Novell International Cryptographic Infrastructure 2.4.0

Security Policy for Solaris 8

www.novell.com

December 23, 2002

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

This product may require export authorization from the U.S. Department of Commerce prior to exporting from the U.S. or Canada.

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.

www.novell.com

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

Access Manager is a registered trademark of Novell, Inc., in the United States and other countries.

ConsoleOne is a trademark of Novell, Inc. NDS is a trademark of Novell, Inc.

NetWare is a registered trademark of Novell, Inc., in the United States and other countries.

Novell is a registered trademark of Novell, Inc., in the United States and other countries.

Novell Client is a trademark of Novell, Inc.

Novell Directory Services is a trademark of Novell, Inc.

Novell International Cryptographic Infrastructure (NICI) is a trademark of Novell, Inc. It includes RSA BSAFE cryptographic software from RSA Security.

Transaction Tracking System is a trademark of Novell, Inc.

TTS is a trademark of Novell, Inc.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

Contents

List of Figures

List of Tables

1 Introduction

The Novell International Cryptographic Infrastructure (NICI) consists of a set of components that have been implemented on a number of different platforms. Versions have been implemented on Novell's NetWare 5.x and 6.x, Microsoft's Windows 2000, Windows NT 4.0, Sun's Solaris, Linux, and AIX. This document describes the Security Policy for NICI version 2.4.0 as it has been implemented for the Solaris 8 EAL4 evaluated platform.

2 Security Requirements

The Novell NICI 2.4.0 Cryptography Library for Solaris 8 conforms to FIPS 140-1 Level 2 as shown in Table 2.1.

Table 2.1: FIPS 140-1 Test Category Levels.

FIPS140-1 Test Category	**Level**
Cryptographic Modules	2
Module Interfaces	2
Roles and Services	2
Finite State Machine Model	2
Physical Security	2
Software Security	2
Operating System Security	2
Key Management	2
Cryptographic Algorithms	2
EMI/EMC	3
Self Tests	2

2.1 Cryptographic Modules

NICI consists of a set of software libraries designed to run on a wide variety of modern operating systems and hardware platforms. This particular Security Policy document pertains to the NICI configuration, running on a Solaris 8 platform. In this configuration, NICI is a shared library (.so). In FIPS 140-1 terms, NICI consists of a set of hardware, software, and firmware that make up a "multi-chip stand-alone module".

The module consists of the following components:

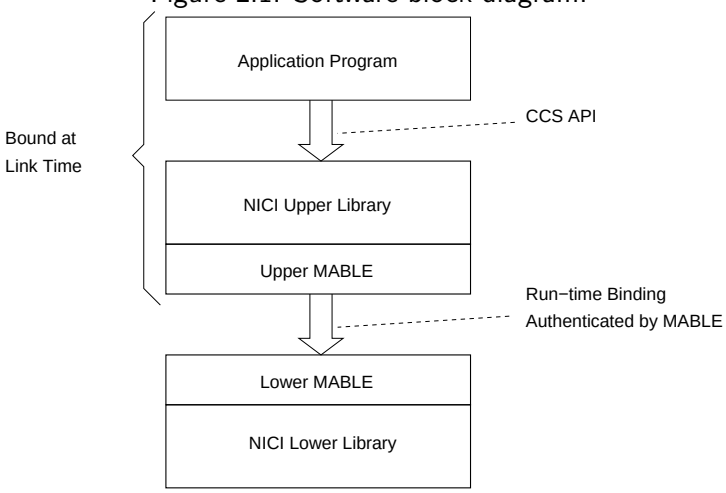
- A C2 TCSEC equivalent system consisting of a hardware platform and operating system software.

  The test system was an EAL4 evaluated configuration of Solaris 8 running on a Sun SPARC Ultra-10. Configuration details are listed in section 3 (Installation Guidance) of this document.

- NICI 2.4.0 for Solaris 8.

  This consists of a matched upper library, which is linked to the application, and a lower library that is installed on the workstation.

