Date:	8 May 2007
To:	T10 Technical Committee
From:	Matt Ball (Quantum) and David Black (EMC) — {{edited by Ralph O. Weber}}
Subject:	SPC-4: Establishing a Security Association using IKEv2

Introduction

This proposal provides a method, named IKEv2-SCSI, for creating a security association using Diffie-Hellman (DH) key establishment based on IETF RFC 4306 "IKEv2" and guidance from NIST SP 800-56A.

A security association provides the infrastructure necessary for sending encrypted messages between the application client and device server, and allows end-point authentication to prevent man-in-the-middle attacks.

References

T10/SSC-3r3c	SCSI-3 Stream Commands.
T10/SPC-4r10	SCSI Primary Commands.
T10/06-225r5	Matt Ball, SSC-3: Key Entry using Encapsulating Security Payload (ESP).
NIST SP 800-56A	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography.
T11/06-157v3	Fibre Channel - Security Protocols (FC-SP)

Differences between IKEv2 and IKEv2-SCSI

The important differences between IKEv2 and IKEv2-SCSI include the following:

a) An SA created by the IKEv2-SCSI protocol is used to directly protect SCSI traffic. There is no concept of child SAs; this is based on a design assumption that SA usage will be infrequent in SCSI command streams;
b) The entity sending SCSI traffic determines what SA is used and what is to be protected via appropriate use of the SAI for the SA. SCSI addresses are not involved in this determination, and hence IKEv2-SCSI does not provide address-based data origin authentication; this functionality is left to SCSI transports, in part because SCSI addresses are transport-specific. SCSI command standards define the uses for SAs and the mechanisms for communicating the applicable SAIs between application clients and device servers;
c) Cryptographic algorithm negotiation has been simplified to use a SCSI device capabilities design approach. The simplification includes removal of IKEv2's proposal concept; the application client chooses algorithms supported by the device server in accordance with the application client's policy and preferences; and
d) Significant portions of IKEv2 have been removed as inapplicable to SCSI. The removed functionality includes Traffic Selectors, NAT Traversal, Remote Configuration, and Compression.

In IKEv2 terminology, the application client is the IKEv2 initiator and the device server is the IKEv2 responder. A device server cannot initiate IKEv2-SCSI.

Revision History

r0 Initial revision with lots of help from Ralph Weber.
r1 Incorporate comments from discussion in November Las Vegas meeting. Major changes:
- Add timeout support (new STV payload). Timeouts are recommended (should) rather than mandatory (shall).
- Change sequencing support to talk about device server discarding state instead of mandatory timeouts.
- Changed most IKEv2-SCSI-specific ASC/ASCQs to new values.
- Require 16 kilobytes of parameter data support.

- Tweak Certificate Encoding field in Certificate Request payload so that it tells the device server whether or not a URL-based certificate format is acceptable to the application client.
- Make support for skipping authentication optional.
- Specify and explain what not to do with the PROGRESS INDICATION sense data.
- Add usage data to SCA payload
- Added the notify (Initial contact only) and delete payloads
- [Added a number of Editor's notes indicating significant additional work to be done ;-).]

r2 Incorporate comments from January Orlando meeting and additional design work. Major changes:
- Use separate SCSI SA Creation Capabilities payload in Device Server Capabilities phase. This removes the erroneous use of Usage Data in the Device Server Capabilities phase.
- Adopt certificate encoding recommendations from RFC 4718 and in addition prohibit Hash and URL certificate formats.
- Add new IANA-allocated values for crypto mechanisms. Remove GMAC as RFC 4543 does not define its use with IKEv2. Adjust vendor-specific ranges to match IANA private use ranges for IKEv2.
- Allow Fibre Channel names as identifiers (for FC-SP certificates).
- Tighten down specification of Delete to reflect removal of Child SAs and avoid problems - it has to be sent on the SA to be deleted because there are no child SAs. Add model clause to explain how Delete works.
- Factor out SA creation command sequencing into a separate subsection. This reduces the amount of text and covers a number of additional error cases.
- Add subsection on how to populate the fields of an SA.
- Renumber IKEv2-SCSI exchange types into IKEv2's private use range. Add additional explanation of IKEv2 header fields, including sequence association checks and errors.
- Lots of other edits and changes (e.g., to remove Editor's notes).

r3 Incorporate comments from March Memphis meeting. Major changes:
- Remove DSS digital signature support. Remove support for encryption without integrity. Finish aligning cryptographic algorithm identifiers with IANA registries for IKEv2.
- Terminology change to SA creation cryptographic command sequence, only allow one at a time per I_T_L Nexus (so device server only has to save one set of active parameters), but return NOT READY if another is attempted instead of aborting the original sequence.
- Adapt to USAGE changes made to SA proposal as part of approving it.
- Add key length values to encryption algorithm table. Add notes about extra salt bytes that CCM and GCM mode take from KEYMAT.

r4 Incorporate comments from April Houston interim meeting. Major changes:
- Change terminology from protocol "phases" to protocol "steps", add summary of protocol steps.
- Distinguish IKEv2-SCSI keys from SA keys for clarity.
- Modify (generally reduce) allowed cryptographic algorithms. GMAC cannot be added because IETF does not support GMAC usage in IKEv2.
- Add AUTH_NONE integrity support (no separate integrity algorithm) for use with AES_CCM and AES_GCM combined mode algorithms that provide integrity.
- Expand SA type (usage) to 2 bytes and reformat SCA payload accordingly. Did not add a second pair of nonces because IKEv2 (RFC 4306) uses the same nonces for the IKEv2 (SA) keys and the keys for the first child SA.

r5 Convert to FrameMaker and edit for T10 style

Unless otherwise indicated additions are shown in blue, deletions in ~~red strikethrough~~, and comments in green.

Proposed Changes in SPC-4 r10

Introduction

The SCSI Primary Commands - 4 (SPC-4) standard is divided into the following clauses and annexes:

Clause 1	is the scope.
Clause 2	enumerates the normative references that apply to this standard.
Clause 3	describes the definitions, symbols, and abbreviations used in this standard.
Clause 4	describes the conceptual relationship between this document and the SCSI-3 Architecture Model.
Clause 5	describes the command model for all SCSI devices.
Clause 6	defines the commands that may be implemented by any SCSI device.
Clause 7	defines the parameter data formats that may be implemented by any SCSI device.
Clause 8	defines the well known logical units that may be implemented by any SCSI device.
Annex A	identifies differences between the terminology used in this standard and previous versions of this standard. (informative)
Annex B	describes the PERSISTENT RESERVE OUT command features necessary to replace the reserve/release management method and provides guidance on how to perform a third party reservation using persistent reservations. (informative)
Annex C	identifies the differences between IKEv2 (see RFC 4306) and the IKEv2-SCSI SA creation protocol defined by this standard.
Annex ~~C~~D	lists code values in numeric order. (informative)
Annex ~~D~~E	lists assigned vendor identifiers. (informative)

2.4 NIST References

…

NIST FIPS 180-2, *Secure Hash Standard*

NIST FIPS 198a, *The Keyed-Hash Message Authentication Code (HMAC)*

{{Note: both of this references are already listed in SPC-4 r10 … to whit …

Copies of the following approved FIPS standards may be obtained through the National Institute of Standards and Technology (NIST) at http://csrc.nist.gov/publications/fips/index.html.

FIPS 140-2, *Annex C: Approved Random Number Generators*
FIPS 180-2 with Change Notice 1 dated February 25, 2004, *Secure Hash Standard*
FIPS 198a, *The Keyed-Hash Message Authentication Code (HMAC)*

end note}}

2.5 IETF References

{{add the following references to those already listed and maintain RFC number order}}

RFC 2404, *The Use of HMAC-SHA-1-96 within ESP and AH*

RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec*

RFC 2437, *PKCS #1: RSA Cryptography Specifications Version 2.0*

RFC 3280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

RFC 3447, *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*

RFC 3602, *The AES-CBC Cipher Algorithm and Its Use with IPsec*

RFC 3526, *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)*

RFC 4106, *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)*

RFC 4306, *Internet Key Exchange (IKEv2) Protocol* {{note: this entry already appears in SPC-4 r10}}

RFC 4309, *Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)*

RFC 4595, *Use of IKEv2 in the Fibre Channel Security Association Management Protocol*

RFC 4718, *IKEv2 Clarifications and Implementation Guidelines*

RFC 4753, *ECP Groups for IKE and IKEv2*

RFC 4754, *IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)*

…

3.1 Definitions

{{insert the following in the proper alphabetical order.}}

3.1.c cryptographic command sequence (CCS): A defined sequence of SECURITY PROTOCOL IN commands (see 6.29) and SECURITY PROTOCOL OUT commands (see 6.30) that realize the cryptographic protocol of a specified security operation (e.g., the SECURITY PROTOCOL IN commands and SECURITY PROTOCOL OUT commands in the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.f) SA (see 3.1.119) creation transaction).

3.1.d cryptographic key: A cryptographically protected complex secret (i.e., not a password) that is known only to a defined and limited set of entities (e.g., one application client and one device server). Two kinds of keys are associated with SA (see 3.1.119) operation: IKEv2-SCSI keys (see 3.1.g) and SA keys (see 3.1.t).

3.1.f IKEv2-SCSI: Internet Key Exchange protocol version 2 for SCSI. See 5.13.4.

3.1.g IKEv2-SCSI keys: Cryptographic keys (see 3.1.d) used to provide security for IKEv2-SCSI operations (e.g., creation and deletion of the SA). IKEv2-SCSI keys that are needed after SA creation is complete are maintained in the MGMT_DATA SA parameter (see 3.1.103).

3.1.h IKEv2-SCSI CCS: The CCS (see 3.1.c) that is the Key Exchange step and Authentication step, if any, of an IKEv2-SCSI (see 3.1.f) SA (see 3.1.119) creation transaction.

3.1.r SA creation transaction: Any sequence of SECURITY PROTOCOL IN commands (see 6.29) and SECURITY PROTOCOL OUT commands (see 6.30), including but not limited a CCS (see 3.1.c), that is used to create an SA between an application client and device server. See .

