

To: INCITS T10 Committee
From: Matt Ball, Quantum Corporation
Date: 6 June 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.

Revision 1 (T10/06-225r1) 18 May 2006:

Changed SPI to SAI.

Added definition for 'encrypted information packet'.

Removed 'Secure Channel'.

Added mention of ANS X9.102 as a standard that could replace AES Key Wrap.

Within **Table Y1 – Set Data Encryption page**, moved the SAI (formerly SPI) and SEQUENCE NUMBER fields into the KEY field and left those as 'reserved' (as they were previously). Table Y1 is now unchanged.

Updated **Table Y9 – key field contents with key format field set to 02h** to include SAI and SEQUENCE NUMBER fields.

Minor edits.

Revision 2 (T10/06-225r2) 6 June 2006:

Moved the security association parameters into a table.

Created additional level 5 subclauses for describing the various key formats.

Added changes to the references section

Added clauses for these topics: Nonces and Key derivation functions

Changed the layout for the 'key reference' format

Changed key-wrap reference to RFC 3394 instead of "AES Key-Wrap (Draft)"

Added table for assigning an ID to the key derivation functions (KDF).

Numerous rewordings.

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"

T11 FC-SP/06-157v2 "Fibre Channel Security Protocols (FC-SP)"

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 800-57 "Recommendation on Key Management"

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 3394 "Advanced Encryption Standard (AES) Key Wrap Algorithm"

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.

Note: NIST is working on an updated version for the 'AES Key Wrap Specification'. This updated key-wrap algorithm will also be included in the ANS X9.102 standard. You can find a draft copy at <http://eprint.iacr.org/2004/340.pdf>. We may consider referencing this new standard if it looks like it will be FIPS 140-3 Approved (FIPS 140-3 will replace 140-2 and should be out this year).

If we need a standards number for Key Wrap, it would be possible to refer to RFC 3394 "Advanced Encryption Standard (AES) Key Wrap Algorithm" (although I'm a little hesitant to use a standard with the name "Request for Comment (RFC)" in the title).

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. ([Proposed changes are in blue](#))

1 Scope

2 Normative references

2.1 Normative references

2.2 Approved references

2.3 References under development

2.4 IETF references

Copies of the following approved IETF standards may be obtained through the Internet Engineering Task Force (IETF) at the World Wide Web site <http://www.ietf.org>.

[RFC 3394, Advanced Encryption Standard \(AES\) Key Wrap Algorithm](#)
[RFC 4303, IP Encapsulating Security Payload \(ESP\)](#)
[RFC 4306, Internet Key Exchange \(IKEv2\) Protocol](#)

2.5 NIST references

Copies of the following approved NIST standards may be obtained through the National Institute of Standards and Technology (NIST) at the World Wide Web site <<http://csrc.nist.gov>>.

NIST AES Key-Wrap (Draft, November 2001)

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*

3 Definitions, acronyms, keywords, and conventions

3.1 Definitions

3.1.a ciphertext: Data that has been encrypted

3.1.b plaintext: Data that is not encrypted

3.1.c encapsulating security payload (ESP): A protocol for sending encrypted and authenticated information that is specified by IETF RFC 4303.

3.1.d encrypted information packet: A packet of encrypted data that passes across the service delivery subsystem and minimally includes an SAI, sequence number, and ICV.

3.1.e key derivation function (KDF): a cryptographic function used to create derived keys based on a set of shared secret and public parameters (see 4.2.20.7).

3.1.f key encryption key (KEK): a key used for encrypting another key.

3.1.g integrity check value (ICV): a message authentication code used to validate the integrity of data passed across the service delivery subsystem. This is the term used by the IETF.

3.1.h security association (SA): A set of parameters that defines a relationship and associated security processing between an application client and device server (see 4.2.20.2).

3.1.i security association identifier (SAI): A 32-bit value that uniquely specifies a particular security association (see 4.2.20.3). This is similar to the IETF's security parameters index (SPI).

3.1.j sequence number (SN): An integer used to order encrypted information packets sent using a security association (see 4.2.20.5).

3.1.k shared key: a symmetric cryptographic key that is shared between a particular application client and device server, and that is not known to any other entity. A shared key is used for encrypting and authenticating encrypted information packets.

