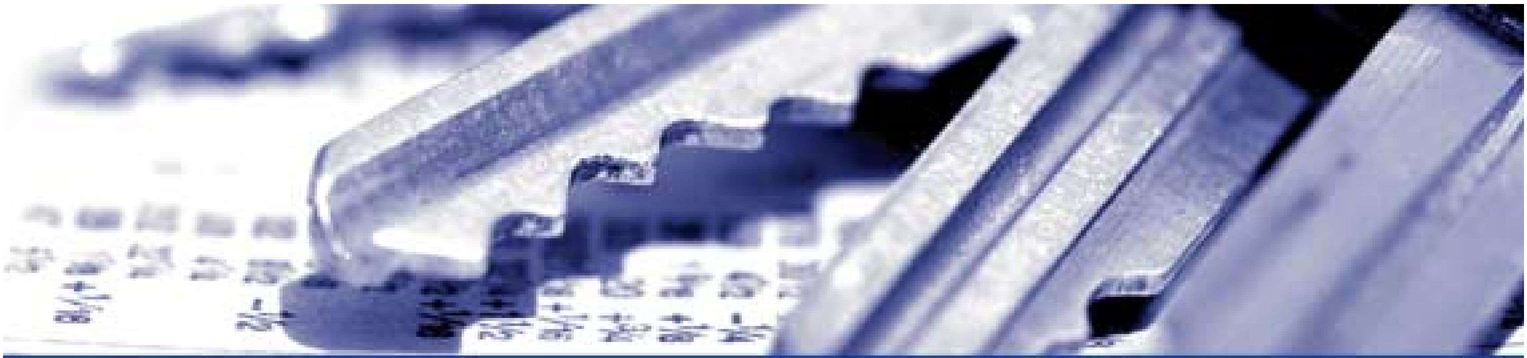

The nShield and payShield modules module security policy

nShield F3 SCSI, nShield F3 SCSI Ultrasign and payShield

nCipher

Date: **13 March 2006**

Version: **1.1.41**

nCipher Corporation Limited makes no warranty of any kind with regard to this information, including, but not limited to, the implied warranties of merchantability and fitness to a particular purpose. nCipher Corporation Limited shall not be liable for errors contained herein or for incidental or consequential damages concerned with the furnishing, performance or use of this material.

Patents

UK Patent GB9714757.3. Corresponding patents/applications in USA, Canada, South Africa, Japan and International Patent Application PCT/GB98/00142.

Contents

Chapter 1: Purpose

The nShield tamper resistant Hardware Security Modules and payShield Payments Modules is a multi-tasking hardware module that is optimized for performing modular arithmetic on very large integers. The nShield and payShield modules module also offers a complete set of key management protocols.

The nShield and payShield modules modules: nShield F3 SCSI, nShield F3 SCSI Ultrasign and payShield are FIPS 140-1 level 3 stand-alone devices. The units are identical in operation and only vary in the processing speed and the support software supplied.

The module runs firmware provided by nCipher. There is the facility for the user to upgrade this firmware. In order to determine that the module is running the correct version of firmware they should use the NewEnquiry service which reports the version of firmware currently loaded. The validated firmware versions are 1.77.100.

The module can be initialized to comply with the roles and services section at either level 2, role based authorization, or level 3, identity based authorization. See “To comply with FIPS 140-1 level 3” on page 17. The initialization parameters are reported by the NewEnquiry and SignModuleState services.

The module offers some general services that do not involve stored keys and are therefore excluded from FIPS 140-1. While the module is executing these services or SEE code it is operating in a non-FIPS 140-1 mode.

The nShield and payShield modules module connects to the host computer via a SCSI bus. The nShield and payShield modules module must be accessed by a custom written application. Full documentation for the nCipher API can be downloaded from the nCipher web site: http://www.ncipher.com.

