UnaliWear, Inc.
Kanega Watch

FIPS 140-2 Cryptographic Module
Non-Proprietary Security Policy

Version: 1.1
Date: January 20, 2017

Table of Contents

1 Introduction 4

1.1 Hardware and Physical Cryptographic Boundary 5
1.2 Software and Logical Cryptographic Boundary 5
1.3 Modes of Operation 6

2 Cryptographic Functionality 7

2.1 Critical Security Parameters 8
2.2 Public Keys 8

3 Roles, Services, and Authentication 9

3.1 Assumption of Roles 9
3.2 Services 9

4 Self-tests 12

5 Physical Security 12

6 Operational Environment 12

7 Mitigation of Other Attacks Policy 12

8 Security Rules and Guidance 12

9 References and Definitions 13

10 Appendix A – Installation Instructions 16

10.1 OpenRTOS INSTALLATION 16
10.2 Kanega Watch FIPS API 17

List of Tables and Figures

Table 1 – Tested Operating Environment 4
Table 2 - Security Level of Security Requirements 5
Table 3 – Ports and Interfaces 5
Figure 1 – Module Block Diagram 6
Table 4 – Approved and CAVP Validated Cryptographic Functions 7
Table 5 – Critical Security Parameters (CSPs) 8
Table 6 – Public Keys 8
Table 7 – Roles Description 9
Table 8 – Authorized Services available in FIPS mode 9
Table 9 – CSP Access Rights within Services 11
Table 10 – Power-on Self-tests 12
Table 11 – References 13
Table 12 – Acronyms and Definitions 14
Table 13 – Source Files 15

1 Introduction

This document defines the Security Policy for the UnaliWear, Inc. Kanega Watch (Software Version 3.9.2) module, hereafter denoted the Module. The Module is a cryptography software library. The Module meets FIPS 140-2 overall Level 1 requirements.

The Module is intended for use by US Federal agencies and other markets that require FIPS 140-2 validated cryptographic functionality. The Module is a software-only module, multi-chip standalone module embodiment; the cryptographic boundary is the collection of object files from the source code files listed in Table 13 – Source Files. console.bin is the compiled binary executable that is the module's cryptographic boundary. No software components have been excluded from the FIPS 140-2 requirements.

Operational testing was performed for the following Operating Environment:

Table 1 – Tested Operating Environment

	Operating System	Processor	Platform
1	OpenRTOS v9.0.0	ATMEL ATSAM4L	Atmel Sam4L8 Xplained Pro

The FIPS 140-2 security levels for the Module are as follows:

Table 2 - Security Level of Security Requirements

Security Requirement	Security Level
Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

1.1 Hardware and Physical Cryptographic Boundary

The physical cryptographic boundary is the general purpose computer where the Module is installed. The Module relies on the computer system where it is running for input/output devices.

Table 3 – Ports and Interfaces

Description	Logical Interface Type
API entry point	Control in
API function parameters	Data in
API return value	Status out
API function parameters	Data out

1.2 Software and Logical Cryptographic Boundary

Figure 1 depicts the Module operational environment.