3.1.s SA generation: Computation and initialization of the SA parameter values (see 3.1.103) required to create an SA (see 3.1.119). This is the final step in creating an SA. It is performed separately by the application client and device server after all SCSI commands required to create an SA have been performed without error.

3.1.t SA keys: Cryptographic keys (see 3.1.d) that are maintained in the USAGE_DATA SA parameter (see 3.1.103) and used to provide security for the operations that use the SA (e.g., encryption of command parameter data).

3.1.u SA participant: An application client or device server that participates in the creation or use of an SA (see 3.1.119).

3.2 Symbols and acronyms

{{insert the following in the proper alphabetical order.}}

CCS	cryptographic command sequence (see 3.1.c)
PKI	Public Key Infrastructure (see RFC 3280)

…

5.13.2 Security associations

5.13.2.1 Principals of ~~security associations~~ SAs

…

5.13.2.2 SA parameters

…

5.13.2.3 Creating a security association

The SECURITY PROTOCOL IN command (see 6.29) and SECURITY PROTOCOL OUT command (see 6.30) security protocols shown in table 46 are used to create SAs. The process of creating an SA establishes the SA parameter (see 5.13.2.2) values as follows:

a) Initial values for:
   A) Both (i.e., application client and device server) sequence numbers set to zero; and
   B) All KEYMAT bytes set to zero;
b) Unchanging values for the lifetime of the SA:
   A) Both SAIs;

B) TIMEOUT;
C) Both nonces;
D) KDF_ID;
E) USAGE_TYPE;
F) USAGE_DATA; and
G) MGMT_DATA;
and

c) Values that are zero upon completion of SA creation:
A) KEY_SEED.

Table 46 — Security protocols ~~that~~ used to create SAs

Security Protocol Code	Description	Reference
~~TBD~~	~~TBD~~	~~TBD~~
zzh	SA creation capabilities	7.7.2
xxh	IKEv2-SCSI	5.13.4

5.13.3 Key derivation functions

…

5.13.4 Using IKEv2-SCSI to create a security association

{{All of 5.13.4, 5.13.5, 5.13.6, and 5.13.7 are new. Additions/deletions markups are not applied in these subclauses.}}

5.13.4.1 Overview

The IKEv2-SCSI protocol is a subset of the IKEv2 protocol (see RFC 4306) that this standard defines for use in the creation and maintenance of an SA (see 3.1.119).

An IKEv2-SCSI SA creation transaction (see 3.1.r) shall only be initiated by the application client.

The IKEv2-SCSI protocol creates two SAs:

a) An SA that protects data sent from the application client to the device server; and
b) An SA that protects data sent from the device server to the application client.

An IKEv2-SCSI SA creation transaction encompasses up to three steps that shall be performed in the following order:

1) Device Server Capabilities step (see 5.13.4.2): The application client determines the device server's cryptographic capabilities;
2) Key Exchange step (see 5.13.4.3): The application client and device server:
   A) Perform a key exchange;
   B) Determine SAIs (see 3.1.120); and
   C) May complete the creation of the SA;
   and
3) Authentication step (see 5.13.4.4): Unless omitted by application client and device server negotiations in the previous steps:

- A) The application client and device server authenticate:
  - a) Each other;
  - b) The key exchange; and
  - c) The capability selection;

  and
- B) Complete the creation of the SA.

The values in the SECURITY PROTOCOL field and the SECURITY PROTOCOL SPECIFIC field in the SECURITY PROTOCOL IN command (see 6.29) and SECURITY PROTOCOL OUT command (see 6.30) identify the step for the IKEv2-SCSI protocol (see 7.7.3.2).

The Key Exchange step and the Authentication step depend on the results from the Device Capabilities step in order to create an SA. Two sets of keys are involved in creation of an SA:

- a) IKEv2-SCSI keys that are created by the IKEv2-SCSI Key Exchange step. These keys are used by the IKEv2-SCSI Authentication step and to delete the SA; and
- b) SA keys created as part of generating the SA. These keys are used by SCSI operations that obtain security from the generated SA.

An application client may or may not:

- a) Proceed to the Key Exchange step after the Device Server Capabilities step; or
- b) Perform a separate Device Server Capabilities step for each IKEv2-SCSI SA creation transaction.

If the device server's capabilities have changed, the Key Exchange step may return an error, and the Authentication step shall return an error. The application client may recover from such errors by repeating the Device Server Capabilities step.

After a Device Capabilities step, the application client performs SA creation by sending a sequence of two or four IKEv2-SCSI commands over a single I_T_L nexus to the device server. These commands constitute an IKEv2-SCSI CCS (see 3.1.h):

1) A Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.3.2);
2) A Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.3.3);
3) An Authentication step SECURITY PROTOCOL OUT command (see 5.13.4.4.2); and
4) An Authentication step SECURITY PROTOCOL IN command (see 5.13.4.4.3).

{{From here to the next comment that indicates the end of text introduced in r5, new text is included in an attempt to codify a no-overlaps rule previously stated in the comparison between IKE and IKEv2-SCSI (now Annex C) and to instantiate most of the other requirements described in 5.13.4.6.}} {{More work is required to instantiate the agreements reached during discussions of 07-226r1}}

The device server shall maintain state for the IKEv2-SCSI CCS from the time the Key Exchange step SECURITY PROTOCOL OUT command is completed with GOOD status until:

- a) The IKEv2-SCSI CCS completes successfully;
- b) A properly ordered command in the IKEv2-SCSI CCS is terminated with a status other than GOOD; or
- c) The number of seconds specified in the IKEV2-SCSI PROTOCOL TIMEOUT field of the SCSI Timeout Values payload (see 7.7.4.14) in the Key Exchange step SECURITY PROTOCOL OUT parameter data elapses and none of the following commands have been received:
  - A) The next command in the IKEv2-SCSI CCS; or
  - B) A REQUEST SENSE command;
- d) Power cycle;
- e) Hard reset;

f) Logical unit reset; or
g) I_T nexus loss.

If the device server receives a SECURITY PROTOCOL OUT or SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., xxh) whose processing is inconsistent with the device server maintained state for the IKEv2-SCSI CCS for the I_T_L nexus (e.g., receipt of a SECURITY PROTOCOL OUT command when a SECURITY PROTOCOL IN command is expected), then:

a) The state maintained for the IKEv2-SCSI CCS shall not be altered; and
b) The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION OPERATION IN PROGRESS.

{{SA CREATION OPERATION IN PROGRESS is a new additional sense code.}}

The device server shall complete the processing of each command in the IKEv2-SCSI CCS before returning status for that command. The application client may determine the progress of the device server in processing a command by sending a REQUEST SENSE command as described in 5.13.7.

While it is processing one command in the IKEv2-SCSI CCS, the device server shall respond to any SECURITY PROTOCOL OUT or SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., xxh) in the manner that this subclause describes for the handling of any such command whose processing is inconsistent with the device server maintained state for the IKEv2-SCSI CCS for the I_T_L nexus.

Except for the cases described in this subclause, the fact that the device server is maintaining IKEv2-SCSI CCS state on a particular I_T_L nexus shall not affect the processing of new commands received on that I_T_L nexus.

{{end new-to-r5 text.}}

{{N.B. It is possible for a man-in-the-middle to hijack SECURITY PROTOCOL IN data. This is possible because the CDB contains insufficient data to verify it is source.}}

If the application client and device server agree to use IKE_AUTH_NONE during the Key Exchange step, the Authentication step is skipped and in this case the IKEv2-SCSI CCS consists of the two Key Exchange step commands.

Use of the Authentication step is negotiated in the Device Server Capabilities step and the Key Exchange step. If both SA participants agree that the Authentication step is not used, then the Authentication step is omitted and SA creation occurs upon the completion of the Key Exchange step. If either SA participant requires that the Authentication step be used, the device server shall not complete SA creation until successful completion of Authentication step.

SA participants should perform the Authentication step unless man-in-the-middle attacks (see 5.13.1.4) are not of concern or are prevented by other means such as physical security of the transport. The Authentication step should be performed if there is any doubt as to whether it is needed.

NOTE x1 - Omission of the Authentication step provides no defense against a man-in-the-middle adversary that is capable of modifying SCSI commands. Such an adversary can insert itself as an intermediary on the created SA without knowledge of the SA participants, thereby completely subverting the intended security. Omission of the Authentication step is only appropriate in environments where the absence of such adversaries is assured by other means, (e.g., a direct physical connection between the systems on which the application client and device server or use of end-to-end security in the SCSI transport security such as FC-SP).

5.13.4.1.1 IKEv2-SCSI Protocol summary {{if this has not been changed to 5.13.4.2 it should have been}}

This subclause graphically summarizes the IKE-v2-SCSI payloads (see 7.7.4) that are exchanged between an application client and a device server during all steps of an IKEv2-SCSI SA creation transaction (see 3.1.r). Each IKEv2-SCSI step (see 5.13.4.1) is shown in a separate figure.

Figure x1 shows the Device Server Capabilities step (see 5.13.4.2). The Device Server Capabilities step consists of a SECURITY PROTOCOL IN command carrying a SCSI SA Creation Capabilities payload (see 7.7.4.12). The IKEv2-SCSI header is not used.

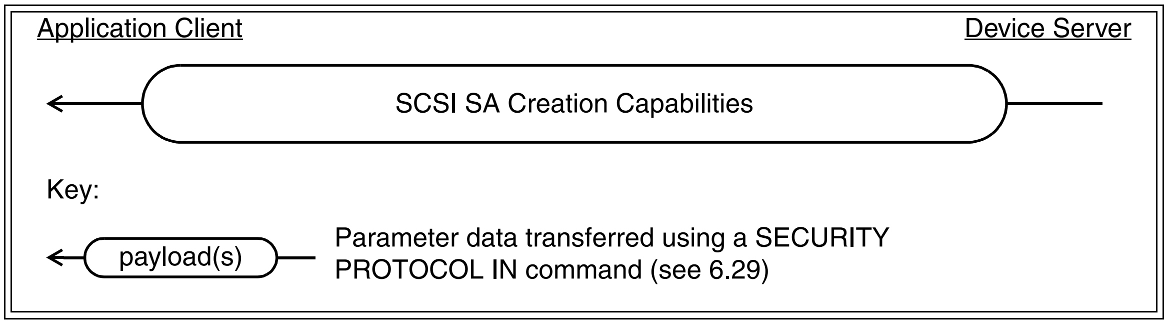
Application Client
Device Server
SCSI SA Creation Capabilities
Key:
payload(s)
Parameter data transferred using a SECURITY PROTOCOL IN command (see 6.29)

Figure x1 — IKEv2-SCSI Device Server Capabilities step

The SCSI SA Creation Capabilities payload indicates the device server's capabilities for SA creation.