3.2 Acronyms

	Concatenation operator
AES	Advanced Encryption Standard
ESP	encapsulating security payload
FIPS	Federal Information Processing Standard
ICV	integrity check value
IETF	Internet Engineering Task Force
KEK	key encryption key
NIST	National Institute of Standards and Technology
RFC	(IETF) request for comment
SA	security association
SP	(NIST) special publication
SAI	security association identifier

4 General Concepts

4.1 Overview

4.2 Sequential-access device model

4.2.19 Data encryption

4.2.20 Encryption of information across the service delivery subsystem

4.2.20.1 Overview

The following subclauses describe cryptographic concepts needed for passing encrypted information packets over the service delivery subsystem. None of the concepts in the following subclauses refer to data stored on the media (see 4.2.19 for those concepts).

NOTE - Many of these concepts are derived from the IETF's encapsulating security payload (ESP) protocol (see IETF RFC 4303 and RFC 4306).

4.2.20.2 Security association

An application client and device server may establish a security association (SA) for passing encrypted information packets (see 3.1.d). An SA is a set of parameters that control the passing of encrypted information packets between the application client and device server. Table 4.1 shows the information that defines an SA.

Table 4.1 – Information that defines a security association

Name	Size (bytes)	Scope	Description	Reference
SAIi	4	public	A security association identifier generated by the application client	4.2.20.3
SAIt	4	public	A security association identifier generated by the device server	4.2.20.3
Ni	16	public	A random nonce generated by the application client	4.2.20.4
Nt	16	public	A random nonce generated by the device server	4.2.20.4
SNi	4	public	The current sequence number for encrypted information packets sent from the application client to the device server	4.2.20.5
SNt	4	public	The current sequence number for encrypted information packets sent from the device server to the application client	4.2.20.5
SKEYSEED	32	private	A special encryption key known only to the application client and device server of this SA. This value is used to generate keying material.	4.2.20.6
KDF_ID	2	public	Key derivation function identifier	4.2.20.7
I_T nexus	-	public	An optional I_T nexus describing which ports are associated with this SA.	SAM-3
Scope description: public: a parameter that may be made available outside of a particular device private: a parameter that shall not be disclosed in plaintext to any external entity				

An application client or device server may maintain a list of security associations internally in an SA database. This database may contain any number of SA entries. It is allowable to destroy an SA entry at any time to make room for a new SA.

An application client or device server shall not pass in plaintext any shared secrets from an SA, such as the SKEYSEED or other derived shared keys, over the service delivery subsystem.

An application client or device server may destroy an SA at any time for any reason. If an I_T nexus is associated with an SA that is destroyed, the device server shall establish for that I_T nexus a Unit Attention condition 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 association identifier

A security association identifier (SAI) is a set of two 32-bit numbers (SAI_i and SAI_t) that uniquely identifies a particular SA. As part of the protocol for establishing an SA, the application client shall generate SAI_i, and the device server shall generate SAI_t. Each side shall pass this SAI to the other participant so that both sides contain the same information within the SA structure. Any encrypted information packet that uses a secure channel shall include the SAI of the receiver (either SAI_i or SAI_t) to identify the SA. The SAI shall be between 256 and $2^{32}-1$, inclusive. The values 0 to 255 are reserved for use by IETF (see RFC 4306).

If a device server receives a command with an invalid SAI_t, 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 ASSOCIATION IDENTIFIER. An SAI is invalid if it does not exist in the recipient's SA database

Editor's Note: INVALID SECURITY ASSOCIATION IDENTIFIER is a new ASQ

4.2.20.4 Nonce

As part of establishing a new SA, each participant shall create a nonce using a cryptographically secure random number generator. The nonce should be generated using a random number generator approved by NIST (see NIST FIPS 140-2 and NIST SP 800-90). The nonce shall never be reused for creating subsequent security associations.

4.2.20.5 Sequence number

A sequence number (SN) is an unsigned 32-bit integer that is used for sequencing encrypted information packets. The main purpose of a sequence number is to prevent a replay attack, in which an attacker sends an old encrypted information packet as new. Both the application client and the device server shall maintain a sequence number for each encrypted information packet in each direction. When a new SA is established, the application client and device server shall set the two sequence numbers to zero. Before sending an encrypted information packet, the sender shall increment the sequence number. Therefore, the first valid sequence number in an encrypted information packet is one. The receiver of an encrypted information packet shall verify that the sequence number is larger than that of all previous encrypted information packets that reference the same SA. 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 sending or receiving an encrypted information packet that uses the highest valid sequence number, the SA shall be destroyed.

