Security Policy

Odyssey® Security Component

Author: Paul Funk

Version: 0.11

Date: February 2, 2006

Revision history

version	date	By	Comments
0.01	01/17/05	Paul Funk	Created
0.02	02/23/05	PJ Kirner	Updated
0.03	04/07/05	Steven Erickson	Updated
0.04	04/28/05	Steven Erickson	Updated
0.05	05/06/05	BKP Security	Minor reformatting
0.06	06/20/05	Steven Erickson	Updated
0.07	06/20/05	BKP Security	Minor reformatting
0.08	08/04/05	Steven Erickson/BKP Security	Response to CMVP comments
0.09	08/25/05	BKP Security	Response to CMVP comments
0.10	12/14/05	BKP Security	Updated product version number
0.11	02/02/06	BKP Security	Updated product version number

Funk Software, Inc.
222 Third Street
Cambridge, MA 02142

617-497-6339

© Copyright 2005, 2006 Funk Software, Inc. All rights reserved. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

Odyssey® is a registered trademark of Funk Software, Inc.

Table of Contents

1. Introduction 2
1.1 Purpose of the Security Policy 2
1.2 Cryptographic Module Definition 2
1.3 Cryptographic Module Description 3
2. Specification of the Security Policy 5
2.1 Identification and Authentication Policy 5
2.2 Access Control Policy 5
2.3 Cryptographic Key Management 7
2.3.1 Key Generation 7
2.3.2 Key Storage 7
2.3.3 Key Protection/Zeroization 7
2.4 Physical Security Policy 8
2.5 API Specification 8
2.6 Self Tests 15
2.7 Operational Environment 16
2.7.1 Assumptions 16
2.7.2 Installation and Initialization 16
2.7.3 Policy 17

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

1. Introduction

This document describes the Cryptographic Module Security Policy for the Odyssey Security Component (OSC). In accordance with the requirements for FIPS 140-2 validation, it specifies the rules under which the OSC must operate.

1.1 Purpose of the Security Policy

A security policy for OSC is defined for the following purposes:

- To meet the requirements for FIPS 140-2 validation at Security Level 1.
- To allow independent evaluation of the OSC with respect to a stated policy.
- To allow independent evaluation of the OSC with respect to the needs of potential users.

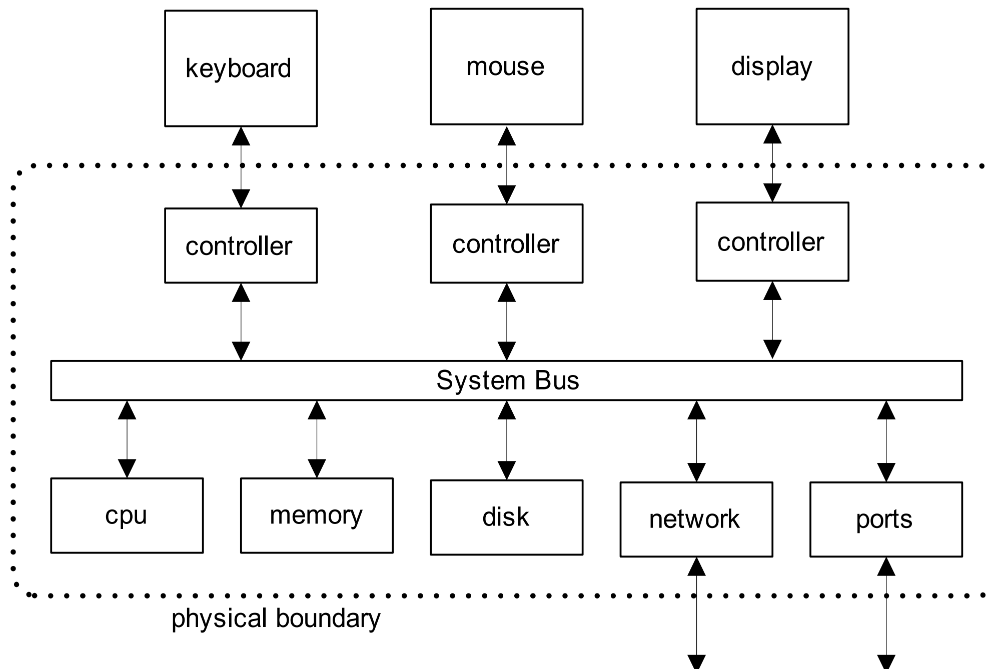
1.2 Cryptographic Module Definition

The OSC is a software module that implements a set of cryptographic algorithms for use by a software application. This Security Policy document details two versions of the OSC:

