BitArmor Secure Cryptographic Engine Version 1.2

Security Policy v 1.0

BitArmor Systems, Inc
401 Liberty Avenue, Suite 1900
Pittsburgh, PA 15222
Phone: 412-682-2200
www.bitarmor.com

FIPS 140-2 Security Policy

I. Introduction

A. Overview

BitArmor offers breakthrough data protection software that protects data wherever it goes. Because it combines persistent file encryption with full disk encryption, BitArmor gives customers a single integrated solution for protecting data at all of its most vulnerable points, like on laptops, on USB drives, in e-mail attachments. BitArmor software helps customers precisely control access to sensitive data so they can achieve regulatory compliance, reduce the liability of publicly disclosing data breaches, and protect valuable intellectual property that is shared inside and outside of their enterprise. Leaders in the Healthcare, Retail, Education, and Legal industries — among others — have chosen BitArmor's easy-to-manage, cost-effective data protection.

B. Purpose

This document is the non-proprietary cryptographic module security policy for the BitArmor Secure Cryptographic Engine from BitArmor Systems, Inc. It serves as a reference description for BitArmor's cryptographic module and its compliance to the FIPS 140-2 security standard. This document was prepared as part of the FIPS 140-2 Security Level 1 validation of BASCE.

C. References

BitArmor Systems, Inc.'s website (www.bitarmor.com) provides detailed information on the company and its DataControl™ security products that utilize proprietary SmartTag™ technology and are covered by BitArmor's No-Breach Guarantee™.

The Federal Information Processing Standards Publication 140-2 (FIPS 140-2), Security Requirements for Cryptographic Modules, specifies the U.S. and Canadian governments' requirements for cryptographic modules. Complete details on the FIPS 140-2 Cryptographic Module Validation Program (CMVP) can be found at csrc.nist.gov/groups/STM/index.html.

D. Conventions

Hereafter, this document refers to the BitArmor Secure Cryptographic Engine as either BASCE, "the cryptographic module", or simply, "the module".

E. Disclosure

This non-proprietary security policy may be reproduced and distributed intact including the ©2009 BitArmor Systems, Inc. copyright notice depicted on page 1 of this document and shown here:

> Copyright ©2009 BitArmor Systems, Inc. All rights reserved. BitArmor, BitArmor Systems, and Smart Tag technology are trademarks of BitArmor Systems, Inc.

II. Cryptographic Module Specification

A. Description

BASCE Version 1.2 is a software module that provides advanced cryptographic functionality to BitArmor DataControl software products, including strong encryption, secure integrity and authentication, and random number generation.

BASCE Version 1.2 consists of two dynamically linked software libraries: FIPSModule.dll, BA_crypto.dll on MS Windows, and libcryptmod.so, libcrypto.so on SUSE Linux Enterprise Server. The cryptographic module Application Program Interface (API) is provided in full by FIPSModule.dll/libcryptmod.so, which in turn loads and uses BA_Crypto.dll/libcrypto.so in a private manner to provide FIPS-Approved cryptographic functionality to BitArmor DataControl client applications.

BASCE Version 1.2 meets the overall requirements of Level 1 security of the FIPS 140-2 standard. Individual security sections of BASCE Version 1.2 are validated at the FIPS 140-2 security levels shown in Table 1 below.

FIPS 140-2 Security Section	Security Level
Cryptographic Module Specification	3
Module Ports and Interfaces	1
Roles, Services and Authentication	1
Finite State Model	1
Physical Security	Not Applicable
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	Not Applicable

TABLE 1: BASCE SECURITY LEVEL VALIDATION PER FIPS 140-2 SECTION

B. Cryptographic Boundary

For FIPS 140-2 purposes, BASCE Version 1.2 (subsequently referred to as just BASCE) is classified as a multi-chip standalone module. BASCE's logical cryptographic boundary is the software library file and its public API. BASCE comprises two such files: FIPSModule.dll and BA_crypto.dll for Microsoft Windows operating systems and libcryptmod.so and libcrypto.so for the SUSE Linux Enterprise Server operating system.

