nFast Cryptographic accelerators security policy
nFast 75/CA-1C, nFast 150/CA-1C, nFast 300/CA-1C

Purpose

The nFast accelerator is a multi-tasking hardware accelerator that is optimized for performing modular arithmetic on very large integers. The nFast accelerator also offers a complete set of key management protocols.

The nFast accelerators nFast 75/CA-1C, nFast 150/CA-1C, nFast 300/CA-1C are FIPS 140-1 level 3 devices.

The units are identical in operation and only vary in the processing speed. The number refers to the approximate number of 1024-bit modular exponentitions a second the model is capable of performing .

The nFast accelerator connects to the host computer via a SCSI-2 bus. The nFast accelerator must be accessed by a custom written application. Full documentation for the nFast API can be downloaded from the nCipher web site: http://www.ncipher.com.

The nFast accelerator stores keys on the hard disk of the host computer in encrypted form called key blobs.

The encryption key is stored as shares on one or more smart cards smart card encrypted under a key that does not leave the module.

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

- Windows NT 4.0 for Intel
- Solaris 2.5, 2.6
- HP-UX 10.10, 10.20
- AIX 4.1, 4.2
- Linux x86
- FreeBSD x86
- NetBSD x86

Windows NT was used for the validation.

Roles

The nFast accelerator defines the following roles:

User

All users initially connect to the nFast accelerator 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 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 authorising the action. Some actions including writing tokens always require a certificate.

nFast Security Officer

The nFast Security Officer is responsible for overall security of the module.

The nFast 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 initialised. Any operation involving a module key or writing a token requires a certificate signed by K_{NSO}.

The nFast Security Officer is responsible for creating key blobs and tokens and distributing these to users.

Junior Security Officer

Where the nFast Security Officer want to delegate responsibility for authorising an action they can create a key pair and give this to their delegate who becomes the JSO. An ACL can then refer to this key, and the JSO is then empowered to sign the certificate authorising 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.

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

Services available to each role

The following table lists the services available to a user in each role and for each service list the additional authorisation required to perform this service.

Non Cryptographic services

The following services do not provide cryptographic functionality.

- Enquiry
- Modular exponentiation
- Exponentiation using CRT
- Random number
- Random prime
- Get slot list
- Get slot information
- Load logical token
- Format token
- Insert software token
- Remove software token
- Get module key list
- Get module signature key
- Fail
- Clear unit
- New client
- Existing client
- Which module
- Merge keys
- Statistics Enumerate Tree
- Statistic Get Values
- Pause For Notifications
- Bignum Operations
- Hash

The Random number and Random prime functions are not used in key generation. They are only used if an external application requires a random number or prime number.

Self test

The self test service is performed automatically when the accelerator is switched on and whenever it is reset by pressing the clear button on the front panel.

FIPS 140-1 services

The following services provide cryptographic functionality

Table 1 Cryptographic services available in each role.

Command/Service	Role			
	Unauth	User	JSO	NSO
Read token data	-	cert	cert	yes
Write token data		cert	cert	yes
Delete token data	-	cert	cert	yes
Load as module key	-	cert	cert	yes
Remove module key	-	cert	cert	yes
Initialise unit	nFast, init	nFast, init	nFast, init	nFast, init
Set Security Officer	nFast, init	nFast, init	nFast, init	nFast, init
Import key	-	level 3 test	level 3 test	level 3 test
Generate key/Generate key pair	-	level 3 test	level 3 test	level 3 test
Duplicate key	-	handle, ACL	handle, ACL	handle, ACL
Get key information	-	handle, ACL	handle, ACL	handle, ACL
Destroy	-	handle, ACL	handle, ACL	handle, ACL
Load blob	-	token or key	token or key	token or key
Make key blob	-	handle, ACL	handle, ACL	handle, ACL
Export key	-	handle, ACL	handle, ACL	handle, ACL
Get application information	-	handle, ACL	handle, ACL	handle, ACL
Set application information	-	handle, ACL	handle, ACL	handle, ACL
Encrypt	-	handle, ACL	handle, ACL	handle, ACL
Decrypt	-	handle, ACL	handle, ACL	handle, ACL
Verify	-	handle, ACL	handle, ACL	handle, ACL
Sign	-	handle, ACL	handle, ACL	handle, ACL
Generate logical token	-	cert	cert	yes
Read share	-	passphrase	passphrase	passphrase
Write share	-	cert	cert	yes
Delete share	-	cert	cert	yes
Change share PIN	-	passphrase	passphrase	passphrase
Channel Open	-	handle, ACL	handle, ACL	handle, ACL
Channel Update	-	channel ID	channel ID	channel ID

Table 1 Cryptographic services available in each role.

