Cryptographic Extension for BREW®

Cryptographic Engine

Version 2.2

Security Policy

80-D8804-1, Rev. D

Qualcomm Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714

18 September, 2007

Table of Contents

1.0 Introduction

This is a non-proprietary Federal Information Processing Standard (FIPS) 140-2 Security Policy for QUALCOMM's Cryptographic Extension for Binary Runtime Environment for Wireless (BREW) Cryptographic Engine.

1.1 Summary

QUALCOMM's Cryptographic Extension for BREW provides services of various cryptographic algorithms such as hash algorithms, encryption schemes, message authentication, and public key cryptography. This Security Policy specifies the rules under which the Cryptographic Extension for BREW Cryptographic Engine must operate. These security rules are derived from the requirements of FIPS 140-2 [1], and related documents [6, 7, 8]. Architecturally, the Cryptographic Extension for BREW comprises two main parts:

- **Cryptographic Engine:** The Cryptographic Engine is written in C and contains all the cryptographic functions. It is compiled separately from the BREW extension code. The cryptographic boundary is drawn around the Cryptographic Engine.
- **BREW extension code:** The BREW extension code provides an interface between the BREW application framework and the Cryptographic Engine. BREW applications must access the Cryptographic Engine via the BREW extension. It is written in C++ and linked with the Cryptographic Engine to create the Cryptographic Extension for BREW.

For the details of the BREW extension interface, including the algorithms available through the BREW extension, please refer to the user documentation of the Cryptographic Extension.

1.2 Purpose

This Security Policy is created to:

- Satisfy a requirement for FIPS 140-2 validation
- Outline the Cryptographic Extension for BREW Cryptographic Engine conformance to FIPS 140-2 Level 1 Security Requirements
- Describe how to configure and operate the Cryptographic Engine in order to comply with FIPS 140-2

1.3 References

[1] NIST *Security Requirements for Cryptographic Modules*, December 3, 2002.

[2] NIST *Security Requirements for Cryptographic Modules, Annex A: Approved Security Functions for FIPS PUB 140-2,* April 3, 2006.

[3] NIST *Security Requirements for Cryptographic Modules, Annex B: Approved Protection Profiles for FIPS PUB 140-2,* November 4, 2004.

[4] NIST *Security Requirements for Cryptographic Modules, Annex C: Approved Random Number Generators for FIPS PUB 140-2,* January 31, 2005.

[5] NIST *Security Requirements for Cryptographic Modules, Annex D: Approved Key Establishment Techniques for FIPS PUB 140-2,* September 12, 2005.

[6] NIST *Derived Test Requirements for FIPS 140-2, Draft,* March 24, 2004.

[7] NIST *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program,* October 5, 2006.

[8] NIST *Frequently Asked Questions for the Cryptographic Module Validation Program,* August 24, 2006.

[9] Qualcomm Incorporated *BREW Application Extensions Overview, 80-D4190-2, Rev. A,* December 16, 2002.

1.4 Revision History

Table 1 identifies and describes revisions to this security policy.

Table 1. Document revision history

Revision	Date	Comments	Author
Rev. D	2007/09/18	Clarified allowed key agreement algorithms and sizes Minor editorial corrections	pbelding
Rev. C	2007/04/23	Revised copyright statement. Fixed some cut and paste errors	ayamada
Rev. B	2007/04/04	Rectified inconsistent use of “module”.	rtsang
	2007/03/21	Added further clarification and some editorial corrections.	ayamada
Rev. A	2007/02/23	Initial draft release.	mrausch
1.9	2007/02/01	• Added QUALCOMM-specific	ayamada