BASCE's physical boundary is the general purpose computer on which the module and client applications run. General purpose computers typically include these basic hardware components:

- Intel x86 processor or equivalent (CPU)
- Memory (RAM)
- Hard disk, CD/DVD-ROM, floppy disk
- Keyboard, mouse, video, Universal Serial Bus (USB) and Ethernet network interfaces
- Video controller
- Serial & parallel ports
- Power supply

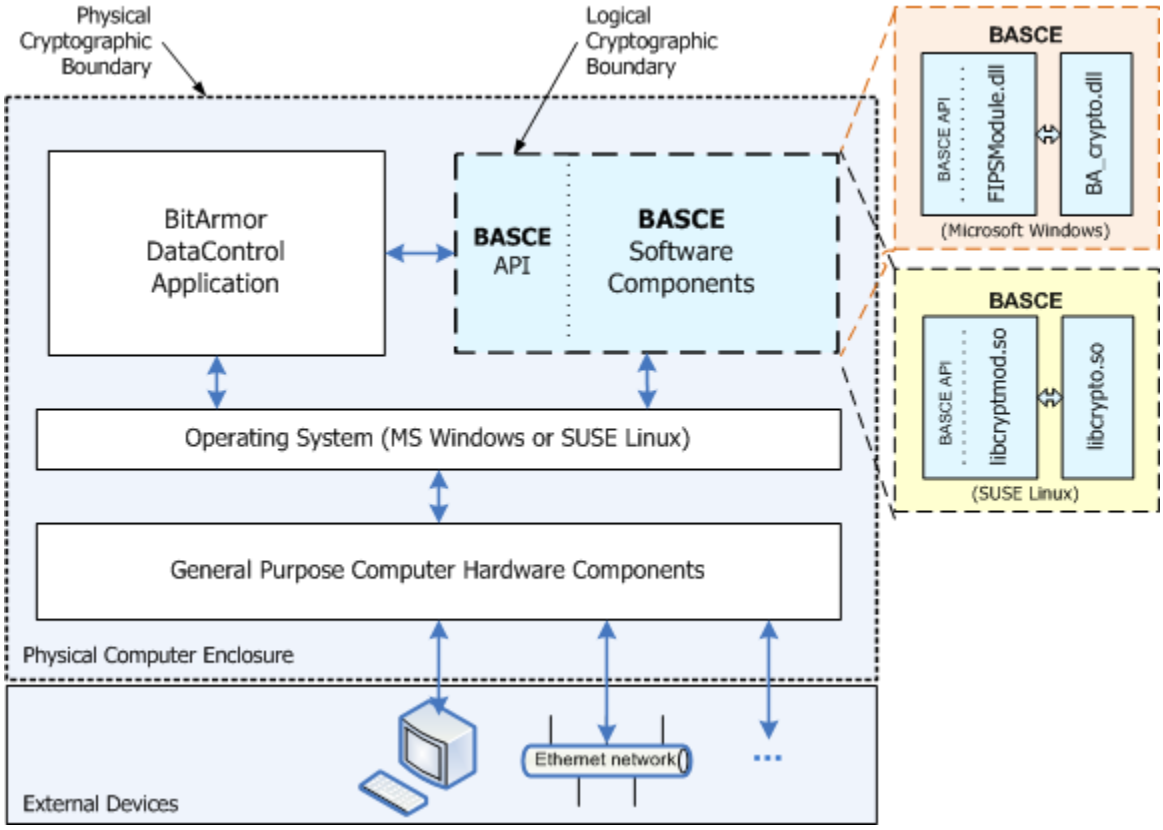

FIGURE 1: PHYSICAL & LOGICAL CRYPTOGRAPHIC BOUNDARIES

The physical and logical cryptographic boundaries of BASCE are shown in Figure 1. The physical computer enclosure contains general purpose computer hardware and is where BASCE and its client-application runs, thus it forms the physical cryptographic boundary of BASCE. The logical cryptographic boundary is the BASCE API and its software components consisting of two dynamic-link library files whose names are operating system dependent. External devices that interface to the general purpose computer (e.g., video, keyboard, mouse, etc.) are outside BASCE's physical cryptographic boundary.