Odyssey Security Component Version 1.2

Odyssey Security Component/Portable Version 1.2

The OSC comprises a pair of binary dynamic link libraries, each compiled from source code written using a combination of C, C++ and assembly language optimizations on specific platforms. A second "portable" version of the product is available using C and C++ (OSC/Portable). One of the binary libraries resides in user space, the other in kernel space. The OSC runs on PCs and handheld devices under the Windows, Windows CE and Linux operating systems. The product was tested on Windows XP (32-bit) and Linux RedHat 9.0 (32-Bit) using an x86-compatible PC. The module does not implement a non-Approved mode of operation.



© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

Figure 1: Hardware Diagram Showing PC Containing Cryptographic Module

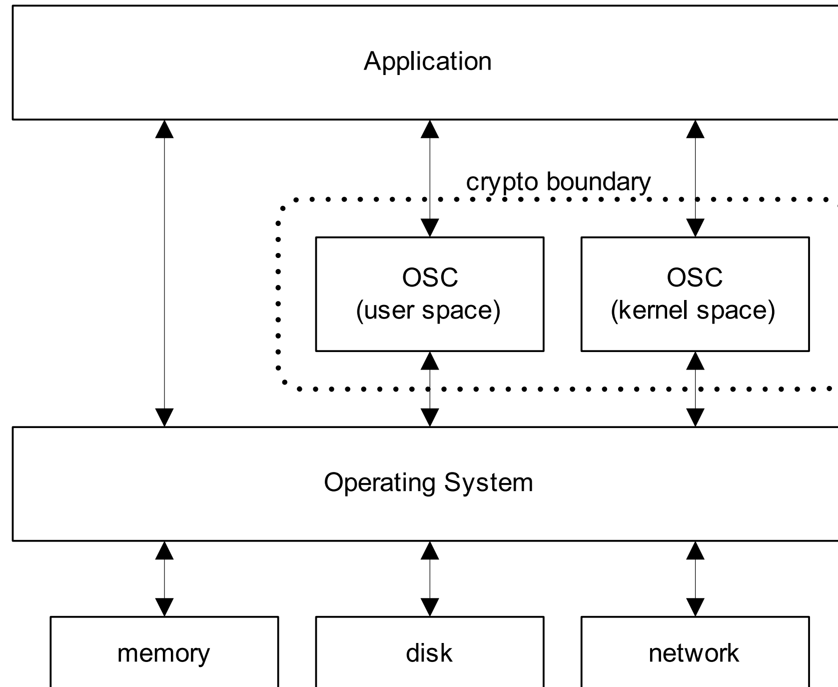


Figure 2: Software Diagram Showing Cryptographic Boundary

1.3 Cryptographic Module Description

The OSC and OSC/Portable provide cryptographic services to external applications residing on the same device and running on the same processor or set of processors. These external applications may be networking applications, in either client, server or peer roles, or may be standalone. The OSC and OSC/Portable allow such applications to integrate cryptography in a well-defined manner, with the benefit of documented APIs, document policies governing use of these APIs, self-test, and external review.

The cryptographic algorithms provided by the OSC and OSC/Portable include the following:

- **Symmetric Encryption/Decryption**

  AES:

  - modes: single block (ECB), CBC, Counter (OSC Certificate #245, OSC/Portable Certificate #246)
  - key sizes: 128, 192, 256 bit

  Triple-DES: (OSC Certificate #331, OSC/Portable Certificate #332)

  - modes: single block (ECB), CBC
  - key sizes: 168 bit

  DES: (OSC Certificate #309, OSC/Portable Certificate #310) transitional phase only – valid until May 19, 2007.

