IPv6 Ready Logo

Phase-2 Interoperability
Test Scenario
IPsec

Technical Document

Revision 2.0.0f

IPv6 Forum *https://www.ipv6forum.org/*
IPv6 Ready Logo Committee *https://www.ipv6ready.org/*

Acknowledgments

IPv6 Forum would like to acknowledge the efforts of the following organizations in the development of this test specification.

- TAHI Project
- University of New Hampshire – Interoperability Laboratory (UNH-IOL)
- IRISA

Table of Contents

Introduction

The IPv6 Forum plays a major role to bring together industrial actors, to develop and deploy the next generation of IP protocols. Contrary to IPv4, which started with a small closed group of implementers, the universality of IPv6 leads to a huge number of implementations. Interoperability has always been considered as a critical feature in the Internet community.

Due to the large number of IPv6 implementations, it is important to provide the market a strong signal proving the level of interoperability across various products. To avoid confusion in the mind of customers, a globally unique logo program should be defined. The IPv6 logo will give confidence to users that IPv6 is currently operational. It will also be a clear indication that the technology will still be used in the future. To summarize, this logo program will contribute to the feeling that IPv6 is available and ready to be used.

Phases of the IPv6 Logo Program

Phase 1
In the first stage, the Logo will indicate that the product includes IPv6 mandatory core protocols and can interoperate with other IPv6 implementations.

Phase 2
The "IPv6 ready" step implies a proper care, technical consensus and clear technical references. The IPv6 ready logo will indicate that a product has successfully satisfied strong requirements stated by the IPv6 Ready Logo Committee (v6RLC).
To avoid confusion, the logo "IPv6 Ready" will be generic. The v6RLC will define the test profiles with associated requirements for specific functionalities.

Phase 3
Same as Phase 2 with IPsec mandated.

Requirements

To obtain the IPv6 Ready Logo Phase-2 for IPsec (IPsec Logo), the Node Under Test (NUT) must satisfy following requirements.

Equipment Type

- End-Node (EN)
  - A node that uses IPsec only for itself. Hosts and Routers can be End-Nodes
- Security Gateway (SGW)
  - A node that can provide IPsec Tunnel Mode for nodes behind it. Routers can be SGWs.
- Passive Node
  - If your device is an End-Node and cannot send ICMP Echo Request, it must play the role of TAR-EN1. Otherwise, it can play either role. In either case choose a device which can send ICMP Echo Request as TAR-EN2.

Security Protocol

NUTs must utilize ESP regardless of the type of the NUT. The IPv6 Ready Logo Program does not test AH.

Mode

The mode requirement depends on the type of NUT.

- End-Node:

  If the NUT is an End-Node, it must pass all of the Transport Mode mode tests. If the NUT supports tunnel mode, it must pass all of the Tunnel Mode tests (i.e. Tunnel mode is an advanced functionality for End-Node NUTs).

- SGW:

  If the NUT is a SGW, it must pass all of the Tunnel Mode tests.

Keying

Previous versions of this test suite required Manual Keying by default, as a minimum requirement. Developments in industry best practices have shown that Manual Keys pose a significant security risk.

According to RFC 7321bis, Section 3:

> Manual Keying is not be used as it is inherently dangerous. Without any keying protocol, it does not offer Perfect Forward Secrecy ("PFS") protection. Deployments tend to never be reconfigured with fresh session keys. It also fails to scale and keeping SPI's unique amongst many servers is impractical. This document was written for deploying ESP/AH using IKE (RFC7298) and assumes that keying happens using IKEv2.
>
> If manual keying is used anyway, ENCR_AES_CBC MUST be used, and ENCR_AES_CCM, ENCR_AES_GCM and ENCR_CHACHA20_POLY1305 MUST NOT be used as these algorithms require IKE.

Following this recommendation, a configuration using Dynamic Keying, facilitated by IKE is used by default, and specifically IKEv2. IKEv1 is obsolete and not supported. Devices which support only Manual Keys will not successfully pass these tests, as the BASIC combined-mode (AEAD) algorithms require Dynamic Keying.

When IKEv2 is used, the encryption keys and Integrity keys are negotiated dynamically. The tester should support the alternative of using IKE with dynamic keys to execute the tests. Manual Keys may be used in tests that have indicated they are acceptable. These tests are run with IKEv2, and if necessary, run again with Manual Keys.

Additionally, this test document includes tests which are specific to IKEv2, providing an additional level of confidence in that protocol's interoperability.

Test Traffic

All tests use ICMP Echo Request and Echo Reply messages by default. ICMP is independent from any implemented application and this adds clarity to the test. If the NUT cannot apply IPsec for ICMPv6 packets, it is acceptable to use other protocols rather than ICMPv6.

In this case, the device must support ICMPv6, TCP, or UDP. The application and port number are unspecified when TCP or UDP packets are used. The test coordinator should support any ports associated with an application used for the test. Applicants must mention the specific protocol and port that was used to execute the tests.

Category

In this document, the tests and algorithms are categorized into two types: BASIC and ADVANCED

ALL NUTs are required to support BASIC. ADVANCED tests are required for all NUTs which support ADVANCED encryption/Integrity algorithms. Each test description contains a Category section. The section lists the requirements to satisfy each test.

Required Tests

The table below indicates the IPv6 Ready requirement level for each test case.

Each test case is made up of individual test parts which use a specific topology. Test parts which are applicable to the DUT device type must be run.

For example, End-node devices must run all End-node vs. End-node test parts, as well as all End-node vs. SGW test parts. SGW devices must run all End-node vs. SGW test parts, as well as all SGW vs. SGW test parts.

Test Case	Title	IPv6 Ready Requirement
IPsec.IO.1.1	Basic Connection	Required
IPsec.IO.1.2	Traffic Selectors	Required
IPsec.IO.1.3	Fragmentation	Required
IPsec.IO.2.1	Authentication	Required
IPsec.IO.2.2	SA: Algorithm Mismatch (NO_PROPOSAL_CHOSEN)	Required
IPsec.IO.2.3	DH Retry (INVALID_KE_PAYLOAD)	Required
IPsec.IO.2.4	Rekeying	Advanced
IPsec.IO.3.1	IKE_SA_INIT Algorithms	Refer to Test Case
IPsec.IO.3.2	IKE_AUTH Algorithms	Refer to Test Case

References

This test specification focus on the following IPsec related RFCs.

Algorithms		
RFC2404	HMAC-SHA1	The Use of HMAC-SHA-1-96 within ESP and AH. C. Madson, R. Glenn. November 1998. (Format: TXT=13089 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC2404)
RFC2410	NULL Encryption	The NULL Encryption Algorithm and Its Use With IPsec. R. Glenn, S. Kent. November 1998. (Format: TXT=11239 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC2410)
RFC2451	ESP CBC	The ESP CBC-Mode Cipher Algorithms. R. Pereira, R. Adams. November 1998. (Format: TXT=26400 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC2451)
RFC3566	AES-XCBC-MAC	The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec. S. Frankel, H. Herbert. September 2003. (Format: TXT=24645 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC3566)
RFC3602	AES-CBC	The AES-CBC Cipher Algorithm and Its Use with IPsec. S. Frankel, R. Glenn, S. Kelly. September 2003. (Format: TXT=30254 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC3602)
RFC3686	AES-CTR	Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP). R. Housley. January 2004. (Format: TXT=43777 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC3686)
RFC4106	GCM with ESP	The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). J. Viega, D. McGrew. June 2005. (Format: TXT=23399 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4106)
RFC4309	AES-CCM	Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP). R. Housley. December 2005. (Format: TXT=28998 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4309)
RFC4543	GMAC with ESP	The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH. D. McGrew, J. Viega. May 2006. (Format: TXT=29818 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4543)
RFC4868	HMAC-SHA256, 384, 512	Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec. S. Kelly, S. Frankel. May 2007. (Format: TXT=41432 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4868)
RFC7634	ChaCha20 Poly1305	ChaCha20, Poly1305, and Their Use in the Internet Key Exchange Protocol (IKE) and IPsec. Y. Nir. August 2015. (Format: TXT=27513 bytes) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC7634)
RFC8221	ESP Req	Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH). P. Wouters, D. Migault, J. Mattsson, Y. Nir, T. Kivinen. October 2017. (Format: TXT=33610 bytes) (Obsoletes RFC7321) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC8221)
RFC8247	IKEv2 Req	Algorithm Implementation Requirements and Usage Guidance for the Internet Key Exchange Protocol Version 2 (IKEv2). Y. Nir, T. Kivinen, P. Wouters, D. Migault. September 2017. (Format: TXT=44739 bytes) (Obsoletes RFC4307) (Updates RFC7296) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC8247)
Architecture		
RFC4301	IPsec Arch	Security Architecture for the Internet Protocol. S. Kent, K. Seo. December 2005. (Format: TXT=262123 bytes) (Obsoletes RFC2401) (Updates RFC3168) (Updated by RFC6040, RFC7619) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4301)
RFC4303	ESP	IP Encapsulating Security Payload (ESP). S. Kent. December 2005. (Format: TXT=114315 bytes) (Obsoletes RFC2406) (Status: PROPOSED STANDARD) (DOI: 10.17487/RFC4303)
RFC4443	ICMPv6	Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. A. Conta, S. Deering, M. Gupta, Ed.. March 2006. (Format: TXT=48969 bytes) (Obsoletes RFC2463) (Updates RFC2780) (Updated by RFC4884) (Status: DRAFT STANDARD) (DOI: 10.17487/RFC4443)
RFC7296	IKEv2	Internet Key Exchange Protocol Version 2 (IKEv2). C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, T. Kivinen. October 2014. (Format: TXT=354358 bytes) (Obsoletes RFC5996) (Updated by RFC7427, RFC7670) (Also STD0079) (Status: INTERNET STANDARD) (DOI: 10.17487/RFC7296)

Test Topology

Topo.EN.EN.1

1. Set global address to TAR-EN1_Link0 and TAR-EN2_Link1 by RA.
2. Make IPsec transport mode or tunnel mode between TAR-EN1 and TAR-EN2.