Command/Service	Role			
	Unauth	User	JSO	NSO
Derive Key	-	handle, ACL	handle, ACL	handle, ACL
Get ACL	-	handle, ACL	handle, ACL	handle, ACL

Table 2 Authorisation required for each service

Code	Description
-	The user can not perform this service in this role.
yes	The user can perform this service in this role without further authorisation.
handle	The user can perform this service if they possess a valid handle for the key. The handle is an arbitrary number generated when the key is loaded.
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.
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.
passphrase	A user can only load a share, or change the share PIN, if they possess the passphrase 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 in this role if they are in possession of a certificate from the nFast Security Officer.
nFast	A user can only perform this service is they are logged into the host computer as the user nFast.
init	Initialising a unit requires physical access to the unit. It destroys all stored keys. Once you have initialised a unit you must define a new NSO before the unit will enter the operational state.
Channel ID	The ChannelUpdate command requires a ChannelID returned by ChannelOpen. The ChannelID must be presented by the user to whom it was issued.
level 3 test	In order to enforce Level 3 compliance for FIPS 140-1 sections 3, 8 and 11: the unit must be initialised with the FIPS_level3_compliance flag set. If this flag is set: the Generate Key, Generate Key Pair and Import commands require authorisation with a certificate signed by the nfast 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.

Services

For more information on each of these services refer to the nFast Technical Reference Manual

Table 3 Description of each service

Service	Description
Enquiry	Returns status information about the accelerator
Modular exponentiation	Returns A^P MOD_N given A P and N
Exponentiation using CRT	Returns A^D MOD PQ given A, D, P and Q
Random number	Generate a random number
Random prime	Generates a random number that passes Rabin Millar primality test.
Pause For Notifications	This is a command which merely waits for a certain period of time before returning which the unit can use to return asynchronous notifications.
Statistics Enumerate Tree	Lists the statistics that can be returned.
Statistic Get Values	Gather statistics from the whole of the nFast system. These statistics do not reveal any cryptographic information.
Bignum Operations	Performs simple arithmetic on big numbers
Hash	This command hashes a message
Generate key	Generates a symmetric key of a given type with a specified ACL and returns a handle. Optionally returns a certificate containing the ACL.
Generate key pair	Generates a key pair of a given type with specified ACLs for each half or the pair. Performs a pairwise consistency check on the key pair. Returns two key handles. Optionally returns certificates containing the ACL
Import key	Loads a key and ACL from the host and returns a handle. If the unit is in its FIPS 140-1 level 3 mode, this service is only available for public keys .
Duplicate key	Creates a second instance of a key with the same ACL and returns a handle to the new instance.
Get key information	Returns the hash of a key for use in ACLs
Destroy	Removes a key
Load blob	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.
Make key blob	Creates a key blob containing the key and returns it.
Export key	Exports a key in plain text. If the unit is in its FIPS 140-1 level 3 mode, this service is only available for public keys .

Table 3 Description of each service

Service	Description
Get application information	Returns the application information stored with a key.
Set application information	Stores information with a key.
Encrypt	Encrypts a plain text with a stored key returning the cipher text.
Decrypt	Decrypts a cipher text with a stored key returning the plain text.
Verify	Verifies a digital signature using a stored key.
Sign	Returns the digital signature of plain text using a stored key.
Generate logical token	Creates a new logical token.
Get slot list	Returns a list of slots present in the accelerator,
Get slot information	Returns information about the token shares stored on the token in a specified slot.
Load logical token	Allocates space for a new logical token
Read share	Reads a share from a token. Once sufficient shares have been loaded recreates token
Write share	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.
Delete share	Removes a share from a smart card or software token.
Change share PIN	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.
Format token	Formats a software token ready for use.
Insert software token	Loads a software token from the host disk and ‘inserts’ it into a slot.
Remove software token	Removes a software token from an emulated slot and writes it to the host disk.
Read token data	Reads a file, but not a logical token, to a smart card or software token.
Write token data	Writes a file, but not a logical token, to a smart card or software token.
Delete token data	Removes a file, but not a logical token, from a smart card or software token.
Load as module key	Loads a key as a module key.
Remove module key	Removes a loaded module key.
Get module key list	Returns a list of the hash of the module keys.
Get module signature key	Returns the handle of the public half of the modules signing key.

