ENDL
TEXAS

Date: 24 August 2006
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.

Since r2 is intended to be the revision presented to CAP, the customary change bar history is not provided.

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.

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*

NIST SP (Special Publication) 800-56A, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*

Copies of the following approved NIST standards may be obtained through the National Institute of Standards and Technology (NIST) at http://csrc.nist.gov/cryptval/shs.htm.

FIPS 180-2 with Change Notice 1 dated February 25, 2004, *Secure Hash Standard*

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 4086, *Randomness Requirements for Security*

3.1 Definitions

…

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 nonce (see 3.1.f) that is a random number. See 5.13.4.

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 5.13.3.2.

3.1.m Security association (SA): A relationship 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 Security 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).

…

3.2 Symbols and acronyms

…

||	concatenation

…

KDF	Key Derivation Function (see 3.1.c)
SA	Security Association (see 3.1.m)
SAI	Security Association Index (see 3.1.p)
SHA	Secure Hash Algorithm (e.g., SHA-1) (see 3.1.s)

…

5.13 Security Features

5.13.1 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		
FFFF 0001h	Concatenation KDF based on SHA-256	5.13.2.1
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 for an equivalent IKEv2 encryption algorithm (see 3.1.52) and the high order 16 bits match the IANA assigned IKEv2 transform type (i.e., 1, Encryption Algorithms).

{{Note: All text from here to the end of this proposal is new}}

5.13.2 Key derivation functions

5.13.2.1 Concatenation KDF based on SHA-256

The security algorithm code FFFF 0001h identifies a KDF that is based on Approved Alternative 1 as specified in NIST SP 800-56A (see 2.4) with the parameters shown in table x1.

Table x1 — NIST SP 800-56A parameters for concatenation KDF based on SHA-256

NIST SP 800-56A Parameter	Value
Fixed Values	
hashlen	256
max_hash_inputlen	256
Auxiliary Function	
Z	SHA-256
Input	
keydatalen	256 × the maximum number of shared keys to be generated (i.e., 256 × the n in the KEY(n) SA parameter (see 5.13.3.2))
OtherInfo AlgorithmID	"INCITS T10 KDF using SHA-256"
OtherInfo PartyUInfo	AC_SAI SA parameter || AC_NONCE SA parameter
OtherInfo PartyVInfo	DS_SAI SA parameter || DS_NONCE SA parameter
OtherInfo SuppPubInfo	A value containing zero bits (i.e., the empty bit string)
OtherInfo SuppPrivInfo	A value containing zero bits (i.e., the empty bit string)

Each 256 bits of the resulting bit string represents one generated cryptographic key (i.e., first 256 bits are stored in the KEY(1) SA parameter, the second 256 bits are stored in the KEY(2) SA parameter, the next 256 bits are stored in the KEY(3) SA parameter, etc.).

After the cryptographic keys are generated, the KEY_SEED SA parameter shall be set to zero (see NIST SP 800-56A).

5.13.3 Security associations

5.13.3.1 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 5.13.3.3). The output of the SA creation protocol is two sets of SA parameters (see 5.13.3.2), one that is stored by the application client and one that is stored 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.

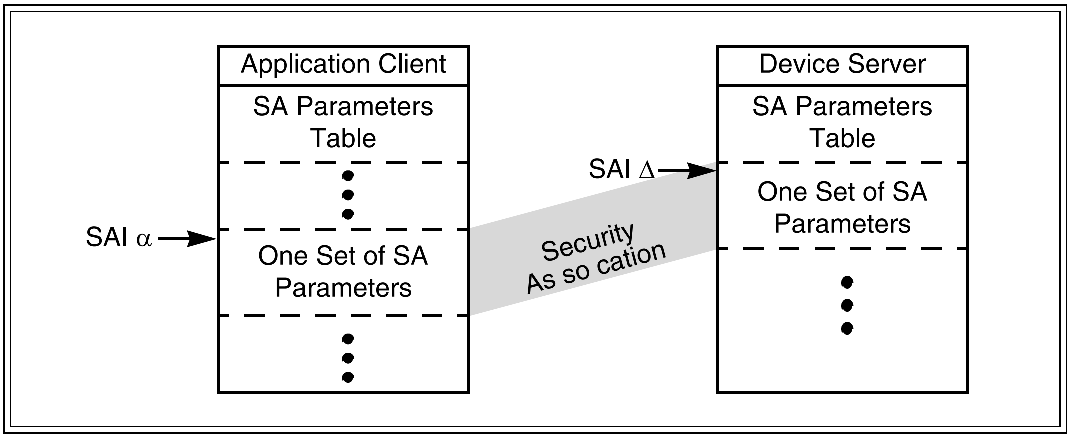

Figure x1 — SA relationships

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).

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

5.13.3.2 SA parameters

Each SAI shall identify at least the SA parameters defined in table x2. Individual security protocols define how the SA parameters are generated and/or used by that security protocol.

Table x2 — Minimum SA parameters (Sheet 1 of 2)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that identify 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

[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 be encrypted whenever they are transferred outside a SCSI device.
[c] SAI values between 0 and 256, 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.

Table x2 — Minimum SA parameters (Sheet 2 of 2)

Name	Description	Size (bytes) [a]		Scope [b]
		Min.	Max.	
SA parameters that are incorporated in messages to prevent message replay attacks.				
AC_SQN	A sequence number that is incremented for each application client message on which a security function is performed.	4	8	Public
DS_SQN	A sequence number that is incremented for each device server message on which a security function is performed.	4	8	Public
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 5.13.4) 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	32	Public
DS_NONCE [d]	A random nonce value that is generated by the device server and used as and input to the key derivation security algorithm specified by the KDF_ID SA parameter during the derivation of an encryption key.	16	32	Public
KEY_SEED [e]	A value that is known only to the application client and device server that are participating in this SA that in combination with the applicable nonce is used to derive all the KEY(n) values using the algorithm specified by the KDF_ID SA parameter.	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.				
KEY(n)	One or more values that are known only to the application client and device server that are participating in this SA that are used in security functions that secure messages.	32	64	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 be encrypted whenever they are transferred outside a SCSI device.

[c] SAI values between 0 and 256, 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.

5.13.3.3 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 5.13.3.2) values as follows:

a) SA lifetime values for both (i.e., application client and device server) SAIs, both nonces, the KEY_SEED, and the KDF_ID SA; and
b) the initial values for both sequence numbers and the initial KEY(n) values.

Table x3 — Security protocols that create SAs

Security Protocol Code	Description	Reference
TBD	TBD	TBD

5.13.4 Random nonces

Random nonces should be generated as specified by RFC 4086.

If the same random number source is used for both nonce generation and the generation of other values (e.g., 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 nonce values that are transmitted outside the SCSI device may provide information about the nonces 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.