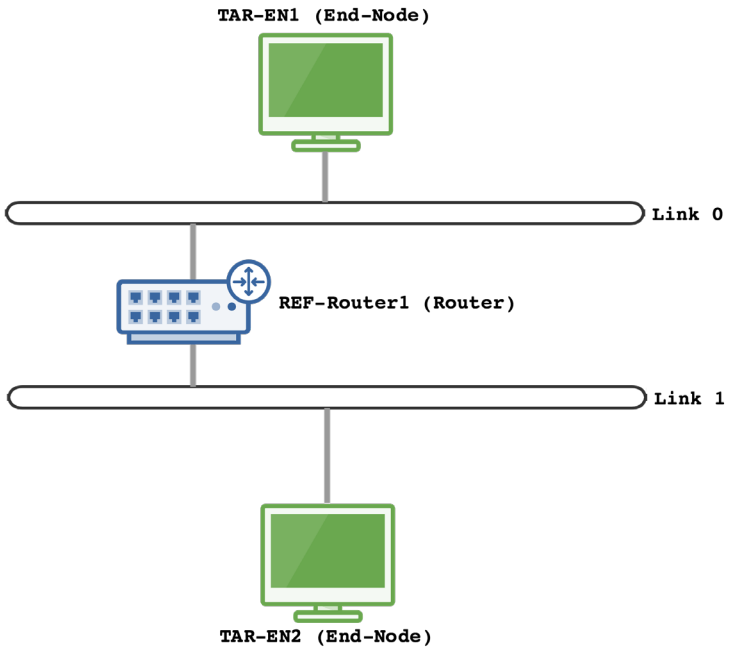

FIGURE 1 – END-NODE V. END-NODE: TRANSPORT AND TUNNEL MODE

Topo.EN.SGW.1

1. Set global address to TAR-EN1_Link0 and REF-Host1_Link2 by RA.
2. Set global address to TAR-SGW1_Link1 and TAR-SGW1_Link2 manually.
3. Set routing table to TAR-SGW1 (REF-Router1_Link1 for Link0)
4. Set routing table to REF-Router1 (TAR-SGW1_Link1 for Link2)
5. Make IPsec tunnel mode between TAR-EN1 and TAR-SGW1.

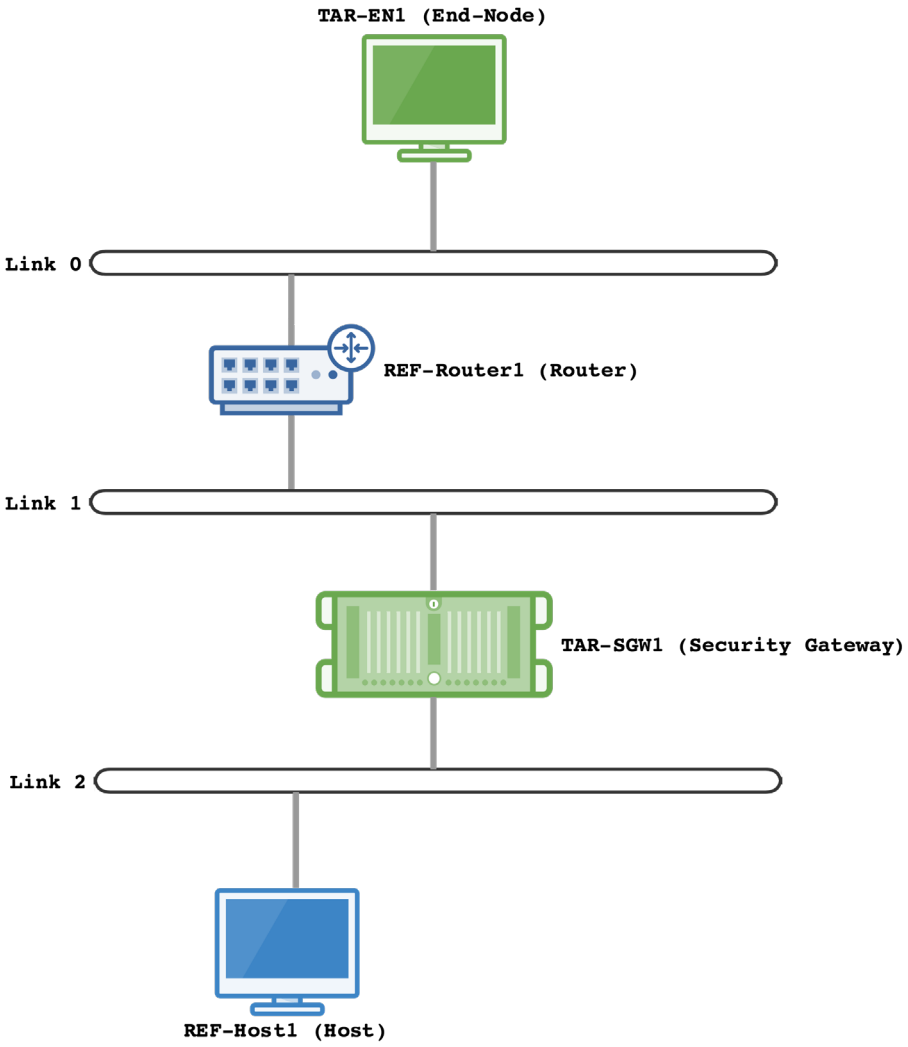

FIGURE 2 - END-NODE V. SGW

Topo.SGW.SGW.1

1. Set global address to REF-Host1_Link0 and REF-Host2_Link3 by RA.
2. Set global address to TAR-SGW1_Link0, TAR-SGW1_Link1, TAR-SGW2_Link2, TAR-SGW2_Link3, REF-Router1_Link1, REF-Router1_Link2 manually.
3. Set routing table to TAR-SGW1 (REF-Router1_Link1 for Link2 and Link3)
4. Set routing table to TAR-SGW2 (REF-Router1_Link2 for Link0 and Link1)
5. Set routing table to REF-Router1 (TAR-SGW1_Link1 for Link0, TAR-SGW2_Link2 for Link3)
6. Make IPsec tunnel mode between TAR-SGW1 and TAR-SGW2.

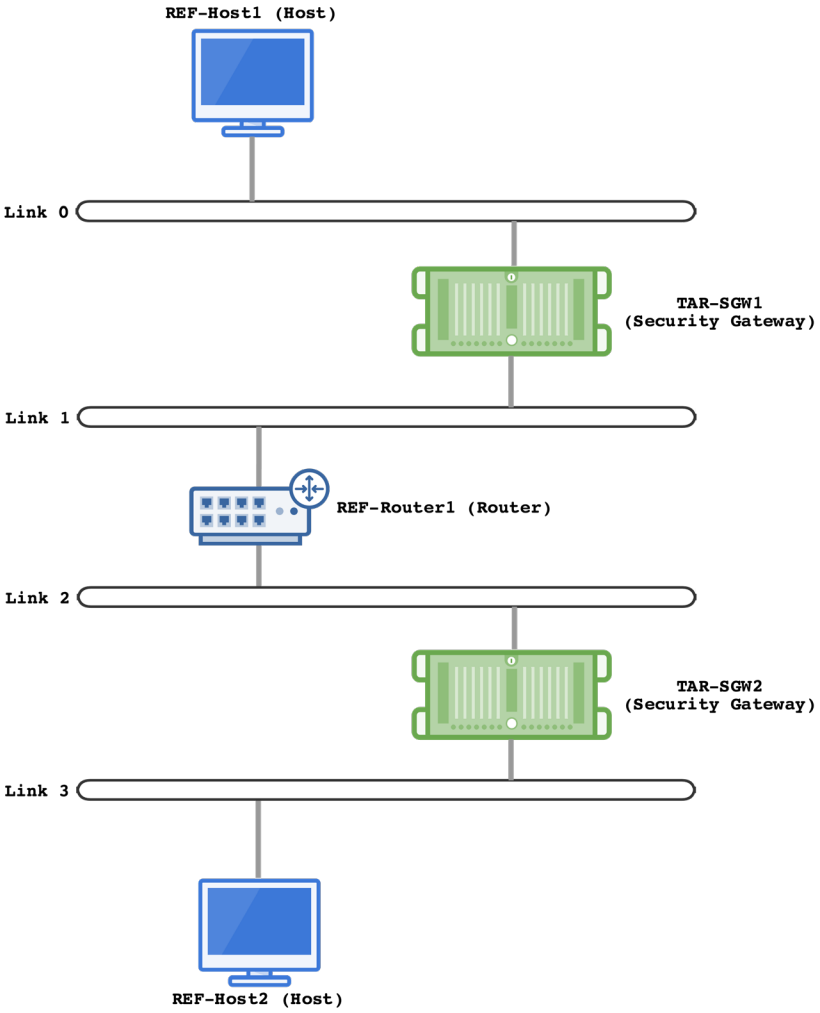

FIGURE 3 – SGW V. SGW

Topo.EN.SGW.2

1. Set global address to TAR-EN1_Link0 and REF-Host1_Link3 by RA.
2. Set global address to TAR-SGW1_Link1 and TAR-SGW1_Link2 manually.
3. Set routing table to TAR-SGW1 (REF-Router1_Link1 for Link0)
4. Set routing table to TAR-SGW1 (REF-Router2_Link2 for Link3)
5. Set routing table to REF-Router1 (TAR-SGW1_Link1 for Link2)
6. Set routing table to REF-Router1 (TAR-SGW1_Link1 for Link3)
7. Set routing table to REF-Router2 (TAR-SGW1_Link2 for Link0)
8. Set routing table to REF-Router2 (TAR-SGW1_Link2 for Link1)
9. Make IPsec tunnel mode between TAR-EN1 and TAR-SGW1.