The nShield and payShield modules module stores keys on the hard disk of the host computer in encrypted form called key blobs.

nShield modules have on-board non-volatile memory. There are new services that enable memory to be allocated as files. Files have Access Control Lists (ACLs,) an ACL can be applied to specific bytes within a file. This enables specific bytes to be used for different purposes. nShield modules have on-board Real-time clock. There are new services to set and read the clock.

nShield modules include a new technology called, the Secure Execution Engine (SEE). This allow users to load a SEE machine. A SEE machine is user written code that implements a specific Software Interrupt interface. This enables users to extend the nCipher API by writing new algorithms or by implementing policy enforcing code on the module. This code is executed in a protected environment. The SEE machine can only access objects stored on the module via the nCipher API. Before a user can send commands to the SEE machine they must create an instance called a SEE World. Each SEE World is treated as a separate user by the module. The services to load a SEE machine and to submit jobs to it and the interface between the nCipher core and the SEE machine are included in the validation. However, the SEE machine itself is excluded from the FIPS 140-1 validation, as this is to be supplied by the user.

The SEE functionality is feature enabled. All payShield modules are supplied with a SEE licence that enables them to run the nCipher Secure payments processing library which supports a number of payments specific algorithms and functions which provide support for 3-D Secure, EMV (Europay, MasterCard, Visa) and PIN processing.

The nShield and payShield modules module can be connected to a computer running one of the following operating systems:

- Windows NT 4.0 for Intel
- Windows 2000
- Solaris
- HP-UX
- AIX
- Linux x86
- FreeBSD x86

Windows NT was used for the validation.

Roles

The nShield and payShield modules module defines the following roles:

User

All users initially connect to the nShield and payShield modules module in the User role. In this role the user can load previously created tokens and use these to load keys from key blobs. Once they have loaded a key they can then use it to perform cryptographic operations as defined by the Access Control List (ACL) stored with the key.

Each key blob contains an ACL that determines what services can be performed on that key. This ACL can require a certificate from a Security Officer authorizing the action. Some actions including writing tokens always require a certificate.

nCipher Security Officer

The nCipher Security Officer (NSO) is responsible for overall security of the module.

The nCipher Security Officer is identified by a key pair, referred to as K_{NSO}. The hash of the public half of this key is stored when the unit is initialized. Any operation involving a module key or writing a token requires a certificate signed by K_{NSO}.

The nCipher Security Officer is responsible for creating the authentication tokens (smart cards) for each user and ensuring that these tokens are physically handed to the correct person.

A user assumes the role of NSO by loading the private half of KNSO and presenting the KeyID for this key to authorize a command.

Junior Security Officer

Where the nCipher Security Officer want to delegate responsibility for authorizing an action they can create a key pair and give this to their delegate who becomes a Junior Security Officer (JSO). An ACL can then refer to this key, and the JSO is then empowered to sign the certificate authorizing the action. The JSO’s keys should be stored on a key blob protected by a token that is not used for any other purpose.

In order to assume the role of JSO the user loads the JSO key and presents the KeyID of this key, and if required the certificate singed by KNSO that delegates authority to the key, to authorize a command.

A JSO can delegate portions of their authority in the same way.

Services available to each role

The module offers non-cryptographic services, which are excluded from FIPS 140-1 and cryptographic services, which are FIPS 140-1 validated. These services are described in the next section.

For more information on each of these services refer to the nCipher Developer’s Guide and nCipher Developer’s Reference.

Non Cryptographic services

Status information

The following services display non cryptographic information about a module and are available to any user without authentication:

- Enquiry
  Returns status information
- Firmware authenticate
  Performs a zero knowledge challenge response protocol that enables a user to ensure that the firmware in the module matches the firmware supplied by nCipher.
- Get module long term key
  Returns the public half of the module’s long term key. This can be used to verify certificates signed with this key.
- Get module signature key
  Returns a handle to the public half of the module’s signing key. this can be used to verify key generation certificates and to authenticate inter module paths.
- Get slot list
  Returns the list of slots available from this module.
- Module Info
  Returns low level status information about the module. This service is designed for use in nCipher’s test routines.
- NoOp
  Does nothing, can be used to determine that the module is responding to commands.
- NVMemList
  Lists a list of files stored in the non-volatile memory.
- Pause for notifications
  This service enables the module to communicate with the nCipher server.
- Statistics enumerate tree
  Lists the statistics available.
- Statistic get values
  Returns a particular statistic.

The status of the module is also indicated by the LED on the front panel.

State	Description
Power Off	Off
Self Test	On
Pre Maintenance	One short flash followed by a long pause
Maintenance	Two short flashes followed by a long pause
Pre Initialisation	One long flash followed by a long pause
Initialisation	Two long flashes followed by a long pause
Operational	On regularly blinking off. The length of time the LED is on reduces with the current load.
Error	Three long flashes followed by a pause, three short flashes followed by a pause, three long flashes followed by a longer pause, an error code and a long pause.