Revision	Date	Comments	Author
		language. • Added non-FIPS algorithms.	
1.8	2007/01/25	Improved the description of the cryptographic module.	ayamada
1.7	2007/01/16	Added further clarification on BREW extension.	ayamada
1.6	2007/01/16	• Further clarified installation and un-installation procedure. • Supplied algorithm certificate numbers.	ayamada
1.5	2007/01/11	Removed "Key generation" line for symmetric ciphers from table	ajanicij
1.4	2007/01/05	• Changed copyright to 2006-2007 • Added description of modification to change history • Added manual installation • Added manual installation to the description of the installation procedure.	ajanicij ajanicijevic
1.3	2007/01/02	Modified to be more consistent with the BREW architecture.	rtsang
1.2	2006/11/03	Modified with corrections after the first review.	ayamada
1.1	2006/10/13	Initial draft	ayamada

2.0 Cryptographic Module Specification

The Cryptographic Extension for BREW Cryptographic Engine is a multiple-chip standalone software cryptographic module that operates with the following commercially available components:

- General-purpose handheld computing and communication device
- Operating System (OS) that runs on the device
- BREW that runs on the device and OS

2.1 Physical Specifications

The device generally consists of the following physical components:

1. CPU (Microprocessor)
2. Memory

a.) Working memory is located on the RAM containing the following spaces:

i. Input/output buffer

ii. Plaintext/ciphertext buffer

iii. Control buffer

Key storage is not deployed in this module.

b.) Program memory is also located on RAM.

3. Data Storage (such as flash memory or hard disk)
4. Display
5. Keys and Buttons
6. Network Interface
7. Serial and/or Parallel Port
8. Battery

Figure 1 illustrates the configuration of the device.