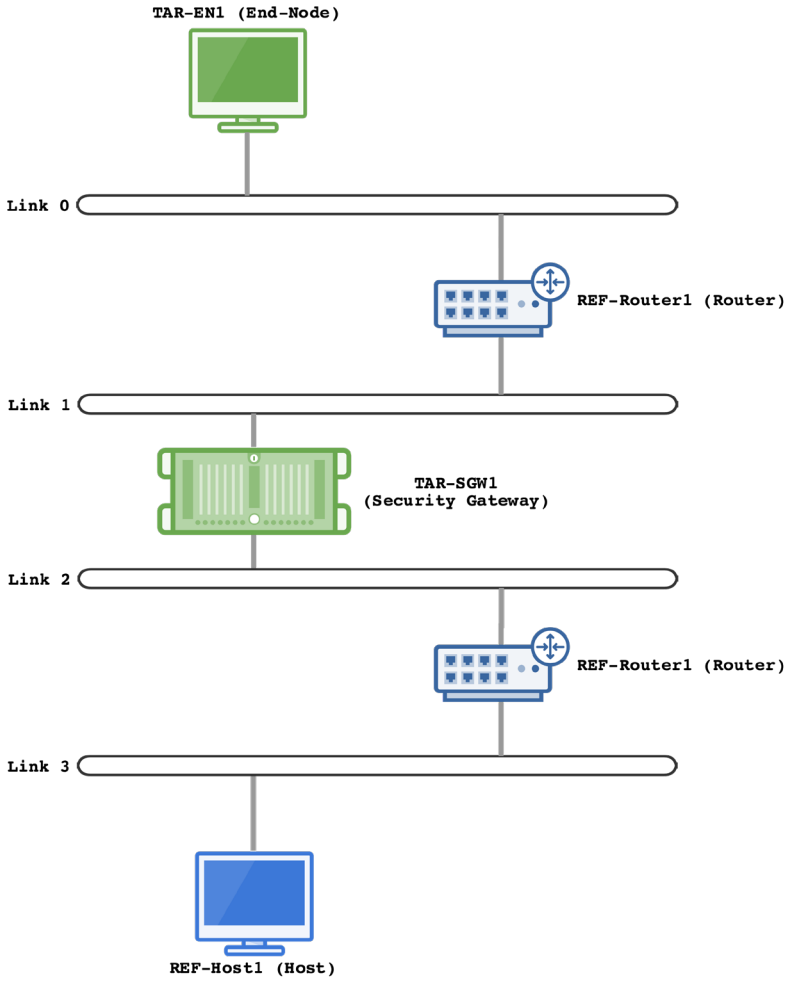

FIGURE 4 - END-NODE V. SGW

Description

Each test scenario consists of the following parts.

Purpose	The 'Purpose' is the short statement describing what the test attempts to achieve. It is usually phrased as a simple assertion of the future or capability to be tested.
Initialization	The 'Initialization' section describes how to initialize and configure the NUT before starting each test. If a value is not provided, then the protocol's default value is used.
Database	The 'Database' section describes the needed configuration for the Policy Database for the test case.
Packets	The 'Packets' section describes the simple format of the packets used in the test. In this document, the packet name is represented in Italic style font.
Procedure	The 'Procedure' describes the step-by-step instructions for carrying out the test.
Observable Results	The 'Observable Results' section describes the expected result. The NUT passes the test if the results described in this section are obtained.
Possible Problems	The 'Possible Problems' section contains a description of known issues with the test procedure, which may affect test results in certain situations.

Common Configurations

This section defines the Common Configurations referenced by various test cases.

Global Security Associations

Unless otherwise specified, the dynamically negotiated settings and algorithms below are used for every test case.

The IKEv2 settings apply for test cases that use 1 or more Security Association, however the Traffic Selectors may change, and are specified in the test case.

IKEv2 is the preferred mechanism for negotiating keys and configuring settings. If necessary, the Manual Settings may be used in the absence of IKEv2, or for debugging.

IKEv2 Settings	
IKE Encryption Algorithm	ENCR_AES_CBC (128-bit)
IKE Integrity Algorithm	AUTH_HMAC_SHA2_256_128
IKE PRF Algorithm	PRF_HMAC_SHA2_256
IKE DH Group	14 (2048-bit MODP Group)
Authentication Method	PSK: IPSECTEST12345678!
ID Type	ID_IPV6_ADDR

ESP	
ESP Encryption Algorithm	ENCR_AES_CBC (128-bit)
ESP Integrity Algorithm	AUTH_HMAC_SHA2_256_128

Cfg.EN.EN.1 (Transport)

Unless otherwise specified, utilize **Topo.EN.EN.1, and Global Security Associations**

Policy 1	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_EN2_LINK1
Mode	Transport
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	TAR_EN2_LINK1
Traffic Selector Protocol/Port	ANY/ANY

Cfg.EN.EN.2 (Tunnel)

Unless otherwise specified, utilize **Topo.EN.EN.1, and Global Security Associations**

Policy 1	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_EN2_LINK1
Mode	Tunnel
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	TAR_EN2_LINK1
Traffic Selector Protocol/Port	ANY/ANY

Cfg.EN.SGW.1

Unless otherwise specified, utilize **Topo.EN.SGW.1, and Global Security Associations**

Policy 1	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_SGW1_LINK1
Mode	Tunnel
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	LINK2
Traffic Selector Protocol/Port	ANY/ANY

Cfg.EN.SGW.2

Unless otherwise specified, utilize **Topo.EN.SGW.2, and Global Security Associations**

Policy 1	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_SGW1_LINK1
Mode	Tunnel
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	LINK2 + LINK3
Traffic Selector Protocol/Port	ANY/ANY

Cfg.SGW.SGW.1

Unless otherwise specified, utilize **Topo.SGW.SGW.1, and Global Security Associations**

Policy 1	
Peer Left	TAR_SGW1_LINK1
Peer Right	TAR_SGW2_LINK2
Mode	Tunnel
Traffic Selector Address Left	LINK0
Traffic Selector Address Right	LINK3
Traffic Selector Protocol/Port	ANY/ANY

Section 1: IPsec/ESP Architecture

Scope

The following tests focus on IPsec and ESP Architecture (RFC 4301/4303).

Overview

Tests in this section exercise interoperability based on the requirements of RFC 4301 and RFC 4303. IKEv2 (RFC7296) is utilized to negotiate connections.

IPsec.IO.1.1: Basic Connection

Purpose:

Verify Devices can establish and use an IPsec Security Association using a basic setup and common topology.

Part A: End-Node v. End-Node Transport Mode

Step	Action	Expected Result
1.	Connect devices according to Configuration **Cfg.EN.EN.1**.	
2.	Transmit an ICMPv6 Echo Request from TAR-EN1 to TAR-EN2, or otherwise cause TAR-EN1 to initiate the IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR-EN1 and TAR-EN2, including IKE_SA_INIT and IKE_AUTH.
3.	If necessary, transmit an ICMPv6 Echo Request from TAR-EN1 to TAR-EN2.	TAR-EN1 transmits a valid ESP ICMPv6 Echo Request as negotiated, and TAR-EN2 transmits a valid ESP ICMPv6 Echo Reply as negotiated.
4.	Reset TAR-EN1 and TAR-EN2	All IKE SAs and IPsec SAs are removed from both devices.
5.	Transmit an ICMPv6 Echo Request from TAR-EN2 to TAR-EN1, or otherwise cause TAR-EN2 to initiate the IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR-EN1 and TAR-EN2, including IKE_SA_INIT and IKE_AUTH.
6.	If necessary, transmit an ICMPv6 Echo Request from TAR-EN2 to TAR-EN1.	TAR-EN2 transmits a valid ESP ICMPv6 Echo Request as negotiated, and TAR-EN1 transmits a valid ESP ICMPv6 Echo Reply as negotiated.

Part B: End-Node v. End-Node Tunnel Mode

Step	Action	Expected Result

7.	Connect devices according to Configuration **Cfg.EN.EN.2**.	
8.	Transmit an ICMPv6 Echo Request from TAR-EN1 to TAR-EN2, or otherwise cause TAR-EN1 to initiate the IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR-EN1 and TAR-EN2, including IKE_SA_INIT and IKE_AUTH.
9.	If necessary, transmit an ICMPv6 Echo Request from TAR-EN1 to TAR-EN2.	TAR-EN1 transmits a valid ESP ICMPv6 Echo Request as negotiated, and TAR-EN2 transmits a valid ESP ICMPv6 Echo Reply as negotiated.
10.	Reset TAR-EN1 and TAR-EN2	All IKE SAs and IPsec SAs are removed from both devices.
11.	Transmit an ICMPv6 Echo Request from TAR-EN2 to TAR-EN1, or otherwise cause TAR-EN2 to initiate the IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR-EN1 and TAR-EN2, including IKE_SA_INIT and IKE_AUTH.
12.	If necessary, transmit an ICMPv6 Echo Request from TAR-EN2 to TAR-EN1.	TAR-EN2 transmits a valid ESP ICMPv6 Echo Request as negotiated, and TAR-EN1 transmits a valid ESP ICMPv6 Echo Reply as negotiated.