FIPSModule.dll (MS Windows) and libcryptmod.so (SUSE Linux Enterprise Server) implement BASCE's API. FIPSModule.dll utilizes BA_CryptoLib.dll (and libcryptmod.so utilizes libcrypto.so) to provide cryptographic functionality, accessing it through private routines only. BASCE

interacts only with the client-application based process that loads it; the module makes no network connections, spawns no new threads, and performs no file output or interprocess communication.

In addition, the module does not store any critical security parameter (CSP) internally — all CSPs are passed by the client-application using references only.

Finally, BASCE is implemented entirely in software therefore physical security is provided solely by the host platform. Accordingly, in conjunction with FIPS 140-2 Security Level 1 requirements, BASCE is not subject to the physical security requirements of the standard.

For the purpose of FIPS 140-2 validation, BASCE was tested on Microsoft Windows XP-Professional SP3 (32-bit) and SUSE Linux Enterprise Server 10 (32-bit) operating systems.

C. Modes of Operation

BASCE will operate in the following two modes of operation:

1. Non-Approved Mode
2. FIPS Mode

BASCE operates in Non-Approved Mode by default upon loading and initialization by a client-application. In order to operate in FIPS Mode, the application must call the CM_setFIPSMode routine, which performs module integrity and known-answer testing. If all tests are successful the module transitions to FIPS Mode, otherwise it transitions to a failed operational state in which no cryptographic functionality is available. FIPS-approved and non-FIPS approved cryptographic algorithms can be run in the Non-Approved Mode.

The CM_setFIPSMode routine takes an integer input parameter called “mode”, which it uses to establish BASCE’s mode of operation. “Mode” must be set to ‘1’ to operate BASCE in FIPS Mode. Once BASCE establishes FIPS Mode operation, its mode cannot be changed to Non-Approved Mode.

Note: All keys used while running BASCE in Non-Approved Mode are not to be used during subsequent operation of BASCE in FIPS Mode. Conversely, all keys used while running in FIPS Mode must not be used in Non-Approved Mode.

BASCE provides zeroization routines for all of its cryptographic data types, which should be utilized to zeroize all client-application CSPs prior to transitioning to FIPS Mode, or out of FIPS Mode. Refer to section G. Roles, Services and Authentication below for details.

D. FIPS-Approved Functionality

BASCE contains the following FIPS-approved cryptographic functionality.

Algorithm	Function	FIPS Std	CAVP Certificate#
AES (ECB, CBC and CTR modes)	Symmetric Cipher	FIPS PUB 197	1101

TDES (ECB, CBC and CTR modes)	Symmetric Cipher	NIST SP800-67	802
SHA-1, SHA-256, SHA-384, SHA-512	Message Digest	FIPS PUB 180-3	1024
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	Message Authentication	FIPS PUB 198-1	614
ANSI X9.31 Appendix A.2.4 PRNG AES-128, AES-256	Deterministic RNG	FIPS PUB 186-2	613

TABLE 2: FIPS APPROVED FUNCTIONALITY

These functions are made available by BASCE in both FIPS and Non-Approved Modes of operation.

E. Non-FIPS Functionality

BASCE contains the following non-FIPS-approved cryptographic functionality.

Algorithm	Function
DES (ECB, CBC, and CTR modes)	Symmetric Cipher
MD5	Message Digest
HMAC-MD5	Message Authentication

TABLE 3: NON-FIPS APPROVED FUNCTIONALITY

These functions are made available by BASCE only when operating in Non-Approved Mode. They are physically blocked by the module while in FIPS Mode and result in an error indicating they are not available in FIPS Mode, when called.