Module state

The following services change the module state.

- Fail
  A test service that causes the module to fail.
- Clear unit
  Clears the module - equivalent to pressing the clear button.
- Initialise unit
  Causes a module in the pre-initialisation state to enter the intiailisation state.
- Maintenance
  Causes a module in the pre-maintenance state to enter the maintenance state.

Note *The nCipher server will only issue these commands if the user is connected with Administrator privileges.*

Arithmetic

The following services perform basic mathematics and are available without authentication:

- Bignum operations
  Performs simple mathematical operations.
- Hash
  Hashes a value
- Modular exponentiation
  Performs a modular expoentiation on values supplied with the command.
- Modular exponentiation using CRT
  Performs a modular expoentiation on values, supplied with the command using Chinese Remainder Theorem.
- Random number
  Generates a random number for use in a application.
- Random prime
  Generates a random prime.

Note *There are separate services for generating keys. These services are designed to enable an application to access the random number source for its own purposes - for example an on-line casino may use* GenerateRandom *to drive its applications.*

FIPS 140-1 services

The following services provide user authentication or cryptographic functionality:

Command/ Service	Role			Description
	Unauth	User / JSO	NSO	
ChangeSharePIN	-	pass phrase	pass phrase	Updates the pass phrase used to encrypt a token share. The pass phrase supplied by the user is not used directly, it is first hashed and then combined with the module key.
ChannelOpen	-	handle, ACL	handle, ACL	Opens a communication channel which can be used for bulk encryption.
ChannelUpdate	-	handle	handle	Performs encryption on a previously opened channel.
CreateBuffer	-	cert	cert	Allocates an area of memory to load data. If the data is encrypted, this service specifies the encryption key and IV used.
CreateSEEWorld	-	handle cert	handle cert	Creates a SEE World, passing in the initialization data stored in a buffer. This command checks the signatures on the buffer. It also specifies whether debugging is allowed or not. Enabling debugging requires a certificate from the nCipher Security Officer.
Decrypt	-	handle, ACL	handle, ACL	Decrypts a cipher text with a stored key returning the plain text.
DeriveKey	-	handle, ACL	handle, ACL	Creates a new key object from a variable number of other keys already stored on the module and returns a handle for the new key.
Destroy	-	handle	handle	Removes an object, if an object has multiple handles as a result of RedeemTicket service, this removes the current handle.
Duplicate	-	handle, ACL	handle, ACL	Creates a second instance of a key with the same ACL and returns a handle to the new instance.
Encrypt	-	handle, ACL	handle, ACL	Encrypts a plain text with a stored key returning the cipher text.
EraseFile	level 2	cert	yes	Removes a file, but not a logical token, from a smart card or software token.
EraseShare	level 2	cert	yes	Removes a share from a smart card or software token.
ExistingClient	yes	yes	yes	Starts a new connection as an existing client.
Export	-	handle, ACL	handle, ACL	Exports a key in plain text. If the unit has been initialized to comply with FIPS 140-1 level 3, this service is only available for public keys.
FeatureEnable	-	cert	cert	Enables a service. This requires a certificate signed by the nCipher Master Feature Enable key.