Part C: End-Node v. SGW

Step	Action	Expected Result
13.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
14.	Transmit an ICMPv6 Echo Request from TAR-EN1 to REF-Host1, or otherwise cause TAR-EN1 to initiate the IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR-EN1 and TAR-SGW1, including IKE_SA_INIT and IKE_AUTH.
15.	If necessary, transmit an ICMPv6 Echo Request from TAR-EN1 to REF-Host1.	TAR-EN1 transmits a valid ESP ICMPv6 Echo

		Request as negotiated. TAR-SGW1 forwards the plain-text Echo Request to REF-Host1. REF-Host1 transmits an ICMPv6 Echo Reply. TAR-SGW1 forwards a valid ESP message containing the Echo Reply to TAR-EN1.
16.	Reset TAR-EN1 and TAR-EN2	All IKE SAs and IPsec SAs are removed from both devices.
17.	Transmit an ICMPv6 Echo Request from REF-Host1 to TAR-EN1, or otherwise cause TAR-SGW1 to initiate the IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR-SGW1 and TAR-EN1, including IKE_SA_INIT and IKE_AUTH.
18.	If necessary, transmit an ICMPv6 Echo Request from REF-Host1 to TAR-EN1.	REF-Host1 transmits an ICMPv6 Echo Request. TAR-SGW1 forwards a valid ESP message containing the Echo Request to TAR-EN1. TAR-EN1 transmits a valid ESP ICMPv6 Echo Reply as negotiated. TAR-SGW1 forwards the plain-text Echo Reply to REF-Host1.

Part D: SGW v. SGW

Step	Action	Expected Result
19.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
20.	Transmit an ICMPv6 Echo Request from REF-Host1 to REF-Host2, or otherwise cause TAR-SGW1 to initiate the IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR-SGW1 and TAR-SGW2, including IKE_SA_INIT and IKE_AUTH.

21.	If necessary, transmit an ICMPv6 Echo Request from REF-Host1 to REF-Host2.	REF-Host1 transmits an ICMPv6 Echo Reply. TAR-SGW1 transmits a valid ESP ICMPv6 Echo Request as negotiated. TAR-SGW2 forwards the plain-text Echo Request to REF-Host2. REF-Host2 transmits an ICMPv6 Echo Reply. TAR-SGW2 forwards a valid ESP message containing the Echo Reply to TAR-SGW1. TAR-SGW1 forwards the plain-text Echo Reply to REF-Host1.
22.	Reset TAR-SGW1 and TAR-SGW2	All IKE SAs and IPsec SAs are removed from both devices.
23.	Transmit an ICMPv6 Echo Request from REF-Host2 to REF-Host1, or otherwise cause TAR-SGW2 to initiate the IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR-SGW1 and TAR-SGW2, including IKE_SA_INIT and IKE_AUTH.
24.	If necessary, transmit an ICMPv6 Echo Request from REF-Host2 to REF-Host1.	REF-Host2 transmits an ICMPv6 Echo Reply. TAR-SGW2 transmits a valid ESP ICMPv6 Echo Request as negotiated. TAR-SGW1 forwards the plain-text Echo Request to REF-Host1. REF-Host1 transmits an ICMPv6 Echo Reply. TAR-SGW1 forwards a valid ESP message containing the Echo Reply to TAR-SGW2. TAR-SGW2 forwards the plain-text Echo Reply to REF-Host2.

Possible Problems:

- None

IPsec.IO.1.2: Traffic Selectors

Purpose:

Verify Devices can establish and use an IPsec Security Associations when Traffic Selectors are specified.

Common Procedure:

Step	Action	Expected Result
1.	Transmit data from TAR 2 to TAR 1, or otherwise cause TAR 2 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
2.	If necessary, re-transmit data from TAR 2 to TAR 1.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.

Part A: TCP - End-Node v. End-Node Transport Mode

Policy P.EN.EN.TCP	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_EN2_LINK1
Mode	Transport
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	TAR_EN2_LINK1
Traffic Selector Protocol/Port	TCP/ANY

Step	Action	Expected Result
1.	Connect devices according to Configuration **Cfg.EN.EN.1**. Substitute Policy **P.EN.EN.TCP** above.	
2.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
3.	Following the Common Procedure above, use TCP to generate the test data.	

Part B: TCP - End-Node v. End-Node Tunnel Mode

Policy P.EN.EN.TCPTUN	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_EN2_LINK1
Mode	Tunnel
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	TAR_EN2_LINK1
Traffic Selector Protocol/Port	TCP/ANY

Step	Action	Expected Result
4.	Connect devices according to Configuration **Cfg.EN.EN.1**. Substitute Policy **P.EN.EN.TCPTUN** above.	
5.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
6.	Following the Common Procedure above, use TCP to generate the test data.	

Part C: TCP - End-Node v. SGW

Policy P.EN.SGW.TCP	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_SGW1_LINK1
Mode	Tunnel
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	LINK2
Traffic Selector Protocol/Port	TCP/ANY

Step	Action	Expected Result
7.	Connect devices according to Configuration **Cfg.EN.SGW.1**. Substitute Policy **P.EN.SGW.TCP** above.	
8.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
9.	Following the Common Procedure above, use TCP to generate the test	

	data using TAR-EN1 and REF-Host1 as data endpoints.	

Part D: TCP - SGW v. SGW

Policy P.SGW.SGW.TCP	
Peer Left	TAR_SGW1_LINK1
Peer Right	TAR_SGW2_LINK2
Mode	Tunnel
Traffic Selector Address Left	LINK0
Traffic Selector Address Right	LINK3
Traffic Selector Protocol/Port	TCP/ANY

Step	**Action**	**Expected Result**
10.	Connect devices according to Configuration **Cfg.SGW.SGW.1**. Substitute Policy **P.SGW.SGW.TCP** above.	
11.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
12.	Following the Common Procedure above, use TCP to generate the test data using REF-Host1 and REF-Host2 as data endpoints.	

Part E: ICMPv6 - End-Node v. End-Node Transport Mode

Policy P.EN.EN.ICMP	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_EN2_LINK1
Mode	Transport
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	TAR_EN2_LINK1
Traffic Selector Protocol/Port	ICMPv6/Echo Request and Echo Reply

Step	**Action**	**Expected Result**
13.	Connect devices according to Configuration **Cfg.EN.EN.1**. Substitute Policy **P.EN.EN.ICMP** above.	

14.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
15.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data.	

Part F: ICMPv6 - End-Node v. End-Node Tunnel Mode

Policy P.EN.EN.ICMPTUN	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_EN2_LINK1
Mode	Tunnel
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	TAR_EN2_LINK1
Traffic Selector Protocol/Port	ICMPv6/Echo Request and Echo Reply

Step	**Action**	**Expected Result**
16.	Connect devices according to Configuration **Cfg.EN.EN.1**. Substitute Policy **P.EN.EN.ICMPTUN** above.	
17.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
18.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data.	

Part G: ICMPv6 - End-Node v. SGW

Policy P.EN.SGW.ICMP	
Peer Left	TAR_EN1_LINK0
Peer Right	TAR_SGW1_LINK1
Mode	Tunnel
Traffic Selector Address Left	TAR_EN1_LINK0
Traffic Selector Address Right	LINK2
Traffic Selector Protocol/Port	ICMPv6/Echo Request and Echo Reply

Step	**Action**	**Expected Result**

19.	Connect devices according to Configuration **Cfg.EN.SGW.1**. Substitute Policy **P.EN.SGW.ICMP** above.	
20.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
21.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data using TAR-EN1 and REF-Host1 as data endpoints.	

Part H: ICMPv6 - SGW v. SGW

Policy P.SGW.SGW.ICMP	
Peer Left	TAR_SGW1_LINK1
Peer Right	TAR_SGW2_LINK2
Mode	Tunnel
Traffic Selector Address Left	LINK0
Traffic Selector Address Right	LINK3
Traffic Selector Protocol/Port	ICMPv6/Echo Request and Echo Reply

Step	**Action**	**Expected Result**
22.	Connect devices according to Configuration **Cfg.SGW.SGW.1**. Substitute Policy **P.SGW.SGW.ICMP** above.	
23.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
24.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data using REF-Host1 and REF-Host2 as data endpoints.	

Possible Problems:

- None

IPsec.IO.1.3: Fragmentation

Purpose:

Verify Devices can establish and use an IPsec Security Associations on links that require Path MTU Discovery and Fragmentation.

Common Procedure:

Step	Action	Expected Result
1.	Transmit data from TAR 2 to TAR 1, or otherwise cause TAR 2 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
2.	If necessary, re-transmit data from TAR 2 to TAR 1.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.
3.	Transmit data that exceeds the configured IPv6 Path MTU from TAR 2 to TAR 1.	An ICMPv6 Packet Too Big Message is emitted.
4.	Transmit data that exceeds the configured IPv6 Path MTU from TAR 2 to TAR 1.	The data is correctly fragmented and processed by all devices.
5.	Step 4 may be repeated as needed to completely discover the Path MTU.	

Part A: End-Node v. End-Node Transport Mode – TAR-EN1 Packet Too Big Processing

Step	Action	Expected Result
1.	Connect devices according to Configuration **Cfg.EN.EN.1**.	
2.	Configure the MTU of Link 0 to 1500 bytes and Link 1 to 1280 bytes.	
3.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
4.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data.	TAR-EN1 processes the ICMPv6 Packet Too Big message, and correctly

		fragments future packets.

Part B: End-Node v. End-Node Transport Mode – TAR-EN2 Packet Too Big Processing

Step	Action	Expected Result
5.	Connect devices according to Configuration **Cfg.EN.EN.1**.	
6.	Configure the MTU of Link 0 to 1280 bytes and Link 1 to 1500 bytes.	
7.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
8.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data.	TAR-EN2 processes the ICMPv6 Packet Too Big message, and correctly fragments future packets.