F. Module Ports and Interfaces

The physical ports of BASCE are those of the general purpose computer system on which it runs, including keyboard, mouse, network, hard disk drive, CD-ROM drive, video, USB and power. BASCE has no direct physical controls, status indicators or electrical characteristics that impact its operation. Logical controls and status indicators take the form of input/output parameters of the module's API.

BASCE's API is its logical interface and the sole means by which client applications communicate with it. All information flows through four logically distinct interfaces as required by FIPS 140-2: *data input*, *data output*, *control input,* and *status output*. Although these interfaces share the same physical port, BASCE's API preserves the interface type semantic using explicit input, output, control, and status parameters. These interfaces are defined in Table 4 below.

Logical Interface	Module Interface	Physical Port
Data Input	API function parameters that contain data values or references to data structures, which are used as input to a module function	Keyboard, hard disk, mouse, CD-ROM, floppy disk, and USB/ parallel/serial/network ports

Data Output	API function parameters that contain data values that are, or references to data structures that refer to, output from a module function	Video monitor, hard disk, floppy disk, and USB/parallel/serial/ network ports
Control Input	API function calls that initialize and modify the control state and operational mode of the module	Keyboard, hard disk, mouse, CD-ROM, floppy disk, and USB/ parallel/serial/network ports
Status Output	API function return values	Video monitor, hard disk, floppy disk, and USB/parallel/serial/ network ports
Power	Not applicable	Power supply

TABLE 4: PORTS AND INTERFACES

G. Roles, Services and Authentication

BASCE supports both the Crypto-Officer and User roles for cryptographic module operation as required by FIPS 140-2. The operator assumes either of the roles based on the operations being performed; BASCE does not provide authentication and does not support concurrent operators.

BASCE supports the following services and access control.

Role	Service	CSP	Access
ADMINISTRATIVE SERVICES			
Crypto-Officer	Install	None	-
	Uninstall	None	-
Crypto-Officer or **User**	Initialize	None	-
	Show Version	None	-
	Show Mode	None	-
	Set FIPS Mode	Module Integrity HMAC-SHA-256 Key	Execute
	Run Self-Tests	Module Integrity HMAC-SHA-256 Key	Execute
	Zeroize	AES, TDES, HMAC-SHA-*Key, PRNG Seed or Seed Key	Write
CRYPTOGRAPHIC SERVICES			
Crypto-Officer or **User**	Symmetric Encryption/Decryption	AES, TDES Symmetric Key	Execute
	Message Digest (SHA-1, SHA-256, SHA-384, SHA-512)	none	Execute
	Keyed Hash (HMAC-SHA-1, -SHA-256, -SHA-384, -SHA-512)	HMAC-SHA-* Key	Execute

	Pseudo Random Number Generation ANSI X9.31 Appendix A.2.4 PRNG (AES-128/-256)	Seed, Seed Key	Write, Execute

TABLE 5: SERVICES AND ACCESS CONTROL

Note that only the Crypto-Officer can install and uninstall BASCE but all other services can be performed by either Crypto-Officer or User role. The administrative services from Table 5 are described in more detail in Table 6 below.

Service	Description	API Function	Input	Output
Initialize	Initialize BASCE to Non-Approved Mode; must be run before any other function	CM_init	None	Integer indicating Success or Failure
Show Version	Obtain BASCE's version number	CM_getFIPSModule Version	None	Major, Minor, Patch, Build number as 32-bit unsigned integer
Set FIPS Mode	Set BASCE operation mode to FIPS Mode	CM_setFIPSMode	Mode	Integer indicating Success or Failure
Show Mode	Obtain BASCE's current operating mode	CM_getFIPSMode	None	Integer indicating FIPS or Non-Approved Mode
Run Self-Tests	Execute all BASCE known answer self-tests and module integrity test, blocking all security function output during operation	CM_runSelfTests	None	Success or Failure; In the event of Failure, transition **BASCE** to the error/failed state
Zeroize	Zeroize BASCE's integrity key	CM_zeroize	None	Zeroize BASCE's HMAC-SHA-256 integrity key
	Zeroize an AES, TDES key	CM_releaseCipher	Cipher Handle	Zeroize and free the client-application's memory structure associated with a symmetric cipher
	Zeroize a SHA message digest	CM_releaseHash	Hash Handle	Zeroize and free the client-application's memory structure associated with a message digest
	Zeroize a PRNG Seed and Seed key	CM_releasePRNG	PRNG Handle and Flag	Zeroize and free the client-application's memory structure associated with the module's PRNG. If Flag is true, zeroize and release the AES 128/256 bit key as well
	Zeroize an HMAC key	CM_releaseHMAC	HMAC key handle	Zeroize and free the client-application's memory associated with an HMAC key