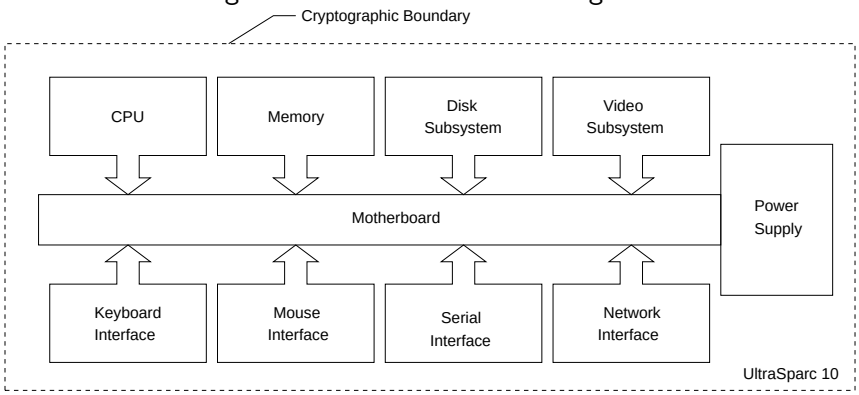
Figure 2.1: Software block diagram.



The cryptographic boundary is defined by the Sun SPARC Ultra-10. Since NICI must be able to store at least one permanent key in order to be able to securely wrap and unwrap other keys, that key is stored in a DES encrypted form per user, encrypted under a key encryption key, protected by the EAL4 operating system's mechanisms. Audit data and stored NICI keys can be zeroized by reformatting the computer's hard drives.

MABLE is the Module Authentication and Binding Library Extensions (patent pending) technology used to authenticate NICI to an application and to provide ongoing binding between an application and NICI as if the application is statically linked to NICI. Upper MABLE is statically linked to an application and contains the challenge generation, certificate verification, and ongoing binding mechanism functions. Lower MABLE is statically linked to NICI and contains the response-to-challenge generation, signature creation, and ongoing binding mechanism functions.

Figure 2.2: Hardware Block Diagram.



2.2 Module Interfaces

FIPS 140-1 defines a cryptographic boundary, and as well as interfaces through which information is allowed to enter and leave the cryptographic boundary. Defining such interfaces is normally straightforward for developers of hardware modules, but developers of software modules are faced with the task of choosing an appropriate set of interface definitions. NICI has the following logical interfaces: data in, data out, control-in, and status out. These interfaces are supported by the API set.

2.2.1 Data Input/Output Interface

FIPS 140-1 requires the definition of Data Input/Output (I/O) and Command/Status interfaces. NICI defines these interfaces through the Controlled Cryptographic Services API. The API provides the means to input and output data. The Data Input/Output interface is active only during the User State.

2.2.2 Command/Status Interface

The FIPS 140-1 Control interface is used to initiate the NICI Module. It is activated by the operating system when an application program asks the operating system to attach NICI and causes it to commence operation. It may also be activated when the operating system commands NICI to shut down. Otherwise, it is active only during the User and Crypto Officer States, when commands are issued via the API set. The Status interface is active only during the User and Crypto Officer States.

2.3 Roles and Services

Novell NICI 2.4.0 is FIPS 140-1 Level 2 compliant for Roles and Services. NICI implements identity-based authentication.

Table 2.2: Roles and Services.

Operation	User Role	Crypto Officer Role
Install NICI		X
Upgrade NICI		X
Configure NICI		X
Zeroize Keys		X
Zeroize Audit Data		X
Encrypt/Decrypt	X	
Generate Keys and Random Data	X	
Sign/Verify	X	

2.3.1 User Role

A “User” is an application program, running as a single or multiple process (perhaps multi-threaded), which has been linked with the Novell NICI interface library. This version of NICI supports multiple processes with different user identities with separation between such multiple instances relying on the

access mechanisms provided by the Solaris 8 operating system. Each instance of NICI has an identity and a separate memory space with access to a unique set of key materials.