Figure x2 shows the Key Exchange step (see 5.13.4.3). The Key Exchange step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.

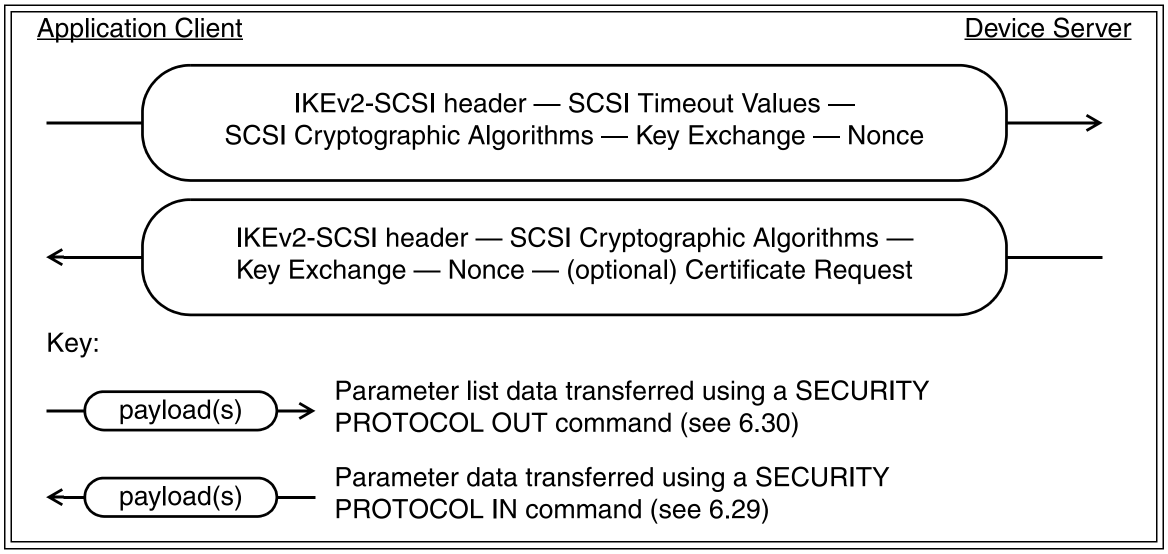
Application Client
Device Server
IKEv2-SCSI header — SCSI Timeout Values — SCSI Cryptographic Algorithms — Key Exchange — Nonce
IKEv2-SCSI header — SCSI Cryptographic Algorithms — Key Exchange — Nonce — (optional) Certificate Request
Key:
payload(s)
Parameter list data transferred using a SECURITY PROTOCOL OUT command (see 6.30)
payload(s)
Parameter data transferred using a SECURITY PROTOCOL IN command (see 6.29)

Figure x2 — IKEv2-SCSI Key Exchange step

The SCSI Timeout Values payload contains timeouts for SA creation and usage. The SCSI Cryptographic Algorithms payloads select and agree on usage of the SA and the cryptographic algorithms for the SA. The Key Exchange and Nonce payloads are part of the key and nonce exchanges that are used to generate SA keys. The optional Certificate Request payload enables the device server to request a certificate from the application client.

Figure x3 shows the Authentication step (see 5.13.4.4). The Authentication step consists of a SECURITY PROTOCOL OUT command followed by a SECURITY PROTOCOL IN command.

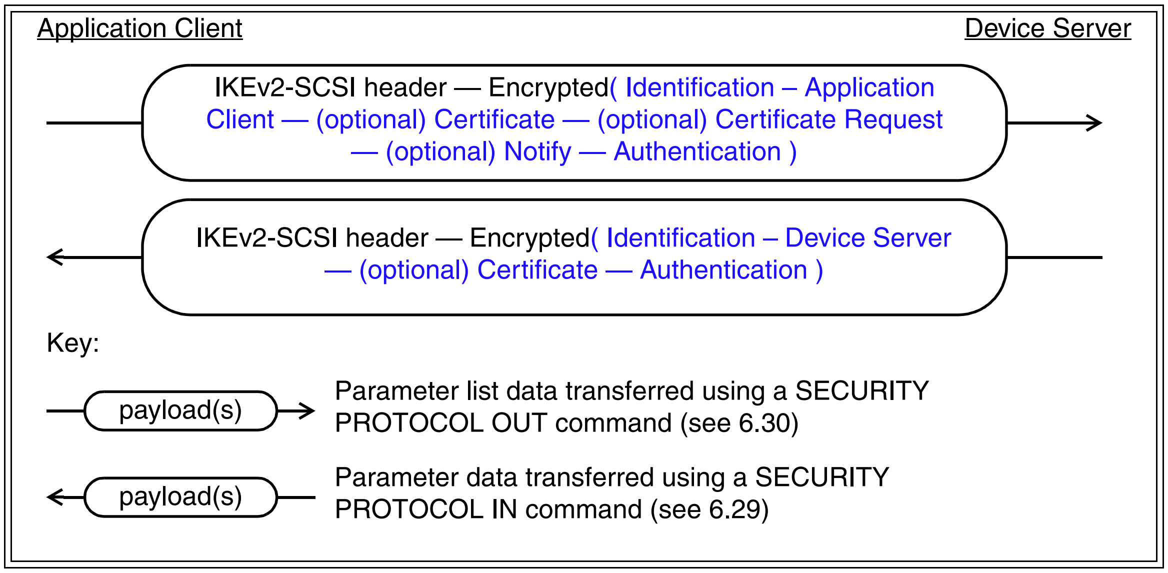
Application Client
Device Server
IKEv2-SCSI header — Encrypted(Identification – Application Client — (optional) Certificate — (optional) Certificate Request — (optional) Notify — Authentication)
IKEv2-SCSI header — Encrypted(Identification – Device Server — (optional) Certificate — Authentication)
Key:
payload(s)
Parameter list data transferred using a SECURITY PROTOCOL OUT command (see 6.30)
payload(s)
Parameter data transferred using a SECURITY PROTOCOL IN command (see 6.29)

Figure x3 — IKEv2-SCSI Authentication step

{{The blue in figure x3 is intended for inclusion in SPC-4 and is used in conformance with the T10 Style Guide to assist viewers capable of rendering color to more easily see the encrypted payloads. It is not normative.}}

All payloads in the Authentication step are protected using the cryptographic algorithms determined by the SCSI Cryptographic Algorithms payloads in the Key Exchange step.

The Identification payloads contain the identities to be authenticated and are not required to be SCSI names or identities. The optional Certificate payloads contain certificates for authentication and the optional Certificate Request payload enables the application client to request a certificate from the device server. The optional Notify payload provides a mean to delete stale SAs between the same SA participants. The Authenticate payloads authenticate not only the SA participants, but also the entire protocol sequence (e.g., the Authenticate payloads prevent a man-in-the-middle attack from succeeding).

5.13.4.2 Device Server Capabilities step

In the Device Server Capabilities step, the application client sends a SECURITY PROTOCOL IN command (see 6.29) with the SECURITY PROTOCOL field set to SA Creation Capabilities (i.e., zzh) and the SECURITY PROTOCOL SPECIFIC field set to 0101h (i.e., read the device server's IKEv2-SCSI cryptographic capabilities).

The device server shall use the authentication algorithm type F9h in the SCSI Cryptographic Algorithms payload (see 7.7.4.13) in the Device Server Capabilities step to report supported IKEv2-SCSI authentication algorithms.

{{The requirements in the following paragraph deserve CAP study. They are not unheard of in SCSI, just uncommon.}}

An authentication algorithm type of IKE_AUTH_NONE is used to indicate that the device server permits the Authentication step to be omitted (see 5.13.4.1). The device server shall not return IKE_AUTH_NONE as an authentication algorithm type in the Device Server Capabilities step unless the device server has been configured to do so by a person who represents the owner of the SCSI target device that contains the device server. The methods for configuring a device server to return IKE_AUTH_NONE are outside the scope of this standard. Device servers shall not be manufactured to return IKE_AUTH_NONE as an authentication algorithm type in the Device Server Capabilities step.

NOTE x2 - The Device Server Capabilities step has no IKEv2 exchange equivalent in RFC 4306. This step replaces most of IKEv2's negotiation by having the application client obtain the supported capabilities from the device server.

5.13.4.3 IKEv2-SCSI Key Exchange step

5.13.4.3.1 Overview

The Key Exchange step consists of an unauthenticated Diffie-Hellman key exchange with nonces (see RFC 4306) and is accomplished as follows:

1) A SECURITY PROTOCOL OUT command (see 5.13.4.3.2);
2) A SECURITY PROTOCOL IN command (see 5.13.4.3.3); and
3) Key exchange completion (see 5.13.4.3.4)

NOTE x3 - The Key Exchange step corresponds to the IKEv2 IKE_SA_INIT exchange in RFC 4306, except that determination of device server capabilities has been moved to the Device Server Capabilities step.

5.13.4.3.2 Key Exchange step SECURITY PROTOCOL OUT command

To send its key exchange message to the device server, the application client sends a SECURITY PROTOCOL OUT command (see 6.30) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., xxh) and the SECURITY PROTOCOL SPECIFIC field set to 0102h. The parameter list consists of an IKEv2-SCSI header (see 7.7.3.3.1) and the following:

{{Are the acronyms needed here (or for any other IKE payload descriptions). See table C.1.}}

1) A SCSI Timeout Values (i.e., STV) payload (see 7.7.4.14);
2) A SCSI Cryptographic Algorithms (i.e., SCA) payload (see 7.7.4.13);
3) A Key Exchange (i.e., KE) payload (see 7.7.4.3); and
4) A Nonce (i.e.; NONCE) payload (see 7.7.4.7).