TABLE 6: DESCRIPTION OF SERVICES

H. Self-Tests

As mentioned in the services above, BASCE employs self-tests to ensure the module's integrity is intact and that it's functioning properly. Self-tests are performed when the Crypto-Officer

requests that the module operate in FIPS Mode, or on demand while BASCE is in FIPS Mode by invoking the Run Self-Tests service. In addition, the FIPS 140-2 Continuous Random Number Generator Test is performed while the module operates in FIPS Mode.

BASCE's self-tests include *module integrity verification* and cryptographic service *known answer tests*. During execution of self-tests, BASCE blocks access to all cryptographic services. If any self-test fails, the module is transitioned to an error state and subsequent calls will return an error code indicating this condition. Such errors can only be recovered by reloading BASCE by restarting the application, reinstalling BASCE, or returning it BitArmor for analysis and repair if the problem persists.

BASCE Module Integrity Verification

Module integrity verification is a self-test that is performed by checking a build (compile) time HMAC-SHA-256 digest against the runtime executables. If the pre-computed value matches the runtime-computed value, then the test succeeds otherwise BASCE transitions to the irrecoverable error state described above.

BASCE Known Answer Tests (KATs)

Known answer tests are tests for which a set of cryptographic (question, answer) pairs are stored in the cryptographic module and utilized by self-tests for comparison of runtime performance with known results. If any KAT self-test fails, BASCE transitions to the irrecoverable error state.

The following table contains self-tests that are incorporated into BASCE.

<table>
<tr><th colspan="3">Cipher Function KAT Self-Tests</th></tr>
<tr><td rowspan="3">AES</td><td rowspan="3">Encrypt
&
Decrypt</td><td>ECB-128, -192, -256</td></tr>
<tr><td>CBC-128, -192, -256</td></tr>
<tr><td>CTR-128, -256</td></tr>
<tr><td rowspan="2">TDES</td><td rowspan="2">Encrypt
&
Decrypt</td><td>ECB-Key Option 2</td></tr>
<tr><td>CBC-Key Option 2</td></tr>
<tr><th colspan="3">Secure Hash Function KAT Self-Tests</th></tr>
<tr><td colspan="2">SHA1</td><td>Compute 20-byte message digest</td></tr>
<tr><td colspan="2">SHA-256</td><td>Compute 32-byte message digest</td></tr>
<tr><td colspan="2">SHA-384</td><td>Compute 48-byte message digest</td></tr>
<tr><td colspan="2">SHA-512</td><td>Compute 64-byte message digest</td></tr>
<tr><th colspan="3">HMAC Function KAT Self-Tests</th></tr>
<tr><td colspan="2">HMAC-SHA-1</td><td>Compute 20-byte Keyed MAC</td></tr>
<tr><td colspan="2">HMAC-SHA-256</td><td>Compute 32-byte Keyed MAC</td></tr>
<tr><td colspan="2">HMAC-SHA-384</td><td>Compute 48-byte Keyed MAC</td></tr>
</table>

HMAC-SHA-512	Compute 64-byte Keyed MAC
PRNG FUNCTION KAT SELF-TESTS	
AES-ECB-128	Compute 16-byte random number
AES-ECB-256	Compute 16-byte random number
MODULE INTEGRITY VERIFICATION SELF-TEST	
HMAC-SHA-256	Compute run-time HMAC and compare with build-time HMAC

TABLE 7: KNOWN ANSWER TESTS

BASCE Conditional Testing

Conditional testing involves comparison of successively-generated random number values at runtime to ensure a collision does not occur. This testing is done on a *continuous* basis while the module operates in FIPS Mode and is referred to as the Continuous Random Number Generator Test by FIPS 140-2. This means that every time a random number is generated, it is compared with the previously generated value for a collision. If a collision ever occurs during module use, BASCE transitions to the error state in the same manner as it does if a self-test fails.