All NICI applications must be installed by the Crypto Officer. Access to the NICI library and configuration files are granted by the Crypto Officer using the Solaris 8 file access mechanisms. All installed applications that are granted access to the NICI shared library (.so) are authenticated users. An authenticated user is able to perform crypto operations via the API set defined in the Controlled Cryptography Services Software Development Specification (CCS) document.

NICI maintains a set of persistent unique keys per Solaris 8 operating system user. The Solaris 8 operating system maintains the separation of these sets of keys. All processes with the same Solaris 8 user ID have access to a unique set of keys with independent key generation capability.

2.3.2 Crypto Officer Role

A single Crypto Officer role is supported in NICI as the "root" defined on the Solaris 8 operating system. Authenticating to the Solaris 8 operating system assigns the Crypto Officer role to the "root" user. The purpose of the Crypto Officer is to setup, configure, and reconfigure the NICI software. In addition, the Crypto Officer can zeroize NICI keys and audit data if required. The Crypto Officer is also the security administrator as defined by the Solaris 8 EAL4 operating system.

2.4 Finite State Machine Model

NICI has an embedded finite state machine that is compliant with the FIPS 140-1 specification. The finite state machine is described fully in a separate document that is submitted during the FIPS 140-1 level 2 validation process.

2.5 Physical Security

As a multiple-chip stand-alone cryptographic module, the workstation enclosure for the EAL4 evaluated system must have tamper evident labels placed in a manner so as to prevent undetected access to the inside of the enclosure. Please refer to section 3, "Installation Guidance" for further details.

2.6 Software Security

All NICI software including executable and data files is protected by the Solaris 8 operating system's access control mechanisms covered by its DAC (Discretionary Access Controls) policy and enforced by TSF (TOE Security Function) installed in accordance with its EAL4 evaluation. The shared object module is protected by file system access controls from unauthorized tampering. Solaris 8 operating system's TSF protects NICI configuration files and run-time memory image from tampering and access. Similarly, the NICI configuration file is protected by the operating system's access control mechanisms.

Table 2.3 lists the critical security parameters and their access rights. "Key Encryption Key" is a DES key embedded in the code (see Section 2.1). "DAC Key" is a HMAC-SHA1 key embedded in the code (see Section 2.11.1).

Table 2.3: Critical Security Parameters (CSP).

CSP	Crypto Officer	User
Key Encryption Key	Read/Write	Read-only
DAC Key	Read/Write	Read-only
Audit Data	Read/Write	Read-only

2.7 Operating System Security

NICI 2.4.0 as evaluated requires Solaris 8 installed in its EAL4 evaluated configuration. See Chapter 3, "Installation Guidance" for further information.

2.8 Cryptographic Key Management

NICI provides cryptographic key management services using secret key (symmetric) and public key (asymmetric) algorithms. Secret keys and private keys are protected from unauthorized disclosure, modification, and substitution. Public keys are protected against unauthorized modification and substitution.

NICI key use policies are comprised of key usage flags (encrypt, wrap, sign, etc.), key types (DES, RSA, AES, etc.), and algorithms (RSA, DES, DSA, etc.). NICI keys are listed in Table 2.4. SENSITIVE is an attribute of a key set at key generation time, and EXTRACT is a key usage flag.

A key management key must have wrap and key management encrypt key usage flags set in order to wrap keys. Key type must also match the algorithm used with a particular key. For instance, a DES key can not be used with the RSA algorithm. These combined constitute the NICI key use policies.

It is the application's responsibility to use the FIPS approved APIs, algorithms, and keys to maintain the FIPS 140-1 mode of operation. Use of any one of the non-FIPS algorithms or non-FIPS approved APIs would invalidate the FIPS mode of operation. Loading of software/firmware other than the validated software/firmware will put the module in the non-FIPS mode of operation.

2.8.1 FIPS Approved Key Generation

The G function in the pseudo-random generator described in FIPS 186-2 is constructed using the SHA-1 hash function with b=512.