The SCSI Timeout Values payload contains the inactivity timeouts that apply to this IKEv2-SCSI SA creation transaction (see 3.1.r) and the SA (see 3.1.119) that is created.

The SCSI Cryptographic Algorithms payload contains the following information about the SA to be created:

a) The cryptographic algorithms selected by the application client; and
b) The usage data (see 7.7.4.13) that is specific to the SA.

{{The following paragraph and note replace the following text from r4: "The cryptographic algorithms shall be selected from the algorithms obtained from the device server in the Device Server Capabilities step (see 5.13.4.2). If the application client is unable to select a set of algorithms that are appropriate for the intended usage of the SA, the application client shall not perform the Key Exchange step and shall not create an SA."}}

If the SCSI Cryptographic Algorithms payload selects cryptographic algorithms that are not returned by the device server in the Device Server Capabilities step (see 5.13.4.2), the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, the additional sense code set to INVALID FIELD IN PARAMETER LIST, the SKSV bit set to one, and SENSE KEY SPECIFIC field set as defined in 4.5.2.4.2.

NOTE x4 - If the application client is unable to select a set of algorithms that are appropriate for the intended usage of the SA, the application client should not perform the Key Exchange step to request the creation of an SA.

The Key Exchange payload contains the application client's Diffie-Hellman value.

The Nonce payload contains the application client's random nonce (see 3.1.95).

5.13.4.3.3 Key Exchange step SECURITY PROTOCOL IN command

{{The following text has been reworded to better match 5.13.4.3.2 and to remove requirements on the application client.}}

If the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.3.2) completes with GOOD status, the application client sends a SECURITY PROTOCOL IN command (see 6.29) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., xxh) and the SECURITY PROTOCOL SPECIFIC field set to 0102h to obtain the device server's key exchange message.

The parameter data returned by the device server in response to the SECURITY PROTOCOL IN command shall contain an IKEv2-SCSI header (see 7.7.3.3.1) and the following:

1) A SCSI Cryptographic Algorithms (i.e., SCA) payload (see 7.7.4.13);
2) A Key Exchange (i.e., KE) payload (see 7.7.4.3);
3) A Nonce (i.e., NONCE) payload (see 7.7.4.7); and
4) Zero or more Certificate Request (i.e., CERTREQ) payloads (see 7.7.4.5).

{{The relationship of the Certificate Request payload above to the Certificate payload in the Authentication step Security Protocol OUT command needs to be clarified.}}

As part of processing of the Key Exchange step SECURITY PROTOCOL IN command, the device server shall:

a) Associate the SECURITY PROTOCOL IN command to the most recently processed Key Exchange step SECURITY PROTOCOL OUT command received on the I_T_L nexus;
b) Return the cryptographic algorithms supplied by the application client in the Key Exchange step SECURITY PROTOCOL OUT command parameter list;
c) Return its SAI (see 3.1.120) in the SCSI Cryptographic Algorithms payload;
d) Return information about the completed the Diffie-Hellman exchange with the Key Exchange payload; and
e) Return its random nonce (see 3.1.95) in the Nonce payload.

The device server may use optional Certificate Request payload(s) to specify its trust anchors list when PKI-based Authentication is being used (see RFC 3280).

{{need to add discussion about SCSI Cryptographic Algorithm payload contents. Device server may copy SP-OUT contents to SP-IN data (and add the device server's SAI per c above). Application client should ignore everything in the SCSI Cryptographic Algorithm payload except the device server's SAI.}}=WorkInProgress

{{The remainder of this subclause probably can be deleted after the other SCSI Cryptographic Algorithm payload contents requirements are codified.}}=WorkInProgress

If the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.3.3) completes with GOOD status, the application client should:

Compare the cryptographic algorithms and usage data (see 7.7.4.13) received in the SCSI Cryptographic Algorithms payload to the cryptographic algorithms and usage data that were requested in the Key Exchange step SECURITY PROTOCOL OUT command. If the algorithms and usage data are not the same, the application client shall report an error in a vendor-specific manner, shall not complete the Key Exchange step (see 5.13.4.3.4), shall not perform the Authentication step (see 5.13.4.4) and shall not create the SA (see 5.13.4.5).

5.13.4.3.4 Key Exchange step completion

If the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.3.3) completes with GOOD status, the SA participants (see 3.1.u) complete the Key Exchange step by:

{{The remainder of this subclause has not been translated.}}

generate SKEYSEED (see RFC 4306) using the specified pseudo-random function before proceeding to the Authentication step, if applicable; and then

use SKEYSEED to generate the following IKEv2-SCSI keys (see RFC 4306):

SK_d: A key used to generate the SA keys. This is recorded as KEY_SEED in the resulting SCSI security association (See TBD).

SK_ai and SK_ar: IKEv2 authentication keys for use in generation of keyed MACs at the application client (SK_ai) and the device server (SK_ar). IKEv2 refers to these as authentication keys, but their function is to provide cryptographic integrity protection for subsequent IKEv2 messages.

SK_ei and SK_er: IKEv2 encryption keys for encryption at the application client (SK_ei) and the device server (SK_er) to protect subsequent IKEv2 messages.

SK_pi and SK_pr: IKEv2 pseudo-random function keys that participate in the generation of the AUTH payloads. These keys cryptographically bind the authenticated identities to this cryptographic exchange.

If the application client selects the IKE_AUTH_NONE value (i.e., F9h) for the authentication algorithm type in the Key Exchange step, and the Key Exchange step completes without errors, the application client shall not perform the Authentication step. In this case, both the application client and the device server shall generate the SA as defined in 5.13.4.5.

The application client shall not select the IKE_AUTH_NONE value as an authentication algorithm type in the Key Exchange step unless:

It is present in the parameter data returned during the Device Server Capabilities step; and

The application client is configured to omit the Authentication step by an administrator.

NOTE – When IKE_AUTH_NONE is used, IKEv2-SCSI has no protection against any man-in-the-middle attacks. The following administrative decisions are security policy decisions that absence of authentication is acceptable, and should only be made with a full understanding of the security consequences of the lack of authentication:

enabling return of the IKE_AUTH_NONE authentication algorithm type in the Device Capabilities step, and

enabling the application client to select IKE_AUTH_NONE in the Key Exchange step

Such decisions should only be made in situations where active attacks on IKEv2-SCSI are not of concern (e.g., direct attachment of initiator and target, end-to-end secure transport channel such as IPsec for iSCSI).

5.13.4.4 IKEv2-SCSI Authentication step

5.13.4.4.1 Overview

The Authentication step performs the following functions:

a) authenticates both the application client and the device server;
b) protects the previous steps of the protocol; and
c) cryptographically binds the authentication and the previous steps to the created SA.

The Authentication step is accomplished as follows:

1) A SECURITY PROTOCOL OUT command (see 5.13.4.4.2); and
2) A SECURITY PROTOCOL IN command (see 5.13.4.4.3).

The parameter data for both commands shall be encrypted and integrity protected using the algorithms and keys determined in the Key Exchange step (see 5.13.4.3.4).

NOTE x5 - The Authentication step corresponds to the IKEv2 IKE_AUTH exchange in RFC 4306.

5.13.4.4.2 Authentication step SECURITY PROTOCOL OUT command

To send its authentication information to the device server, the application client sends a SECURITY PROTOCOL OUT command (see 6.30) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., xxh) and the SECURITY PROTOCOL SPECIFIC field set to 0103h. The parameter data consists of the IKEv2-SCSI header (see 7.7.3.3.1) and an Encrypted payload (see 7.7.4.11) that contains the following:

1) An Identification – Application Client (i.e., ID) payload (see 7.7.4.4);
2) Zero or more Certificate (i.e., CERT) payloads (see 7.7.4.5);
3) Zero or more Certificate Request (i.e., CERTREQ) payloads (see 7.7.4.5);
4) Zero or one Notify (i.e., N-IC), payload (see 7.7.4.8); and
5) An Authentication (i.e., AUTH) payload (see 7.7.4.6).

{{The relationship of the Certificate Request payload above to the Certificate payload in the Authentication step Security Protocol IN command needs to be clarified.}}

The device server shall decrypt and verify the integrity of the Encrypted payload using the algorithms and keys determined in the Key Exchange step (see 5.13.4.3.4).

The application client:

a) send its identity with the ID payload;
b) send knowledge of the secret corresponding to ID; and
c) integrity protect the prior steps using the AUTH Authentication payload.

The application client may send its Certificate(s) in Certificate payload(s) as described in RFC 4306. The application client may send a list of its trust anchors in Certificate Request payload(s) as described in RFC 4306. If any

Certificate payloads are included in the parameter data, the first Certificate payload shall contain the public key used to verify the Authentication payload.

The application client and device server may use different authentication methods. The use of Certificate and Certificate Request payloads may differ between the Authentication step SECURITY PROTOCOL OUT command and the Authentication step SECURITY PROTOCOL IN command.

{{Several rewrites applied in the next three paragraphs between r4 and r5}}

The application client uses the Notify payload to send an initial contact notification to the device server. The initial contact notification specifies that the application client has no stored state for any SAs with the device server other than the SA that is being created.

In response to receipt of an initial contact notification, the device server should delete all other SAs that were authenticated with a SECURITY PROTOCOL OUT command that contained the same Identification - Application Client payload data as that which is present in the SECURITY PROTOCOL OUT command that the device server is processing.

If the device server deletes other SAs in response to an initial contact notification, it shall do so only after the successful completion of the Authentication step (see 5.13.4.4). If an error occurs during the Authentication step, the device server shall ignore the initial contact notification.

If the device server is unable to proceed with SA creation for any reason (e.g., the verification of the Authentication payload fails), the SECURITY PROTOCOL OUT command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to an appropriate value. The additional sense code AUTHENTICATION FAILED shall be used when verification of the Authentication payload fails, or when authentication fails for any other reason.

