To: INCITS T10 Committee
From: Matt Ball, Quantum Corporation
Date: 4 May 2006
Subject: SSC-3: Using NIST AES Key-Wrap for Key Establishment

Revision History

Revision 0 (T10/06-225r0):
Posted to the T10 web site on 4 May 2006.

Related Documents

T10/SSC-3 r2 "SCSI Stream Commands 3"
T10/06-172r1 (Paul Entzel, Quantum Corp.) "Add commands to control data encryption"
T10/06-103r1 (David Black, EMC) "Encrypt keys for transfer to device"
NIST AES Key-Wrap (Draft, November 2001)
NIST FIPS 140-2 "Security Requirements for Cryptographic Modules"
NIST FIPS 140-2, Annex D "Approved Key Establishment Techniques"
NIST FIPS 140-2 IG "Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program"
NIST FIPS 180-2 “Secure Hash Standard (SHS)”
NIST FIPS 197 "Announcing the Advanced Encryption Standard (AES)"
NIST SP 800-38B "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication"
NIST SP 800-56A "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"
NIST SP 800-90 "Recommendation for Random Number Generation Using Deterministic Random Bit Generators"
IEEE 802.11i "Amendment 6: Medium Access Control (MAC) Security Enhancements"
IETF RFC 4303 "IP Encapsulating Security Payload (ESP)"
IETF RFC 4306 “Internet Key Exchange (IKEv2) Protocol”

General

NOTE: This is a proposal against SSC-3, with modifications from T10/06-172r1.

The purpose of this proposal is to provide a way for the application client to pass an encrypted key to the device server (note: this is different than passing an *encryption* key; the goal is to encrypt the encryption key). This feature has some of the following benefits:

- It is possible to hide the encryption key from an eavesdropper. Generally speaking, the encryption key is more valuable than the data, so it is important to keep this information safe.
- To comply with NIST FIPS 140-2, it is necessary to encrypt the key before passing it to the device server. Currently, there is no method in SSC-3 to provide a FIPS-compliant solution.

Currently, there are two FIPS 140-2 approved methods for key establishment (see <http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexd.pdf>):

- Key Management using ANSI X9.17 (NIST FIPS 171)
- NIST AES Key-Wrap (See <http://csrc.nist.gov/CryptoToolkit/kms/AES_key_wrap.pdf>)

The first method, FIPS 171, was withdrawn by NIST on February 8, 2005. This pretty much leaves us with 'AES Key-Wrap'. Accordingly, this proposal provides a method for using AES Key-Wrap to establish a key within a device server.

- The AES Key-Wrap algorithm requires the use of a shared *session key* for wrapping the key. David Black proposed in T10/06-103r1 an outline for using Diffie-Hellman for creating such a session key. This proposal does not duplicate this work, but instead assumes that a *security association* (SA) is already established.

This proposal borrows several concepts from the IETF, IEEE, and NIST. The message format is very similar to that of RFC 4303, "IP Encapsulating Security Payload (ESP)". The *key derivation function* (KDF) is taken from a recent standard by NIST (SP 800-56A). The reader is encouraged to become familiarized with these documents and other from the ‘Related Documents’ section.

Proposed Changes

Editor's Note: These proposed changes apply to SSC-3 r2 (2006/03/01) as updated by T10/06-172r2. Some headings are used only to keep the numbering consistent.

1 Scope

2 Normative references

3 Definitions, acronyms, keywords, and conventions

3.1 Definitions

3.1.a **data at rest:** Data that is stored on a storage medium.

3.1.b **data in flight:** Data that passes across the interface bus.

3.1.c **key derivation function:** a cryptographic function used to create derived keys based on a set of shared secret and public parameters.

3.1.d **integrity check value (ICV):** a message authentication code used to validate the integrity of data passed across the interface. This is the term used by the IETF.

3.1.e **security association (SA)**: A set of parameters that defines a secure channel between an application client and device server (see 4.2.20.2).

3.1.f **secure channel:** An encrypted communication channel between the application client and device server that uses a set of shared secret keys.