2.8.2 Key Distribution

NICI key distribution capabilities comply with FIPS 171 options 1 (key exchange role), 4 (MAC), 5 (key and IV generation), 6 (key generation techniques), and 14 (send IV).

NICI has TripleDES and RSA key management keys. The TripleDES and RSA key management keys are generated and used in FIPS mode. NICI uses DES-MAC to ensure integrity of persistent keys in FIPS mode. NICI uses digital signatures to sign certificates. NICI does not use RSA for data encryption in FIPS mode.

Table 2.4: NICI Keys.

Key Name	Key Type / Algorithm	Key Usages	Description
STORAGE	TripleDES	WRAP, UNWRAP, SENSITIVE	Key-wrapping key. NICI-generated and NICI-maintained. FIPS approved.
SESSION	DES, TripleDES	WRAP, UNWRAP, SENSITIVE	Key-wrapping key per connection between a client and a server. NICI-generated, present while the connection is active. FIPS approved.
CA	RSA	SIGN, VERIFY, SENSITIVE	NICI's machine-unique CA key-pair. NICI-generated and NICI-maintained. Not FIPS approved.
PARTITION	DES, TripleDES	WRAP, UNWRAP, SENSITIVE	Security Domain Keys, key wrapping only. NICI-generated and NICI-maintained. FIPS approved.
FOREIGN	Any	Any, EXTRACT, not SENSITIVE	Generator unknown, maybe NICI. Not FIPS approved.
Other	DES, TripleDES, AES	WRAP, UNWRAP, ENCRYPT, DECRYPT, not EXTRACT	NICI-generated. FIPS approved.
Other	HMAC-SHA1, DSA, RSA ANSI X9.31 (vendor-affirmed)	SIGN, VERIFY, not EXTRACT	NICI-generated. FIPS approved.
Other	RSA (encryption), RSA (key distribution)	WRAP, UNWRAP, ENCRYPT, DECRYPT	NICI-generated. Not FIPS approved.

2.8.3 Key Entry and Output

NICI does not possess a manual key entry method; all keys are entered electronically. Aside from the Crypto Officer's role in distributing configuration data (used under the control of the Crypto Officer at installation time), all keys are entered under the User's control via the API interface.

Typical key entry to NICI is done via key unwrapping, i.e., by decrypting the key value, and verifying the integrity of the attributes associated with the key. NICI maintains a storage key that is usable only for key wrapping for this purpose.

In FIPS mode, raw key entry (key injection) and output (key extraction) are not allowed.

2.8.4 Key Storage

When keys have been unwrapped within the confines of the NICI cryptographic module boundary, they are kept in plaintext form. Keys in memory are protected by the EAL4 operating system.

NICI provides key wrapping as a secure way of transferring keys in and out of NICI. NICI maintains a persistent TripleDES key wrapping key to applications (See STORAGE key on Table 2.4). No means is provided to unauthorized applications to obtain this key-management key.

NICI uses RSA digital signatures to sign certificates and to encrypt keys in FIPS mode. NICI does not use RSA for data encryption in FIPS mode.

2.8.5 Key Destruction

When the particular NICI context associated with the usage of a set of keys is closed, all keys associated with that context within NICI are zeroized and destroyed in memory. When NICI itself is closed within a given process all keys in all contexts are zeroized.

In order to destroy the audit data and NICI storage keys, the Crypto Officer must perform a complete reformatting of the hard disk, thoroughly scrubbing the disk to make certain there is no readable residue.

