ENDL
TEXAS

Date: 15 March 2007
To: T10 Technical Committee
From: Ralph O. Weber
Subject: Security Association Model for SPC-4

Overview

A critical element of data encryption and integrity checking algorithms is an entity called an SA (Security Association) that the participating pair of endpoints represent using a pair of indices.

- In most of the security world, a SA index is know as an SPI (Security Parameters Index).
- Because of the long-standing usage of the acronym SPI in SCSI, this proposal uses SAI (Security Association Index) as the SCSI equivalent of the security-traditional SPI.

SA information (i.e., the security parameters) are never transmitted in their entirety in any of the usual SPC-4 suspects (CDBs, parameter data, etc.). A SA is represented by two sets of parameters, one stored internally at each of the two participating endpoints. This situation produces an unusual SCSI model challenge that can only be covered by some carefully crafted model text, which is the goal of this proposal.

Revision History

r0 Original revision
r1 Made changes requested by Matt Ball and David Black.
r2 Made more changes requested by Matt Ball and David Black.
r3 Made changes requested by Gerry Houlder and Bob Nixon.
r4 Made changes requested by the September 2006 CAP working group.
r5 Change KEY(n) to KEYMAT as recommended by David Black on the T10 Reflector (17 September 2006), and modify random nonce definition so that SSC-3 can reference random number generation rules in SPC-4.
r6 Made changes requested by the November 2006 SSC and CAP working groups. Also forced DS_SAI values to be unique across all I_T nexuses.
r7 Made changes requested by the January 2007 CAP working group. Also added an SA Parameter for SA-specific management data (e.g., SA data needed to delete the SA).
r8 Made changes requested by the March 2007 CAP working group.

Changes made in r8 are identified with change bars.

Definition of nonce

The definition of nonce has produced substantial discussions during the development of this proposal. Coordinating the SPC-4 nonce definition with the one already present in OSD and OSD-2 has been tricky. For reference, the OSD definition is reproduced here with the proposed changes indicated.

3.1.23 nonce: A value that is used one and only one time ~~and thus~~ to provide uniqueness to a value (e.g., a secure cryptographic key) in whose derivation it participates or to uniquely ~~identifies~~ identify a single instance of something (e.g., a timestamp ~~an individual OSD command, or one credential~~) ~~transacted~~ exchanged between an application client~~,~~ and a device server~~, and security manager~~.

Proposed SPC-4 Changes

Most of the text shown below is new SPC-4 material shown in black. If a subclause contains old and new material colors and strikeouts are used to identify changes and notes that are not intended for inclusion in SPC-4.

2.4 NIST References

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

NIST SP (Special Publication) 800-38C, *Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*

Copies of the following approved NIST 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)*

2.5 IETF References

Copies of the following approved IETF standards may be obtained through the Internet Engineering Task Force (IETF) at www.ietf.org.

…

RFC 2104, *HMAC: Keyed-Hashing for Message Authentication*
RFC 3566, *The AES-XCBC-MAC-96 Algorithm and Its Use With IPsec*
RFC 3766, *Determining Strengths For Public Keys Used For Exchanging Symmetric Keys*
RFC 4086, *Randomness Requirements for Security*
RFC 4306, *Internet Key Exchange (IKEv2) Protocol*
RFC 4434, *The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE)*

3.1 Definitions

…

3.1.b hashed message authentication code (HMAC): A type of message authentication code that is calculated using the cryptographic hash function as defined in FIPS 198a (see 2.4) in combination with a secret key.

3.1.c key derivation function (KDF): An algorithm that is used to derive cryptographic keying material from a shared secret and other information.

3.1.f nonce: A value that is used one and only one time to provide uniqueness to a value (e.g., a secure cryptographic key) in whose derivation it participates or to uniquely identify a single instance of something (e.g., a timestamp) exchanged between an application client and a device server.

3.1.h random nonce: A secure random number (see 4.6) that has a negligible chance of repeating and whose uses include providing significant uniqueness and randomness in cryptographic calculations.

3.1.l SA parameters: The parameters stored by both an application client and a device server that are associated with one SA (see 3.1.m) and identified by a pair of SAIs (see 3.1.p). See 5.13.2.2.

3.1.m Security association (SA): A relationship and associated security processing between an application client and device server that is used to apply security functions (e.g., data integrity checking, data encryption) to data that is transferred in either direction. See 5.13.