Part C: End-Node v. End-Node Tunnel Mode – TAR-EN1 Packet Too Big Processing

Step	Action	Expected Result
9.	Connect devices according to Configuration **Cfg.EN.EN.1**.	
10.	Configure the MTU of Link 0 to 1500 bytes and Link 1 to 1280 bytes.	
11.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
12.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data.	TAR-EN1 processes the ICMPv6 Packet Too Big message, and correctly fragments future packets.

Part D: End-Node v. End-Node Tunnel Mode – TAR-EN2 Packet Too Big Processing

Step	Action	Expected Result
13.	Connect devices according to Configuration **Cfg.EN.EN.1**.	

14.	Configure the MTU of Link 0 to 1280 bytes and Link 1 to 1500 bytes.	
15.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
16.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data.	TAR-EN2 processes the ICMPv6 Packet Too Big message, and correctly fragments future packets.

Part E: End-Node v. SGW – Topology 1 – TAR-EN1 Packet Too Big Processing

Step	Action	Expected Result
17.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
18.	Configure the MTU of Link 0 to 1500 bytes and Link 1 to 1280 bytes.	
19.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
20.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data, using TAR-EN1 and REF-Host1 as data endpoints.	TAR-EN1 processes the ICMPv6 Packet Too Big message, and correctly fragments future packets.

Part F: End-Node v. SGW – Topology 1 – TAR-SGW1 Packet Too Big Processing

Step	Action	Expected Result
21.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
22.	Configure the MTU of Link 0 to 1280 bytes and Link 1 to 1500 bytes.	
23.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
24.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data, using TAR-EN1 and REF-Host1 as data endpoints.	TAR-SGW1 processes the ICMPv6 Packet Too Big message, and

		correctly fragments future packets.

Part G: End-Node v. SGW – Topology 2 – Encrypted Packet Too Big Processing

Step	Action	Expected Result
25.	Connect devices according to Configuration **Cfg.EN.SGW.2**.	
26.	Configure the MTU of Link 2 to 1280 bytes. All other links are 1500 byte MTU.	
27.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
28.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data, using TAR-EN1 and REF-Host1 as data endpoints.	TAR-SGW1 encrypts and forwards the ICMPv6 Packet Too Big message as it would any other matching data. TAR-EN1 processes the Encrypted ICMPv6 Packet Too Big message, and correctly fragments future packets.

Part F: SGW v. SGW – TAR-SGW1 Packet Too Big Processing

Step	Action	Expected Result
29.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
30.	Configure the MTU of Link 0 to 1500 bytes and Link 1 to 1280 bytes.	
31.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
32.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data.	TAR-SGW1 processes the ICMPv6 Packet Too Big message, and correctly fragments future packets.

Part G: SGW v. SGW – TAR-SGW2 Packet Too Big Processing

Step	Action	Expected Result
1.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
2.	Configure the advertised MTU of Link 0 to 1280 bytes and Link 1 to 1500 bytes.	
3.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
4.	Following the Common Procedure above, use ICMPv6 Echo Request to generate the test data.	TAR-SGW2 processes the ICMPv6 Packet Too Big message, and correctly fragments future packets.

Possible Problems:

- None

Section 2: IKEv2 Architecture

Scope

The following tests focus on IKEv2 Architecture (RFC 7296).

Overview

Tests in this section exercise interoperability based on the requirements of RFC 7296.

IPsec.IO.2.1: Authentication

Purpose:

Verify Devices can successfully authenticate using Pre-shared Keys and RSA Digital signatures, as well as detect a failure in authentication.

Common Procedure:

Step	Action	Expected Result
1.	Transmit data from TAR 1 to TAR 2, or otherwise cause TAR 1 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
2.	If necessary, re-transmit data from TAR 1 to TAR 2.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.
3.	Reset TAR-EN1 and TAR-EN2	All IKE SAs and IPsec SAs are removed from both devices.
4.	Transmit data from TAR 2 to TAR 1, or otherwise cause TAR 2 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
5.	If necessary, re-transmit data from TAR 2 to TAR 1.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.

Part A: PSK - End-Node v. End-Node Tunnel Mode

Step	Action	Expected Result
1.	Connect devices according to Configuration **Cfg.EN.EN.2**.	

2.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
3.	Observe Results	The negotiation completes successfully, and all data is encrypted and exchanged without error.

Part B: PSK - End-Node v. SGW

Step	Action	Expected Result
4.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
5.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
6.	Observe Results	The negotiation completes successfully, and all data is encrypted and exchanged without error.

Part C: PSK - SGW v. SGW

Step	Action	Expected Result
7.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
8.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
9.	Observe Results	The negotiation completes successfully, and all data is encrypted and exchanged without error.

Part D: PSK Mismatch - End-Node v. End-Node Tunnel Mode

Step	Action	Expected Result
10.	Connect devices according to Configuration **Cfg.EN.EN.2**.	
11.	Configure TAR 1 with a PSK that is different from TAR 2.	
12.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
13.	Observe Results	The negotiation **DOES NOT** complete successfully. The logs or console should indicate an authentication error (Notify Type 24). No encrypted IPsec/ESP data is transferred.

Part E: PSK Mismatch - End-Node v. SGW

Step	Action	Expected Result
14.	Connect devices according to Configuration **Cfg.EN.SGW.1.**	
15.	Configure TAR 1 with a PSK that is different from TAR 2.	
16.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
17.	Observe Results	The negotiation **DOES NOT** complete successfully. The logs or console should indicate an authentication error (Notify Type 24). No encrypted IPsec/ESP data is transferred.

Part F: PSK Mismatch - SGW v. SGW

Step	Action	Expected Result

18.	Connect devices according to Configuration **Cfg.SGW.SGW.1.**	
19.	Configure TAR 1 with a PSK that is different from TAR 2.	
20.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
21.	Observe Results	The negotiation **DOES NOT** complete successfully. The logs or console should indicate an authentication error (Notify Type 24). No encrypted IPsec/ESP data is transferred.

Part G: RSA Digital Signature - End-Node v. End-Node Tunnel Mode

Step	Action	Expected Result
22.	Connect devices according to Configuration **Cfg.EN.EN.2**.	
23.	Configure the devices to use the RSA Digital Signature (1) Authentication Method for both DUT and TN1, in place of Shared Key (3). Use a common, locally created Test CA to attest and sign the device certificates. Use ID_IPV6_ADDR as the IKEv2 Identification Type, if available. Otherwise, choose another available ID Type. The types may be different.	
24.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
25.	Observe Results	The negotiation completes successfully, and all data is encrypted and exchanged without error.

Part H: RSA Digital Signature - End-Node v. SGW

Step	Action	Expected Result
26.	Connect devices according to Configuration **Cfg.EN.SGW.1.**	
27.	Configure the devices to use the RSA Digital Signature (1) Authentication Method for both DUT and TN1, in place of Shared Key (3). Use a common, locally created Test CA to attest and sign the device certificates. Use ID_IPV6_ADDR as the IKEv2 Identification Type, if available. Otherwise, choose another available ID Type. The types may be different.	
28.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
29.	Observe Results	The negotiation completes successfully, and all data is encrypted and exchanged without error.

Part I: RSA Digital Signature - SGW v. SGW

Step	Action	Expected Result
30.	Connect devices according to Configuration **Cfg.SGW.SGW.1.**	
31.	Configure the devices to use the RSA Digital Signature (1) Authentication Method for both DUT and TN1, in place of Shared Key (3). Use a common, locally created Test CA to attest and sign the device certificates.	

	Use ID_IPV6_ADDR as the IKEv2 Identification Type, if available. Otherwise, choose another available ID Type. The types may be different.	
32.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
33.	Observe Results	The negotiation completes successfully, and all data is encrypted and exchanged without error.

Possible Problems:

- None

IPsec.IO.2.2 SA: Algorithm Mismatch (NO_PROPOSAL_CHOSEN)

Purpose:

Verify Devices can detect and react to mismatching SA Proposals.

Common Procedure:

Step	Action	Expected Result
1.	Transmit data from TAR 1 to TAR 2, or otherwise cause TAR 1 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
2.	If necessary, re-transmit data from TAR 1 to TAR 2.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.
3.	Reset TAR-EN1 and TAR-EN2	All IKE SAs and IPsec SAs are removed from both devices.
4.	Transmit data from TAR 2 to TAR 1, or otherwise cause TAR 2 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
5.	If necessary, re-transmit data from TAR 2 to TAR 1.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.

Part A: IKE_SA_INIT - End-Node v. End-Node Tunnel Mode

Step	Action	Expected Result
1.	Connect devices according to Configuration **Cfg.EN.EN.2**.	
2.	Configure TAR 1 with an IKE_SA Suite that is different from TAR 2.	

3.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
4.	Observe Results	The IKE_SA_INIT negotiation **DOES NOT** complete successfully. The logs or console should indicate a no proposal chosen error (Notify Type 14). No encrypted IPsec/ESP data is transferred.

Part B: IKE_SA_INIT - End-Node v. SGW

Step	Action	Expected Result
5.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
6.	Configure TAR 1 with an IKE_SA Suite that is different from TAR 2.	
7.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
8.	Observe Results	The IKE_SA_INIT negotiation **DOES NOT** complete successfully. The logs or console should indicate a no proposal chosen error (Notify Type 14). No encrypted IPsec/ESP data is transferred.

Part C: IKE_SA_INIT - SGW v. SGW

Step	Action	Expected Result
9.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
10.	Configure TAR 1 with an IKE_SA Suite that is different from TAR 2.	