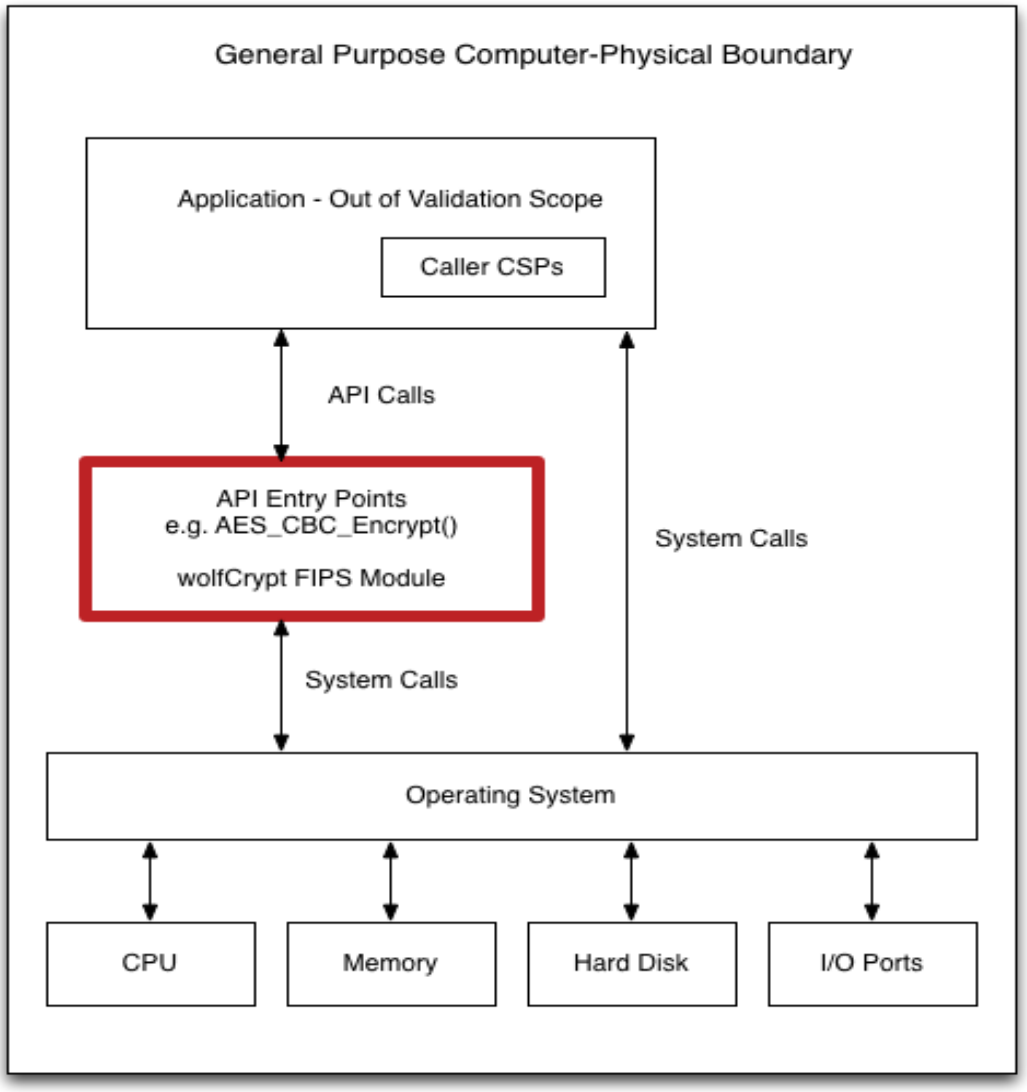

Figure 1 – Module Block Diagram

The above diagram shows the Logical Boundary highlighted in red contained within the Physical Boundary. The Logical Boundary contains all FIPS API entry points. The Logical Boundary is invoked by the Application through the API Calls.

1.3 Modes of Operation

The Module supports a FIPS Approved mode of operation only. FIPS Approved algorithms are listed in Table 4.

For installation instructions, see Appendix A – Installation Instructions.

The conditions for using the module in an Approved mode of operation are:

1. The module is a cryptographic library and it is intended to be used with a calling application. The calling application is responsible for the usage of the primitives in the correct sequence.

2. The keys used by the module for cryptographic purposes are determined by the calling application. The calling application is required to provide keys in accordance with FIPS 140-2 requirements.

The command “wolfCrypt_GetStatus_fips()” can be called to verify that the module is in the FIPS mode.

2 Cryptographic Functionality

The Module implements the FIPS Approved cryptographic functions listed in the tables below.

Table 4 – Approved and CAVP Validated Cryptographic Functions

Algorithm	Description	Cert #
AES	[FIPS 197, SP 800-38A] Functions: Encryption, Decryption Modes: CBC, CTR Key sizes: 256 bits	4012
HMAC	[FIPS 198-1] Functions: Generation, Verification SHA sizes: SHA-256	2617
SHA	[FIPS 180-4] Functions: Message Digest SHA sizes: SHA-256	3310

2.1 Critical Security Parameters

All CSPs used by the Module are described in this section. All usage of these CSPs by the Module (including all CSP lifecycle states) is described in the services detailed in Section 3. The CSP names correspond to the API parameter inputs.