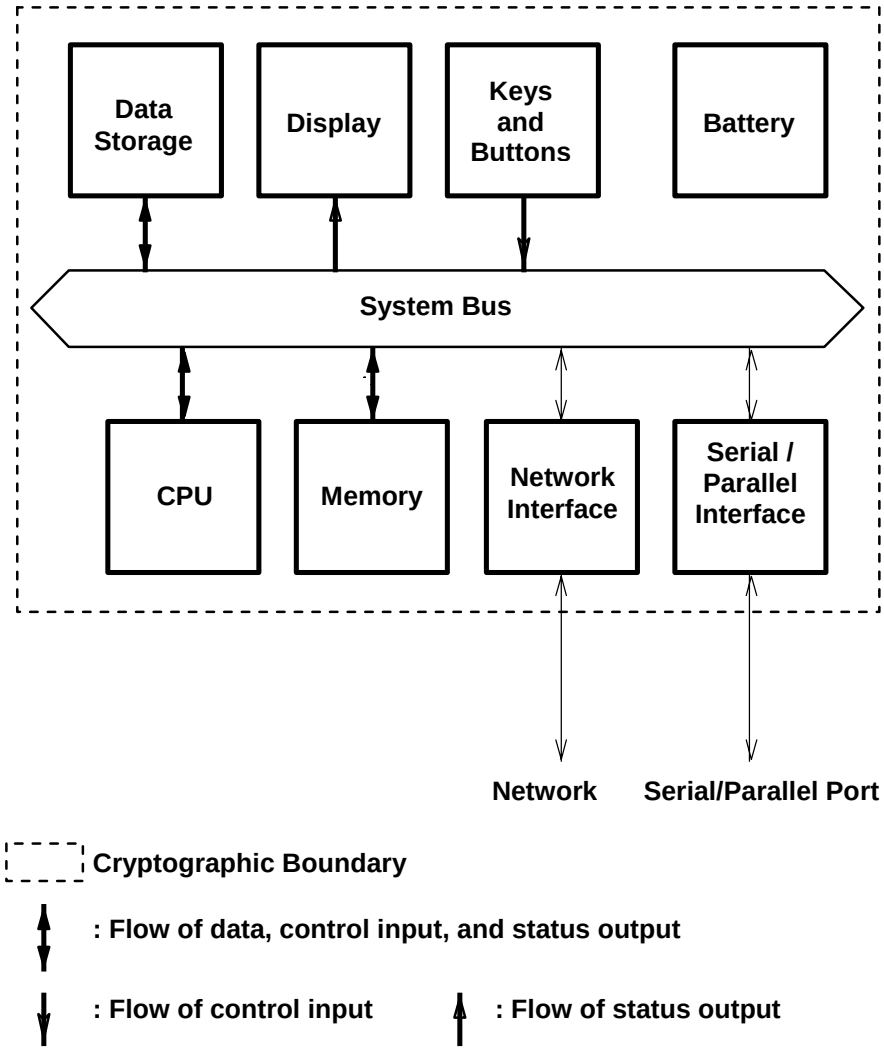

Figure 1. Cryptographic Module Hardware Block Diagram

2.2 Computer Hardware, OS, and BREW

The Cryptographic Engine is tested on a LG VX9800 phone with an ARM processor running BREW 3.1 over the LG Firmware OS T98VZV50. BREW SDK 3.1.2 is used to create and test the Cryptographic Engine. The module runs on equivalent BREW versions on various processors, hardware, and OS, while maintaining its compliance to the FIPS 140-2 Level 1 requirements.

2.3 Software Specifications

The Cryptographic Extension for BREW components are manufactured by QUALCOMM Inc. and Certicom Corp., providing services to the C++ computer language users. The interface into the Cryptographic Extension for BREW is via Application Programmer's Interface (API) function calls. These function calls provide the interface to the cryptographic services, for which the parameters and return codes provide the control input and status output, as shown in Figure 2.

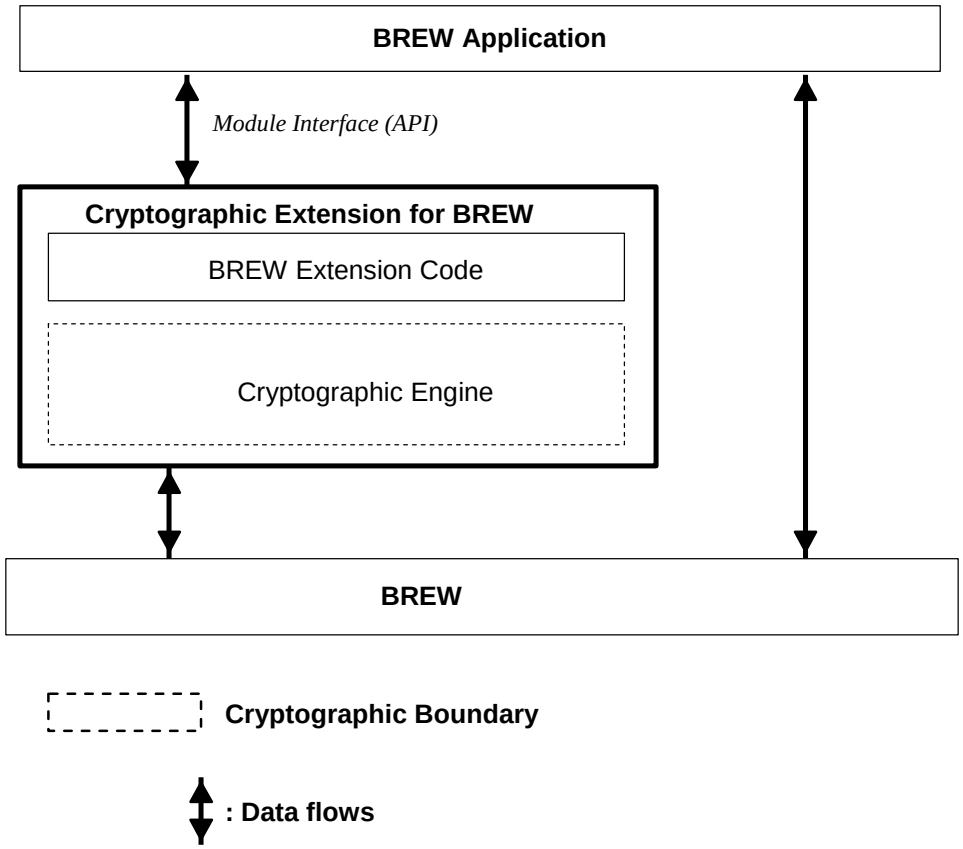

Figure 2. Cryptographic Module Software Block Diagram

3.0 Cryptographic Module Ports and Interfaces

The physical and logical interfaces are summarized in Table 2.