Command/ Service	Role			Description
	Unauth	User / JSO	NSO	
ForeignTokenCmd	-	handle	handle	Sends an ISO-7816 command to a smart card over the channel opened by ForeignTokenOpen.
ForeignTokenOpen	-	FE, cert	FE	Opens a channel to foreign smart card that accepts ISO-7816 commands. This service cannot be used if the smart card has been formatted using FormatToken. The channel is closed when the card is removed from the reader, or if the handle is destroyed.
FormatToken	level 2	cert	yes	Formats a smart card or software token ready for use.
GenerateKey	level 2	cert	yes	Generates a symmetric key of a given type with a specified ACL and returns a handle. Optionally returns a certificate containing the ACL.
GenerateKeyPair	level 2	cert	yes	Generates a key pair of a given type with specified ACLs for each half or the pair. Performs a pair wise consistency check on the key pair. Returns two key handles. Optionally returns certificates containing the ACL
GenerateKLF	-	FE	FE	Generates a new long term key.
GenerateLogicalToken	level 2	cert	yes	Creates a new logical token, which can then be written as shares to smart cards or software tokens
GetACL	-	handle, ACL	handle, ACL	Returns the ACL for a given handle.
GetAppData	-	handle, ACL	handle, ACL	Returns the application information stored with a key.
GetChallenge	yes	yes	yes	Returns a random nonce that can be used in certificates
GetKeyInfo	-	handle	handle	Superseded by GetKeyInfoEx
GetKeyInfoEx	-	handle	handle	Returns the hash of a key for use in ACLs
GetLogicalTokenInfo	-	handle	handle	Superseded by GetLogicalTokenInfoEx
GetLogicalTokenInfoEx	-	handle	handle	Returns the token hash and number of shares for a logical token
GetRTC	yes	yes	yes	Returns the time according to the on-board real-time clock
GetShareACL	yes	yes	yes	Returns the access control list for a share
GetSlotInfo	yes	yes	yes	Returns status of the physical token in a slot. Enables a user to determine if the correct token is present before issuing a ReadShare command.
GetTicket	-	handle	handle	Gets a ticket - an invariant identifier - for a key. This can be passed to another client or to a SEE World which can redeem it using RedeemTicket
ImpathGetInfo	-	handle	handle	Returns information about an impath

Command/ Service	Role			Description
	Unauth	User / JSO	NSO	
ImpathKXBegin	yes	yes	yes	Creates a new inter-module path and returns the key exchange parameters to send to the peer module.
ImpathKXFinish	-	handle	handle	Completes an impath key exchange. Require the key exchange parameters from the remote module.
ImpathReceive	-	handle	handle	Decrypts application data with the impath decryption key.
ImpathSend	-	handle	handle	Encrypts application data with the impath encryption key.
Import	level 2	cert	yes	Loads a key and ACL from the host and returns a handle. Exports a key in plain text. If the unit has been initialized to comply with FIPS 140-1 level 3, this service is only available for public keys.
InsertSoftToken	yes	yes	yes	Loads a software token from the host disk and 'inserts' it into a slot. Automatically run by the server.
LoadBlob	-	handle	handle	Loads a key that has been stored in a key blob. The user must first have loaded the token or key used to encrypt the blob.
LoadBuffer	-	handle	handle	Loads signed data into a buffer. Several load buffer commands may be required to load all the data, in which case it is the responsibility of the client program to ensure they are supplied in the correct order. Requires the handle of a buffer created by CreateBuffer
LoadLogicalToken	yes	yes	yes	Allocates space for a new logical token - the individual shares can then be assembled using ReadShare or SwallowShare. Once assembeld the token can be used in LoadBlob or MakeBlob commands
MakeBlob	-	handle, ACL	handle, ACL	Creates a key blob containing the key and returns it.
NewClient	yes	yes	yes	Returns a client id.
NVMemAlloc	-	cert	yes	Allocates an area of non-volatile memory as a file and sets the ACLs for this file.
NVMemFree	-	cert	yes	Frees a file stored in non-volatile memory.
NVMemOp	-	cert	ACL	Performs an operation on a file stored in non-volatile memory. Operations include: read, write, increment, decrement, etc.
ReadFile	yes	yes	yes	Reads a file, but not a logical token, to a smart card or software token.
ReadShare	yes	yes	yes	Reads a share from a physical token. Requires Once sufficient shares have been loaded recreates token