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

notes: may only be used for interoperation with legacy systems.

modes: single block (ECB), CBC

key sizes: 56 bit

- **Asymmetric Encryption/Decryption for key wrapping**

  RSA: (PKCS #1)

  notes: This is a non-approved algorithm which is allowed for use in FIPS 140-2 mode for key transport purposes.

  key sizes: 1024, 1536, 2048, 2072, 4096 bit

- **Message Digests**

  SHS: (OSC Certificate #322, OSC/Portable Certificate #323)

  digest sizes: 160, 224, 256, 384, 512 bit

- **Message Authentication**

  HMAC (OSC Certificate #53, OSC/Portable Certificate #55)

  digest sizes: 160, 224, 256, 384, 512 bit

  CCM-AES (OSC Certificate #2, OSC/Portable Certificate #3)

  key sizes: 128, 192, 256 bit

- **Digital Signature/Verification**

  DSA: (OSC Certificate #133, OSC/Portable Certificate #135)

  notes: DSA parameter generation uses the algorithm specified in FIPS 186-2 to create and verify the PQG parameters.

  key sizes: 512, 576, 640, 704, 768, 832, 896, 960, 1024 bit

  RSA: (PKCS#1) (OSC Certificate #61, OSC/Portable Certificate #62)

  key sizes: 1024, 1536, 2048, 2072, 4096 bit

- **Symmetric Key Generation**

  General purpose FIPS 186-2 PRNG (OSC Certificate #79, OSC/Portable Certificate #84) used to generate DES, Triple-DES, AES and HMAC keys.

- **Asymmetric Key Generation**

  RSA key generation (PKCS #1)

  notes: This is a non-approved algorithm which is allowed for use in FIPS 140-2 mode.

  key sizes: 1024, 1536, 2048, 2072, 4096 bit

  FIPS 186-2 DSA key generation (OSC Certificate #133, OSC/Portable Certificate #135)

  key sizes: 512, 576, 640, 704, 768, 832, 896, 960, 1024 bit

- **Pseudo-Random Number Generation**

  General purpose FIPS 186-2 PRNG (OSC Certificate #79, OSC/Portable Certificate #84)

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

- **Key Agreement**

  Diffie-Hellman key agreement

  notes: This is a non-approved algorithm which is allowed for use in FIPS 140-2 mode.

2. Specification of the Security Policy

2.1 Identification and Authentication Policy

The OSC neither identifies nor authenticates the user that accesses it, regardless of role. This is acceptable for FIPS 140-2 l Level 1 validation only.

Role	Type of Authentication	Authentication Data
User	n/a	n/a
Cryptographic Officer	n/a	n/a

Table 1: Roles and Required Identification and Authentication

Authentication Mechanism	Strength of Mechanism
None	n/a

Table 2: Strengths of Authentication Mechanisms

2.2 Access Control Policy

The OSC and OSC/Portable support the following roles: User and Cryptographic Officer. Determination of role is based on the service chosen. An operator invoking User-specific services is in the User role; otherwise, the operator is in the Cryptographic Officer role.

The Critical Security Parameters (CSPs) defined for the OSC consist of cryptographic keys and random numbers used as seeding material.

An operator in either role can perform the following operations on CSPs: read, write and execute.

Each service's API indicates the type of access to CSPs defined by that API. When a CSP is used by the API call to perform particular services, read and execute access is indicated. When a CSP is generated, modified or deleted by the API call, write access is indicated.

Approved Service	CSPs	Certificate Number (OSC/OSC-Portable)	Accessible Roles	Type of Access
Symmetric Encryption/Decryption				
AES	AES Key	245,246	User	read, execute

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

Triple-DES	Triple-DES Key	331, 332	User	read, execute
DES	DES Key	309, 310	User	read, execute
Asymmetric Encryption/Decryption for Key Wrapping				
RSA Encryption	RSA Encryption Public Key	N/A	User	read, execute
RSA Decryption	RSA Decryption Private Key	N/A	User	read, execute
Message Digest				
SHS	N/A	322, 323	User	read, execute
Message Authentication				
AES-CCM	AES-CCM key	2, 3	User	read, execute
HMAC	HMAC key	53, 55	User	read, execute
Digital Signature/Verification				
RSA Verify	RSA Public Key	61,62	User	read, execute
RSA Sign	RSA Private Key	61, 62	User	read, execute
DSA Verify	DSA Public Key	133, 135	User	read, execute
DSA Sign	DSA Private Key	133, 135	User	read, execute
Symmetric Key Generation				
FIPS 186-2 PRNG	FIPS 186-2 PRNG Seed and Seed Key	79, 84	User	read, execute
Asymmetric Key				