2.9 Cryptographic Algorithms

NICI 2.4.0 supports the following FIPS approved algorithms:

1. DSA (FIPS 186-2)
2. DES and Triple DES (FIPS 46-3 and 81)
3. SHA-1 (FIPS 180-1)
4. RSA signature (X9.31)
5. AES (FIPS 197)
6. HMAC-SHA-1 (FIPS 198)

Non-FIPS approved algorithms that also are supported include:

7. Diffie-Hellman (PKCS#3)
8. RSA encryption/decryption (PKCS#1, RFC 2437)
9. MD2 (RFC 1319)
10. MD4 (RFC 1320)
11. MD5 (RFC 1321)
12. HMAC (RFC 2104)
13. RC2 (RFC 2268)
14. RC4
15. RC5 (RFC 2040)
16. CAST128 (RFC 2144)

17. Password Based Encryption, six algorithms (PKCS#12)
18. UNIX Crypt
19. LMdigest (CIFS)
20. TLS-KeyExchange-RSASign (RFC 2246)
21. NetWarePassword (Novell)

When only FIPS approved algorithms (numbers 1-6) are used, NICI is functioning in FIPS mode. If any non-FIPS approved algorithm (numbers 7-21) is used, NICI is running in non-FIPS mode. It is the application programmer's responsibility to enforce FIPS and Non-FIPS modes of operation.

2.10 EMI/EMC

The EAL4 evaluated system complies with EMI/EMC requirements.

2.11 Self-Tests

NICI conforms to the FIPS 140-1 Level 2 requirements for self-test.

The required start-up self-tests are performed every time the NICI is started by the operating system, prior to transitioning to the User state. If the self-tests do not run correctly, NICI will not start, and an error indication will be returned via the API.

2.11.1 Software Integrity Tests

NICI satisfies the requirements for FIPS 140-1 Level 2 for Power-up Self-Tests.

Cryptographic Algorithms Test

Known answer tests are performed for DES, TDES, AES, HMAC-SHA-1, RSA, and DSA upon startup. Pair-wise consistency tests are performed for RSA and DSA upon startup.

Software/Firmware Test

NICI complies with FIPS 140-1 by storing a DAC for the NICI shared library when the module is installed. This DAC is under the control of the Crypto Officer and is protected by the Solaris 8 EAL4 operating system security. The DAC for the shared library (.so) is calculated using an embedded key at initialization and compared with the stored version. NICI fails initialization if the DAC does not match. NICI is using HMAC-SHA1 to compute the DAC.

2.11.2 Conditional Self Tests

The following tests are performed as specified for each test:

Pair-Wise Consistency Tests (for public/private key pairs)

When a public/private key pair is generated the key pair is tested for pair-wise consistency. The public key is used to encrypt a plaintext value and checked to ensure that an identity mapping did not occur, and then the private key is used to decrypt that value and the value compared to the original. If the values are not identical, the tests fails. If the keys are to be used only for the calculation of a signature, then the consistency is tested by the calculation and verification of a signature. These tests are applied to RSA and DSA keys.

Continuous Random Number Test

The module performs continuous random number generator tests as dictated by FIPS 140-1. Pseudorandom numbers are generated using approved FIPS 186-2 (Appendix 3.1) standard. The random number generator generates blocks of 160 bits.

3 Installation Guidance

3.1 FIPS 140-1 Level 2 Installation Requirements

For NICI version 2.4.0 for Solaris 8 to be compliant with the FIPS 140-1 Level 2 specification the following requirements must be met:

1. NICI must be installed on a EAL4 evaluated computing platform according to Solaris 8.0 Security Release Notes" Sun document number s8.0_125 (see `http://www.sun.com/software/-security/securitycert/docs/SRN_1.0.pdf`) .
2. NICI must be installed using the standard NICI 2.4.0 Installation Program to insure that file permissions are correctly set.
3. The EAL4 evaluated system hardware must have tamper evident labels applied such that removable covers or other parts may not be removed without leaving evidence that an intrusion has taken place. These labels must be kept securely under the control of the security officer.

3.2 Evaluated Configuration

NICI 2.4.0 was evaluated in the following configuration:

1. EAL4 evaluated computing platform consisting of a Sun SPARC Ultra-10 with Solaris 8 installed as specified by Solaris 8.0 Security Release Notes" Sun document number s8.0_125 (see `http://www.sun.com/software/security/securitycert/docs/SRN_1.0.pdf`) .
2. NICI was installed using the standard installation program.
3. The labels used were Bay Area Labels Voidable Mylar Labels. As shown in figure 3.1, two labels were applied to secure the removeable cover. These labels were left in place for at least 24 hours prior to the test in accordance with the manufacturer's specifications.