Command/ Service	Role			Description
	Unauth	User / JSO	NSO	
RedeemTicket	ticket	ticket	ticket	Gets a handle in the current name space for the object referred to by a ticket. Redeeming a ticket authorizes the user to use that key.
RemoveKM	-	cert	yes	Removes a loaded module key.
RemoveSoftToken	server	server	server	Removes a software token from an emulated slot and writes it to the host disk.
SEEJob	-	cert	yes	Sends a command to a SEE World.
SetACL	-	handle, ACL	handle, ACL	Sets the ACL for an existing key. The existing ACL for the key must allow the operation.
SetAppData	-	handle, ACL	handle, ACL	Stores information with a key.
SetKM	-	cert	yes	Loads a key as a module key.
SetKNSO	nFast, init	nFast, init	-	Superseded by SetNSOPerm. Although this command can still be used it cannot authorize recently added services added.
SetNSOPerm	nFast, init	nFast, init	-	Loads a key hash as the nCipher Security Officer’s Key and sets the security policy to be followed by the module. This can only be performed while the unit is in the initialisation state.
SetRTC	-	cert	yes	Sets the real-time clock.
SetSEEMachine	-	cert handle	handle	Loads the contents of a buffer (created by CreateBuffer / LoadBuffer) as the SEE machine for this module. This command checks the signatures on the buffer.
Sign	-	handle, ACL	handle, ACL	Returns the digital signature of plain text using a stored key.
SignModuleState	-	handle, ACL	handle, ACL	Signs a certificate describing the modules security policy, as set by SetNSOPerm
SquirtShare	-	handle, ACL	handle, ACL	Reads a logical token share and encrypts it under an impath key for transfer to another module where it can be loaded using SwallowShare
SwallowShare	-	handle, encrypted share	handle, encrypted share	Takes a share encrypted with SquirtShare and uses it to recreate the logical token. Requires the handle for the impath that encrypted the share.
TraceSEEWorld	-	cert	yes	Reads debugging output from a SEE World.
Verify	-	handle, ACL	handle, ACL	Verifies a digital signature using a stored key.

Command/ Service	Role			Description
	Unauth	User / JSO	NSO	
WriteFile	-	cert	yes	Writes a file, but not a logical token, to a smart card or software token.
WriteShare	-	cert	yes	Writes a new share to a smart card or software token. The number of shares that can be created is specified when the token is created. All shares must be written before the token is destroyed.

Code	Description
-	The user can not perform this service in this role.
yes	The user can perform this service in this role without further authorization.
handle	The user can perform this service if they possess a valid handle for the resource: key, channel, impath, token, buffers, SEEWorld. The handle is an arbitrary number generated when the object is created. The handle for an object is specific to the user that created the object. The ticket services enable a user to pass an ID for an object they have created to another user or SEEWorld.
ACL	The user can only perform this service with a key if the ACL for the key permits this service. The ACL may require that the user present a certificate signed by a Security Officer or another key.
token or key	A user can only load a key blob if they have previously loaded the token or key used to encrypt the blob.
pass phrase	A user can only load a share, or change the share PIN, if they possess the pass phrase used to encrypt the share. The module key with which the Pass Phrase was combined must also be present.
cert	A user can only perform this service if they are in possession of a certificate from the nCipher Security Officer. This certificate can be tied to a specific key, and hence to a specific smart card.
FE	This service is not available on all modules. It must be enabled using the FeatureEnable service before it can be used.
nFast	A user can only perform this service is they are logged into the host computer as the user nFast.
level 2	The module can be initialized to comply with FIPS 140-1 level 3 by setting the fIPS_level3_compliance flag. If this flag is set: the Generate Key, Generate Key Pair and Import commands require authorization with a certificate signed by the nCipher Security Officer. the Import command fails if you attempt to import a key of a type that can be used to Sign or Decrypt messages. the Generate Key, Generate Key Pair, Import and Derive Key operations will not allow you to create an ACL for a secret key that allows the key to be exported in plain text.

Keys

For each type of key used by the nShield and payShield modules module, the following section describes the access that a user has to the keys.

The nShield and payShield modules module refers to keys by their handle, an arbitrary number, or by its SHA-1 hash.

Security Officer's key

The nCipher Security officer's key must be set as part of the initialisation process. This is a public/private key pair that the nCipher Security Officer uses to sign certificates to authorize key management and other secure operations. The SHA-1 hash of the public half of this key pair is stored in the module EEPROM. The nCipher Security Officer is responsible for this key. If it is generated by the module it may be exported as a key blob under a token.

The public half of this key is included as plain text in certificates.

Long term signing key

The nShield and payShield modules module can store a 160 bits in the EEPROM provided by the Dallas 4320. This data is combined with a discrete log group stored in the module firmware to produce a DSA key. This key can be set by the GenerateKLF service, it can be used to sign a module state certificate using the SignModuleState service and the public value retrieved by the non-cryptographic service Get Long Term Key.

This key is not used to encrypt any other data. It only serves to provide a cryptographic identity for a module that can be included in a PKI certificate chain. nCipher may issue such certificates to indicate that a module is a genuine nCipher module.

