The nForce module security policy

nForce 150 PCI, nForce 300 PCI

nCipher

nCipher

Date: **16 January 2004**

Version: **1.0.22**

Patents

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

Purpose

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

The nForce modules: nForce 150 PCI, nForce 300 PCI are FIPS 140-1 level 2 embedded devices. The units are identical in operation and only vary in the processing speed.

The units are identical in operation and only vary in the processing speed.

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 1.71.15 and 1.71.91.

The nForce module connects to the host computer via a PCI bus. The nForce 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 nForce module stores keys on the hard disk of the host computer in encrypted form called key blobs.

The nForce 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 nForce module defines the following roles:

User

All users initially connect to the nForce 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 authorising 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 initialised. 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 authentication tokens (smart cards) for each user. The security officer must establish the actual identity of the user before they given them the authentication token.

Junior Security Officer

Where the nCipher Security Officer want to delegate responsibility for authorising 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 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 Junior Security Officer can delegate portions of their authority in 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 module 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:

Cryptographic services available in each role.

Command/Service	Role			
	Unauth	User	JSO	NSO
Change share PIN	-	pass phrase	pass phrase	pass phrase
Channel Open	-	handle, ACL	handle, ACL	handle, ACL
Channel Update	-	channel ID	channel ID	channel ID
Decrypt	-	handle, ACL	handle, ACL	handle, ACL
Delete share	-	cert	cert	yes
Delete token data	-	cert	cert	yes
Derive Key	-	handle, ACL	handle, ACL	handle, ACL
Destroy	-	handle, ACL	handle, ACL	handle, ACL
Duplicate key	-	handle, ACL	handle, ACL	handle, ACL
Encrypt	-	handle, ACL	handle, ACL	handle, ACL
Export key	-	handle, ACL	handle, ACL	handle, ACL
Generate key/Generate key pair	-	level 3 test	level 3 test	level 3 test
Generate logical token	-	cert	cert	yes
Get ACL	-	handle, ACL	handle, ACL	handle, ACL
Get application information	-	handle, ACL	handle, ACL	handle, ACL
Get key information	-	handle, ACL	handle, ACL	handle, ACL
Import key	-	level 3 test	level 3 test	level 3 test
Initialise unit	nFast, init	nFast, init	nFast, init	nFast, init
Load as module key	-	cert	cert	yes
Load blob	-	token or key	token or key	token or key

Cryptographic services available in each role.

Command/Service	Role			
	Unauth	User	JSO	NSO
Make key blob	-	handle, ACL	handle, ACL	handle, ACL
Read share	-	pass phrase	pass phrase	pass phrase
Read token data	-	cert	cert	yes
Remove module key	-	cert	cert	yes
Set application information	-	handle, ACL	handle, ACL	handle, ACL
Set Security Officer	nFast, init	nFast, init	nFast, init	nFast, init
Sign	-	handle, ACL	handle, ACL	handle, ACL
Verify	-	handle, ACL	handle, ACL	handle, ACL
Write share	-	cert	cert	yes
Write token data		cert	cert	yes

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. The handle for an object is specific to the user that created the object. There are services which allow 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.
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.

Authorisation required for each service

Code	Description
cert	A user can only perform this service in this role 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.
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 nCipher Security Officer 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	The module can beinitialised to comply with FIPS 140-1 level 3 roles and services and key management by setting the fIPS_level3_compliance flag. If this flag is set: the Generate Key, Generate Key Pair and Import commands require authorisation 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.

Services

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

Description of each service

Service	Description
Bignum Operations	Performs simple arithmetic on big numbers
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.
Channel Open	Opens a communication channel which can be used for bulk encryption.