{{AUTHENTICATION FAILED is a new additional sense code}}

5.13.4.4.3 Authentication step SECURITY PROTOCOL IN command

To obtain the device server's authentication information, the application client then sends a SECURITY PROTOCOL IN command (see 6.29) with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., xxh) and the SECURITY PROTOCOL SPECIFIC field set to 0103h. The parameter data consists of the IKEv2-SCSI header (see 7.7.3.3.1) and an Encrypted payload (see 7.7.4.11) that contains the following:

1) An Identification – Device Server (i.e., ID) payload (see 7.7.4.4);
2) Zero or more Certificate (i.e., CERT) payloads (see 7.7.4.5); and
3) A receiver Authentication (i.e., AUTH) payload (see 7.7.4.6).

The Encrypted payload shall apply encryption and integrity protection to all the payloads it contains using the algorithms and keys determined in the Key Exchange step (see 5.13.4.3.4).

The device server:

a) sends its identity with the ID payload;
b) authenticates its identity; and
c) protects the integrity of the prior step messages with the Authentication payload.

The device server may return its Certificate(s) in Certificate payload(s) as described in RFC 4306. If any Certificate payloads are included in the parameter data, the first Certificate payload shall contain the public key used to verify the Authentication payload.

{{The second sentence below needs a rewrite. What it's trying to say is that the application client and device server independently decide whether to use CERTREQ, what to put in it, and how to respond to CERTREQ or its absence.}}

The application client and device server may use different authentication methods. The use of Certificate and Certificate Request payloads may differ between the Authentication step SECURITY PROTOCOL OUT command and the Authentication step SECURITY PROTOCOL IN command.

After returning GOOD status for the SECURITY PROTOCOL IN command, the device server shall generate the SA as described in 5.13.4.5.

The application client should verify the Authentication payload as described in 7.7.4.6. The Certificate payload(s) are used as part of this verification for PKI-based authentication. If the Authentication payload is verified and no other error occurs the application client should generate the SA as described in 5.13.4.5.

If the application client is unable to proceed with SA creation for any reason (e.g., the verification of the AUTH payload fails), the application client should:

a) Not use the SA for any additional activities; and
b) Shall delete the SA pair as described in 5.13.6.

5.13.4.5 SA generation

The application client and the device server shall initialize the SA parameters (see 3.1.103) as follows:

a) AC_SAI shall be set to the value in the IKE_SA APPLICATION CLIENT SAI field in the IKEv2-SCSI header (see 7.7.3.3) in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.3.2);
b) DC_SAI shall be set to the value in the IKE_SA DEVICE SERVER SAI field in the IKEv2-SCSI header in the Key Exchange step SECURITY PROTOCOL IN command (see 5.13.4.3.3);
c) TIMEOUT shall be set to the IKEV2-SCSI SA INACTIVITY TIMEOUT field in the SCSI Timeout Values payload (see 7.7.4.14) in the Key Exchange step SECURITY PROTOCOL OUT command;
d) AC_SQN shall be set to one;
e) DC_SQN shall be set to one;
f) AC_NONCE shall be set to the value of the NONCE DATA field in the Nonce payload (see 7.7.4.7) in the Key Exchange step SECURITY PROTOCOL OUT command;
g) DS_NONCE shall be set to the value of the NONCE DATA field in the Nonce payload received from the device server in the Key Exchange step SECURITY PROTOCOL IN command;
h) KEY_SEED shall be set to the value of SK_d computed as part of Key Exchange step completion (see 5.13.4.3.4);
i) KDF_ID shall be set as described in 5.13.4.3.4;
j) KEYMAT should be set to zero;
k) USAGE_TYPE shall be set to the value in the SA TYPE field in the SCSI Cryptographic Algorithms payload (see 7.7.4.13) in the Key Exchange step SECURITY PROTOCOL OUT command;
l) USAGE_DATA shall contain at least the following values from of the usage data field in the SCSI Cryptographic Algorithms payload in the Key Exchange step SECURITY PROTOCOL OUT command:
   A) The USAGE DATA field;
   B) The ALGORITHM IDENTIFIER field (see 7.7.4.12.2) in the SCSI Cryptographic Algorithm descriptor (see 7.7.4.12.2) for the ENCR algorithm type;
   C) The KEY LENGTH field (see 7.7.4.12.2.2) in the ALGORITHM ATTRIBUTES field in the SCSI Cryptographic Algorithm descriptor for the ENCR algorithm type; and
   D) The ALGORITHM IDENTIFIER field (see 7.7.4.12.2) in the SCSI Cryptographic Algorithm descriptor for the INTEG algorithm type;

   and
m) MGMT_DATA shall contain at least the following values:

A) From the SCSI Cryptographic Algorithms payload (see 7.7.4.13) in the Key Exchange step SECURITY PROTOCOL OUT command:
   a) The ALGORITHM IDENTIFIER field (see 7.7.4.12.2) in the SCSI Cryptographic Algorithm descriptor (see 7.7.4.12.2) for the ENCR algorithm type;
   b) The KEY LENGTH field (see 7.7.4.12.2.2) in the ALGORITHM ATTRIBUTES field in the SCSI Cryptographic Algorithm descriptor for the ENCR algorithm type;
   c) The ALGORITHM IDENTIFIER field (see 7.7.4.12.2) in the SCSI Cryptographic Algorithm descriptor for the INTEG algorithm type;
B) From the Key Exchange step completion (see 5.13.4.3.4);
   a) The value of SK_ei; and
   b) The value of SK_ai;

   and
C) The next value of the message id field in the IKEv2-SCSI header.

NOTE x6 - The inclusion of the algorithm identifiers and key length in USAGE_DATA SA parameter enables the SA to apply the same encryption and integrity algorithms that IKEv2-SCSI negotiated to later uses.

5.13.4.6 IKEv2-SCSI CCS

{{I believe everything stated in the next paragraph is covered in the revised 5.13.4.1.}}

~~A new CCS shall be initiated by a Key Exchange step SECURITY PROTOCOL OUT command that is not terminated with CHECK CONDITION and the device server shall allocate a new device server SAI as part of processing that command. The application client shall issue the commands in an IKEv2-SCSI CCS in order on the same I_T_L Nexus. The IKEv2-SCSI protocol timeout in the STV payload of the Key Exchange step SECURITY PROTOCOL OUT command shall be the amount of time that the device server is required to wait for the next command in the IKEv2-SCSI CCS; the application client should ensure that this timeout does not expire before it issues that next command. The device server maintains information for SA creation (see 5.13.4.5) while an IKEv2-SCSI CCS is in progress.~~

{{Completing the work agreed upon in 07-226r1 discussion will cover everything from the next paragraph that really needs to be done.}}

~~The CCS is identified by the device server SAI and the application client SAI received in the Key Exchange step SECURITY PROTOCOL out command. The CCS for the Authentication step SECURITY PROTOCOL OUT command is determined by matching the SAIs in the IKEv2-SCSI header with these two SAIs. The two SECURITY PROTOCOL IN commands are associated with the CCS identified by the preceding SECURITY PROTOCOL OUT command on the same I_T_L nexus and shall return device server parameter data for that CCS; the application client shall check that the received parameter data is for the correct CCS. If any command that is part of a CCS is terminated with CHECK CONDITION status for any reason, that command shall not advance the CCS to expect the next command in the sequence.~~

{{Need to move the next paragraph to the AUTH OUT subclause.}}=WorkInProgress

The device server shall check the IKEv2-SCSI header and the integrity checksum data in the Encrypted payload of an Authentication phase SECURITY PROTOCOL OUT command before performing any checks of data contained in the Encrypted payload. If the contents of the IKEv2-SCSI header or the integrity checksum data in the Encrypted payload cause the Authentication phase SECURITY PROTOCOL OUT command to be terminated with CHECK CONDITION status, the terminated command shall have no effect on the CCS in progress. If the contents of the Encrypted payload cause the Authentication phase SECURITY PROTOCOL OUT command to be terminated with CHECK CONDITION status with a sense key other than NOT READY, the CCS shall be terminated and the SA shall not be generated.

{{I believe everything stated in the next paragraph is covered in the revised 5.13.4.1.}}

~~An IKEv2-SCSI CCS shall be terminated and the SA shall not be generated when the IKEv2-SCSI protocol timeout expires before the next command that is part of the CCS is received, and the device server discards the SA creation information in response to the timeout expiration. If multiple SECURITY PROTOCOL IN commands for the same phase are received, this timeout shall be measured from completion of the first instance of the command.~~

{{Incorporating the SECURITY PROTOCOL IN kindness feature describe in 07-226r1 will accomplish everything the following note intends to describe.}}

~~NOTE – Termination of a CCS in response to terminating a SECURITY PROTOCOL IN command with CHECK CONDITION status would create a security vulnerability due to the absence of security checks on CDB for the SECURITY PROTOCOL IN command. If the application client is unable to issue a SECURITY PROTOCOL IN command that is not terminated with CHECK CONDITION status, the CCS will time out, and this is the only means of terminating the CCS.~~

{{Whatever has not been done right in this revision regarding the next several paragraphs will be resolved by implementing the agreements reached during the 07-226r1 discussion.}}

{{It is not possible to detect a 'different IKEv2-SCSI CCS' especially for SECURITY PROTOCOL IN commands. Otherwise, I believe the key intentions of the following paragraph have been included in 5.13.4.1 as follows. The commands in an IKEv2-SCSI CCS are processed one at a time. Processing of the next command in the CCS cannot begin until the device server has completed processing of the previous command. The application client always knows which command is being processed, and therefore it knows the progress of the overall CCS. If the application client wants to know the progress of an individual command while the device server is busy processing that command, the application client may send a REQUEST SENSE command as per 5.13.7.}}

~~If any command that is part of an IKEv2-SCSI CCS is received on an I_T_L Nexus while a different IKEv2-SCSI CCS is in progress, that command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to IKEv2-SCSI OPERATION IN PROGRESS. The sense data should include sense key specific data for the NOT READY sense key that contains a progress indication field indicating the progress of the IKEv2-SCSI CCS (e.g., the IKEv2-SCSI CCS is 25% complete if one command out of four has been performed). 5.13.7 applies to any progress indication that reports progress information beyond the number of commands processed.~~

~~[Editor's Note: IKEv2-SCSI OPERATION IN PROGRESS is a new ASC/ASCQ - suggest 00h/1Eh]~~

{{I believe everything stated in the next paragraph is covered in the revised 5.13.4.1.}}

~~If an application client abandons an incomplete IKEv2-SCSI CCS, the protocol timeout in case a) above enables the device server to discard its SA creation information. The device server shall enforce a maximum value for the protocol timeout.~~

{{move this to the SCSI Timeout Values payload subclause.}}=WorkInProgress

NOTE: The maximum value for the protocol timeout should be long enough to allow the application client to continue the IKEv2-SCSI CCS, but short enough that if an incomplete IKEv2-SCSI CCS is abandoned, the device server will discard the state for that IKEv2-SCSI CCS and become available to for another IKEv2-SCSI CCS without excessive delay.