Module signing key

When the nShield and payShield modules module is initialized it automatically generate a key pair that it uses to sign certificates. The private half of this pair is stored internally in EEPROM and never released. The pubic half is revealed in plaintext, or encrypted as a key blob under some other key. This key is only ever used to verify that a certificate was generated by a specified module.

Module keys

Module keys are used to protect tokens. The nShield and payShield modulesmodule generates the first module key K_{M0} when it is initialized. This module key is guaranteed never to have been known outside this module.

The Security Officer can load further module keys. These can be generated by the module or may be loaded from an external source. Setting a key as a module key stores the key in EEPROM.

Module keys can not be exported once they have been assigned as module keys. They may only be exported on a key blob protected by an archiving key when they are initially generated.

Logical tokens

A logical token is a Triple DES key used to protect key blobs.

While loaded in the module logical tokens are stored in the object store.

Tokens keys are never exported from the module, except on physical tokens or software tokens. These are protected by a module key and optional pass phrase. Additionally a logical token may be shared over several physical and/or software tokens, with each share protected by a share key.

Logical tokens can be shared between one or more physical token. The properties for a token define how many shares are required to recreate the logical token. Shares can only be generated when a token is created. A share may be further protected using a pass phrase. The pass phrase is combined with a module key, the share number and the smart card id to form a unique Triple DES key used to encrypt the share. This Triple DES key is not stored on the module. It is recalculated every time share is loaded.

Impath keys

An impath is a secure channel between two modules.

To set up an impath two modules perform a validated key-exchange, currently using Diffie Helman but with provision to use other key exchange algorithms if any are added to the module.

The key exchange parameters are signed by the modules signing key Once the modules have validated the signatures the module derives four symmetric keys using a protocol based on SHA-1. Currently symmetric keys are Triple DES.

The four keys are used for encryption, decryption, MAC creation, MAC validation. The protocol ensures that the key Module 1 uses for encryption is used for decryption by module 2.

All impath keys are stored as objects in the object store in SRAM.

Key objects

Keys used for encryption, decryption, signature verification and digital signatures are stored in the module as objects in the object store in SRAM. All key objects are identified by a random identifier that is specific to the user and session.

All key objects are stored with an Access control List or ACL. The ACL specifies what operations can be performed with this key.

Whenever you generate a key or import a key in plain text you must specify a valid ACL for that key type. The ACL can be changed using the SetACL service. The ACL can only be made more permissive if the original ACL includes the ExpandACL permission.

Key objects may be exported as key blobs if there ACL permits this service. Each blob stores a single key and an ACL. The ACL specifies what operations can be performed with this copy of the key. The ACL stored with the blob must be at least as restrictive as the ACL associated with the key object from which the blob was created. When you load a key blob, the new key object takes its ACL from the key blob. Working key blobs are encrypted under a logical token. Key objects may also be exported as key blobs under an archiving key.

Key objects can only be exported in plain text if their ACL permits this operation. If the module has been initialized to comply with FIPS 140-1 level 3 the ACL for a private or secret key cannot include the export as plain service.

A user may pass a key to another user - or to a SEE World - using the ticketing mechanism. The GetTicket mechanism takes a key identifier and returns a ticket. This ticket refers to the key identifier - it does not include any key data. A ticket can be passed as a byte block to the other user who can then use the RedeemTicket service to obtain a key identifier for the same object that is valid for their session. As the new identifier refers to the same object the second user is still bound by the original ACL.

Session keys

Keys used for a single session are generated as required by the module. They are stored along with their ACL as objects in the object store.

Certificate signing keys

The ACL associated with a key object can call for a certificate to be presented to authorize the action. The required key can either be the nCipher Security Officer's key or any other key. Keys are specified in the ACL by an identifying key SHA-1 hash.

Certain services can require certificates signed by the nCipher Security Officer.

Firmware Integrity Key

All firmware is signed using a DSA key pair. A module only loads new firmware if the signature decrypts and verifies correctly.

The private half of this key is stored at nCipher.

The public half is included in all firmware. The firmware is stored in flash memory when the module is switched off, this is copied to SRAM as part of the start up procedure.

Firmware Confidentiality Key

All firmware is encrypted to prevent casual decompilation.