3.1.p Security association index (SAI): A number representing the parameters for a security association as stored internally by the application client or device server. In other security models, this value is called the security parameters index (SPI). See 5.13.

3.1.s Secure hash algorithm (SHA): A secure hash algorithm (e.g., SHA-1) specified in FIPS 180-2 with Change Notice 1 dated February 25, 2004 (see 2.4).

3.1.x Secure random number: A random number that is generated in ways that protect it from security attacks. See 4.6.

…

3.2 Symbols and acronyms

…

||	concatenation
AES	Advanced Encryption Standard
HMAC	Hashed Message Authentication Code (see 3.1.b)
IKEv2	Internet Key Exchange version 2 (see RFC 4306)
KDF	Key Derivation Function (see 3.1.c)
SA	Security Association (see 3.1.m)
SAI	Security Association Index (see 3.1.p)
SHA-1	Secure Hash Algorithm, 160 bits (see 3.1.s)
SHA-256	Secure Hash Algorithm, 256 bits (see 3.1.s)
SHA-384	Secure Hash Algorithm, 384 bits (see 3.1.s)
SHA-512	Secure Hash Algorithm, 512 bits (see 3.1.s)

…

{{Note: Add symbols alphabetically by description. Add acronyms alphabetically by acronym.}}

4.6 Secure random numbers

Secure Random numbers should be generated as specified by RFC 4086 (e.g., see FIPS 140-2 Annex C: Approved Random Number Generators).

If the same random number source is used generate random numbers for multiple purposes (e.g., nonces and secret keys), interactions between the two shall not be allowed to compromise secrecy. If the value sequence generated by the common random number source is predictable to any degree, then the random number values that are transmitted outside the SCSI device may provide information about the random number values that the SCSI device maintains internally, based on the reasonable assumption that an adversary knows the order in which the random numbers are obtained from the common random number source. SCSI devices shall eliminate sources of such predictability.

Compliance with RFC 4086 is one method for achieving the required independence between random number values.

…

5.13 Security Features

5.13.1 Security goals and threat model

…

5.13.1.4 Types of attacks

…

There are a wide variety of active attacks (e.g., spoofing, replay, insertion, deletion, and modification of communications). Man-in-the-middle attacks are a ~~sinister~~ class of active attacks that involve the attacker inserting itself in the middle of communication, enabling it to intercept all communications without the knowledge of the communicating parties for various purposes (e.g., insertion, deletion, replay, modification and/or inspection via decryption of the communications). ~~the purpose of insertion, deletion, and/or modification of the communications.~~

{{Note: All text from here to 5.13.4 is new}}

5.13.2 Security associations

5.13.2.1 Principals of security associations

Before an application client and device server begin applying security functions (e.g., data integrity checking, data encryption) to messages (i.e., data that is transferred in either direction between them), they perform a security protocol to create at least one SA (see 5.13.2.3). The result of the SA creation protocol is two sets of SA parameters (see 5.13.2.2), one that is maintained by the application client and one that is maintained by the device server.

In this model, SAs decouple the process of creating a security relationship from its usage in processing security functions. This decoupling allows either the creation or the usage of an SA to be upgraded in response to changing security threats without requiring both processes to be upgraded concurrently.

Figure x1 shows the relationship between application clients and device servers with respect to SAs.