11.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
12.	Observe Results	The IKE_SA_INIT negotiation **DOES NOT** complete successfully. The logs or console should indicate a no proposal chosen error (Notify Type 14). No encrypted IPsec/ESP data is transferred.

Part D: IKE_AUTH - End-Node v. End-Node Tunnel Mode

Step	Action	Expected Result
13.	Connect devices according to Configuration **Cfg.EN.EN.2**.	
14.	Configure TAR 1 with an IPSEC_SA Suite that is different from TAR 2.	
15.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
16.	Observe Results	The IKE_AUTH negotiation **DOES NOT** complete successfully. The logs or console should indicate a no proposal chosen error (Notify Type 14). No encrypted IPsec/ESP data is transferred.

Part E: IKE_AUTH - End-Node v. SGW

Step	Action	Expected Result
17.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
18.	Configure TAR 1 with an IPSEC_SA Suite that is different from TAR 2.	

19.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
20.	Observe Results	The IKE_AUTH negotiation **DOES NOT** complete successfully. The logs or console should indicate a no proposal chosen error (Notify Type 14). No encrypted IPsec/ESP data is transferred.

Part F: IKE_AUTH - SGW v. SGW

Step	Action	Expected Result
21.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
22.	Configure TAR 1 with an IPSEC_SA Suite that is different from TAR 2.	
23.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
24.	Observe Results	The IKE_AUTH negotiation **DOES NOT** complete successfully. The logs or console should indicate a no proposal chosen error (Notify Type 14). No encrypted IPsec/ESP data is transferred.

Possible Problems:

- None

IPsec.IO.2.3: DH Retry (INVALID_KE_PAYLOAD)

Purpose:

Verify Devices can reattempt IKE_SA keying material generation when DH groups public keys are mismatched.

Common Procedure:

Step	Action	Expected Result
1.	Transmit data from TAR 1 to TAR 2, or otherwise cause TAR 1 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
2.	If necessary, re-transmit data from TAR 1 to TAR 2.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.

Part A: End-Node v. End-Node Tunnel Mode – TAR-EN1 Retransmits Key

Step	Action	Expected Result
1.	Connect devices according to Configuration **Cfg.EN.EN.2**.	
2.	Configure TAR-EN1 to propose both DH14 and DH19, preferring DH19.	
3.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
4.	Observe Results	During IKE_SA_INIT, TAR 1 proposes a DH key using group 19. TAR 2 transmits an invalid KE payload error (NOTIFY Type 17). TAR 1 retransmits the IKE_SA_INIT Request using DH 14. The remainder of the negotiation completes

		successfully, and IPsec ESP data is encrypted and exchanged without error.

Part B: End-Node v. End-Node Tunnel Mode – TAR-EN2 Retransmits Key

Step	Action	Expected Result
5.	Connect devices according to Configuration **Cfg.EN.EN.2**.	
6.	Configure TAR-EN2 to propose both DH14 and DH19, preferring DH19.	
7.	Following the Common Procedure above, substitute TAR-EN2 as TAR 1 and TAR-EN1 as TAR2.	
8.	Observe Results	During IKE_SA_INIT, TAR 1 proposes a DH key using group 19. TAR 2 transmits an invalid KE payload error (NOTIFY Type 17). TAR 1 retransmits the IKE_SA_INIT Request using DH 14. The remainder of the negotiation completes successfully, and IPsec ESP data is encrypted and exchanged without error.

Part C: End-Node v. SGW – TAR-EN1 Retransmits Key

Step	Action	Expected Result
9.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
10.	Configure TAR-EN1 to propose both DH14 and DH19, preferring DH19.	

11.	Following the Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR 2.	
12.	Observe Results	During IKE_SA_INIT, TAR 1 proposes a DH key using group 19. TAR 2 transmits an invalid KE payload error (NOTIFY Type 17). TAR 1 retransmits the IKE_SA_INIT Request using DH 14. The remainder of the negotiation completes successfully, and IPsec ESP data is encrypted and exchanged without error.

Part D: End-Node v. SGW – TAR-SGW1 Retransmits Key

Step	Action	Expected Result
13.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
14.	Configure TAR-SGW1 to propose both DH14 and DH19, preferring DH19.	
15.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-EN1 as TAR 2.	
16.	Observe Results	During IKE_SA_INIT, TAR 1 proposes a DH key using group 19. TAR 2 transmits an invalid KE payload error (NOTIFY Type 17). TAR 1 retransmits the IKE_SA_INIT Request using DH 14. The remainder of the negotiation completes successfully, and IPsec ESP data is encrypted

		and exchanged without error.

Part E: SGW v. SGW – TAR-SGW1 Retransmits Key

Step	Action	Expected Result
17.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
18.	Configure TAR-SGW1 to propose both DH14 and DH19, preferring DH19.	
19.	Following the Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR 2.	
20.	Observe Results	During IKE_SA_INIT, TAR 1 proposes a DH key using group 19. TAR 2 transmits an invalid KE payload error (NOTIFY Type 17). TAR 1 retransmits the IKE_SA_INIT Request using DH 14. The remainder of the negotiation completes successfully, and IPsec ESP data is encrypted and exchanged without error.

Part F: SGW v. SGW – TAR-SGW2 Retransmits Key

Step	Action	Expected Result
21.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
22.	Configure TAR-SGW2 to propose both DH14 and DH19, preferring DH19.	
23.	Following the Common Procedure above, substitute TAR-SGW2 as TAR 1 and TAR-SGW1 as TAR 2.	

24.	Observe Results	During IKE_SA_INIT, TAR 1 proposes a DH key using group 19. TAR 2 transmits an invalid KE payload error (NOTIFY Type 17). TAR 1 retransmits the IKE_SA_INIT Request using DH 14. The remainder of the negotiation completes successfully, and IPsec ESP data is encrypted and exchanged without error.

Possible Problems:

- The NUT or interoperable device may not support more than 1 DH group. In this case, an interoperable device that supports more than one group should be selected, and the test cases requiring the NUT to support more than 1 DH group may be omitted.

IPsec.IO.2.4: Rekeying

Purpose:

Verify Devices can reattempt IKE_SA keying material generation when DH groups public keys are mismatched.

IKE_SA Rekeying Common Procedure:

Step	Action	Expected Result
1.	Transmit data from TAR 1 to TAR 2, or otherwise cause TAR 1 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
2.	If necessary, re-transmit data from TAR 1 to TAR 2.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.
3.	Continue to transmit data regularly, until the lifetime of the IKE_SA expires for TAR 1.	TAR 1 should initiate a CREATE_CHILD_SA exchange.
4.	Continue to transmit data regularly, until the lifetime of the IKE_SA expires for TAR 1.	The IKE iSPI and rSPI values in the CREATE_CHILD_SA are different from the values used during the INITIAL Exchange, and first CREATE_CHILD_SA Exchange.

IPSEC_SA Rekeying Common Procedure:

Step	Action	Expected Result
1.	Transmit data from TAR 1 to TAR 2, or otherwise cause TAR 1 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
2.	If necessary, re-transmit data from TAR 1 to TAR 2.	TAR 2 transmits a valid ESP packet as

		negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.
3.	Continue to transmit data regularly, until the lifetime of the IPSEC_SA expires for TAR 1.	TAR 1 should initiate a CREATE_CHILD_SA exchange.
4.	Continue to transmit data regularly, until.	The ESP SPIs used in traffic between TAR 1 and TAR 2 are different from the values used prior to the CREATE_CHILD_SA Exchange.

Part A: IKE_SA - End-Node v. End-Node Tunnel Mode – TAR-EN1 Rekeys

Step	Action	Expected Result
1.	Connect devices according to Configuration **Cfg.EN.EN.2**.	
2.	Configure TAR-EN1 to enable Rekeying of the IKE_SA and configure the lifetime of the IKE_SA to expire prior to TAR-EN2.	
3.	Following the IKE_SA Rekeying Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
4.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part B: IKE_SA - End-Node v. End-Node Tunnel Mode – TAR-EN2 Rekeys

Step	Action	Expected Result
5.	Connect devices according to Configuration **Cfg.EN.EN.2**.	

6.	Configure TAR-EN2 to enable Rekeying of the IKE_SA and configure the lifetime of the IKE_SA to expire prior to TAR-EN1.	
7.	Following the IKE_SA Rekeying Common Procedure above, substitute TAR-EN2 as TAR 1 and TAR-EN1 as TAR2.	
8.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part C: IKE_SA - End-Node v. SGW – TAR-EN1 Rekeys

Step	Action	Expected Result
9.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
10.	Configure TAR-EN1 to enable Rekeying of the IKE_SA and configure the lifetime of the IKE_SA to expire prior to TAR-SGW1.	
11.	Following the IKE_SA Rekeying Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
12.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part D: IKE_SA - End-Node v. SGW – TAR-SGW1 Rekeys

Step	Action	Expected Result
13.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	

14.	Configure TAR-SGW1 to enable Rekeying of the IKE_SA and configure the lifetime of the IKE_SA to expire prior to TAR-EN1.	
15.	Following the IKE_SA Rekeying Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-EN1 as TAR2.	
16.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part E: IKE_SA - SGW v. SGW – TAR-SGW1 Rekeys

Step	Action	Expected Result
17.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
18.	Configure TAR-SGW1 to enable Rekeying of the IKE_SA and configure the lifetime of the IKE_SA to expire prior to TAR-SGW2.	
19.	Following the IKE_SA Rekeying Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
20.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part F: IKE_SA - SGW v. SGW – TAR-SGW2 Rekeys

Step	Action	Expected Result
21.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	

22.	Configure TAR-SGW2 to enable Rekeying of the IKE_SA and configure the lifetime of the IKE_SA to expire prior to TAR-SGW1.	
23.	Following the IKE_SA Rekeying Common Procedure above, substitute TAR-SGW2 as TAR 1 and TAR-SGW1 as TAR2.	
24.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part G: IPSEC_SA - End-Node v. End-Node Tunnel Mode – TAR-EN1 Rekeys

Step	Action	Expected Result
1.	Connect devices according to Configuration **Cfg.EN.EN.2**.	
2.	Configure TAR-EN1 to enable Rekeying of the IPSEC_SA and configure the lifetime of the IPSEC_SA to expire prior to TAR-EN2.	
3.	Following the IPSEC_SA Rekeying Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-EN2 as TAR2.	
4.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part H: IPSEC_SA - End-Node v. End-Node Tunnel Mode – TAR-EN2 Rekeys

Step	Action	Expected Result
5.	Connect devices according to Configuration **Cfg.EN.EN.2**.	