Table 3 Description of each service

Service	Description
Initialise	Puts the unit into the initialisation state, clearing all stored keys. Requires physical access to the unit.
Set Security Officer	Loads a key hash as the Security Officer's Key. This can only be performed while the unit is in the initialisation state.
Fail	Causes the unit to enter the error state, for test purposes.
Clear unit	Resets the unit removing all temporary keys, but not module keys. Performs power on self tests.
New client	Returns a client id.
Existing client	Starts a new connection as an existing client.
Which module	Returns the list of modules on which a key is loaded.
Merge keys	Where one key is loaded on several modules with different handles, returns a new handle that can be used to refer to any of these handles.
Channel Open	Opens a communication channel which can be used for bulk encryption.
Channel Update	performs encryption on a previously opened channel.
Derive Key	Creates a new key object from a variable number of other keys already stored on the accelerator and returns a KeyId for the new key.
Get ACL	Returns the ACL for a given KeyID.

Keys

For each type of key used by the nFast accelerator, the following section describes the access that a user has to the keys.

The nFast refer to keys by their handle, an arbitrary number, or by its SHA-1 hash.

Accelerator signing key - private half

This key is used to sign certificates created by accelerator. The user has no access to this key in any role.

Accelerator signing key - public half

This key is used to verifying certificates created by accelerator. Users can export this in plain text in any role.

Module Key

These keys are used for encrypting tokens. Any user may export a list of SHA-1 hashes of the module keys to determine whether a required key is present. The nFast Security Officer can load keys as module keys and remove module keys. The Security Officer can export the key for in an encrypted from for archival before it is loaded as a module key.

Security Officers key - private half

This key is used to sign certificates authorising other users to perform tasks. The NSO can output this key as a key blob. No other user can access this key.

Security Officers key - public half

This key is used to verify certificates. The nFast Security Officer can export this as a key blob or in plain text. The key is in plain text in certificates so other users who are given a certificate may see this key.

Token

Tokens are never output in plain text. Any user can retrieve a list of SHA-1 hashes of the tokens they have loaded. They have no access to tokens loaded by other users. Tokens are exported as encrypted shares. The nFast Security Officer can create tokens under any module key. Junior Security Officer's can create tokens if they have a certificate from the nFast Security Officer. This certificate can restrict the module keys under which the JSO can create keys.

Working keys

Keys used for encryption, decryption, applying and verifying signatures are stored in key blobs. A user can access a key only if they possess the key blob and can load the token under which the blob was encrypted. The blob contains an ACL that lists the services that can be performed with the key.

Note *In order to comply with FIPS 140-1 level 3: all secret and private keys must have ACLs that prevent output in plain text.*

An ACL can require a certificate authorising a service. The keys used to sign these certificates are themselves stored in key blobs. The Security Officer loads these keys in the same way as any other key.

Users can only access keys that they have loaded; they cannot access keys loaded by other users.

Rules

Identification and authentication

All communication with the nFast accelerator 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 accelerator the user must first log on to the host computer and start an nFast enabled application. The application connects to the nFast server, this connection is given a client ID, a 120-bit arbitrary number.

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 sufficient shares in the token, the required pass phrases and the module key are loaded in the accelerator.

Module keys are stored in EEPROM in the accelerator. They can only be loaded or removed by the nFast Security Officer, who is identified by a public key pair created when the accelerator is initialised. It is not possible to change the Security Officer's key without reinitialising the accelerator, 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 authorisation limits. If a hard limit is reached that service can no longer be performed on that key. If a per-authorisation limit is reached the user must reauthorise 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.

Object re-use

All objects stored in the accelerator are referred to by a handle. The accelerator'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 accelerators enforce strict type checking. When a handle is released the memory allocated to it is actively zeroed.

Error conditions

If the nFast accelerator cannot complete a command due to a temporary condition, the accelerator 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 nFast encounter 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 nFast accelerator 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.

To comply with FIPS 140-1 level 3

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

To initialise an accelerator

1. Fit the initialisation link and restart the accelerator
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 smart cards or 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 accelerator. 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 authorisation at this stage.

12 Remove the initialisation link and restart the accelerator.

To create a new user

1 Create a logical token.

2 Write one or more shares of this token onto smart cards or 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 authorise 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 nFast Security Officer's key that:
- includes the hash of this key as the certifier
- authorises 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 authorise 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 smart cards or 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 nFast Security officer's key includes the hash of this key as the certifier
- authorises the actions GenerateKey, GenerateKeyPair
- authorises the actions GenerateLogicalToken, WriteShare and MakeBlob, these may be limited to a particular module key.

6 Give the Junior Security Officer the smart card or 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 smart card or 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 authorisation 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.