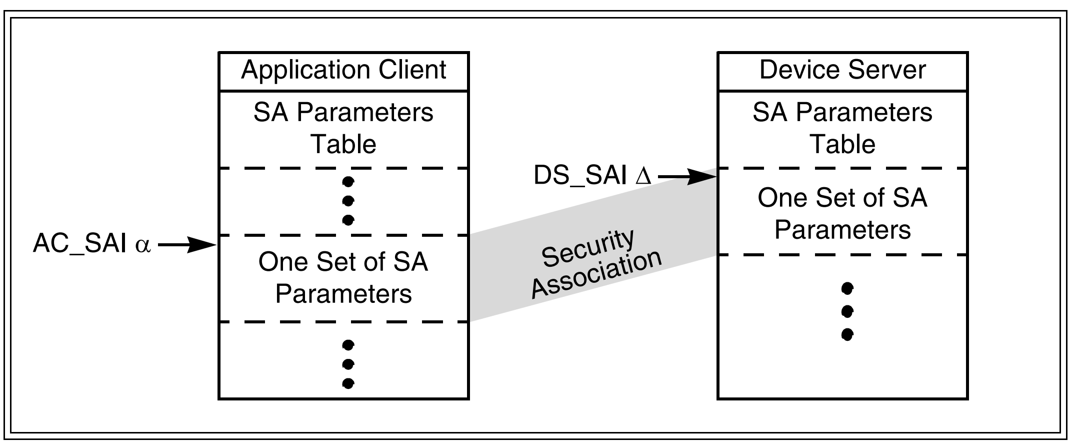

Figure x1 — SA relationships

In both the application client and the device server, the SA parameters are modelled as being stored in an indexed array and the SAI identifies one set of SA parameters within that array. The application client and device server are not required to store the parameters for any given SA in the same array locations. In order to support this implementation flexibility, a single SA is modelled as having two different SAI values (i.e., one for the application client and one for the device server).

The device server shall maintain a single SA parameters table for all I_T nexuses.

SAs shall not be preserved across a power cycle, hard reset, or logical unit reset. SAs shall not be affected by an I_T nexus loss.

5.13.2.2 SA parameters

Each SAI shall identify at least the SA parameters defined in table x1. Individual security protocols define how the SA parameters are generated and/or used by that security protocol.

Table x1 — Minimum SA parameters (part 1 of 3)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that identify and manage the SA.				
AC_SAI	The SAI used by the application client to identify the SA. [c]	4	4	Public
DS_SAI	The SAI used by the device server to identify the SA. [c]	4	4	Public
TIMEOUT	The number of seconds that may elapse after the completion of an SA access operation (i.e., SA creation or SA usage by a command) before the device server should discard the state associated with this SA (e.g., the SA parameters). If SA state is discarded because no SA access operations are received during the specified interval, the device server shall respond to further attempts to access the SA as if the SA had never been created. This parameter shall not be set to zero.	4	4	Public
SA parameters that are incorporated in messages to prevent message replay attacks.				
AC_SQN	A sequence number that is incremented for each response message received by an application client on which a security function is performed and used to detect replay attacks (see 5.13.1.4).	8	8	Public
DS_SQN	A sequence number that is incremented for each request message received by a device server on which a security function is performed and used to detect replay attacks (see 5.13.1.4).	8	8	Public

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.
[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.
[c] SAI values between 0 and 255, inclusive, are reserved.
[d] Nonce SA parameters shall be at least half the size of KEY_SEED SA parameter.
[e] The number of bits of entropy in the KEY_SEED should be as close to the number of bits in the KEY_SEED as possible (see RFC 3766).

Table x1 — Minimum SA parameters (part 2 of 3)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that are used by security functions to derive the secret keys that are applied to messages (e.g., for encryption).				
AC_NONCE [d]	A random nonce (see 3.1.h) value that is generated by the application client and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
DS_NONCE [d]	A random nonce value that is generated by the device server and used as an input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	64	Public
KEY_SEED [e]	A value that is known only to the application client and device server that are creating this SA that in combination with the applicable nonce is used to derive the KEYMAT value. The KEY_SEED shall be set to zero as part of completing the SA creation function.	16	64	Secret
KDF_ID	A security algorithm (see 5.13.2) coded value that identifies the KDF used by the application client and device server.	4	4	Public
SA parameters that are used by security functions to secure messages between the application client and device server.				
KEYMAT	A value that is known only to the application client and device server that are participating in this SA that may be subdivided into one or more key values that are used in security functions that secure messages. The contents of KEYMAT depend on the USAGE_TYPE SA parameter value.	14	1 024	Secret

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.
[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.
[c] SAI values between 0 and 255, inclusive, are reserved.
[d] Nonce SA parameters shall be at least half the size of KEY_SEED SA parameter.
[e] The number of bits of entropy in the KEY_SEED should be as close to the number of bits in the KEY_SEED as possible (see RFC 3766).

Table x1 — Minimum SA parameters (part 3 of 3)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that are used by SA management functions.				
USAGE_TYPE	A coded value (see table x2) that indicates the how the SA is used.	2	2	Public
USAGE_DATA	Information associated with how the SA is used (e.g., cryptograph algorithms and key sizes). The contents of USAGE_DATA depend on the USAGE_TYPE SA parameter value.	0	1 024	Public
MGMT_DATA	SA data that is used in ways defined by the SA creation protocol to perform SA management functions (e.g., deletion of the SA).	0	1 024	Protocol specific

[a] These size values are guidelines. Specific security protocols may place more exacting size requirements on SA parameters.
[b] Public SA parameters may be transferred outside a SCSI device unencrypted. Secret SA parameters shall not be transferred outside a SCSI device. Fields within a protocol specific SA parameter are Shared or Secret as defined by the applicable SA creation protocol.
[c] SAI values between 0 and 255, inclusive, are reserved.
[d] Nonce SA parameters shall be at least half the size of KEY_SEED SA parameter.
[e] The number of bits of entropy in the KEY_SEED should be as close to the number of bits in the KEY_SEED as possible (see RFC 3766).