Figure 3.1: Tamper-evident Label Placement on a Sun SPARC Ultra-10.

Appendix A

FIPS Mode CCS API Definitions

For complete descriptions, please refer to the *Controlled Cryptography Services Software Development Specifications* document available from Novell.

API	Description
CCS_Init	Initializes the CCS library
CCS_Shutdown	Closes the CCS library
CCS_GetInfo	Return information about the CCS interface
CCS_GetPolicyInfo	Determines the policy constraints on key attributes for a given type and usage
CCS_GetKMStrength	Returns the key management strength level
CCS_GetRandom	Returns a random number
CCS_GetAlgorithmInfo	Obtain information about a specific algorithm
CCS_GetAlgorithmList	Obtain information about the algorithms available in the system.
CCS_GetMoreAlgorithmInfo	Obtain variable-length information about an algorithm.
CCS_CreateContext	Create a cryptography context.
CCS_DestroyContext	Destroy a cryptography context.
CCS_DestroyObject	Destroy a CCS object.
CCS_FindObjectsInit	Initialize a search for objects that match a template.
CCS_FindObjects	Continue a search for objects that match a template.
CCS_GetAttributeValue	Obtain the value of one or more object attributes.
CCS_SetAttributeValue	Modify the values of one or more object attributes.
CCS_DataEncryptInit	Initialize a data encryption operation.
CCS_Encrypt	Encrypt single-part data.

CCS_EncryptUpdate	Continue a multi-part encryption operation.
CCS_EncryptFinal	Finish a multi-part encryption operation.
CCS_DataDecryptInit	Initialize a data decryption operation.
CCS_Decrypt	Decrypt encrypted data in a single part.
CCS_DecryptUpdate	Continue a multi-part decryption operation.
CCS_DecryptFinal	Finish a multi-part decryption operation.
CCS_DigestInit	Initialize a message-digesting operation.
CCS_Digest	Digest data in a single part.
CCS_DigestUpdate	Continue a multi-part message-digesting operation.
CCS_DigestFinal	Finish a multi-part message-digesting operation.
CCS_SignInit	Initialize a signature operation.
CCS_Sign	Sign data in a single part.
CCS_SignUpdate	Continue a multi-part signature operation.
CCS_SignFinal	Finish a multi-part signature operation.
CCS_VerifyInit	Initialize a verification operation.
CCS_Verify	Verify data in a single part.
CCS_VerifyUpdate	Continue a multi-part verification operation.
CCS_VerifyFinal	Finish a multi-part verification operation.
CCS_GenerateKey	Generate a secret key.
CCS_GenerateKeyPair	Generate a public-key/private-key pair.
CCS_WrapKey	Wrap (i.e. encrypt) a key for storage or distribution external to CCS.
CCS_UnwrapKey	Unwrap (i.e. decrypt) a key.
CCS_LoadCertificate	Load a public-key certificate, verify its signature and load the resulting public key.
CCS_LoadSelfSignedCertificate	Load a self-signed public-key certificate, verify its signature and load the resulting public key.
CCS_LoadUnverifiedCertificate	Load a public-key certificate and the resulting public key without verifying the certificate signature.
CCS_GenerateCertificate	Create and sign a public-key certificate.
CCS_GenerateCertificateFromRequest	Create and sign a public-key certificate whose public key is provided by a PKCS#10 Certification Request.
CCS_GetLocalCertificate	Return a public-key certificate or local portion of the certification path for one of the NICI-predefined public keys.
CCS_GetCertificate	Return a public-key certificate or complete certification path for one of the NICI-predefined public keys.