4.2.20.6 Generating keying material for the SA

All encrypted information packets associated with a given SA shall use shared keys for encrypting and validating information across the service delivery subsystem. These shared keys are a function of a shared key-seed (SKEYSEED), the SAI values, and the nonce values. To generate a particular shared key, Table 4.2 shows all the shared keys, along with their respective indices and descriptions.

Table 4.2 - Description of shared keys

Name	Index	Description
SK_d	1	derived key used to generate a new SKEYSEED during a re-keying operation
SK_ai	2	used for authenticating ESP messages from the application client to the device server
SK_at	3	used for authenticating ESP messages from the device server to the application client
SK_ei	4	used for encrypting ESP messages from the application client to the device server
SK_et	5	used for encrypting ESP messages from the device server to the application client
SK_pi	6	used for authenticating an SA (see RFC 4306)
SK_pt	7	used for authenticating an SA (see RFC 4306)
SK_kwei	8	used for encryption of NIST AES key-wrap payloads
SK_kwai	9	used for authentication of NIST AES key-wrap payloads

Name	Index	Description
Legend:		
		SK = shared key
		d = derived
		a = authentication (integrity check value)
		e = encryption
		i = initiator (application client)
		t = target (device server)
		kw = key-wrap

To generate the shared keys, an application client or device server shall use the negotiated key derivation function (KDF) for an SA, providing the index as an input. The size of each shared key shall match the size of the corresponding hash function used by the KDF. See 4.2.20.7 for more details on key derivation functions.

The shared keys shall not be transmitted across the service delivery subsystem in plaintext.

4.2.20.7 Key derivation function

A key derivation function (KDF) is a cryptographic primitive that computes the shared keys using the SKEYSEED, SAI, and nonce values as input. An application client and device server shall negotiate a key derivation function identifier (KDF_ID) from Table 4.1.

Table 4.3 - Key derivation functions

Code	Description	Reference
0000h	Reserved	
0001h	NIST SP 800-56A: Concatenation Key Derivation Function (Approved Alternative 1) using SHA-256	4.2.20.8
0002h-0FFFh	Reserved	
1000h-FFFFh	Vendor specific	

An application client or device server only needs to generate the shared keys that might be used.

If a particular function requires a shared key shorter than the full shared key size, then that function shall use the leftmost bits (i.e. most significant) to create a smaller shared key.

4.2.20.8 Key derivation function using NIST SP 800-56A

A KDF_ID value of 0001h shall indicate the key derivation function as specified in NIST SP 800-56A under the clause entitled "Concatenation Key Derivation Function (Approved Alternative 1)", with the following qualifiers:

Auxiliary Function:

- 1) H: SHA-256 hashing function as specified in NIST FIPS 180-2

Input:

- 1) Z: SKEYSEED
- 2) keydatalen: 256 * 9 (i.e. the hash size times the maximum number of shared keys)
- 3) OtherInfo:

- AlgorithmID: The empty string "".
- PartyUInfo: SAl_i || N_i (that is, the concatenation of SAl_i and N_i)
- PartyVInfo: SAl_t || N_t (that is, the concatenation of SAl_t and N_t)
- SuppPubInfo: The empty string "".
- SuppPrivInfo: The empty string "".

When generating a particular output from this KDF, the value 'counter' shall be set to the corresponding Index within Table 4.2. Each shared key that is generated by this KDF shall have a size of 256 bits.

Immediately after an application client or device server generates all the shared keys, it shall set all instances of the SKEYSEED variable to zero, as required by NIST SP 800-56A.

- 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

8.5.3.2.1 Set data encryption page format

Table Y1 shows the parameter list format of the Set Data Encryption page.