CONDITIONAL TESTING	
ANSI X9.31 with AES-128/-256 PRNG	Continuous Random Number Generator Test

TABLE 8: CONDITIONAL TESTING

I. Physical Security

BASCE is a software-only cryptographic module and therefore FIPS 140-2 physical security requirements do not apply.

J. Operational Environment

As a dynamically-linked library, BASCE extends the functionality of the client-application that links/loads it. Therefore the user/operator of BASCE is defined to be its client-application for FIPS 140-2 purposes.

This is relevant to the FIPS 140-2 Security Level 1 requirement that cryptographic modules must operate in a single-user mode, operational environment. Since BASCE is a software library, there is only one application that accesses it and nothing further has to be done at the operating system level to meet the FIPS 140-2 requirement.

K. Cryptographic Key Management

BASCE does not support long term key storage and all cryptographic key input to BASCE is through references to client-application memory structures. Any use of cryptographic keys that requires temporary storage within the module uses local function variables that are de-allocated at function termination.

BASCE uses cryptographic keys for symmetric cipher encrypt/decrypt functions AES and TDES, generating HMAC-SHA-1/-256/-384/-512, and as a seed key for its PRNG. Cryptographic keys can be encrypted using AES-128, AES-192, AES-256, or TDES and they can be zeroized by overwriting their memory with zeros using the BASCE API.

If the client-application imports or exports keys outside the physical cryptographic boundary, it must use a FIPS Approved encryption method and import or export them in encrypted form.

Cryptographic keys are stored in memory until they are zeroized using the API. The keys and CSPs that BASCE uses are listed in Table 9 below.

Cryptographic Key/CSP	Size in Bits	Source	Module Input	Module Output	Zeroization Function
PRNG Seed	128	BASCE or Client-Application	Plaintext	Plaintext	CM_releasePRNG
PRNG Seed Key	128 or 256	Client-Application	Plaintext	-	CM_releasePRNG
AES Key	128, 192 or 256		Plaintext	-	CM_releaseCipher
TDES Keys (option 2)	128		Plaintext	-	CM_releaseCipher
HMAC-SHA-256 Module Integrity Key	256	BASCE	-	-	CM_zeroize
HMAC Key	Any	Client-Application	Plaintext	-	CM_releaseHMAC

TABLE 9: CRYPTOGRAPHIC KEYS AND CSPS

CSP Output Controls

Note, that as required by FIPS 140-2, two independent, internal actions are required in order for the module to output any CSP. The PRNG Seed is the only CSP that may be output by BASCE and there are two routines that can output it – CM_initPRNG and CM_getPRNGRandom.

During initialization of the PRNG data structure, CM_initPRNG will generate and output a PRNG seed if one is not input (i.e., generated and supplied by the client-application). Subsequent random number generation requires re-computing and outputting the PRNG seed via CM_getPRNGRandom.

Output of the PRNG seed is controlled by two flags – a global module flag and an input parameter flag for each function above. In order for a function to output the PRNG seed, both flags must be set to 'TRUE'; otherwise the function will fail with an error indicating that CSP output was blocked.

The two independent, internal actions are:

1. First, the client-application must indicate that PRNG CSP output is required during module initialization by setting CM_init's Boolean input parameter "outputCSP" to 'TRUE'. This action sets the global module flag to 'TRUE'.
2. Second, when the client-application invokes PRNG initialization and random number generation services, it must again indicate that CSP output is requested through the corresponding function's input parameter "outputCSP", setting it to 'TRUE' as well.