The USAGE_TYPE SA parameter shall be one of the values shown in table x2.

Table x2 — USAGE_TYPE SA parameter values

Value	Description	Reference
0000h - 0080h	Reserved	
0081h	Tape Data Encryption	SSC-3
0082h - FFFFh	Reserved	

5.13.2.3 Creating a security association

The SECURITY PROTOCOL IN command (see 6.27) and SECURITY PROTOCOL OUT command (see 6.28) security protocols shown in table x3 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
   F) MGMT_DATA;
   and
c) Values that are zero upon completion of SA creation:
   A) KEY_SEED.

Table x3 — Security protocols that create SAs

Security Protocol Code	Description	Reference
TBD	TBD	TBD

5.13.3 Key derivation functions

5.13.3.1 Overview

Table x4 summarizes the key derivation functions defined by this standard.

Table x4 — Key derivation functions summary

Security Algorithm Code (see table 44)	Description	Reference
0002 0002h	IKEv2-based iterative HMAC KDF based on SHA-1	5.13.3.3
0002 0005h	IKEv2-based iterative HMAC KDF based on SHA-256	5.13.3.3
0002 0006h	IKEv2-based iterative HMAC KDF based on SHA-384	5.13.3.3
0002 0007h	IKEv2-based iterative HMAC KDF based on SHA-512	5.13.3.3
0002 0004h	IKEv2-based iterative KDF based on AES-128 in XCBC mode	5.13.3.4

5.13.3.2 IKEv2-based iterative KDF

In principle, KEYMAT is generated by applying IFUNC to STRING using KEY_SEED to produce the requested number of KEYMAT bits. In equation notation, the operation is as follows:

KEYMAT = IFUNC(KEY_SEED, STRING)

However, accomplishing this may require multiple applications of IFUNC because the number of bits output by IFUNC is not sufficient to meet the number of KEYMAT bits that are to be produced.

The IKEv2-based (see RFC 4306) iterative technique for applying IFUNC is as follows:

1) Initialize PREV_OUTPUT to a null string (i.e., a string that contains no bits);
2) Repeat the following function for values of N that increment from one by one to a maximum of 255 or until the total number of bits returned by all invocations of IFUNC equals or exceeds the number of KEYMAT bits that are to be produced, whichever occurs first:

   T_N = IFUNC(KEY_SEED, (PREV_OUTPUT || STRING || a byte containing the value N))

   and
3) Concatenate the T_N values (e.g., T_1 || T_2 || T_3) and return as many of the resulting bits as specified by the number of KEYMAT bits that are to be produced input parameter, starting with the first bit in T_1.

Protocols that call the IFUNC function to generate KEYMAT should ensure that the number of KEYMAT bits requested does not cause N to exceed 255. If N reaches 256, then:

1) The requested number of KEYMAT bits is not returned by IFUNC; and
2) The request to produce KEYMAT shall be terminated with an error.

5.13.3.3 HMAC-based KDFs

If the KDF_ID is one of those shown in table x5, the key derivation function is a combination of the:

a) The HMAC function defined in FIPS 198a (see 2.4);
b) The secure hash function shown in table x5 for the specified KDF_ID value; and
b) IKEv2-based iterative KDF technique (see 5.13.3.2).

The technique requires the following inputs from or related to the SA parameters:

a) AC_SAI;
b) DC_SAI;
c) AC_NONCE;
d) DC_NONCE;
e) KEY_SEED; and
f) KEYMAT size, in bits.

The USAGE_TYPE SA parameter and USAGE_DATA SA parameter (see 5.13.2.2) specify the KEYMAT size as part of the security protocol that performs SA creation (see 5.13.2.3).

The IKEv2-based iterative KDF technique is applied with the following inputs:

a) IFUNC is the HMAC function defined in FIPS 198a with the translation of inputs names shown in table x5;
b) KEY_SEED is the KEY_SEED SA parameter; and

c) STRING store the concatenated contents of the following SA parameters:
   1) AC_NONCE;
   2) DC_NONCE;
   3) AC_SAI; and
   4) DC_SAI.