{{I believe the following paragraph is overtaken by the definition of how to delete an IKEv2-SCSI SA (see 5.13.6) and the various recommendations that application clients delete SAs they do not plan to use. I further believe it is impractical to ask the inactivity timeout to satisfy both this concern and the normal concerns of an SA user.}}

~~If an application client abandons a complete IKEv2-SCSI CCS (e.g., due to an Authentication failure at step 4) above or a parameter data error at step 2) above when the Authentication step is omitted, the device server will~~

~~create an SA that will never be used. This orphan SA can be removed via use of the SA inactivity timeout in the STV payload to detect that the SA is not being used (see 7.7.4.13). This is a consideration in selection of SA inactivity timeout values.~~

5.13.5 Abandoning an IKEv2-SCSI SA creation

{{This entire subclause is new and probably not fully integrated in other text. It is, however, referenced in the draft 7.7.3.3.1.}} {{The content of this subclause will need to be updated based on agreements reached during the disucssion of 07-226r1.}}

The occurrence of errors in either the application client or the device server may require that an IKEv2-SCSI CCS (see 3.1.c) be abandoned.

A device server shall indicate that it has abandoned an IKEv2-SCSI CCS by terminating the next IKEv2-SCSI CCS command it receives (see 5.13.4.1) with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense key set to any value except SA CREATION OPERATION IN PROGRESS.

An application client should not abandon an IKEv2-SCSI CCS when the next command in the CCS is a SECURITY PROTOCOL IN command. Instead, the application client should send the appropriate SECURITY PROTOCOL IN command and then abandon the IKEv2-SCSI CCS.

An application client should specify that it has abandoned an IKEv2-SCSI CSS by:

1) Sending an IKEv2-SCSI Delete command (see 5.13.6), and
2) If GOOD status is not returned in response to the IKEv2-SCSI Delete command, sending SECURITY PROTOCOL OUT command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., xxh) and the SECURITY PROTOCOL SPECIFIC field set to 0103h and the INTTR bit set to zero in the IKEv2-SCSI header (see 7.7.3.3.1).

5.13.6 Deleting an IKEv2-SCSI SA

When an SA is deleted, both sets of SA parameters (see 5.13.2.2) are deleted as follows:

1) The application client uses the information in its SA parameters to prepare an IKEv2-SCSI Delete command that requests deletion of the device server's SA parameters;
2) The application client deletes its SA parameters and any associated data;
3) The application client sends the IKEv2-SCSI Delete command prepared in step 1) to the device server;
4) In response to the IKEv2-SCSI Delete command, the device server deletes its SA parameters and any associated data.

The IKEv2-SCSI Delete command is a SECURITY PROTOCOL OUT command with the SECURITY PROTOCOL field set to IKEv2-SCSI (i.e., xxh) and the SECURITY PROTOCOL SPECIFIC field set to 0104h. The parameter data consists of the IKEv2-SCSI header (see 7.7.3.3.1) and an Encrypted payload (see 7.7.4.11) that contains the one Delete payload (see 7.7.4.9).

The Delete payload should conform to the requirements described in 7.7.4.9.

The Encrypted payload shall be encrypted and integrity checked using the MGMT_DATA SA parameter for the SA that is being deleted.

If the device server is able to valid the integrity checking information and decrypt the Encrypted payload, and locate the specified SA, it shall delete its SA parameters and any associated data. If the device server is unable to process the SECURITY PROTOCOL OUT command parameter list, the command shall be terminated with a CHECK CONDITION status, with the sense key and additional sense code set as described in 7.7.5.

5.13.7 Security progress indication

The cryptographic calculations required by some security protocols can consume a significant amount of time in the device server. If the device server receives a SECURITY PROTOCOL OUT command or SECURITY PROTOCOL IN command that it is unable to process because required calculations are not complete, then the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to LOGICAL UNIT NOT READY, OPERATION IN PROGRESS. The sense data should include sense key specific data for the NOT READY sense key (i.e., a PROGRESS INDICATION field indicating the progress of the device server in performing the necessary cryptographic calculations).

The device server shall not use the progress indication to report the detailed progress of cryptographic computations that may take a variable amount of time based on their inputs. The device server may use the progress indication to report synthetic progress that does not reveal the detailed progress of the computation (e.g., divide a constant expected time for the computation by 10 and advance the progress indication in 10% increments based solely on the time).

The requirements in this subclause apply to implementations of Diffie-Hellman computations and operations involving public or asymmetric keys (e.g., RSA) that optimize operations on large numbers based on the values of inputs (e.g., a computational step may be skipped when a bit or set of bits in an input is zero). A progress indication that advances based on the computation structure (e.g., count of computational steps) may reveal the time taken by content-dependent portions of the computation, and reveal information about the inputs.

When cryptographic calculations are in progress, the sense data specified in this subclause shall be returned in response to a REQUEST SENSE command.

{{New model subclause text ends here. Additions/deletions markups resume.}}

5.13.47 Security algorithm codes

…

6.29 SECURITY PROTOCOL IN command

~~6.29.1 SECURITY PROTOCOL IN command description~~

The SECURITY PROTOCOL IN command (see table 193) is used to retrieve security protocol information (see 6.29.2) or the results of one or more SECURITY PROTOCOL OUT commands (see 6.30).

Table 193 — SECURITY PROTOCOL IN command
{{no changes in table 193 contents}}

The SECURITY PROTOCOL field (see table 194) specifies which security protocol is being used.

Table 194 — SECURITY PROTOCOL field in SECURITY PROTOCOL IN command

Code	Description	Reference
00h	Security protocol information	~~6.29.2~~ 7.7.1
01h - 06h	Defined by the TCG	3.1.140
07h - 1Fh	Reserved	
20h	Tape Data Encryption	SSC-3
	{{insert two new rows (suggest 40h and 41h) and adjust reserved values accordingly}}	
zzh	SA Creation Capabilities	7.7.2
xxh	IKEv2-SCSI	7.7.3
21h - EDh	Reserved	
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE 1667
EFh	ATA Device Server Password Security	TBD
F0h - FFh	Vendor Specific	

Editors Note 1 - ROW: SECURITY PROTOCOL field code values 21h – 2Fh are tentatively reserved for SSC-x uses.

The contents of the SECURITY PROTOCOL SPECIFIC field depend on the protocol specified by the SECURITY PROTOCOL field (see table 194).

A 512 increment (INC_512) bit set to one specifies that the ALLOCATION LENGTH field (see 4.3.4.6) expresses the maximum number of bytes available to receive data in increments of 512 bytes (e.g., a value of one means 512 bytes, two means 1 024 bytes, etc.). Pad bytes may or may not be appended to meet this length. Pad bytes shall have a value of 00h. An INC_512 bit set to zero specifies that the ALLOCATION LENGTH field expresses the number of bytes to be transferred.

Indications of data overrun or underrun and the mechanism, if any, for processing retries depend on the protocol specified by the SECURITY PROTOCOL field (see table 194).

Any association between a previous SECURITY PROTOCOL OUT command and the data transferred by a SECURITY PROTOCOL IN command depends on the protocol specified by the SECURITY PROTOCOL field (see table 194). If the device server has no data to transfer (e.g., the results for any previous SECURITY PROTOCOL OUT commands are not yet available), the device server may transfer data indicating it has no other data to transfer.

The format of the data transferred depends on the protocol specified by the SECURITY PROTOCOL field (see table 194).

The device server shall retain data resulting from a SECURITY PROTOCOL OUT command, if any, until one of the following events is processed:

a) Transfer of the data via a SECURITY PROTOCOL IN command from the same I_T_L nexus as defined by the protocol specified by the SECURITY PROTOCOL field (see table 194);
b) Logical unit reset (See SAM-4); or

c) I_T nexus loss (See SAM-4) associated with the I_T nexus that sent the SECURITY PROTOCOL OUT command.

If the data is lost due to one of these events the application client may send a new SECURITY PROTOCOL OUT command to retry the operation.

~~6.29.2 Security protocol information description~~

~~6.29.2.1 Overview~~

~~...~~ {{Move the entire contents of 6.29.2 to 7.7.1.}} ~~...~~

~~6.29.2.4.3 Attribute certificate description~~

~~RFC 3281 defines the certificate syntax for certificates consistent with X.509v2 Attribute Certificate Specification. Any further restrictions beyond the requirements of RFC 3281 are yet to be defined by T10.~~

6.30 SECURITY PROTOCOL OUT command

The SECURITY PROTOCOL OUT command (see table 198) is used to send data to the logical unit. The data sent specifies one or more operations to be performed by the logical unit. The format and function of the operations depends on the contents of the SECURITY PROTOCOL field (see table 199). Depending on the protocol specified by the SECURITY PROTOCOL field, the application client may use the SECURITY PROTOCOL IN command (see 6.29) to retrieve data derived from these operations.

Table 198 — SECURITY PROTOCOL OUT command
{{no changes in table 198 contents}}

The SECURITY PROTOCOL field (see table 199) specifies which security protocol is being used.