Note that in cases where the PRNG seed is initially generated by the client-application, CSP output controls are still required in order to obtain the updated seed values computed by the module during random number generation.

Also note that if a PRNG seed is not supplied by the client-application initially, one will be generated via **CryptGenRandom** on Windows operating systems and **/dev/random** on SUSE Linux.

L. Design Assurance

BitArmor Systems utilizes Perforce Server (v2006.2) for configuration management of product source code and documentation. Perforce supports authentication, access control, and logging. Refer to www.perforce.com for more information.

M. Mitigation of Other Attacks

BASCE does not provide any security mechanisms in addition to those required by FIPS 140-2 security requirements for cryptographic modules.

III. FIPS 140-2 Compliant Operation

A. Crypto-Officer Guidance

Installation and Initialization

BASCE is a component of the BitArmor DataControl security product and is not an end-user product itself. The software libraries that make it up are installed and uninstalled as part of a larger product installation procedure that is documented in the product installation guide.

Upon loading by the client-application, BASCE must be initialized by calling CM_init before it can be used. This routine initializes BASCE's control structures and establishes a Non-Approved mode of operation.

BitArmor DataControl applications specify the mode of operation at run-time by calling CM_setFIPSMode, the Set FIPS Mode service, and they determine the current mode of operation by calling CM_getFIPSMode, the Show Mode service.

Specifically, an application calls CM_setFIPSMode using the CM_FIPS_LEVEL1 value as an input argument in order to establish FIPS Mode. This produces a result status output of success or failure:

```
// Status indicator variable
int status = CM_FAILURE;
// Initialize the module for CSP output and set FIPS Mode
if( (status = CM_init(TRUE)) == CM_SUCCESS ) {
        if( (status = CM_setFIPSMode(CM_FIPS_LEVEL1)) == CM_SUCCESS ) {
                // Operating in FIPS Mode…
        } else {
                // Could not establish FIPS Mode…
        }
} //end
```

Additionally, an application can confirm BASCE's operating mode with the Show Mode service:

```
// Determine the module's mode of operation
If( (status = CM_getFIPSMode()) == CM_FIPS_LEVEL1 ) {
        //FIPS Mode…
} else {
        //Non-Approved Mode…
} //end
```

Zeroization

BASCE's API provides zeroization routines that BitArmor DataControl applications use to zeroize their in-memory CSPs (recall that BASCE stores no CSPs internally). It is recommended that client applications zeroize all CSPs prior to program termination and immediately prior to a transition from Non-Approved to FIPS Mode operation if any CSP has been used in Non-Approved Mode. (Note that BASCE does not allow transitions from FIPS Mode to Non-Approved Mode.)

Each cryptographic security function of BASCE has an associated data type that the client-application must instantiate and use with BASCE's API. Client applications pass references to data type instances in their BASCE function calls for initialization/allocation, subsequent use of the available cryptographic functionality, zeroization and de-allocation. BASCE initializes them by allocating the appropriate amount of memory and initial values based on the cryptographic function type, mode, key length, etc. The client-application then assigns data values and performs further operations in a similar manner until it finishes with a particular cryptographic function. When the client-application deems a CSP no longer necessary, it should zeroize and de-allocate the memory associated with it using the appropriate BASCE zeroization function call as described in Table 6 above. (Zeroization can be invoked by Crypto-Officer or User role.)

Note that zeroizing the module integrity key (via CM_zeroize) will prevent a client-application from subsequently setting FIPS Mode. This action zeroizes the HMAC-SHA-256 key used by BASCE for Module Integrity Verification in memory only.

Management

The Crypto-Officer is not required to perform any maintenance or module management after it is installed. All module management is performed at the application level.

B. User Guidance

BitArmor DataControl applications utilize BASCE to provide FIPS-approved cryptographic functionality to their end-users. BitArmor DataContol applications are programmed to utilize BASCE's services in a manner that is consistent and correct according to BASCE's detailed design and API specifications. End users of BitArmor DataControl do not need to manage, or service BASCE. End users interact with the application and do not interface directly with BASCE.