6.	Configure TAR-EN2 to enable Rekeying of the IPSEC_SA and configure the lifetime of the IPSEC_SA to expire prior to TAR-EN1.	
7.	Following the IPSEC_SA Rekeying Common Procedure above, substitute TAR-EN2 as TAR 1 and TAR-EN1 as TAR2.	
8.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part I: IPSEC_SA - End-Node v. SGW – TAR-EN1 Rekeys

Step	Action	Expected Result
9.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	
10.	Configure TAR-EN1 to enable Rekeying of the IPSEC_SA and configure the lifetime of the IPSEC_SA to expire prior to TAR-SGW1.	
11.	Following the IPSEC_SA Rekeying Common Procedure above, substitute TAR-EN1 as TAR 1 and TAR-SGW1 as TAR2.	
12.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part J: IPSEC_SA - End-Node v. SGW – TAR-SGW1 Rekeys

Step	Action	Expected Result
13.	Connect devices according to Configuration **Cfg.EN.SGW.1**.	

14.	Configure TAR-SGW1 to enable Rekeying of the IPSEC_SA and configure the lifetime of the IPSEC_SA to expire prior to TAR-EN1.	
15.	Following the IPSEC_SA Rekeying Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-EN1 as TAR2.	
16.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part K: IPSEC_SA - SGW v. SGW – TAR-SGW1 Rekeys

Step	Action	Expected Result
17.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	
18.	Configure TAR-SGW1 to enable Rekeying of the IPSEC_SA and configure the lifetime of the IPSEC_SA to expire prior to TAR-SGW2.	
19.	Following the IPSEC_SA Rekeying Common Procedure above, substitute TAR-SGW1 as TAR 1 and TAR-SGW2 as TAR2.	
20.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Part L: IKE_SA - SGW v. SGW – TAR-SGW2 Rekeys

Step	Action	Expected Result
21.	Connect devices according to Configuration **Cfg.SGW.SGW.1**.	

22.	Configure TAR-SGW2 to enable Rekeying of the IPSEC_SA and configure the lifetime of the IPSEC_SA to expire prior to TAR-SGW1.	
23.	Following the IPSEC_SA Rekeying Common Procedure above, substitute TAR-SGW2 as TAR 1 and TAR-SGW1 as TAR2.	
24.	Observe Results	The negotiation completes successfully, keys are renegotiate, and all data is encrypted and exchanged without error.

Possible Problems:

- None

Section 3: Algorithms

Scope:

The following test cases verify a device correctly utilizes ESP for different algorithms.

Overview:

Tests in this section verify that a node properly process and transmit based on the Algorithms and Security Policy Database and Security Association Database.

Common Configurations

IKE_SA_INIT Algorithm List

The test case parts itemized below are used in this section, and referred to by each test case.

Proposal	ENCR – Type 1	PRF – Type 2	INTEG – Type 3	D-H – Type 4
IKESA.1	ENCR_AES_CBC (128-bit)	PRF_HMAC_SHA2_256	AUTH_HMAC_SHA2_256_128	14
IKESA.2	ENCR_AES_CBC (256-bit)	PRF_HMAC_SHA2_256	AUTH_HMAC_SHA2_256_128	14
IKESA.3	ENCR_AES_CBC (256-bit)	PRF_HMAC_SHA2_512	AUTH_HMAC_SHA2_512_256	14
IKESA.4	ENCR_CHACHA20_POLY1305	PRF_HMAC_SHA2_256	Not Specified or NONE (0)	14
IKESA.5	ENCR_AES_GCM_16	PRF_HMAC_SHA2_256	Not Specified or NONE (0)	14
IKESA.6	ENCR_AES_CCM_8	PRF_HMAC_SHA2_256	Not Specified or NONE (0)	14
IKESA.7	ENCR_AES_CBC (128-bit)	PRF_HMAC_SHA1	AUTH_HMAC_SHA1_96	14
IKESA.8	ENCR_AES_CBC (128-bit)	PRF_AES128_XCBC	AUTH_AES_XCBC_96	14
IKESA.9	ENCR_AES_CBC (128-bit)	PRF_HMAC_SHA2_256	AUTH_HMAC_SHA2_256_128	19

D-H (Type 4) – Groups

- 14 – 2048-bit MODP Group
- 19 – 256-bit random ECP Group

IKE_AUTH (IPSEC_SA/ESP) Algorithm List

The test case parts itemized below are used in this section, and referred to by each test case.

Proposal	ENCR – Type 1	INTEG – Type 3
IPSECSA.1	ENCR_AES_CBC (128-bit)	AUTH_HMAC_SHA2_256_128
IPSECSA.2	ENCR_AES_CBC (256-bit)	AUTH_HMAC_SHA2_256_128
IPSECSA.3	ENCR_AES_CBC (256-bit)	AUTH_HMAC_SHA2_512_256
IPSECSA.4	ENCR_NULL	AUTH_HMAC_SHA2_256_128
IPSECSA.5	ENCR_NULL	AUTH_AES_XCBC_96
IPSECSA.6	ENCR_NULL	AUTH_HMAC_SHA1_96
IPSECSA.7	ENCR_AES_CCM_8 (128-bit)	Not Specified or NONE (0)
IPSECSA.8	ENCR_AES_GCM_16 (128-bit)	Not Specified or NONE (0)
IPSECSA.9	ENCR_AES_GCM_16 (256-bit)	Not Specified or NONE (0)
IPSECSA.10	ENCR_NULL_AUTH_AES_GMAC (128-bit)	Not Specified or NONE (0)
IPSECSA.11	ENCR_NULL_AUTH_AES_GMAC (256-bit)	Not Specified or NONE (0)
IPSECSA.12	ENCR_CHACHA20_POLY1305	Not Specified or NONE (0)

IPsec.IO.3.1: IKE_SA_INIT Algorithms

Purpose:

Verify various algorithms for the IKE_SA negotiated during IKE_SA_INIT.

IKE_SA_INIT Algorithm List

The test case parts itemized below are used in this section, and referred to by each test case.

Proposal	ENCR – Type 1	PRF – Type 2	INTEG – Type 3	DH – Type 4	Requirement Level
IKESA.1	ENCR_AES_CBC (128-bit)	PRF_HMAC_SHA2_256	AUTH_HMAC_SHA2_256_128	14	MUST
IKESA.2	ENCR_AES_CBC (256-bit)	PRF_HMAC_SHA2_256	AUTH_HMAC_SHA2_256_128	14	MUST
IKESA.3	ENCR_AES_CBC (256-bit)	PRF_HMAC_SHA2_512	AUTH_HMAC_SHA2_512_256	14	SHOULD+
IKESA.4	ENCR_CHACHA20_POLY1305	PRF_HMAC_SHA2_256	Not Specified or NONE (0)	14	SHOULD
IKESA.5	ENCR_AES_GCM_16	PRF_HMAC_SHA2_256	Not Specified or NONE (0)	14	MUST
IKESA.6	ENCR_AES_CCM_8	PRF_HMAC_SHA2_256	Not Specified or NONE (0)	14	SHOULD (IoT)
IKESA.7	ENCR_AES_CBC (128-bit)	PRF_HMAC_SHA1	AUTH_HMAC_SHA1_96	14	MUST-
IKESA.8	ENCR_AES_CBC (128-bit)	PRF_AES128_XCBC	AUTH_AES_XCBC_96	14	SHOULD (IoT)
IKESA.9	ENCR_AES_CBC (128-bit)	PRF_HMAC_SHA2_256	AUTH_HMAC_SHA2_256_128	19	SHOULD

D-H (Type 4) – Groups

- 14 – 2048-bit MODP Group
- 19 – 256-bit random ECP Group

Notes:
Test in this part should be named according to the SA selected from the **IKE_SA_INIT Algorithm List**.

For example: IPsec.IO.3.1.A.IKESA.3 would refer to IPsec.IO.3.1, End-Node vs. End-Node Transport Mode, using ENCR_AES_CBC (256-bit), PRF_HMAC_SHA2_512, AUTH_HMAC_SHA2_512_256, 14.

Common Procedure:
Complete the following procedure for each of the supported/mandatory combinations in the table below.

Step	Action	Expected Result

1.	Transmit data from TAR 2 to TAR 1, or otherwise cause TAR 2 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR 2, including IKE_SA_INIT and IKE_AUTH.
2.	If necessary, re-transmit data from TAR 2 to TAR 1.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.