Description of each service

Service	Description
Channel Update	performs encryption on a previously opened channel.
Clear unit	Resets the unit removing all temporary keys, but not module keys. Performs power-up self-tests.
Decrypt	Decrypts a cipher text with a stored key returning the plain text.
Delete share	Removes a share from a smart card or software token.
Delete token data	Removes a file, but not a logical token, from a smart card or software token.
Derive Key	Creates a new key object from a variable number of other keys already stored on the module and returns a KeyId for the new key.
Destroy	Removes a key
Duplicate key	Creates a second instance of a key with the same ACL and returns a handle to the new instance.
Encrypt	Encrypts a plain text with a stored key returning the cipher text.
Enquiry	Returns status information about the module
Existing client	Starts a new connection as an existing client.
Exponentiation using CRT	Returns A^D MOD PQ given A, D, P and Q
Export key	Exports a key in plain text. If the unit has been initialised to comply with FIPS 140-1 level 3 roles and services and key management, this service is only available for public keys.
Fail	Causes the unit to enter the error state, for test purposes.
Format token	Formats a software token ready for use.
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
Generate logical token	Creates a new logical token.
Get ACL	Returns the ACL for a given KeyID.
Get application information	Returns the application information stored with a key.

Description of each service

Service	Description
Get key information	Returns the hash of a key for use in ACLs
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.
Get slot information	Returns information about the token shares stored on the token in a specified slot.
Get slot list	Returns a list of slots present in the module,
Hash	This command hashes a message
Import key	Loads a key and ACL from the host and returns a handle. If the unit has been initialised to comply with FIPS 140-1 level 3 roles and services and key management, this service is only available for public keys.
Initialise	Puts the unit into the initialisation state, clearing all stored keys. Requires physical access to the unit.
Insert software token	Loads a software token from the host disk and 'inserts' it into a slot.
Load as module key	Loads a key as a module 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.
Load logical token	Allocates space for a new logical token
Make key blob	Creates a key blob containing the key and returns it.
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.
Modular exponentiation	Returns $A^P \text{ MOD}_N$ given A P and N
New client	Returns a client id.
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.
Random number	Generate a random number
Random prime	Generates a random number that passes Rabin Millar primality test.

Description of each service

Service	Description
Read share	Reads a share from a token. Once sufficient shares have been loaded recreates token
Read token data	Reads a file, but not a logical token, to a smart card or software token.
Remove module key	Removes a loaded module key.
Remove software token	Removes a software token from an emulated slot and writes it to the host disk.
Set application information	Stores information with a key.
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.
Sign	Returns the digital signature of plain text using a stored key.
Statistic Get Values	Gather statistics from the whole of the nCipher system. These statistics do not reveal any cryptographic information.
Statistics Enumerate Tree	Lists the statistics that can be returned.
Verify	Verifies a digital signature using a stored key.
Which module	Returns the list of modules on which a key is loaded.
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.
Write token data	Writes a file, but not a logical token, to a smart card or software token.

Keys

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

The nForce module refers to keys by their handle, an arbitrary number, or by its SHA-1 hash.

Module signing key - private half

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

Module signing key - public half

This key is used to verifying certificates created by module. 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 nCipher Security Officer can load keys as module keys and remove module keys. The Security Officer can export the key in an encrypted form for archival before it is loaded as a module key.

Security Officer's key - private half

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

Security Officer's key - public half

This key is used to verify certificates. The nCipher 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 nCipher Security Officer can create tokens under any module key. Junior Security Officer's can create tokens if they have a certificate from the nCipher 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.

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 normally only access keys that they have loaded; they cannot access keys loaded by other users. However, 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.

Rules

Identification and authentication

All communication with the nForce 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, as listed in the table *Authorisation required for each service* on page 5. 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 initialised. It is not possible to change the Security Officer's key without reinitialising 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 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. 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 nForce 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 nForce 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

All security relevant components of the nForce module are covered by epoxy resin.

Security Boundary

The security boundary is the PCB.

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

- jumper block
- PCI connector

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

To comply with FIPS 140-1 level 3 roles and services and key management

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 roles and services 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 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 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 nCipher 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 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
- 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 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 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.

Algorithms

nForce modules support the following 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

Non-FIPS approved algorithms

CAST 5

Rijndael

Arc Four (compatible with RC4)

El Gamal

Diffie-Helman

MD5

MD2

RIPEMD 160

HMAC (MD2, MD5, SHA-1, SHA-192, SHA-256, RIPEMD160)

XOR (used in some key derivation mechanisms).