Table 2. Logical and physical interfaces

FIPS 140-2 Interface	Logical Interface	Physical Interface
Data Input	API	Wireless modem
Data Output	API	Wireless modem
Control Input	API	Keys and buttons
Status Output	Return Code	Display
Power Input	Initialization function	Not applicable (battery is included)
Maintenance	Not supported	Not supported

4.0 Roles, Services, and Authentication

4.1 Roles

The Cryptographic Engine supports Crypto Officer and User Roles, meeting FIPS 140-2 Level 1 requirements. These roles are enforced by this Security Policy. The Crypto Officer has the responsibility for installing the Cryptographic Engine (see Table 3). In order to operate the module securely, it is the Crypto Officer and User's responsibility to confine calls to those methods that have been FIPS 140-2 approved. Thus, in the approved mode of operation, all Roles shall confine themselves to calling FIPS Approved algorithms, as marked in Table 3.

4.2 Services

The Cryptographic Engine supports many cryptographic algorithms. The set of cryptographic algorithms supported by the Cryptographic Engine are provided in Table 3.

- TDES, AES, SHS (SHA-1, SHA-224, SHA-256, SHA-384, and SHA- 512), HMAC-SHS (HMAC-SHA-1, HMAC-SHA-224, HMAC-SHA256, HMAC- SHA-384, and HMAC-SHA-512), RNG, DSA, RSA PKCS #1, and ECDSA algorithms have been validated to comply with FIPS.
- The Cryptographic Engine also supports FIPS allowed key establishment techniques (key agreement and key wrapping) DH, ECDH, ECMQV, and RSA PKCS #1.
- DES, DESX, ARC4, MD2, MD5, and HMAC-MD5 are supported as non FIPS-approved algorithms. In order to operate the module in compliance with FIPS, these algorithms should not be used.

The BREW extension code does not expose the symmetric key generation and RNG seeding functions. The RNG is seeded by using the BREW "`AEECLSID RANDOM`" class.

4.3 Operator Authentication

The Cryptographic Engine does not deploy an authentication mechanism. The roles of Crypto Officer and User are implicitly selected by the operator.

Table 3. Roles, Services and Access

Service	Crypto Officer	User	Keys and CSPs	Access
Installation, etc.				
Installation	X			
Uninstallation	X			
Self-tests	X	X	ECDSA public key	Use
Show status	X	X		
Symmetric Ciphers				
Key generation	X	X	AES, TDES keys	Create, Read
Encrypt	X	X	AES, TDES keys	Use
Decrypt	X	X	AES, TDES keys	Use
Hash Algorithms and Message Authentication				
Hashing	X	X		
Message Authentication	X	X	HMAC keys	Use
Random Number Generation				
Seeding	X	X	Seed	Use
Request	X	X		
Digital Signature				
Key pair generation	X	X	RSA, DSA, ECDSA key pairs	Create, Read
Sign	X	X	RSA, DSA, ECDSA private keys	Use
Verify	X	X	RSA, DSA, ECDSA public keys	Use
Key Agreement				
Key pair generation	X	X	ECDH, ECMQV, DH key pairs	Create, Read
Shared secret generation	X	X	ECDH, ECMQV, DH key pairs	Use
Key Wrapping				

Key pair generation	X	X	RSA key pairs	Create, Read
Wrap	X	X	RSA public keys	Use
Unwrap	X	X	RSA private keys	Use