Table 199 — SECURITY PROTOCOL field in SECURITY PROTOCOL OUT command

Code	Description	Reference
00h	Reserved	
01h - 06h	Defined by the TCG	3.1.140
07h - 1Fh	Reserved	
20h	Tape Data Encryption	SSC-3
	{{insert one new row (suggest 41h) and adjust reserved values accordingly}}	
xxh	IKEv2-SCSI	7.7.6
21h - EDh	Reserved	
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE 1667
EFh	ATA Device Server Password Security	TBD
F0h - FFh	Vendor Specific	

Editors Note 2 - ROW: SECURITY PROTOCOL field code values 21h – 2Fh are tentatively reserved for SSC-x uses.

The contents of the SECURITY PROTOCOL SPECIFIC field depend on the protocol specified by the SECURITY PROTOCOL field (see table 199).

A 512 increment (INC_512) bit set to one specifies that the TRANSFER LENGTH field (see 4.3.4.4) expresses the number of bytes to be transferred in increments of 512 bytes (e.g., a value of one means 512 bytes, two means 1 024 bytes, etc.). Pad bytes shall be appended as needed to meet this requirement. Pad bytes shall have a value of 00h. A INC_512 bit set to zero specifies that the TRANSFER LENGTH field indicates the number of bytes to be transferred.

Any association between a SECURITY PROTOCOL OUT command and a subsequent SECURITY PROTOCOL IN command depends on the protocol specified by the SECURITY PROTOCOL field (see table 199). Each protocol shall define whether:

a) The device server shall complete the command with GOOD status as soon as it determines the data has been correctly received. An indication that the data has been processed is obtained by sending a SECURITY PROTOCOL IN command and receiving the results in the associated data transfer; or
b) The device server shall complete the command with GOOD status only after the data has been successfully processed and an associated SECURITY PROTOCOL IN command is not required.

The format of the data transferred depends on the protocol specified by the SECURITY PROTOCOL field (see table 199).

…

7.7 Security protocol parameters

7.7.1 Security protocol information description

{{Move the entire contents of 6.29.2 here}}

7.7.2 SA creation capabilities

{{All text from here to the end of this proposal is new. Additions/deletions markups are not applied in these subclauses.}}

7.7.2.1 Overview

7.7.2.2 CDB description

7.7.2.3 IKEv2-SCSI Device Server Capabilities step parameter data format

{{Is IKEv2-SCSI header really not used with SCSI SA Creation Capabilities? ROW to verify to his satisfaction.}}

7.7.3 IKEv2-SCSI

7.7.3.1 Overview

7.7.3.2 CDB description

{{Reflect Annex C item a) in the description of the SECURITY PROTOCOL IN CDB and item b) in the description of the SECURITY PROTOCOL OUT CDB.}}

7.7.3.3 IKEv2-SCSI parameter data format

7.7.3.3.1 Overview

{{Need to verify correct SAI values in AUTH OUT command.}}=WorkInProgress

Table E1 shows the parameter data format used by a SECURITY PROTOCOL IN command and the parameter list format used a SECURITY PROTOCOL OUT command with a security protocol field set to IKEv2-SCSI (i.e., xxh).

{{insert table E1}}=WorkInProgress

The IKE_SA APPLICATION CLIENT SAI field contains the value that will become the AC_SAI SA parameter (see 5.13.2.2) when the SA is generated (see 5.13.4.5). The AC_SAI is chosen by the application client to uniquely identify its representation of the SA that is being negotiated. If the device server receives an IKEv2-SCSI header with the IKE_SA APPLICATION CLIENT SAI field set to zero, the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SA CREATION PARAMETER VALUE INVALID.

{{I believe the following IKE_SA APPLICATION CLIENT SAI field requirements are not necessary. See notes in 5.13.4.6.}}

~~The application client shall use this field to associate the parameter data in a SECURITY PROTOCOL IN COMMAND with the IKEv2-SCSI CCS in progress on the I_T_L nexus (see 5.13.4.6). If the application client cannot make this association, it shall abandon the CCS and shall not create an SA (see 5.13.4.6)~~

Except in the Key Exchange step SECURITY PROTOCOL OUT command (see 5.13.4.3.2), The IKE_SA DEVICE SERVER SAI field contains the value that will become the DC_SAI SA parameter when the SA is generated. The DC_SAI is chosen by the device server in accordance with the requirements in 5.13.2.1 to uniquely identify its representation of the SA that is being negotiated. In the Key Exchange step SECURITY PROTOCOL OUT command the IKE_SA DEVICE SERVER SAI field is reserved.

{{I believe the following IKE_SA DEVICE SERVER SAI field requirements are not necessary. See notes in 5.13.4.6.}}

~~The device server shall use this field to associate the Authentication step SECURITY PROTOCOL OUT command with the IKEv2-SCSI CCS in progress on the I_T_L nexus (see 5.13.4.6). If the device server cannot make this association, the command shall be terminated with CHECK CONDITION status, with the sense key set to NOT READY, and the additional sense code set to IKEv2-SCSI OPERATION IN PROGRESS, see 5.13.4.6.~~

The NEXT PAYLOAD field contains code (see table x1) that identifies the first IKE payload that follows the IKEv2-SCSI header.

Table x1 — NEXT PAYLOAD field

Code	IKE Payload Name	Support requirements in SECURITY PROTOCOL …		Reference
		IN	OUT	
00h	No Next Payload	Mandatory		7.7.4.2
01h - 20h		Reserved		
21h	Security Association	Reserved [a]		RFC 4306
22h	Key Exchange	Mandatory		7.7.4.3
23h	Identification – Application Client	Reserved	Mandatory	7.7.4.4
24h	Identification – Device Server	Mandatory	Reserved	7.7.4.4
25h	Certificate	Optional		7.7.4.5
26h	Certificate Request	Optional		7.7.4.5
27h	Authentication	Mandatory		7.7.4.6
28h	Nonce	Mandatory		7.7.4.7
29h	Notify [b]	Reserved	Mandatory	7.7.4.8
2Ah	Delete	Reserved	Mandatory	7.7.4.9
2Bh	Vendor ID	Mandatory		7.7.4.10
2Ch	Traffic Selector – Application Client	Reserved		RFC 4306
2Dh	Traffic Selector – Device Server	Reserved		RFC 4306
2Eh	Encrypted	Mandatory		7.7.4.11
2Fh	Configuration	Reserved		RFC 4306
30h	Extensible Authentication	Reserved		RFC 4306
31h - 7Fh		Restricted		RFC 4306
80h	SCSI SA Creation Capabilities	Mandatory		7.7.4.12
81h	SCSI Cryptographic Algorithms	Mandatory		7.7.4.13
82h	SCSI Timeout Values	Mandatory		7.7.4.14
83h - BFh		Reserved		
C0h - FFh	Vendor Specific			

[a] The Security Association payload type value is not used in IKEv2-SCSI. The SCSI Cryptographic Algorithms payload (i.e., 81h) is used instead.

[b] The Notify payload is used only to carry an Initial Contact notification. All other notifications defined in RFC 4306 are reserved.

The MAJOR VERSION field shall contain the value 2h. If a device server receives an IKE header with a MAJOR VERSION field containing a value other than 2h, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to SA CREATION PARAMETER VALUE INVALID.

The MINOR VERSION field is reserved.

The EXCHANGE TYPE field is reserved.

The initiator (INTTR) bit shall be set to:

a) One for SECURITY PROTOCOL OUT commands; and
b) Zero for SECURITY PROTOCOL IN commands.

If a device server receives an IKEv2-SCSI header with the INTTR bit set to zero, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to SA CREATION PARAMETER VALUE INVALID. If an application client receives an IKEv2-SCSI header with the INTTR bit set to one, it should abandon the SA being created (see 5.13.5).

The VERSION bit is reserved.

{{The remainder of this subclause has not yet been translated.}}

A response (rspns) bit set to one indicates that this parameter data is in response to a previous command with the same message id. A rspns bit set to zero indicates that this is parameter data is not associated with any previous message id of the same value. The RSPNS bit shall be set to zero for SECURITY PROTOCOL OUT commands and shall be set to one for SECURITY PROTOCOL IN commands.

The message id field contains an incrementing value that identifies a particular message (SECURITY PROTOCOL OUT command) and response (SECURITY PROTOCOL IN command) pair. The first message id in the Key Exchange step shall be zero. The application client shall increment the message id for each subsequent message. The device server shall respond with the same message id that the application client used in the initial command and shall set the RSPNS bit to one. Neither the application client nor the device server shall process an IKEv2-SCSI payload that contains a lower message id than the largest one previously seen (see RFC 4306).

If the device server receives a SECURITY PROTOCOL OUT command with an invalid message id field in its parameter data, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to IKEv2-SCSI PARAMETER VALUE INVALID.

If the application client receives an invalid message id field in the parameter data for a SECURITY PROTOCOL IN command, the application client shall abandon the CCS and shall not create an SA (see 5.13.4.6).

The length field shall contain the total number of bytes to be transferred for this IKEv2-SCSI message, including the header and all the IKE payloads.

The IKE payloads (variable) field contains one or more IKE payloads (see table F1).

7.7.4 IKE Payloads

7.7.4.1 Overview

7.7.4.2 No Next payload

7.7.4.3 Key Exchange payload

{{Reflect Annex C item s) in this subclause.}}

7.7.4.4 Identification payload

7.7.4.5 Certificate and Certificate Request payloads

7.7.4.6 Authentication payload

7.7.4.7 Nonce payload

{{Reflect Annex C item q) in this subclause.}}

7.7.4.8 Notify payload

7.7.4.9 Delete payload

7.7.4.10 Vendor ID payload

7.7.4.11 Encrypted payload

7.7.4.12 SCSI SA Creation Capabilities payload

7.7.4.12.1 Overview

7.7.4.12.2 SCSI Cryptographic Algorithms descriptor

7.7.4.12.2.1 Overview

7.7.4.12.2.2 Encryption Algorithm (ENCR) identifiers

7.7.4.12.2.3 Pseudo-random Function (PRF) identifiers

7.7.4.12.2.4 Integrity Algorithm (INTEG) identifiers

7.7.4.12.2.5 Diffie-Hellman Group (D-H) identifiers

7.7.4.12.2.6 IKE Authentication Algorithm Type (IKE-AUTH) identifiers

7.7.4.13 SCSI Cryptographic Algorithms payload

7.7.4.14 SCSI Timeout Values payload

7.7.5 Translating IKEv2 errors

{{The requirements in this subclause are considerably more specific than in r4.}}

IKEv2 (see RFC 4306) defines an error reporting mechanism based on the Notify payload. This standard translates such error reports into the device server and application actions defined in this subclause.