3.1.g **security parameters index (SPI):** A 32-bit value that uniquely specifies a particular security association (see 4.2.20.3).

3.1.h **sequence number:** An integer used to order messages send over a secure channel (see 4.2.20.4).

3.2 Acronyms

AES	Advanced Encryption Standard
FIPS	Federal Information Processing Standard
ICV	integrity check value
IETF	Internet Engineering Task Force
NIST	National Institute of Standards and Technology
RFC	request for comment
SA	security association
SP	special publication (from NIST)
SPI	security parameters index

4 General Concepts

4.1 Overview

4.2 Sequential-access device model

4.2.19 Data encryption

Editor's note: All the following sections are new, up to clause 5

4.2.20 Encryption of information across the interface

4.2.20.1 Overview

This section describes cryptographic concepts for encrypting information over the interface, also known as data in flight. None of the concepts in this section refer to data stored on the media, also called data at rest (see 4.2.19).

4.2.20.2 Security association

An application client and device server may establish a security association (SA) for passing encrypted information across the interface. An SA is a set of parameters that control the passing of secure information between the application client and device server. These parameters include the following:

a) A security parameters index (SPIi) generated by the application client (see 4.2.20.3).

b) A security parameters index (SPIt) generated by the device server (see 4.2.20.3).

c) A random nonce (Ni) generated by the application client.

d) A random nonce (Nt) generated by the device server.

e) The current sequence number for messages sent from the application client to the device server (see 4.2.20.4).

f) The current sequence number for messages sent from the device server to the application client (see 4.2.20.4).

g) A shared key seed, called SKEYSEED. This is a special encryption key known only to the application client and device server of this SA. By using the SKEYSEED, two SPI values, and two nonce values, it is possible to generate keying material for this SA (See 4.2.20.5).

h) An optional I_T nexus describing which application client and device server is associated with this SA.

i) A version number. For this standard, the version number shall be 1.

j) A negotiated key derivation function (KDF). For this version, the only allowed KDF is specified by NIST SP 800-56A (see 4.2.20.5).

An application client or device server maintains a list of security associations internally in an SA database. This database may contain any number of SA entries.

An application client or device server shall not pass in plaintext any shared secrets, such as the SKEYSEED or other derived shared keys, over the interface.

An application client or device server may destroy an SA at any time for any reason. In this event, it is necessary to establish a new SA before sending encrypted messages across the interface. If an I_T nexus is associated with an SA that is destroyed, the device server shall establish a Unit Attention condition for the I_T Nexus with the additional sense set to SECURITY ASSOCIATION DESTROYED.

Editor's Note: SECURITY ASSOCIATION DESTROYED is a new ASQ

If a SA is destroyed, all the data within the SA structure shall be internally cleared.

4.2.20.3 Security parameters index

A security parameters index (SPI) is a 32-bit number that uniquely describes an SA. Both the application client and device server are responsible for establishing their SPI and ensuring its uniqueness. As part of the protocol for establishing an SA, both the application client and device server shall create their own SPI that is unique within their context, and shall pass this SPI to the other participant so that both sides contain the same information within the SA structure. Any encrypted message that uses a secure channel shall include the SPI of the receiver to identify the SA. The SPI shall be between 256 and 2^{32}-1, inclusive.

If a recipient receives a command with an invalid SPI, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID SECURITY PARAMETERS INDEX. An SPI is invalid if it does not exist in the recipient's SA database

Editor's Note: INVALID SECURITY PARAMETERS INDEX is a new ASQ

4.2.20.4 Sequence number

A sequence number is an unsigned 32-bit integer that is used for sequencing messages. The main purpose of a sequence number is to prevent a replay attack, in which an attacker sends an old message as a new message. Both the application client and the device server shall maintain a sequence number for each message in each direction. After establishing a new SA, the application client and device server shall set the two sequence numbers to zero. Before sending a message, the sender shall increment the sequence number. Therefore, the first valid sequence number in a message is one. The receiver of a message shall verify that the sequence number is larger than that of the previous message. If this is not true, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID SEQUENCE NUMBER.