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

Generation				
RSA Key Generation	RSA Key Pair	N/A	User	write
DSA Key Generation	DSA Key Pair	133, 135	User	write
Diffie-Hellman key generation	Diffie-Hellman key	N/A	User	write
Pseudo-Random Number Generation				
FIPS 186-2 PRNG	FIPS 186-2 PRNG Seed and Seed Key	79, 84	User	read, execute
Key Agreement				
Diffie-Hellman key exchange	Diffie-Hellman key	N/A	User	read, execute
Other Services				
Enable FIPS Mode	N/A	N/A	Crypto Officer	N/A
Show Status	N/A	N/A	Crypto Officer	N/A
Run Self Tests	HMAC Software integrity key	53, 55	Crypto Officer	read

Table 3: FIPS 140-2 Approved Services Authorized for Roles

2.3 Cryptographic Key Management

2.3.1 Key Generation

The cryptographic module supports generation of DSA, RSA, and Diffie-Hellman (DH) public and private keys, as well as the generation of symmetric keys for AES, DES, Triple-DES and HMAC.

2.3.2 Key Storage

Public and private keys are provided to authorized operators of the cryptographic module, and any key is destroyed when the operator indicates that the key is no longer being used. No persistent or long-term storage of keys is provided by the cryptographic module. If the operator wished to store keys they would be responsible for doing so outside of the cryptographic module.

2.3.3 Key Protection/Zeroization

All key data exists in data structures allocated within the cryptographic module, and can only be returned to an authorized user using the defined API. The operating system protects system memory and process space from access by unauthorized users. The operator of the cryptographic module should follow the steps outlined in the documentation to ensure sensitive data is protected by zeroizing the data from memory when it is no longer needed.

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

2.4 Physical Security Policy

The physical security of the cryptographic module is provided by the PC that it is being used on. Physical Security requirements for FIPS 140-2 Level 1 modules are not applicable.

2.5 API Specification

The following is a specification of the API provided by the Cryptographic Module. The API specification is the same for both OSC and OSC/Portable as well as for both the user and kernel component. Please see the OSC and OSC/Portable product documentation for a more detailed description of the API.

```
int OS_CALL

    odFIPS_GetAPIVersion(uint32_t* pnFipsMinAPIVersion, uint32_t*
pnFipsAPIVersion, uint32_t* pnFipsAPIVTableSize);

int OS_CALL odFIPS_EnableFIPSMode( const char *pszPathToFIPSModule);

int OS_CALL odFIPS_DisableFIPSMode();

OD_FIPS_STATE OS_CALL odFIPS_GetState();

int OS_CALL odFIPS_GetError();

int OS_CALL odFIPS_PRNG_Init(OD_FIPS_PRNG_CTX *pContext,
        size_t nContextSize, const void *pXKey, size_t nXKeySize,
        uint32_t nCreationFlags);

int OS_CALL odFIPS_PRNG_Iterate(OD_FIPS_PRNG_CTX *pContext,
        void *pOut, const void *pXSEED, size_t nXSEEDSize);

int OS_CALL odFIPS_PRNG_Generate(OD_FIPS_PRNG_CTX *pContext,
        void *pOutData, size_t nDataSize);

void OS_CALL odFIPS_PRNG_Term(OD_FIPS_PRNG_CTX *pContext);

int OS_CALL odFIPS_SHA_Init(OD_FIPS_SHA_CTX *pContext,
        size_t nContextSize, size_t nDigestSize);

int OS_CALL odFIPS_SHA_Update(OD_FIPS_SHA_CTX *pContext,
        const void *pData, size_t nDataSize);

int OS_CALL odFIPS_SHA_Final(OD_FIPS_SHA_CTX *pContext,
        void *pMessageDigest);

void OS_CALL odFIPS_SHA_Term(OD_FIPS_SHA_CTX *pContext);

int OS_CALL odFIPS_SHA(const void *pData, size_t nDataSize,
        void *pMessageDigest, size_t nDigestSize);

int OS_CALL odFIPS_HMAC_SHA_Init(OD_FIPS_HMAC_SHA_CTX *pContext,
        size_t nContextSize, size_t nDigestSize);

int OS_CALL odFIPS_HMAC_SHA_AddKey(OD_FIPS_HMAC_SHA_CTX *pContext,
        const void *pKey, size_t nKeySize);

int OS_CALL odFIPS_HMAC_SHA_Update(OD_FIPS_HMAC_SHA_CTX *pContext,
```