The encryption key is stored at nCipher's offices and is in the firmware.

The firmware is stored in flash memory when the module is switched off, this is copied to SRAM as part of the start up procedure.

nCipher Master Feature Enable Key

For commercial reasons not all nCipher modules offer all services. Certain services must be enabled separately. In order to enable a service the user presents a certificate signed by the nCipher Master Feature Enable Key. This causes the module to set the appropriate bit in the Dallas EEPROM.

The private half of this key is stored at nCipher's offices.

The public half is included in the firmware. The firmware is stored in flash memory when the module is switched off, this is copied to SRAM as part of the start up procedure.

Rules

Identification and authentication

All communication with the nShield and payShield modules module is performed via a server program running on the host machine, using standard inter process communication, using sockets in UNIX operating system, named pipes under Windows NT.

In order to use the module the user must first log on to the host computer and start an nCipher enabled application. The application connects to the nCipher server, this connection is given a client ID, a 120-bit arbitrary number.

Before performing any service the user must present the correct authorization. Where several stages are required to assemble the authorization, all the steps must be performed on the same connection.

Access Control

Keys are stored on the host computer's hard disk in an encrypted format, known as a key blob. In order to load a key the user must first load the token used to encrypt this blob.

Tokens can be divided into shares. Each share can be stored on a smart card or software token on the computer's hard disk. These shares are further protected by encryption with a pass phrase and a module key. Therefore a user can only load a key if they possess the physical smart cards containing sufficient shares in the token, the required pass phrases and the module key are loaded in the module.

Module keys are stored in EEPROM in the module. They can only be loaded or removed by the nCipher Security Officer, who is identified by a public key pair created when the module is initialized. It is not possible to change the Security Officer's key without re initializing the module, which clears all the module keys, thus preventing access to all other keys.

The key blob also contains an Access Control List that specifies which services can be performed with this key, and the number of times these services can be performed. These can be hard limits or per authorization limits. If a hard limit is reached that service can no longer be performed on that key. If a per-authorization limit is reached the user must reauthorize the key by reloading the token.

All objects are referred to by handle. Handles are cross-referenced to client IDs. If a command refers to a handle that was issued to a different client, the command is refused. Services exist to pass a handle between clientIDs.

Object re-use

All objects stored in the module are referred to by a handle. The module's memory management functions ensure that a specific memory location can only be allocated to a single handle. The handle also identifies the object type, and all of the modules enforce strict type checking. When a handle is released the memory allocated to it is actively zeroed.

Error conditions

If the nShield and payShield modules module cannot complete a command due to a temporary condition, the module returns a command block with no data and with the status word set to the appropriate value. The user can resubmit the command at a later time. The server program can record a log of all such failures.

If the nShield and payShield modules module encounters an unrecoverable error it enters the error state. This is indicated by the status LED flashing in the Morse pattern SOS. As soon as the unit enters the error state all processors stop processing commands and no further replies are returned. In the error state the unit does not respond to commands. The unit must be reset.

Physical security

The nShield module is enclosed in a steel case made of three pieces of 1 mm steel.

The unit is sealed with tamper evident seals from Advantage technology.

In order to prevent access to the unit via the ventilation slots, the slots in the base and lid are offset so that there is no straight line path though the two sets of slots.

All components within the security boundary of the nShield module are covered by epoxy resin.

Security Boundary

The security boundary is the steel case.

The following items are excluded from FIPS 140-1 validation as they are not security relevant.

- SCSI termination fuse
- power connector
- jumper block
- smart card interface board
- anti-spark capicators

Code written for SEE machines are excluded from the FIPS 140-1 validation. The validation shows that there is a secure interface between the cryptographic core and the SEE machine, and that a malicious SEE machine cannot gain access to objects protected by the cryptographic core.

To comply with FIPS 140-1 level 3

The nCipher enabled application must perform the following services, for more information refer to the nCipher Security Officer’s Guide and Technical Reference Manual.

