance supported [MS-IKEE]	"FRAGMENTATION"	40 48 B7 D5 6E BC E8 85 25 E7 DE 7F 00 D6 C2 D3
NAT-T supported [MS-IKEE]	"RFC 3947"	4A 13 1C 81 07 03 58 45 5C 57 28 F2 0E 95 45 2F
Negotiation Discovery supported [MS-IKEE]	"MS-Negotiation Discovery Capable"	FB 1D E3 CD F3 41 B7 EA 16 B7 E5 BE 08 55 F1 20

<5> Section 1.7: Vendor ID Payload. Implemented in Windows Vista and Windows Server 2008. The following Vendor ID payload is sent by the Microsoft IKE v1 implementation when the initiator or responder supports both IKE v1 and the Authenticated Internet Protocol:

Common Name	String Representation	Wire Representation (MD5 hash of string)
Authenticated Internet Protocol supported	MS-MamieExists	21 4C A4 FA FF A7 F3 2D 67 48 E5 30 33 95 AE 83

<6> Section 2.1: UDP ports. Implemented in Windows Vista and Windows Server 2008. The Authenticated Internet Protocol runs on UDP ports 500 and 4500. The UDP ports are not configurable.

<7> Section 2.2.1: Version field handling. Implemented in Windows Vista and Windows Server 2008. Packets with a minor version number larger than 0 are accepted.

<8> Section 2.2.3.1: Error codes. Implemented in Windows Vista and Windows Server 2008. The Authenticated Internet Protocol logs the failure to the Security Event Log.

<9> Section 2.2.3.2.2: In the Windows Vista and Windows Server 2008 implementations of the Authenticated Internet Protocol, the responder adds 8 bytes of **Initialization Vector** after the seqNUM field if the GSS-API exchange has completed. The presence of the **Initialization Vector** is indicated by the length of the Crypto payload (16 bytes if the **Initialization Vector** is present, 8 bytes otherwise).

<10> Section 2.2.3.5: Error codes. Implemented in 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].

<11> Section 3.1.2.1: Negotiation Retransmission timer. Implemented in Windows Vista and Windows Server 2008. See section 3.1.5 and its associated Windows behavior information for details on Windows implementation of retransmission timers.

<12> Section 3.1.3: Cryptographic parameters. The Authenticated Internet Protocol is implemented in Windows Vista and Windows Server 2008 only. Microsoft implements the following algorithms:

Message Authentication Algorithm	Key Length (Bytes)	Operating Systems
NULL [RFC3505]	N/A	Windows Vista, Windows Server 2008
HMAC-SHA1-96 [RFC2404]	20	Windows Vista, Windows Server 2008
HMAC-MD5-96 [RFC2403]	16	Windows Vista, Windows Server 2008
AES-GMAC with 128, 192, and 256 bits keys [RFC4543]	16, 24, and 32	Windows Server 2008
SHA-256 [SHA256]	32	Windows Server 2008

Encryption Algorithm	Key Length (Bytes)	Operating Systems
NULL [RFC3505]	N/A	Windows Vista, Windows Server 2008
DES-CBC [RFC2405]	8	Windows Vista, Windows Server 2008
TripleDES-CBC [RFC2451]	24	Windows Vista, Windows Server 2008
AES-CBC with 128, 192, and 256 bits keys [RFC3602]	16, 24, and 32	Windows Vista, Windows Server 2008
AES-GCM with 128, 192, and 256 bits keys [RFC4106]	16, 24, and 32	Windows Server 2008

Diffie-Hellman	Operating Systems
default 768-bit MODP group [RFC2409]	Windows Vista, Windows Server 2008
alternate 1024-bit MODP group [RFC2409]	Windows Vista, Windows Server 2008
2048-bit MODP group [RFC3526]	Windows Vista, Windows Server 2008
ECP_256 [ECP]	Windows Vista, Windows Server 2008
ECP_384 [ECP]	Windows Vista, Windows Server 2008

<13> Section 3.1.5: Negotiation retransmission timer, notify retransmission timer, responder time out, NAT-T keep-alive timer, QM rekey timer. Implemented in Windows Vista and Windows Server 2008.