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

```
        const void *pData, size_t nDataSize);
int OS_CALL odFIPS_HMAC_SHA_Final(OD_FIPS_HMAC_SHA_CTX *pContext,
        void *pMessageDigest);
void OS_CALL odFIPS_HMAC_SHA_Term(OD_FIPS_HMAC_SHA_CTX *pContext);
int OS_CALL odFIPS_HMAC_SHA(const void *pKey, size_t nKeySize,
        const void *pData, size_t nDataSize, void *pMessageDigest,
        size_t nDigestSize);
int OS_CALL odFIPS_DES_Init(OD_FIPS_DES_KEY_CTX *pKeyScheduleContext,
        size_t nKeyScheduleContextSize, const void *pKey);
int OS_CALL odFIPS_DES_ecb_encrypt(
        OD_FIPS_DES_KEY_CTX *pKeyScheduleContext,
        const void *pInputData, void *pOutputData, int enc);
int OS_CALL odFIPS_DES_ecb3_encrypt(
        OD_FIPS_DES_KEY_CTX *pKeyScheduleContext1,
        OD_FIPS_DES_KEY_CTX *pKeyScheduleContext2,
        OD_FIPS_DES_KEY_CTX *pKeyScheduleContext3,
        const void *pInputData, void *pOutputData, int enc);
int OS_CALL
    odFIPS_DES_cbc_encrypt(
        OD_FIPS_DES_KEY_CTX *pKeyScheduleContext,
        const void *pInputData,
        void *pOutputData,
        size_t nDataLength,
        const void *pIV,
        void *pIVOut,
        int enc);
int OS_CALL
    odFIPS_DES_ede3_cbc_encrypt(
        OD_FIPS_DES_KEY_CTX *pKeyScheduleContext1,
        OD_FIPS_DES_KEY_CTX *pKeyScheduleContext2,
        OD_FIPS_DES_KEY_CTX *pKeyScheduleContext3,
        const void *pInputData,
        void *pOutputData,
        size_t nDataLength,
        const void *pIV,
        void *pIVOut,
        int enc);
```

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

```
void OS_CALL odFIPS_DES_Term(
        OD_FIPS_DES_KEY_CTX *pKeyScheduleContext);
int OS_CALL odFIPS_AES_Init_with_encrypt_key(
        OD_FIPS_AES_KEY_CTX *pKeyScheduleContext,
        size_t nKeyScheduleContextSize, const void *pKey,
        size_t nKeyBits);
int OS_CALL odFIPS_AES_Init_with_decrypt_key(
        OD_FIPS_AES_KEY_CTX *pKeyScheduleContext,
        size_t nKeyScheduleContextSize,
        const void *pKey, size_t nKeyBits);
int OS_CALL
    odFIPS_AES_cbc_encrypt(
        const OD_FIPS_AES_KEY_CTX *pKeyScheduleContext,
        const void *pInputData,
        void *pOutputData,
        size_t nDataLength,
        const void *pIV,
        void *pIVOut,
        int enc);
int OS_CALL odFIPS_AES_ecb_encrypt(
        const OD_FIPS_AES_KEY_CTX *pKeyScheduleContext,
        const void *pInputData, void *pOutputData,
        size_t nDataLength, int enc);
int OS_CALL
    odFIPS_AES_ctr_encrypt(
        const OD_FIPS_AES_KEY_CTX *pKeyScheduleContext,
        const void *pInputData,
        void *pOutputData,
        size_t nDataLength,
        const void *pCounter,
        void *pCounterOut);
void OS_CALL odFIPS_AES_Term(
        OD_FIPS_AES_KEY_CTX *pKeyScheduleContext);
int OS_CALL odFIPS_AES_CCM_Init(
        OD_FIPS_AES_CCM_CTX *pContext,
        size_t nContextSize, const void *pKey,
        size_t nKeyLength);
```