Editor's Note: INVALID SEQUENCE NUMBER is a new ASQ.

A sequence number is not allowed to wrap. The last valid sequence number is 2^{32}-1, after which it is necessary to establish a new SA before sending any more encrypted messages.

4.2.20.5 Generating keying material for the SA

All secure messages associated with a given SA shall use derived secret shared keys for encrypting and validating data across the interface bus. These derived secret keys are a function of a shared key-seed (SKEYSEED), the SPI values, and the nonce values. The following function shows the method for generating all the shared keys:

{SK_d || SK_ai || SK_at || SK_ei || SK_et} = kdf(SKEYSEED, SPIi || Ni || SPIt || Nt)

where

|| is the concatenation operator, which consecutively joins data in big-endian order

SK_d is the derived key used to generate a new SKEYSEED during a re-keying operation

SK_ai is the shared key used for authenticating messages from the application client to the device server

SK_at is the shared key used for authenticating messages from the device server to the application client

SK_ei is the shared key used for encrypting messages from the application client to the device server

SK_et is the shared key used for encrypting messages from the device server to the application client

kdf is a cryptographically secure 'key derivation function' (see below)

SKEYSEED is the shared key seed for this SA

Ni, Nr are the nonce values from the application client and device server, respectively

SPIi, SPIt are the SPI values from the application client and device server, respectively

For reference, here are the mnemonics for the SK_ abbreviations:

a) d = derived

b) a = authentication (integrity check value)

c) e = encryption

d) i = initiator (application client)

e) t = target (device server)

For this version, the following specifications apply (subsequent versions may allow additional parameters):

a) All keys shall be 256 bits in length. This includes the various SK_ shared keys and the SKEYSEED.

b) The key derivation function (kdf) shall be specified by the document NIST SP 800-56A under the section "5.8.1 Concatenation Key Derivation Function (Approved Alternative 1)". This function shall be based on the SHA-256 cryptographic hash function, as specified by NIST FIPS 180-2 "Secure Hash Standard". The output length of the kdf shall be 1280 bits (=5*256, representing all five derived keys).

c) All nonce values shall be 128 bits in length. These values should be created using a cryptographically secure pseudo-random number generator. In particular, if the same random number generator is used to create both the nonce values and encryption keys, knowledge of the nonce shall not compromise the security of the encryption key.

d) Each SPI value shall be 32 bits in length and shall be between 256 and 2^{32}-1, inclusive. The value zero is reserved to mean "no SA", and the values 1 to 255 are reserved for future use.

A device need only compute the shared keys that are required for a particular algorithm. For example, to implement the NIST AES Key-Wrap algorithm, it is only necessary to compute SK_ai and SK_ei.

5 Explicit address command descriptions for sequential-access devices

6 Implicit address command descriptions for sequential-access devices

7 Common command descriptions for sequential-access devices

8 Parameters for sequential-access devices

8.1 Diagnostic parameters

8.2 Log Parameters

8.3 Mode Parameters

8.4 Vital product data (VPD) parameters

8.5 Security protocol parameters

8.5.1 Security protocol overview

8.5.2 SECURITY PROTOCOL IN command specifying Tape Data Encryption security protocol

8.5.3 SECURITY PROTOCOL OUT command specifying Tape Data Encryption security protocol

8.5.3.1 SECURITY PROTOCOL OUT command specifying Tape Data Encryption security protocol overview

8.5.3.2 Set Data Encryption page

Table Y1 shows the parameter list format of the Set Data Encryption page. (new changes are in **blue**)

Table Y1 – Set Data Encryption page

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0010h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (m-3)						
3								(LSB)
4	SCOPE			Reserved				LOCK
5	Reserved					CKOD	CKORP	CKORL
6	ENCRYPTION MODE							
7	DECRYPTION MODE							
8	ALGORITHM INDEX							
9	KEY FORMAT							
10	SECURITY PARAMETERS INDEX (SPI)							
13								
14	SEQUENCE NUMBER							
17								
18	(MSB)	KEY LENGTH (n-19)						
19								(LSB)
20	KEY							
n								
n+1	KEY-ASSOCIATED DATA DESCRIPTORS LIST							
m								

(Text removed)

The KEY FORMAT field indicates the format of the value in the KEY field. Values for this field are described in table Y5.