Table 5 – Critical Security Parameters (CSPs)

CSP	Description / Usage
HMAC Key	KeyedHash key .This key is greater or equal to 112 bits.
AES EDK	AES (256) encrypt/decrypt key

2.2 Public Keys

Table 6 – Public Keys

Key	Description / Usage
None	No public keys are used in this build

3 Roles, Services, and Authentication

3.1 Assumption of Roles

The Module supports two distinct operator roles, User and Cryptographic Officer (CO). The cryptographic module does not provide an authentication or identification method of its own. The CO and the User roles are implicitly identified by the service requested.

Table 7 lists all operator roles supported by the Module. The Module does not support a maintenance role or bypass capability.

Table 7 – Roles Description

Role ID	Role Description	Authentication Type	Authentication Data
CO	The Cryptographic Officer Role is assigned the Zeroize service. The installation of the module should also be done by the CO.	None	None
User	The User Role is assigned all services except Zeroize.	None	None

3.2 Services

All services implemented by the Module are listed in the tables below with a description of service CSP access.

Table 8 – Authorized Services available in FIPS mode

Service	Description	Role
Module Reset (Self-test)	Reset the Module by restarting the application calling the Module. Does not access CSPs.	User
Show status	Functions that give module status feedback. Does not access CSPs. The calling application may use the wolfCrypt_GetStatus_fips() API to determine the current status of the Module. A return code of 0 means the Module is in a state without errors. Any other return code is the specific error state of the Module.	User
Zeroize	Functions that destroy CSPs. All services automatically overwrite memory bound CSPs. Cleanup of the stack is the duty of the application. Restarting the general purpose computer clears all CSPs in RAM.	CO

Service	Description	Role
Symmetric encrypt/decrypt	Used to encrypt and decrypt data using AES-256 EDK. CSPs passed in by the application API's are listed in this public document: https://www.wolfssl.com/wolfSSL/Docs-wolfssl-manual-18-1-wolfcrypt-api-aes.html	User
Message digest	Used to generate a SHA-256 message digest. Does not access CSPs. API's are listed in this public document: https://www.wolfssl.com/wolfSSL/Docs-wolfssl-manual-18-23-wolfcrypt-api-sha256.html	User
Keyed hash	Used to generate or verify data integrity with HMAC-SHA256. The HMAC Key is passed in by the application. API's are listed in this public document: https://www.wolfssl.com/wolfSSL/Docs-wolfssl-manual-18-18-wolfcrypt-api-hmac.html	User

Table 9 defines the relationship between access to CSPs and the different module services. The modes of access shown in the table are defined as:

- R = Read: The module reads the CSP. The read access is typically performed before the Module uses the CSP.
- E = Execute: The module executes using the CSP.
- Z = Zeroize: The module zeroizes the CSP.

Table 9 – CSP Access Rights within Services

Service	CSPs	
	HMAC Key	AES EDK
Module Reset (Self-test)	-	-
Show Status	-	-
Zeroize	Z	Z
Symmetric encrypt/decrypt	-	R,E,Z
Message digest	-	-
Keyed hash	R,E,Z	-

4 Self-tests

Each time the Module is powered up, it tests that the cryptographic algorithms still operate correctly and that sensitive data have not been damaged. The Module provides a default entry point to automatically run the power on self-tests compliant with IG 9.10. Power on self–tests are available on demand by reloading the Module.

On power-on or reset, the Module performs the self-tests described Table 10. All KATs must complete successfully prior to any other use of cryptography by the Module. If one of the KATs fails, the Module enters the self-test failure error state. To recover from an error state, reload the Module into memory.