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

```
int OS_CALL odFIPS_AES_CCM_InitMessage(
        OD_FIPS_AES_CCM_CTX *pContext,
        const void *pIV, size_t nIVLength,
        size_t nHeaderLength, size_t nMessageLength,
        size_t nTagLength);
int OS_CALL odFIPS_AES_CCM_AuthenticateHeader(
        OD_FIPS_AES_CCM_CTX *pContext,
        const void *pHeader, size_t nHeaderLength);
int OS_CALL odFIPS_AES_CCM_AuthenticateData(
        OD_FIPS_AES_CCM_CTX *pContext,
        const void *pData, size_t nDataLength);
int OS_CALL
    odFIPS_AES_CCM_CryptData(
        OD_FIPS_AES_CCM_CTX *pContext,
        const void *pData,        void *pDataOut,
        size_t nDataLength);
int OS_CALL odFIPS_AES_CCM_ComputeMessageAuthenticationTag(
        OD_FIPS_AES_CCM_CTX *pContext,
        void *pTag, size_t nTagLength);
void OS_CALL odFIPS_AES_CCM_Term(
        OD_FIPS_AES_CCM_CTX *pContext);
int OS_CALL
    odFIPS_AES_CCM_EncryptMessage(
        OD_FIPS_AES_CCM_CTX *pContext,
        const void *pIV, size_t nIVLength,
        const void *pHeader, size_t nHeaderLength,
        const void *pMessage, void *pMessageOut,
        size_t nMessageLength, void *pTag,
        size_t nTagLength);
int OS_CALL
    odFIPS_AES_CCM_DecryptMessage(
        OD_FIPS_AES_CCM_CTX *pContext,
        const void *pIV, size_t nIVLength,
        const void *pHeader, size_t nHeaderLength,
        const void *pMessage, void *pMessageOut,
        size_t nMessageLength, const void *pTag,
        size_t nTagLength);
```

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

```
int OS_CALL odFIPS_AES_KeyWrap_Encrypt(
        OD_FIPS_AES_KEY_CTX *pKeyScheduleContext,
        void *pOutBuffer, const void *pInBuffer,
        size_t nInBufferSize, const void *pIV);
int OS_CALL odFIPS_AES_KeyWrap_Decrypt(
        OD_FIPS_AES_KEY_CTX *pKeyScheduleContext,
        void *pOutBuffer, size_t nOutBufferSize,
        const void *pInBuffer, const void *pIV);
int OS_CALL odFIPS_DH_Init(
        OD_FIPS_DH_CTX *pContext,
        size_t nContextSize, OD_FIPS_PRNG_CTX *pPRNGContext,
        const void *pGenerator, size_t nGeneratorLength,
        const void *pPrime, size_t nPrimeLength);
int OS_CALL
    odFIPS_DH_GenerateParameters(
        OD_FIPS_DH_CTX *pContext,
        void *pPrimeOut, size_t nPrimeLength);
int OS_CALL  odFIPS_DH_CalculatePublic(
        OD_FIPS_DH_CTX *pContext,
        void *pPublic, size_t* pnOutPublicLength,
        size_t nPublicMaxLength, const void *pPrivateNonce,
        size_t nPrivateNonceLength);
int OS_CALL odFIPS_DH_CalculateSharedKey(
        OD_FIPS_DH_CTX *pContext,
        void *pSharedKey, size_t* pnOutSharedKeyLength,
        size_t nSharedKeyMaxLength, const void *pPeerPublic,
        size_t nPeerPublicLength);
void OS_CALL odFIPS_DH_Term(OD_FIPS_DH_CTX *pContext);
int OS_CALL
    odFIPS_RSA_InitWithPublicKey(
        OD_FIPS_RSA_CTX *pContext, size_t nContextSize,
        OD_FIPS_PRNG_CTX *pPRNGContext, const void *pN,
        size_t nNLength, const void *pE, size_t nELength);
int OS_CALL
    odFIPS_RSA_InitWithPrivateKey(
        OD_FIPS_RSA_CTX *pContext,
        size_t nContextSize,
```