Table x5 — HMAC-based key derivation functions

FIPS 198a inputs selected by KDF_ID	KDF_ID (see table x4)			
	0002 0002h	**0002 0005h**	**0002 0006h**	**0002 0007h**
H (i.e., hash function)	SHA-1 (see table x6)	SHA-256 (see table x6)	SHA-384 (see table x6)	SHA-512 (see table x6)
B (i.e., hash input block size) [a]	64	64	128	128
L (i.e., hash output block size) [a]	20	32	48	64
t (i.e., MAC size) [a]				
K (i.e., key)	KEY_SEED SA parameter			
text	STRING as defined in this subclause and used in 5.13.3.2			

[a] In accordance with FIPS 198a, all sizes are shown in bytes.

Details of the hash functions that act as inputs to the FIPS 198a HMAC function are shown in table x6.

Table x6 — Hash functions used by HMAC based on KDF_ID

KDF_ID (see table x4)	Function	Description
0002 0002h	SHA-1	HMAC input *H* is the SHA-1 secure hash function defined in FIPS 180-2 with Change Notice 1 (see 2.4).
0002 0005h	SHA-256	HMAC input *H* is the SHA-256 secure hash function defined in FIPS 180-2 with Change Notice 1.
0002 0006h	SHA-384	HMAC input *H* is the SHA-384 secure hash function defined in FIPS 180-2 with Change Notice 1.
0002 0007h	SHA-512	HMAC input *H* is the SHA-512 secure hash function defined in FIPS 180-2 with Change Notice 1.

5.13.3.4 AES-XCBC-PRF-128 IKEv2-based iterative KDF

If the KDF_ID is 0002 0004h, the key derivation function is a combination of the:

a) AES-XCBC-PRF-128 secure hash function defined in RFC 4434 (see 2.5) and RFC 3566; and
b) IKEv2-based iterative KDF technique (see 5.13.3.2).

The technique requires the following inputs from or related to the SA parameters:

a) AC_SAI;
b) DC_SAI;
c) AC_NONCE;
d) DC_NONCE;
e) KEY_SEED; and
f) The number of KEYMAT bits that are to be produced.

The IKEv2-based iterative KDF technique is applied with the following inputs:

a) IFUNC is the AES-XCBC-PRF-128 secure hash function with the translation of inputs names shown in table x7;
b) KEY_SEED is the KEY_SEED SA parameter; and
c) STRING store the concatenated contents of the following SA parameters:
   1) AC_NONCE;
   2) DC_NONCE;
   3) AC_SAI; and
   4) DC_SAI.

Table x7 — RFC 3566 parameter translations KDF based on AES-XCBC-PRF-128

RFC 3566 Parameter	Translation
K (i.e., key)	KEY_SEED SA parameter
M (i.e., message)	STRING as defined in this subclause and used in 5.13.3.2

{{Note: The following subclauses already appear in SPC-4, and modifications are shown.}}

~~5.13.2 Security algorithm codes~~

5.13.4 Security algorithm codes

Table 44 lists the security algorithm codes used in security protocol parameter data.

Table 44 — Security algorithm codes

Code	Description	Reference
Encryption Algorithms		
0001 0010h [a]	AES-CCM with a 16 byte MAC	NIST SP 800-38C
0001 0014h [a]	AES-GCM with a 16 byte MAC	NIST SP 800-38D
KDF Algorithms		
0002 0002h [a]	IKEv2-based iterative HMAC KDF based on SHA-1	5.13.3.3
0002 0004h [a]	IKEv2-based iterative KDF based on AES-128 in CBC mode	5.13.3.4
0002 0005h [a]	IKEv2-based iterative HMAC KDF based on SHA-256	5.13.3.3
0002 0006h [a]	IKEv2-based iterative HMAC KDF based on SHA-384	5.13.3.3
0002 0007h [a]	IKEv2-based iterative HMAC KDF based on SHA-512	5.13.3.3
Other Algorithms		
0000 0400h - 0000 FFFFh	Vendor specific	
All other values	Reserved	

[a] The lower order 16 bits of this code value are assigned to match an IANA assigned value, if any, for an equivalent IKEv2 encryption algorithm (see 3.1.53) and the high order 16 bits match the IANA assigned IKEv2 transform type (i.e., 1~~,~~ – Encryption Algorithms, 2 – Pseudo-random Functions).