The nToken security policy

nCIPHER™

Date: **13 March 2006**

Version: **1.4.28**

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 nToken is a FIPS 140-2 level 2 module, with level 3 physical security. It is designed to protect a single signing key used to identify a host. Use of this key proves to an nCipher netHSM that the session was instigated by a client running on that host.

When a user attempts to connect to a nCipher netHSM, the nCipher server will connect to the nToken as a user and have the nToken sign the messages that are required to authenticate the host to the netHSM.

Although the nToken uses a cut down version of the nCore API used by other nCipher modules. Most of the services available on standard nCipher modules are disabled on nTokens. The nCipher server knows which modules are nTokens and will not send commands from users to the nToken.

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 is 2.18.15.

The nToken connects to the host computer via a PCI bus. The nToken must be accessed by a custom written application.

The nToken can be connected to a computer running one of the following operating systems:

- Windows 2000 and 2003
- Solaris
- HP-UX
- AIX
- Linux x86
- FreeBSD x86

Linux was used for the validation.

Initialising the nToken

When the module is initialised it generates a random AES key for use as a module key. This key is stored in the modules EEPROM and is never revealed.

In order to enroll an nToken the administrator uses the nTokenEnroll utility to

1. generate a DSA key pair
2. store the private half as a key blob protected by the module key.
3. export a certificate containing the public key and the nTokens Electronic Serial number to the nCipher netHSM.
4. display the SHA-1 hash of the key on the host computer's display.

Using the nToken

Once the nToken is enrolled - whenever the nCipher server is started it loads the key blob created when the module was initialised obtaining a key ID for the signing key.

Then whenever a user starts a new session, the nCipher server has the module sign a message confirming the identity of the nCipher sever and sends this message to the netHSM. The netHSM verifies the signature to determine whether to accept the commands.

Upgrading Firmware

Although nCipher do not expect that the user will ever need to update the firmware in an nToken, nCipher provide a utility to perform this.

Verifying firmware

The user can use the fwcheck utility provided by nCipher to verify that the module has been programmed with valid firmware. This utility takes a copy of the signed and encrypted firmware file and performs a a zero knowledge challenge response protocol based on HMAC that enables a user to ensure that the firmware in the module matches the firmware supplied by nCipher.

nCore API

The nCipher server knows that the module is an nToken and will not use it for any other commands.

Keys

For each type of key used by the nToken, the following section describes the access that a user has to the keys. The nToken refers to keys by their handle, an arbitrary number, or by its SHA-1 hash.

Module key

The Module key is an AES key used to protect other keys. This key is generated by the module key when it is initialized. The module key is guaranteed never to have been known outside this module.

Authentication Key

When the nToken is enrolled it generates a DSA key pair used for signature generation. The public half of this key is exported in plain text and has to be transferred to the netHSM.

The private half is encrypted under the module key and exported an an nCipher format key blob which is stored on the host computer's hard disk.

In order to use the signature key, the user loads the key blob. When the key is loaded in the module it is stored in DRAM and identified by a random identifier that is specific to the user and session. The user can then have the module sign messages by providing this identifier.

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 SDRAM as part of the start up procedure.

Firmware Confidentiality Key

All firmware is encrypted using Triple DES 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 SDRAM as part of the start up procedure.

nCipher Master Feature Enable Key

The nToken uses the same firmware as the nCipher nForce and nShield modules. However most functionality is disabled using nCipher's Feature Enable Mechanism. This controls features using certificates signed by the nCipher Master Feature Enable Key. Presenting such a certificate causes the module to set the appropriate bit in the Dallas EEPROM.

The nCipher Master Feature Enable Key is a DSA key pair, The private half of this key pair is stored at nCipher's offices. The public half of the key pair is included in the firmware. The firmware is stored in flash memory when the module is switched off, this is copied to SDRAM as part of the start up procedure.

Roles

The module has three roles:

- Administrator
  Generates or replaces the Module key. This step is performed by nCipher before the module is shipped, but can be repeated by the customer. Generates the signing key. To assume the administrator role you run the nTokenEnroll utility. This creates a key blob that is used to authenticate the user and exports the public key that can be used to verify messages.
- User
  The user role uses the key to sign messages.
  To assume the user role you present the key blob generated by the administrator. If this blob load correctly you are returned an ObjectID for the signing key.
  To sign a message you send the Sign command with the KeyID and message. The module returns the signed message.

Services

The following services are provided by the nToken..

Service	Key type	Role	Description
Generate Key	AES	Admin	The module automatically generates a AES key used as the Module key. This key is never exported.
Generate Key	DSA	Admin	The module generates a DSA key pair, used to sign messages.
Wrap Key	AES + HMAC	Admin	The private DSA key is wrapped as an nCipher key blob. Wrapping uses AES CBC mode for encryption and HMAC-SHA-1 for integrity.
Export	DSA public	Admin	The public half of the DSA key pair is exported in plain text.
Hash	SHA-1	Admin	The module exports the SHA-1 hash of the DSA private key. The hash is constructed from the public components, the hash of the public and private halves are therefore identical. This fact can be used by someone relying on the signature to check that the correct public key is being used to verify the message.
Unwrap Key	AES + HMAC	User	The user presents the wrapped private key at the start of a session. If the key unwraps and the MAC verifies, the user is authorized.
Sign	DSA private	User	Once the user has loaded the private key they can use it to sign messages.
Zero	DSA private	Admin/User	Clears all unwrapped keys. The Module key can be zeroed by repeating the Key Generation Service.
Show Status		Admin/User	Displays status information.

Rules

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 nToken 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 nToken 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.

Security Boundary

The physical security boundary is the plastic jig that contains the potting on both sides of the PCB.

All cryptographic components are covered by potting.

The following items are excluded from FIPS 140-2 validation as they are not security relevant.

- status LED
- PCI connector
- mode links
- clear button

Status information

The module has an LED that indicates the overall state of the module.

The module also returns a status message in the reply to every command. This indicates the status of that command.

Physical security

All security critical components of the nToken are covered by epoxy resin.

The module has a clear button. Pressing this button put the module into the self-test state, clearing all stored key objects and running all self-tests. The long term security critical parameters, module keys, module signing key and Security Officer's key can be cleared by regenerating the Module key.

Checking the module

To ensure physical security, the administrator should regularly examine the epoxy resin security coating for obvious signs of damage.

Strength of functions

Attacking Object IDs

A user is authenticated by a key blob, which is encrypted by a 256-bit AES key with integrity provided by a 160-bit HMAC key. It is therefore almost impossible to spoof this authentication.

An attacker may however get the module to sign a message with the stored key by guessing the client and key key identifiers.

Connections are identified by a ClientID, a random 32 bit number.

Objects are identified by an ObjectID again this is a random 32 bit number.

In order to randomly gain access to a key loaded by another user you would need to guess two random 32 bit numbers. The module can process about 2^{16} commands per minute - therefore the chance of succeeding within a minute is $2^{16} / 2^{64} = 2^{-48}$.

Algorithms

FIPS approved algorithms:

AES	Certificate #15
Triple DES	Certificate #34
	Double and triple length keys Approved for Federal Government Use - Modes are CBC and ECB
DSA	Certificate #113
SHA-1	Certificate #255
HMAC-SHA-1	Certificate #3
RSA	PKCS1.5 Signature Generation and Verification Certificate #16
Random Number Generator	(FIPS 186-2 Change Notice 1 SHA-1) Certificate #20