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

```
        OD_FIPS_PRNG_CTX *pPRNGContext,
        const void *pN, size_t nNLength,
        const void *pE, size_t nELength,
        const void *pP, size_t nPLength,
        const void *pQ, size_t nQLength,
        const void *pD, size_t nDLength,
        const void *pDMP1, size_t nDMP1Length,
        const void *pDMQ1, size_t nDMQ1Length,
        const void *pIQMP, size_t nIQMPLength);
int OS_CALL odFIPS_RSA_InitAndGenerateKey(
        OD_FIPS_RSA_CTX *pContext,
        size_t nContextSize, OD_FIPS_PRNG_CTX *pPRNGContext,
        size_t nBitsInModulus_N, uint32_t nEValue);
int OS_CALL
    odFIPS_RSA_ComputeAccelerationParameters(
        OD_FIPS_RSA_CTX *pContext);
int OS_CALL
    odFIPS_RSA_GetKeyData(
        OD_FIPS_RSA_CTX *pContext,
        void *pN, size_t* pnOutNLength,
        void *pE, size_t* pnOutELength,
        void *pP, size_t* pnOutPLength,
        void *pQ, size_t* pnOutQLength,
        void *pD, size_t* pnOutDLength,
        void *pDMP1, size_t* pnOutDMP1Length,
        void *pDMQ1, size_t* pnOutDMQ1Length,
        void *pIQMP, size_t* pnOutIQMPLength);
int OS_CALL odFIPS_RSA_Sign(
        OD_FIPS_RSA_CTX *pContext,
        int eDigestType, const void *pDigest,size_t nDigestLength,
        void* pSignature,size_t* pnOutSignatureLength);
int OS_CALL odFIPS_RSA_Verify(
        OD_FIPS_RSA_CTX *pContext,
        int eDigestType,
        const void *pDigest,size_t nDigestLength,
        const void *pSignature,size_t nSignatureLength);
int OS_CALL odFIPS_RSA_Encrypt(
```

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it’s entirety without any modification.

```
        OD_FIPS_RSA_CTX *pContext,
        const void *pPlainText,size_t nPlainTextLength,
        int nPaddingType,
        void* pCipherText,size_t* pnOutCipherTextLength);
int OS_CALL
    odFIPS_RSA_Decrypt(
        OD_FIPS_RSA_CTX *pContext,
        const void *pCipherText,size_t nCipherTextLength,
        int nPaddingType,
        void* pPlainText,size_t* pnOutPlainTextLength);
int OS_CALL
    odFIPS_RSA_EncryptPrivate(
        OD_FIPS_RSA_CTX *pContext,
        const void *pPlainText,size_t nPlainTextLength,
        int nPaddingType,
        void* pCipherText,size_t* pnOutCipherTextLength);
int OS_CALL
    odFIPS_RSA_DecryptPublic(
        OD_FIPS_RSA_CTX *pContext,
        const void *pCipherText,size_t nCipherTextLength,
        int nPaddingType,
        void* pPlainText,size_t* pnOutPlainTextLength);
void OS_CALL odFIPS_RSA_Term(OD_FIPS_RSA_CTX *pContext);
int OS_CALL odFIPS_DSA_Init(
        OD_FIPS_DSA_CTX *pContext,
        size_t nContextSize,
        OD_FIPS_PRNG_CTX *pPRNGContext,
        const void *pP,size_t nPLength,
        const void *pQ,size_t nQLength,
        const void *pG,size_t nGLength,
        const void *pPublicKey,size_t nPublicKeyLength,
        const void *pPrivateKey,size_t nPrivateKeyLength);
int OS_CALL
    odFIPS_DSA_InitAndGenerateParameters(
        OD_FIPS_DSA_CTX *pContext,
        size_t nContextSize,
        OD_FIPS_PRNG_CTX *pPRNGContext,
```

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

```
        size_t nNumberOfBits,
        const void* pSeed,
        size_t nSeedLength,
        void* pSeedOut,
        size_t *pC,
        void *pH);
int OS_CALL odFIPS_DSA_GenerateNewKey(
        OD_FIPS_DSA_CTX *pContext);
int OS_CALL
    odFIPS_DSA_GetKeyData(
        OD_FIPS_DSA_CTX *pContext,
        void *pP, size_t* pnOutPLength,
        void *pQ, size_t* pnOutQLength,
        void *pG, size_t* pnOutGLength,
        void *pPublicKey, size_t* pnOutPublicKeyLength,
        void *pPrivateKey, size_t* pnOutPrivateKeyLength);
int OS_CALL
    odFIPS_DSA_Sign(
        OD_FIPS_DSA_CTX *pContext,
        const void *pDigest, size_t nDigestLength,
        void* pSignature, size_t* pnOutSignatureLength,
        void *pR, void *pS);
int OS_CALL
    odFIPS_DSA_Verify(
        OD_FIPS_DSA_CTX *pContext,
        const void *pDigest, size_t nDigestLength,
        const void *pSignature, size_t nSignatureLength);
void OS_CALL odFIPS_DSA_Term(
        OD_FIPS_DSA_CTX *pContext);
```

2.6 Self Tests

The cryptographic module performs a number of self tests to verify the integrity of the cryptographic module, and cryptographic algorithms that it contains. The following is a summary of the types of self tests that are available in the cryptographic module.