Negotiation retransmission timer: 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 described in section 3.1.10.

Notify retransmission timer: 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 described in section 3.1.10.

Responder time-out timer: The responder deletes its state if it does not receive a message from the initiator within 60 seconds.

NAT-T keep-alive timer: The timer expires after 20 seconds.

QM rekey timer: The timer expires after 60 seconds and all old QM SAs are deleted.

<14> Section 3.1.8.2.1: Encrypted flag verification. Implemented in Windows Vista and Windows Server 2008. The Windows Vista and Windows Server 2008 implementation does not verify that the encrypted flag not is set.

<15> Section 3.1.8.2.1: Message ID field verification. Implemented in Windows Vista and Windows Server 2008. The Windows Vista and Windows Server 2008 does not verify that the message ID field is zero.

<16> Section 3.1.8.2.1: Nonce generation. Implemented in Windows Vista and Windows Server 2008. These nonces are 32-byte cryptographically strong random numbers. These numbers are generated using a FIPS-140-compliant random number generator internal to the Windows operating system (as specified in [FIPS140]).

<17> Section 3.1.8.2.1: Proposal selection. Implemented in Windows Vista and Windows Server 2008. The responder looks up the list of proposals that it would send to this peer if the responder were the initiator, and it chooses the first such proposal.

<18> Section 3.1.8.2.1: Message ID field verification. Implemented in Windows Vista and Windows Server 2008. The Windows Vista and Windows Server 2008 does not verify that the message ID field is zero.

<19> Section 3.1.8.2.1: Authentication Methods. Implemented in Windows Vista and Windows Server 2008. The Windows implementation responds with all valid authentication methods.

<20> Section 3.1.8.2.5: Encrypted flag verification. Implemented in Windows Vista and Windows Server 2008. The Windows Vista and Windows Server 2008 implementation does not verify that the encrypted flag is not set.

<21> Section 3.1.8.3: Fast quick mode. Implemented in Windows Vista and Windows Server 2008. Fast QM is always performed if possible.

<22> Section 3.1.8.4: Extended mode policy option. Implemented in Windows Vista and Windows Server 2008. Extended mode is configured through a policy option [MSWFPSDK].

<23> Section 3.1.8.4: Message ID field verification. Implemented in Windows Vista and Windows Server 2008. The Windows Vista and Windows Server 2008 does not verify that the message ID field is one.

<24> Section 3.1.9.6: SA time out on QM rekey. Implemented in Windows Vista and Windows Server 2008. See section 6 #10 and section 3.1.5 for details on the rekey timer.

<25> Section 3.1.10: IP address plug and play notifications, DoS protection threshold, Responder Throttle, Shutdown, Policy change, and SA deletion. Implemented in Windows Vista and Windows Server 2008:

IP address plug and play notifications: The Authenticated Internet Protocol reacts to IP address plug and play notifications. If an IP address on the systems is deleted, the Authenticated Internet Protocol deletes all the corresponding SAs and sends a Notify payload of Notify type NOTIFY_STATUS to the peer for each deleted SA.

DoS Protection Threshold: The threshold is 500 negotiations.

Responder Throttle: The receiver drops incoming new negotiation requests from a peer if any of the following conditions are met:

More than 20 main mode security associations (MM SAs) established to the peer

More than 35 in-progress negotiations to the peer

Shutdown: Upon shutdown of the AuthIP protocol daemon, the Authenticated Internet Protocol sends delete notification messages (Notify payload of Notify type NOTIFY_STATUS) for all SAs and

retransmits each notification message at most once if it is not acknowledged within the negotiation retransmission timer time-out period.

Policy change: Upon policy change, the Authenticated Internet Protocol revalidates all SAs against the new policy and sends delete notification for the SAs that do not match the new policy.

SA deletion: The Windows implementation exports APIs to enumerate and delete Authenticated Internet Protocol SAs. Upon deletion, the protocol sends the corresponding delete notifications.

<26> Section 4.3: Error codes. Implemented in Windows Vista and Windows Server 2008. The Authenticated Internet Protocol logs the failure to the Security Event Log.

7 Index

Q

R

S

T

U

V

W