To initialise a module

1. Fit the initialisation link and restart the module
2. Use the Initialise command to enter the Initialisation state.
3. Generate a key pair to use a Security Officer’s key.
4. Generate a logical token to use to protect the Security Officer’s key.
5. Write one or more shares of this token onto software tokens.
6. Export the private half of the Security Officer’s key as a key blob under this token.
7. Export the public half of the Security Officer’s key as plain text.
8. Use the Set Security Officer service to set the Security Officer’s key and the operational policy of the module. In order to comply with FIPS 140-1 level 3 you must set at least the following flags:
   - NSOPerms_ops_ReadFile
   - NSOPerms_ops_WriteFile
   - NSOPerms_ops_EraseShare
   - NSOPerms_ops_EraseFile
   - NSOPerms_ops_FormatToken
   - NSOPerms_ops_GenerateLogToken
   - NSOPerms_ops_SetKM
   - NSOPerms_ops_RemoveKM
   - NSOPerms_ops_StrictFIPS140
9. Keep the tokens and key blobs safe.
10. You can create extra module keys in order to distinguish groups of users.
11. You may want to create working keys and user authorization at this stage.
12. Remove the initialisation link and restart the module.

To create a new user

1. Create a logical token.
2. Write one or more shares of this token onto software tokens.
3. For each key the user will require, export the key as a key blob under this token.
4. Give the user any pass phrases used and the key blob.

To authorize the user to create keys

1. Create a new key, with an ACL that only permits UseAsSigningKey. This action may need to be authenticated.
2. Export this key as a key blob under the users token.
3. Create a certificate signed by the nCipher Security Officer’s key that:
   - includes the hash of this key as the certifier
   - authorizes the action GenerateKey or GenerateKeyPair depending on the type of key required.
   - if the user needs to store the keys, enables the action MakeBlob, limited to their token.
4. Give the user the key blob and certificate.

To authorize a user to act as a Junior Security Officer

1. Generate a logical token to use to protect the Junior Security Officer’s key.
2. Write one or more shares of this token onto software tokens
3. Create a new key pair,
   - Give the private half an ACL that permits Sign and UseAsSigningKey.
   - Give the public half an ACL that permits ExportAsPlainText
4. Export the private half of the Junior Security Officer’s key as a key blob under this token.
5. Export the public half of the Junior Security Officer’s key as plain text.
   - Create a certificate signed by the nCipher Security officer’s key includes the hash of this key as the certifier
   - authorizes the actions GenerateKey, GenerateKeyPair
   - authorizes the actions GenerateLogicalToken, WriteShare and MakeBlob, these may be limited to a particular module key.
6. Give the Junior Security Officer the software token, any pass phrases used, the key blob and certificate.

To authenticate a user to use a stored key

1. Use the LoadLogicalToken service to create the space for a logical token.
2. Use the ReadShare service to read each share from the software token.
3. Use the LoadBlob service to load the key from the key blob.
4. The user can now perform the services specified in the ACL for this key.

Note *To assume Security Officer role load the Security Officer’s key using this procedure. The Security Officer’s key can then be used in certificates authorising further operations.*

To authenticate a user to create a new key

1. If you have not already loaded your user token, load it as above.
2. Use the LoadBlob service to load the authorization key from the key blob.
3. Use the KeyId returned to build a signing key certificate.
4. Present this certificate with the certificate supplied by the security officer with the GenerateKey, GenerateKeyPair or MakeBlob command.

Algorithms

FIPS approved algorithms:

DES	Certificate #24
DES MAC	Compliant with FIPS 113 Verified implementation functions: genblk_signbegin,genblk_verifybegin, genblk_macdata, genblk_signend,genblk_verifyend.
Triple DES	Certificate #34 Double and triple length keys Approved for Federal Government Use - Modes are CBC and ECB
Triple DES MAC	Compliant with FIPS 113 Verified implementation functions: genblk_signbegin,genblk_verifybegin, genblk_macdata, genblk_signend,genblk_verifyend
DSA/SHA-1	Certificate #11
RSA	Vendor Affirmed to PKCS #1
HMAC SHA-1	Vendor Affirmed

Non-FIPS approved algorithms

- CAST 5 (RFC2144)
- CAST 6 (RFC2612)
- Rijndael (AES)
- Blowfish
- Twofish
- Serpent
- Arc Four (compatible with RC4)
- El Gamal
- Diffie-Helman
- MD5
- MD2
- SHA-256
- SHA-384
- SHA-512
- RIPEMD 160
- HMAC (MD2, MD5, SHA-256, SHA-384, SHA-512, RIPEMD160)
- XOR (used in some key derivation mechanisms).