Power-Up Tests:

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it’s entirety without any modification.

- Module Integrity Check, which is performed by verifying HMAC applied to all binary components
- SHA-1 Self Test
- SHA 224, 256, 386, 512 bit Self Test
- DES Self Test
- Triple-DES Self Test
- AES Self Test
- RSA Self Test
- DSA Self Test
- Random Number Generator Self Test
- HMAC SHA-1 Self Test
- HMAC SHA 224, 256, 386, 512 bit Self Test
- AES-CCM Self Test

Conditional Tests:

- Continuous Random Number Generator Test
- RSA Key Generation Conditional Pairwise Consistency Check
- DSA Key Generation Conditional Pairwise Consistency Check

2.7 Operational Environment

2.7.1 Assumptions

The following are assumptions concerning the environment in which the OSC is to be used:

- It is infeasible to read or modify the module's code or data memory space in an unauthorized manner.
- The module is initialized to FIPS-140-2 mode, as described below.

2.7.2 Installation and Initialization

To install and initialize the OSC for FIPS 140-2-compliant operation, the following steps must be performed:

The operating system must be configured to operate in single user mode.

The odFIPS_EnableFIPSMode function needs to be called in order to put the module into FIPS mode.

For installation on Windows systems, the binary dynamic link library (odFIPS.dll and odFIPS.dll.icv) that is delivered as part of the OSC must be copied locally to the machine on which the OSC is to be used. The kernel mode driver (odFIPS.sys) must be installed into the running operating system as a boot loadable driver. The kernel loadable module can be installed onto a windows system by copying odFIPS.sys and odFIPS.sys.icv to

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

\windows\system32\drivers. The CreateService Windows system call should be used to load the driver into the running operating system.

Step 1: Copy odFIPS.sys and odFIPS.sys.icv to \windows\system32\drivers

Step 2: Load the driver into kernel space by using the Windows system call CreateService

For installation on Linux systems, the binary shared object (odFIPS.so and odFIPS.so.icv) that is delivered as part of the OSC must be copied locally to the machine on which the OSC is to be used. The kernel mode driver (odFIPS.o and odFIPS.o.icv) is a kernel loadable module that must be loaded into the running kernel. The odFIPS.o kernel module should be loaded into the running system by using the insmod utility available on Linux systems.

Step 1: Load the kernel module

```
>/sbin/insmod odFIPS.o
```

Step 2: Create the character device for use during the module integrity check

```
> major=`cat /proc/devices | awk "\\$2==\"odFIPS\" {print \\$1}"`
> mknod /dev/odFIPS0 c $major 0
> ln -sf odFIPS0 /dev/odFIPS
```

Step 3: Module Integrity Check

The odFIPS.o kernel mode driver can not access the file system from kernel space, so the necessary file operations must be completed in user space and sent to kernel space through a character based device control channel. odFipsHelper is a utility that reads a file from disk and writes it to /dev/odFIPS. Before the FIPS mode can be enabled in the Linux kernel module, the following two calls must be made to provide the kernel with the necessary module integrity check data

```
> odFipsHelper odFips.o
> odFipsHelper odFips.o.icv
```

The execution steps for Linux kernel module, including the module integrity check are performed by the install_odfips.sh shell script and do not require user intervention. This shell script is a part of the module and is located within the cryptographic boundary of the module.

The OSC must be initialized and self-tests must be run to enter FIPS-140-2 mode.

2.7.3 Policy

To achieve the FIPS 140-2 mode of operation, the following policy must be adhered to:

- No more than one human operator at a time may use the OSC or an application that uses the OSC.

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.

- No more than one instance of the OSC may be loaded into memory for execution on a single machine.
- The machine must not be configured to allow virtualization of memory in which code or data of the OSC resides onto any persistent storage other than storage local to that machine.

© Copyright 2005 Funk Software, Inc. This document may be copied without Funk Software, Inc. explicit permission provided that it is copied in it's entirety without any modification.