Table 4. Supported Algorithms and Standards

	Algorithm	FIPS approved or allowed	Cert number
Block Ciphers	TDES (ECB, CBC, CFB64, OFB64) [FIPS 46-3]	X	#488
	AES (ECB, CBC, CFB128, OFB128, CTR) [FIPS 197]	X	#473
	DES (ECB, CBC, CFB64, OFB64)		
	DESX (ECB, CBC, CFB64, OFB64)		
Stream Cipher	ARC4		
Hash Functions	SHA-1 [FIPS 180-2]	X	#541
	SHA-224 [FIPS 180-2]	X	#541
	SHA-256 [FIPS 180-2]	X	#541
	SHA-384 [FIPS 180-2]	X	#541
	SHA-512 [FIPS 180-2]	X	#541
	MD5 [RFC 1321]		
	MD2 [RFC 1115]		
Message Authentication	HMAC-SHA-1 [FIPS 198]	X	#230
	HMAC-SHA-224 [FIPS 198]	X	#230
	HMAC-SHA-256 [FIPS 198]	X	#230
	HMAC-SHA-384 [FIPS 198]	X	#230
	HMAC-SHA-512 [FIPS 198]	X	#230
	HMAC-MD5 [RFC 2104]		
RNG	ANSI X9.62 RNG [ANSI X9.62]	X	#256
Digital Signature	DSS [FIPS 186-2]	X	#194
	ECDSA [FIPS 186-2, ANSI X9.62]	X	#42
	RSA PKCS1-v1 5	X	#194

	[PKCS #1 v2.1]		
Key Agreement	DH [ANSI X9.42]	X	
	ECDH [ANSI X9.63]	X	
	ECMQV [ANSI X9.63]	X	
Key Wrapping	RSA PKCS1-v1 5 [PKCS #1 v2.1]	X	
	RSA OAEP [PKCS #1 v2.1]	X	

5.0 Finite State Model

The Finite State Model contains the following states:

- Installed/Uninitialized
- Initialized
- Self-Test
- Idle
- Crypto Officer/User
- Error

5.1 State Transition Features

The following step identifies important features of the state transition:

1. When the Cryptographic Engine is installed by the Crypto Officer, the module is in the Installed/Uninitialized state.
2. After the Cryptographic Engine is loaded on the memory along with the application, the module turns to the Initialization state when the initialization command is applied to the module.
3. The Cryptographic Engine transits to the Self-Test state automatically, running the Power-up Tests. While in the self-test state, all data output via the data output interface is prohibited.
    - If success occurs, the module enters Idle state.
    - If failure, the module enters Error state and the module is disabled. From the Error state the Crypto Officer may need to re-install to attempt correction.
4. From the Idle state (which is only entered if self-tests have succeeded), the module can transit to the Crypto Officer/User state when an API function is called.
5. When the API function has completed successfully, the state returns to Idle.
6. If the Conditional Test (Continuous RNG Test or Pair-wise Consistency Test) fails, the state transits to Error state and the module is disabled.
7. When On-demand Self-test is executed, the module enters the Self-Test state.
    - If success occurs, the module enters Idle state.
    - If failure occurs, the module enters Error state and the module is disabled.
8. When the de-initialization command is executed, the module returns to the Installed/Uninitialized state.

6.0 Physical Security

Physical security is not applicable to this software module at Level 1 Security.

7.0 Operational Environment

The Cryptographic Engine is designed for mobile phones, which are single user devices, thus always in single user mode.

8.0 Cryptographic Key Management

The Cryptographic Engine provides the underlying functions to support FIPS 140-2 Level 1 key management. The user will select FIPS-approved algorithms and will handle keys with appropriate care to build up a system that complies with FIPS 140-2. It is the Crypto Officer and User's responsibility to select FIPS 140-2 validated algorithms, as identified in Table 4.

8.1 Key Generation

The Cryptographic Engine provides FIPS 140-2 compliant key generation. The underlying random number generation uses a FIPS Approved method, the ANSI X9.62 RNG [4].

8.2 Key Establishment

The Cryptographic Engine provides the following FIPS allowed key establishment techniques [5], as shown in Table 5.

1. EC Diffie-Hellman (ECDH)
2. ECMQV
3. Diffie-Hellman (DH)
4. RSA PKCS1-v1 5
5. RSA OAEP

The RSA key wrapping techniques above are based on the PKCS #1 v2.1 standard, and are used to transport keys.