Table Y5 – KEY FORMAT field values

Code	Description
00h	The KEY field contains the key to be used to encrypt or decrypt data.
01h	The KEY field contains a vendor specific key reference
02h	NIST AES Key-Wrap
02h – BFh	Reserved
C0h – FFh	Vendor specific

If the KEY FORMAT field is 00h the KEY field contains the key in an algorithm specific format. Table Y8 defines the format of the key in the KEY field. Both the SPI and SEQUENCE NUMBER fields shall be zero for this key format.

Table Y8 – KEY field contents with KEY FORMAT field set to 00h

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)			key				
n								(LSB)

The KEY LENGTH field indicates the length of the key field in bytes. Certain key formats require that the KEY LENGTH have a particular alignment. If the application client violates this alignment, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID KEY LENGTH ALIGNMENT.

Editor's note: INVALID KEY LENGTH ALIGNMENT is a new sense code.

If the KEY FORMAT field is 01h, the KEY field shall contain 8 bytes of T10 vendor identification (see SPC-4) followed immediately by a vendor specific key reference identifying the key to be used to encrypt or decrypt data. If the KEY field contains a vendor specific key reference that is unknown to the device server, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to VENDOR SPECIFIC KEY REFERENCE NOT FOUND.

The SECURITY PARAMETERS INDEX (SPI) field is the 32-bit number that identifies the security association (SA) used for encrypting the key, assuming the key is encrypted. If the key is not encrypted, the SPI field shall be zero. See 4.2.20.3 for more details.

The SEQUENCE NUMBER field is a 32-bit integer used for preventing replay attacks. If the key format does not use an SA, then this number shall be zero. Otherwise, the requirements from 4.2.20.4 shall apply.

If the KEY FORMAT field of Table Y1 is 02h, the KEY field shall contain a wrapped key, as specified by the NIST document 'AES Key Wrap Specification (16 Nov 2001)' (AES Key-wrap). The SPI field shall reference the security association (SA) and corresponding shared keys SK_ei and SK_ai. Each of these shared keys shall contain exactly 256 bits. The value SK_ei shall provide the Key Encryption Key (KEK) for the AES Key-Wrap algorithm. After performing the AES key-wrap algorithm, the wrapped-key shall expand in size by 8 bytes. These extra 8 bytes provide an integrity check for the unwrapping process. Table Y9 shows the layout of the KEY field for KEY FORMAT 02h.

Table Y9 – KEY field contents with KEY FORMAT field set to 02h

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)	AES KEY-WRAPPED KEY						
n+7								(LSB)
n+8	(MSB)	INTEGRITY CHECK VALUE (ICV)						
n+23								(LSB)

The device server shall verify the sequence number and report an error if a non-increasing number is detected (see 4.2.20.4).

For KEY FORMAT 02h, The KEY field of Table Y1 shall also contain a 128-bit integrity check value (ICV) immediately following the wrapped key. The ICV shall be computed over the SPI, SEQUENCE NUMBER, KEY LENGTH, and KEY fields using the CMAC authentication mode, as specified in NIST SP 800-38B. The device server shall invoke the CMAC algorithm using the AES block cipher with a 256-bit key provided by SK_ai (see 4.2.20.5).

The device server shall validate the ICV and report an error if the integrity-check fails. If the device server is unable to validate the ICV, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID INTEGRITY CHECK VALUE

The device server shall ensure that the KEY LENGTH is a multiple of 8 bytes, as specified by the AES Key-wrap specification (see KEY LENGTH description, above). The device server shall also validate the integrity of the wrapped key. If the device server is unable to validate the key, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID INTEGRITY CHECK VALUE.

Editor's note: INVALID INTEGRITY CHECK VALUE is a new sense code.

The AES Key-wrap and ICV algorithms expand the key size by 24 bytes. Therefore, the final unwrapped key size shall be equal to the KEY LENGTH field minus 24 bytes.