Table Y1 – Set Data Encryption page

Bit	7	6	5	4	3	2	1	0
Byte								
0	(MSB)	PAGE CODE (0010h)						(LSB)
1								
2	(MSB)	PAGE LENGTH (m-3)						(LSB)
3								
4	SCOPE			Reserved			LOCK	
5	Reserved				CKOD	CKORP	CKORL	
6	ENCRYPTION MODE							
7	DECRYPTION MODE							
8	ALGORITHM INDEX							
9	KEY FORMAT							
10	Reserved							
17								
18	(MSB)	KEY LENGTH (n-19)						(LSB)
19								
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	Reference
00h	The KEY field contains the key to be used to encrypt or decrypt data.	8.5.3.2.2
01h	The KEY field contains a vendor specific key reference	8.5.3.2.3
02h	NIST AES Key-Wrap	8.5.3.2.4
03h – BFh	Reserved	N/A
C0h – FFh	Vendor specific	N/A

The KEY LENGTH field indicates the length of the key field in bytes.

(Move KEY-ASSOCIATED DATA DESCRIPTORS LIST text here)

8.5.3.2.2 Plaintext key format

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.

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

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

~~The KEY LENGTH field indicates the length of the key field in bytes.~~

8.5.3.2.3 Key reference format

~~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.~~

If the KEY FORMAT field is 01h, the KEY field shall contain the information as defined by Table Y8a.

Table Y8a – KEY field contents with KEY FORMAT field set to 01h

Bit	7	6	5	4	3	2	1	0
Byte								
20	(MSB)							
	T10 VENDOR IDENTIFICATION							
27	(LSB)							
28	(MSB)							
	VENDOR SPECIFIC KEY REFERENCE							
n	(LSB)							

The T10 VENDOR IDENTIFICATION field shall contain 8 bytes of T10 vendor identification (see SPC-4).

The VENDOR SPECIFIC KEY REFERENCE field shall include vendor-specific information used for identifying the key that encrypts or decrypts data. If the VENDOR SPECIFIC KEY REFERENCE 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.

8.5.3.2.4 NIST AES key wrap format

If the KEY FORMAT field of Table Y1 is 02h, the KEY field shall contain a wrapped key, as specified by IETF RFC 3394 (AES Key-wrap). 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	7	6	5	4	3	2	1	0
Byte								
20	(MSB)	SECURITY ASSOCIATION IDENTIFIER (SAIt)						(LSB)
23								
24	(MSB)	SEQUENCE NUMBER						(LSB)
27								
28	(MSB)	AES KEY-WRAPPED KEY						(LSB)
n-16								
n-15	(MSB)	INTEGRITY CHECK VALUE (ICV)						(LSB)
n								

The SAIt field shall reference the security association (SA) that contains the shared keys SK_kwei and SK_kwai. SK_kwai provides the key for generating the integrity check value (ICV) and SK_kwei provides the Key Encryption Key (KEK) for the AES Key-Wrap algorithm (see 4.2.20.3).

The SEQUENCE NUMBER field shall contain the next SNt value for the referenced SA. The device server shall verify the sequence number and report an error if a non-increasing number is detected (see 4.2.20.5).

The AES KEY-WRAPPED KEY field shall contain the wrapped key, according to the NIST AES Key-wrap algorithm.

NOTE - After performing the AES key-wrap algorithm, the wrapped-key expands in size by 8 bytes. These extra 8 bytes provide an integrity check for the unwrapping process.

The device server shall ensure that the length of the AES KEY-WRAPPED KEY field is a multiple of 8 bytes, as specified by the AES Key-wrap specification (i.e. the device server shall ensure that the KEY LENGTH field minus 32 bytes is a multiple of 8 bytes). 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.

The INTEGRITY CHECK VALUE (ICV) field shall contain a 128-bit ICV that is computed, in order, over the KEY LENGTH, SAIt, SEQUENCE NUMBER, and AES KEY-WRAPPED 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_kwai (see 4.2.20.6). If the calculated ICV does not match the value in the ICV field, 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 device server shall also validate the integrity of the wrapped key, according to the NIST AES Key-wrap algorithm. If this integrity check fails, 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.

NOTE - The length of the final unwrapped key equals the KEY LENGTH field minus 32 bytes. These 32 bytes correspond to the total length of the SAIt, SEQUENCE NUMBER, and INTEGRITY CHECK VALUE fields, plus the additional 8 bytes created by the AES Key-Wrap algorithm.