Table 5. FIPS allowed establishment techniques

This technique. . .	supports. . .	and provides. . .
FIPS Mode		
ECDH key agreement technique	elliptic curve size from 163 bits to 521 bits	between 80 and 256 bits of security strength.
ECMQV key agreement technique	elliptic curve size from 163 bits to 521 bits	between 80 and 256 bits of security strength.
DH key agreement technique	modulus size from 1024 bits to 15360 bits	between 80 and 256 bits of security strength.
RSA implementation	modulus size from 1024 bits to 15360 bits	between 80 and 256 bits of security strength.
Non-FIPS Mode		
DH key agreement technique	modulus size from 512 bits to 1023 bits	between 56 and 79 bits of security strength.
RSA implementation	modulus size from 512 bits to 1023 bits	between 56 and 79 bits of security strength.

8.3 Key Entry and Output

It is not allowed to import or export keys, including a seed key, and other security sensitive information outside of physical boundary in plaintext format. For key entry and export, users must ensure to deploy appropriate encryption method using FIPS Approved encryption algorithms in the Cryptographic Engine or any other FIPS validated cryptographic module.

8.4 Key Storage

The Cryptographic Engine is a low-level cryptographic toolkit, and as such does not provide key storage.

8.5 Zeroization of Keys

The Cryptographic Engine functions zeroize all intermediate security sensitive material. All CSPs are zeroized when they are no longer needed by calling destroy functions. Destruction of CSP is enforced in a manner such that missed destruction will make the Cryptographic Engine no longer functional.

9.0 Self-Tests

9.1 Power-up Tests

Self-tests are initiated automatically by the module at start-up. The following tests are applied:

- **Known Answer Tests (KATs):** KATs are performed on TDES, AES, SHS, HMAC-SHS, RNG, and RSA PKCS #1 v1.5 Signature Algorithm. For DSA and ECDSA, Pair-wise Consistency Test is used.
- **Software Integrity Tests:** The software integrity test deploys ECDSA signature validation to verify the integrity of the module.

9.2 On-Demand Self-Tests

On-demand self tests may be invoked by the Cryptographic Officer or User by invoking a function, which is described in the Crypto Officer and User Guide in Appendix A.

9.3 Conditional Tests

The Continuous RNG Test is executed on all RNG generated data, examining the first 160 bits of each requested random generation for repetition. This continuous testing ensures that the RNG is not stuck at any constant value. Also, upon each generation of a RSA, DSA, or ECDSA key pair, the generated key pair is tested for their accuracy by generating a signature and verifying the signature on a given message as a pair-wise Consistency Test.

9.4 Failure of Self-Tests

Failure of the self-tests will place the cryptographic module in the Error state, wherein no cryptographic operations can be performed. If any self-test fails, the cryptographic module will output an error code.

10.0 Design Assurance

10.1 Configuration Management

A configuration management system for the cryptographic module is employed and is described in a document to the testing lab. The configuration management system uses the Concurrent Versioning System (CVS) to track the configurations. The CVS is also used to maintain the documentation, and each document contains version number, revision history, and a list of references.

10.2 Delivery and Operation

Please refer to Section A.1 of Crypto Officer and User Guide in Appendix A to review the steps necessary for the secure installation and initialization of the cryptographic module.

10.3 Development

Detailed design information and procedures are described in documentation submitted to the testing laboratory. This toolkit is designed and developed using high level language C, for C and C++ users. The low level language assembly is used to optimize lower level operations. Development for the cryptographic module is carried out in a multi-platform environment. Certicom is using the CVS revision control system to control revisions. Development of new versions and major features are performed on a branch of the software, and these branches merged back into the trunk after testing and review, but the branch is maintained to perform post-release maintenance.

10.3.1 Test process

Releases are first tested in engineering, via an automated daily procedure—the daily build. Releases are then committed to the product candidate repository. These releases are then sent to the Quality Assurance (QA) department, who must run their own tests before they move them into the product branch. Only the QA department can move candidates into products.