Appendix B

Non-FIPS Mode CCS API Definitions

For complete descriptions, please refer to the *Controlled Cryptography Services Software Development Specifications* document available from Novell.

API	Description
CCS_EncryptRestart	Reinitialize an encryption operation.
CCS_DecryptRestart	Reinitialize a decryption operation.
CCS_Obfuscate	Obfuscates an input string.
CCS_DeObfuscate	De-obfuscates an input string.
CCS_pbeEncrypt	Encrypt data in a single part using a password and password-based algorithm as described in PKCS#5 or PKCS#12.
CCS_pbeDecrypt	Decrypt data in a single part using a password and password-based algorithm as described in PKCS#5 or PKCS#12.
CCS_pbeSign	Generate signature for input data in a single part using a password and password-based algorithm as described in PKCS#12.
CCS_pbeVerify	Verify input data and its signature in a single part using a password and password-based algorithm as described in PKCS#12.
CCS_pbeShroudPrivateKey	Encrypt a PKCS#8 private key using a password and password-based algorithm as described in PKCS#5 or PKCS#12.
CCS_pbeUnshroudPrivateKey	Decrypt and load an encrypted PKCS#8 private key using the password and the password-based algorithm as described in PKCS#5 or PKCS#12.
CCS_LoadPFXPrivateKeyWithPassword	Loads zero or more private keys encrypted in a password from a PKCS#12 PFX structure. See PKCS#12 document for details. Only PKCS#8 private keys are supported.

CCS_LoadPFXCertificateWithPassword	Loads zero or more X.509 certificates and public keys in those certificates from a PKCS#12 structure. The certificates either can be encrypted in a safe bag or can be in plain form. See PKCS#12 and RFC 2459 documents for details.
CCS_DigestRestart	Reinitialize a message-digesting operation.
CCS_SignRestart	Reinitialize a signature operation.
CCS_VerifyRestart	Reinitialize a verification operation.
IKE_Sign	Sign using an IKE Authentication Phase 1 authentication algorithm. The algorithms and mechanisms are described in RFC 2409: The Internet Key Exchange.
IKE_Verify	Verify using an IKE Authentication Phase 1 authentication algorithm. The algorithms and mechanisms are described in RFC 2409: The Internet Key Exchange.
CCS_InjectKey	This is the raw (i.e., plaintext) key injection function that is used for legacy applications with raw key access, and required to use NICI with their existing raw keys.
CCS_ExtractKey	Extract attributes of a key, including its value (NICI_A_KEY_VALUE) attribute.
CCS_GenerateKeyExchangeParameters	This is the parameter generation stage of a key agreement algorithm.
CCS_KeyExchangePhase1	This is the phase 1 of a key exchange algorithm.
CCS_KeyExchangePhase2	This is the phase 2 of a key exchange algorithm.

Appendix C

Abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
CA	Certification Authority
CAST	A block encryption algorithm
CCS	Controlled Cryptography Services
CIFS	Common Internet File System
CSP	Critical Security Parameters
DAC	Discretionary Access Controls
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
EAL	Evaluation Assurance Level
EMI/EMC	Electromagnetic Interference/Electromagnetic Compatibility
HMAC	keyed-Hash Message Authentication Code
IETF	Internet Engineering Task Force
IV	Initialization Vector
LMdigest	Lan Manager message digest algorithm
MABLE	Module Authentication and Binding Library Extensions
MAC	Message Authentication Code
MD2/4/5	Message Digest algorithms
NDS	Novell Directory Services
NICI	Novell International Cryptographic Infrastructure
PFX	Personal inFormation eXchange syntax
PKCS	Public Key Cryptography Standards
RC2/4/5	Encryption algorithms
RFC	IETF Request For Comments
RSA	Rivest-Shamir-Adleman public key algorithm
SHA	Secure Hash Algorithm
SPARC	Scalable Processor ARChitecture
TCSEC	Trusted Computer System Evaluation Criteria
TLS	Transport Level Security
TOE	Target Of Evaluation
TSF	TOE Security Functions