If a device server is required by IKEv2 to report an error using a Notify payload, the device server shall translate error into a CHECK CONDITION status with the sense key and additional sense code shown in table x2. The device server shall terminate the SECURITY PROTOCOL OUT command (see 6.30) that transferred the parameter list in which the IKE requirements for one or more payloads (see 7.7.4) require use of the Notify payload to report an error. The SECURITY PROTOCOL OUT command shall be terminated as described in this subclause. The device server shall not report the IKE errors described in this subclause by terminating a SECURITY PROTOCOL IN command (see 6.29) with a CHECK CONDITION status.

Table x2 — IKEv2 Notify payload error translations for IKEv2-SCSI

IKEv2 (see RFC 4306)		**IKEv2-SCSI**	
Error Type	**Description**	**Additional sense code**	**Sense key**
0000h	Reserved		
0001h	UNSUPPORTED_CRITICAL_ PAYLOAD	SA CREATION PARAMETER NOT SUPPORTED	ILLEGAL REQUEST
0004h	INVALID_IKE_SPI	SA CREATION PARAMETER VALUE INVALID	
0005h	INVALID_MAJOR_VERSION		
0007h	INVALID_SYNTAX [a]		
0009h	INVALID_MESSAGE_ID		
000Bh	INVALID_SPI	SA CREATION PARAMETER VALUE INVALID [b]	
000Eh	NO_PROPOSAL_CHOSEN k)	SA CREATION PARAMETER VALUE INVALID	
0011h	INVALID_KE_PAYLOAD k)		
0018h	AUTHENTICATION_FAILED	AUTHENTICATION FAILED	ABORTED COMMAND
0022h - 0027h	See RFC 4306 [d]	n/a	n/a
2000h - 3FFFh	Vendor Specific		
All others	Restricted		

[a] This sense key and additional sense code shall be returned for a syntax error within an Encrypted payload (see 7.7.4.11) regardless of IKEv2 requirements to the contrary.

[b] SA CREATION PARAMETER VALUE INVALID shall be used for an invalid SAID in an IKEv2-SCSI SECURITY PROTOCOL IN or SECURITY PROTOCOL OUT. The additional sense code for an invalid SAID in all other commands is specified by the appropriate command standard (see 3.1.17).

[c] An application client recovers by restarting processing with the Device Capabilities step (see 5.13.4.2) to rediscover the device server's capabilities.

[d] These IKEv2 Error Types are used for features that are not supported by IKEv2-SCSI SA creation.

{{All additional sense codes in table x2 are new.}}

If an application client detects an IKEv2 error that RFC 4306 requires to be reported with a Notify payload, the application client should notify the device server by deleting the SA (see 5.13.6).

7.7.6 IKEv2-SCSI SECURITY PROTOCOL OUT parameters

7.7.6.1 Overview

7.7.6.2 CDB description

Annex C

(Informative)

IKEv2 protocol details and variations for IKEv2-SCSI

{{All of Annex C is new. Additions/deletions markups are not applied in this annex.}}

The IKEv2 protocol details and variations specified in RFC 4306 apply to IKEv2-SCSI (i.e., this standard) as follows:

a) Any SECURITY PROTOCOL IN command with an allocation length of up to 16 384 bytes is not terminated with an error due to the number of bytes to be transferred;
b) Any SECURITY PROTOCOL OUT command with a transfer length of up to 16 384 bytes is not terminated with an error due to the number of bytes transferred;
c) The timeout and retransmission mechanisms defined in RFC 4306 are not used by this standard (i.e., retransmission is performed by the applicable SCSI transport protocol);
d) Each SCSI command used by this standard completes by conveying a status from the device server to the application client;
e) This standard uses the MESSAGE ID field for sequencing (see RFC 4306), but only for SA creation (see 7.7.3.1);
f) The IKEv2 header EXCHANGE TYPE field is reserved in this standard because equivalent information is transferred in the SECURITY PROTOCOL OUT command and SECURITY PROTOCOL IN command CDBs; {{very new text}}
g) The IKEv2 header VERSION bit is reserved in this standard; {{very new text}}
h) This standard uses the pseudo-random functions (PRF) functions defined by RFC 4306; {{very new text}}
i) The key derivation functions defined and used by this standard (see 5.13.3) are equivalent to the PRF+ found in RFC 4306; {{very new text}}
j) The SA creation transactions (see 3.1.r) defined by this standard are not overlapped. If an application client attempts to start a second SA creation transaction before the first is completed, the offending command is terminated with CHECK CONDITION status (see 5.13.4.1), but this does not affect the SA creation transaction that is already in progress;
k) The NO_PROPOSAL_CHOSEN and INVALID_KE_PAYLOAD notify error types are replaced by the SA CREATION PARAMETER VALUE INVALID additional sense code (see 7.7.5) because IKEv2-SCSI has a different negotiation structure . As defined in RFC 4306, an IKEv2 initiator shall offer one or more proposals to a responder without knowing what is acceptable to the responder, and shall likewise choose a DH group without knowing whether it is acceptable to the responder. These two notify error types allow the responder to inform the initiator that one or more of its choices are not acceptable. In contrast, an IKEv2-SCSI application client obtains the device server capabilities in the Device Capabilities step (see 5.13.4.2) and selects algorithms from them in the Key Exchange step (see 5.13.4.3). An error can only occur if the application client has made an invalid selection, hence the SA CREATION PARAMETER VALUE INVALID description; {{very new text}}
l) IKEv2 version numbers (see RFC 4306) are used by this standard (see 7.7.3.3.1), but the ability to respond to an unsupported version number with the highest version number to be used is not supported, and this standard does not include checks for version downgrade attacks;
m) IKEv2 cookies (see RFC 4306) are not used by this standard;
n) IKEv2 cryptographic algorithm negotiation (see RFC 4306) is replaced by the Device Server Capabilities step (see 5.13.4.2) and the Key Exchange step (see 5.13.4.3) (i.e., the IKEv2 proposal construct is not used by this standard);
o) In this standard an SA is rekeyed by replacing it with a new SA:
    A) CHILD_SAs are not used by this standard;
    B) The RFC 4306 discussion of CHILD_SAs does not apply to this standard;
    C) Coexistence of the original SA and the new SA that is achieved for rekeying purposes by restricting the device server's ability to delete SAs to the following cases: {{very new text}}
        a) Expiration of a timeout (see 7.7.4.14);

        b) Processing of an IKEv2-SCSI Delete function (see 5.13.6); and
        c) Responding to an initial contact notification (see 7.7.4.8);
        and
    D) IKEv2 does not support rekeying notification for IKE_SAs, therefore this standard does not support rekeying notification;
p) Traffic Selectors (see RFC 4306) are not used by this standard;
q) The requirements in RFC 4306 on nonces are be followed for the random nonces (see 3.1.95) defined by this standard;
r) The RFC 4306 requirements on address and port agility are specific to the user datagram protocol and the IP protocol and do not apply to this standard;
s) This standard allows Diffie-Hellman exponential reuse and reuse of analogous Diffie-Hellman public values for Diffie-Hellman mechanisms not based on exponentiation as specified in RFC 4306. The freshness and randomness of the random nonces are critical to the security of IKEv2-SCSI when Diffie-Hellman exponentials and public values are reused (see RFC 4306);
t) Keys for the Authentication step are generated as specified in RFC 4306;
u) This standard uses a slightly modified version of the authentication calculations in RFC 4306 (see 7.7.4.6);
v) The RFC 4306 sections that describe the following features are not used by this standard:
    A) Extensible authentication protocol methods;
    B) Generating keying Material for CHILD_SAs;
    C) Rekeying an IKE SA using CREATE_CHILD_SA;
    D) Requesting an internal address;
    E) Requesting the peer's version;
    F) IPComp;
    G) NAT traversal; and
    H) Explicit congestion notification;
    and
w) IKEv2 Error Handling (see RFC 4306) is replaced by the use of CHECK CONDITION status and sense data by this standard. See 7.7.5 for details of how errors reported in the Notify payload are translated to sense data.

{{Table C.1 is new to r5 and replaces a column in the r4 version of table x1.}}

Where this standard uses IKE payload names (see 7.7.3.3) RFC 4306 uses the shorthand notation shown in table C.1.

Table C.1 — IKE payload names shorthand

IKE payload name in this standard [a]	**RFC 4306 shorthand** [b]
Security Association	SAi or SAr
Key Exchange	KEi or KEr
Identification – Application Client	IDi
Identification – Device Server	IDr
Certificate	CERTi or CERTr
Certificate Request	CERTREQi
Authentication	AUTHi or AUTHr
Nonce	NONCEi or NONCEr
Notify	N-ICi or N-ICr
Delete	Di
Vendor ID	Vi or Vr
Traffic Selector – Application Client	TSi
Traffic Selector – Device Server	TSr
Encrypted	Ei or Er
Configuration	CPi or CPr
Extensible Authentication	EAPi or EAPr

[a] To facilitate future enhancements, all IKE payloads are listed in this table, but not all entries in this table are used in this standard.

[b] In RFC 4306 the lowercase i indicates initiator and r indicates receiver. In this standard, the initiator is the application client and all such IKE payloads (e.g., KEi) appear in a SECURITY PROTOCOL OUT parameter list. The receiver is always the device server in this standard and all such IKE payloads (e.g., AUTHr) appear in SECURITY PROTOCOL IN parameter data.

{{The proposed SPC-4 changes end here.}}

Summary of new additional sense codes

recommend 00h/1Eh for — SA CREATION OPERATION IN PROGRESS
recommend 74h/30h for — SA CREATION PARAMETER NOT SUPPORTED
recommend 74h/10h for — SA CREATION PARAMETER VALUE INVALID
recommend 74h/40h for — AUTHENTICATION FAILED