10.3.1.1 Daily

The software is built automatically on over 45 platforms (servers to embedded devices and PDAs) and the regression tests are then run. In addition, daily integration builds are performed to check the use of the cryptographic library as used in the higher level protocol products.

10.3.1.2 Weekly

Code test coverage and code quality tools (purify and insure) are run on the product. These tests are conducted weekly, as they are very extensive tests that take longer than one day to complete. Weekly benchmarks are also automatically performed on a representative subset of devices.

10.4 Guidance Documents

Crypto Officer Guide and User Guide are provided in Appendix A. This appendix outlines the operations for Crypto Officer and User to ensure the security of the module.

11.0 Mitigation of Other Attacks

The Cryptographic Engine implements mitigation of the following attacks:

- Timing attack on RSA
- Attack on biased private key of DSA

11.1 Timing Attack on RSA

When employing Montgomery computations, timing effects allow an attacker to tell when the base of exponentiation is near the secret modulus, leaking information concerning the secret modulus.

In order to mitigate this attack, the bases of exponentiation are randomized by a novel technique that requires no inversion to remove (unlike other blinding methods e.g. BSAFE Crypto-C User Manual v 4.2). Note that remote timing attacks are practical:

http://crypto.stanford.edu/ dabo/papers/ssl-timing.pdf

11.2 Attack on Biased Private Key of DSA

The standards for choosing ephemeral values in El-Gamal type signatures introduce a slight bias. Means to exploit these biases were presented to ANSI by D. Bleichenbacher.

In order to mitigate this attack, the bias in the RNG is reduced to levels which are far below the Bleichenbacher attack threshold. Change Notice 1 of FIPS 186-2 is published to mitigate this attack:

http://csrc.nist.gov/CryptoToolkit/tkdigsigs.html

Appendix A Crypto Officer And User Guide

A.1 Installation

As described in [9], a BREW extension can be downloaded manually from the development system. The Crypto Officer should download manually Cryptographic Extension for BREW, which contains the Cryptographic Engine, to develop applications. In order to execute a secure installation of the Cryptographic Extension for BREW, along with the Cryptographic Engine, the Crypto Officer must follow the procedure described in this section.

A.2 Uninstallation

To uninstall, remove Cryptographic Extension for BREW from the development environment.

A.3 Commands

A.3.1 Initialization

`sbg22_FIPS140Initialize()`

This function runs a series of self-tests on the module. These tests examine the integrity of the module, and the correct operation of the cryptographic algorithms. If these tests are successful, a value of `SB_SUCCESS` will be returned and the module will be enabled.

This function is not accessible directly from BREW applications. It can be accessed via the `ICRYPTOBASE_Init()` macro.

A.3.2 De-initialization

`sbg22_FIPS140Deinitialize()`

This function de-initializes the module. This function is not accessible directly from BREW applications. It can be accessed via the `ICRYPTOBASE_Uninit()` macro.

A.4 Self-Tests

sbg22_FIPS140RunTest()

This function runs a series of self-tests, and return `SB_SUCCESS` if the tests are successful. These tests examine the integrity of the module, and the correct operation of the cryptographic algorithms. If these tests fail, the module will be disabled. Section A.3 of this document describes how to recover from the disabled state.

This function is not accessible directly from BREW applications. It can be accessed via the `ICRYPTOBASE_SelfTest()` macro.

A.5 Show Status

sbg22_FIPS140GetState()

This function will return the current state of the module. This function is not accessible directly from BREW applications. It can be accessed via the `ICRYPTOBASE_GetState()` macro.

A.6 Disable Status

When the Cryptographic Engine becomes disabled, you can attempt to return the module to the Installed by calling `ICRYPTOBASE_Uninit()`, and then to initialize the module using `ICRYPTOBASE_Init()`.

- If the initialization is successful, the module will be recovered.
- If this attempt fails, please contact QUALCOMM Support.