	IKESA								
Configuration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Cfg.EN.EN.1	M	M	S+	S	M	S	M-	S	S
Cfg.EN.EN.2	M	M	S+	S	M	S	M-	S	S
Cfg.EN.SGW.1	M	M	S+	S	M	S	M-	S	S
Cfg.SGW.SGW.1	M	M	S+	S	M	S	M-	S	S

Possible Problems:

- None

IPsec.IO.3.2: IKE_AUTH Algorithms

Purpose:

Verify various algorithms for the IPSEC_SA (ESP) negotiated during IKE_AUTH.

IKE_AUTH (IPSEC_SA/ESP) Algorithm List

The test case parts itemized below are used in this section, and referred to by each test case.

Proposal	ENCR – Type 1	INTEG – Type 3	Requirement Level
IPSECSA.1	ENCR_AES_CBC (128-bit)	AUTH_HMAC_SHA2_256_128	MUST
IPSECSA.2	ENCR_AES_CBC (256-bit)	AUTH_HMAC_SHA2_256_128	MUST
IPSECSA.3	ENCR_AES_CBC (256-bit)	AUTH_HMAC_SHA2_512_256	SHOULD+
IPSECSA.4	ENCR_NULL	AUTH_HMAC_SHA2_256_128	MUST
IPSECSA.5	ENCR_NULL	AUTH_AES_XCBC_96	SHOULD (IoT)
IPSECSA.6	ENCR_NULL	AUTH_HMAC_SHA1_96	MUST-
IPSECSA.7	ENCR_AES_CCM_8 (128-bit)	Not Specified or NONE (0)	SHOULD (IoT)
IPSECSA.8	ENCR_AES_GCM_16 (128-bit)	Not Specified or NONE (0)	MUST
IPSECSA.9	ENCR_AES_GCM_16 (256-bit)	Not Specified or NONE (0)	MUST
IPSECSA.10	ENCR_NULL_AUTH_AES_GMAC (128-bit)	Not Specified or NONE (0)	MUST
IPSECSA.11	ENCR_NULL_AUTH_AES_GMAC (256-bit)	Not Specified or NONE (0)	MUST
IPSECSA.12	ENCR_CHACHA20_POLY1305	Not Specified or NONE (0)	SHOULD

Notes:
Test in this part should be named according to the SA selected from the **IKE_AUTH (IPSEC_SA/ESP) Algorithm List**.

For example: IPsec.IO.3.2.A.IPSECSA.3 would refer to IPsec.IO.3.2, End-Node vs. End-Node Transport Mode, using ENCR_AES_CBC (256-bit), AUTH_HMAC_SHA2_512_256.

Common Procedure:

Step	Action	Expected Result
1.	Transmit data from TAR 2 to TAR 1, or otherwise cause TAR 2 to initiate IKEv2 negotiation.	A 4 message initial IKEv2 Exchange between TAR 1 and TAR

		2, including IKE_SA_INIT and IKE_AUTH.
2.	If necessary, re-transmit data from TAR 2 to TAR 1.	TAR 2 transmits a valid ESP packet as negotiated, and TAR 1 transmits a valid ESP packet in response as negotiated.

	IPSECSA											
Configuration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Cfg.EN.EN.1	M	M	S+	M	S	M-	S	M	M	M	M	S
Cfg.EN.EN.2	M	M	S+	M	S	M-	S	M	M	M	M	S
Cfg.EN.SGW.1	M	M	S+	M	S	M-	S	M	M	M	M	S
Cfg.SGW.SGW.1	M	M	S+	M	S	M-	S	M	M	M	M	S

M – MUST
S – SHOULD

Possible Problems:

- None

Modification Record

Version	Date	Editor	Modification
2.0.0	2018-09-20	Timothy Carlin	Added IKEv2 Interoperability Test Cases Restructured Test Sections
2.0.0	2017-02-24	Timothy Carlin	Renumber and reorganized Test Sections Create Common Configurations Added CHAHA20-POLY1305 to ADVANCED encryption algorithms Changed AES-CBC(128-bit) and NULL from ADVANCED to BASIC encryption algorithms Changed 3DES-CBC from BASIC to ADVANCED encryption algorithms Added AES-GCM(128-bit) to BASIC encryption algorithms Added AES-CBC (192-bit), AES-CBC(256-bit), AES-GCM(192-bit), and AES-GCM(256-bit) to ADVANCED encryption algorithms Changed HMAC-SHA-256 from ADVANCED to BASIC Integrity algorithms Added AES-GMAC(128-bit) to BASIC Integrity algorithms Added HMAC-SHA-384, HMAC-SHA-512, AES-GMAC(192-bit), and AES-GMAC(256-bit) to ADVANCED Integrity algorithms Added test cases for ESP=AES-CBC(128-bit) HMAC-SHA-256 (Section 5.1.13, 5.2.13, 5.3.13, 5.4.13) Added test cases for ESP=AES-CBC HMAC-SHA-384 (Section 5.1.14, 5.2.14, 5.3.14, 5.4.14) Added test cases for ESP=AES-CBC(256-bit) HMAC-SHA-512 (Section 5.1.15, 5.2.15, 5.3.15, 5.4.15) Added test cases for ESP=AES-GCM NULL (Section 5.1.16, 5.2.16, 5.3.16, 5.4.16), RFC 4106 “The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)” Added test cases for ESP=NULL AES-GMAC (Section 5.1.17, 5.2.17, 5.3.17, 5.4.17), RFC 4543 “The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH Modified formatting and fixed typos
1.11.0	2011-05-10	Timothy Carlin	Change test sequence of Section 5.3.11 (Section 5.3.11 uses new test topology for End-Node vs. SGW Tunnel Mode Test 2) Removed NULL Integrity tests Typos and Bug fixes
1.10.0	2010-05-10		Support Integrity Algorithm HMAC-SHA-256 in RFC 4868 (Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec) (Section 5.1.12, 5.2.12, 5.3.12, 5.4.12)
1.9.2	2017-03-11		Add Fragmentation test cases (Section 5.1.11, 5.2.11, 5.3.11, 5.4.11) Editorial fix at Appendix-A Section 1.2 Added the description of keying information file at Appendix-A Required Data Added file lists needed to be submitted at Appendix-A Section 1.3 Clarified the interoperable device requirement at REQUIREMENTS section

1.9.1	2009-01-06	Support the passive node which doesn't have ping6 application (as Possible Problems in Section 5.1.8, 5.3.8, 5.4.8)
1.9.0	2008-12-09	Support RFC 4312 (The Camellia Cipher Algorithm and Its Use With IPsec) (Section 5.1.7, 5.2.7, 5.3.7, 5.4.7) Use IPv6 prefix defined in RFC 3849 for the documentation
1.5.2	2007-10-11	Remove ESN test cases (Section 5.1.8, 5.2.8, 5.3.8, 5.4.8)
1.5.1	2007-06-19	Correct subsection in Section 5.3
1.5.0	2007-05-27	Support IPsec v3
1.4.3	2005-10-06	Update Appendix
1.4.2	2005-09-30	Change ping direction for tunnel tests between END-Nodes
1.4.1	2005-09-22	Editorial fix
1.4	2005-03-01	Change Keys
1.3	2004-12-21	Correct Require table
1.2	2004-11-29	Add concept of End-Node rather than Host Add criteria Editorial fix
1.1	2004-09-30	
1.0	2004-09-24	

Appendix A: Manual Settings Disallowed

The below algorithms are inherently insecure when used with static keys. The quotes below reference the applicable sections describing this for each algorithm.

AES-CCM

According to RFC 4309, Section 2:

> AES CCM employs counter mode for encryption. As with any stream cipher, reuse of the same IV value with the same key is catastrophic. An IV collision immediately leaks information about the plaintext in both packets. For this reason, it is inappropriate to use this CCM with statically configured keys. Extraordinary measures would be needed to prevent reuse of an IV value with the static key across power cycles. To be safe, implementations MUST use fresh keys with AES CCM. The Internet Key Exchange (IKE) [IKE] protocol or IKEv2 [IKEv2] can be used to establish fresh keys.

Therefore, Manual Keys MUST NOT be used with this algorithm, and devices that do not support IKEv2 will FAIL this test case.

AES-GCM

According to RFC4106, Section 2:

> Because reusing an nonce/key combination destroys the security guarantees of AES-GCM mode, it can be difficult to use this mode securely when using statically configured keys. For safety's sake, implementations MUST use an automated key management system, such as the Internet Key Exchange (IKE) [RFC2409], to ensure that this requirement is met.

Therefore, Manual Keys MUST NOT be used with this algorithm, and devices that do not support IKEv2 will FAIL this test case

AES-GMAC

According to RFC4106, Section 2:

> Because reusing an nonce/key combination destroys the security guarantees of AES-GCM mode, it can be difficult to use this mode securely when using statically configured keys. For safety's sake, implementations MUST use an automated key management system, such as the Internet Key Exchange (IKE) [RFC2409], to ensure that this requirement is met.

Therefore, Manual Keys MUST NOT be used with this algorithm, and devices that do not support IKEv2 will FAIL this test case.

ChaCha20-Poly1305

According to RFC7634, Section 2:

> The Internet Key Exchange Protocol generates a bitstring called KEYMAT using a pseudorandom function (PRF). That KEYMAT is divided into keys for encryption, message authentication, and whatever else is needed. The KEYMAT requested for each ChaCha20-Poly1305 key is 36 octets. The first 32 octets are the 256-bit ChaCha20 key, and the remaining 4 octets are used as the Salt value in the nonce.

Also, from Section 5:

> The most important security consideration in implementing this document is the uniqueness of the nonce used in ChaCha20. The nonce should be selected uniquely for a particular key, but unpredictability of the nonce is not required. Counters and LFSRs are both acceptable ways of generating unique nonces.

Therefore, Manual Keys MUST NOT be used with this algorithm, and devices that do not support IKEv2 will FAIL this test case.

****