Table 10 – Power-on Self-tests

Test Target	Description
Software Integrity	HMAC-SHA-256
AES	KATs: Encryption, Decryption Modes: CBC Key sizes: 256 bits
HMAC	KAT (includes SHA-256 KAT) SHA sizes: SHA-256

5 Physical Security

The FIPS 140-2 Area 5 Physical Security requirements do not apply because the Module is a software module.

6 Operational Environment

The tested environments place user processes into segregated spaces. A process is logically removed from all other processes by the hardware and Operating System. Since the Module exists inside the process space of the application this environment implicitly satisfies requirement for a single user mode. See Table 1 for specific operating environment details.

7 Mitigation of Other Attacks Policy

The Module is not intended to mitigate against attacks that are outside the scope of FIPS 140-2.

8 Security Rules and Guidance

The Module design corresponds to the Module security rules. This section documents the security rules enforced by the cryptographic module to implement the security requirements of this FIPS 140-2 Level 1 module.

1. The Module provides two distinct operator roles: User and Cryptographic Officer.
2. Power-on self-tests do not require any operator action.

3. Data output is inhibited during self-tests, zeroization, and error states.
4. Status information does not contain CSPs or sensitive data that if misused could lead to a compromise of the Module.
5. There are no restrictions on which keys or CSPs are zeroized by the zeroization service.
6. The calling application is the single operator of the Module.
7. The Module does not support manual key entry.
8. The Module does not have any external input/output devices used for entry/output of data.
9. The module does not support key generation.

9 References and Definitions

The following standards are referred to in this Security Policy.

Table 11 – References

Abbreviation	Full Specification Name
[FIPS140-2]	*Security Requirements for Cryptographic Modules*, May 25, 2001
[SP800-131A]	*Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*, January 2015

Table 12 – Acronyms and Definitions

Acronym	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CO	Cryptographic Officer
CSP	Critical Security Parameter
DES	Data Encryption Standard
DRBG	Deterministic Random Bit Generator
FIPS	Federal Information Processing Standard
HMAC	Keyed-Hash Message Authentication Code
RSA	Rivest, Shamir, and Adleman Algorithm
TDES (DES3)	Triple-DES
SHA	Secure Hash Algorithm
RNG	Random number generator
THREAD_LS	Thread Local Storage
MD(4/5)	Message Digest (4/5)
ASN	Abstract Syntax Notation

The source code files in Table 13 create the object files of the Kanega Watch module on the supported operating environment. console.bin is the compiled binary executable.

Table 13 – Source Files

Source File Name	Description
aes.c	AES algorithm
fips.c	FIPS entry point and API wrappers
fips_test.c	Power on Self Tests
hmac.c	HMAC algorithm
sha256.c	SHA-256 algorithm
wolfcrypt_first.c	First FIPS function and Read Only address
wolfcrypt_last.c	Last FIPS function and Read Only address

10 Appendix A – Installation Instructions

This Appendix describes using Kanega Watch in FIPS 140-2 mode as a software component. The intended audience is Users and Crypto Officers using/needing FIPS software.

10.1 OpenRTOS INSTALLATION

Kanega Watch in FIPS mode for OpenRTOS requires the wolfCrypt FIPS library version 3.9.2. The wolfCrypt FIPS releases can be obtained with a link provided by wolfSSL through direct email.

To verify the fingerprint of the package, calculate the SHA256 sum using a FIPS 140-2 validated cryptographic module. The following command serves as an example:

```
shasum a 256 wolfssl3.9.2commercialfipsfreertos.7z
```

Compare the sum to the sum provided with the package. If for some reason the sums do not match stop using the module and contact wolfSSL.

To unpack the bundle:

```
7za x wolfssl3.9.2commercialfipsfreertos.7z
```

When prompted, enter the password. The password is provided in the distribution email.

The wolfCrypt FIPS for OpenRTOS bundle contains the wolfSSL library and the wolfCrypt FIPS library. To build wolfCrypt with FIPS for OpenRTOS:

1. Build and link the library against the application or pull the source code and header files into the project with these preprocessor definitions:
    - FREERTOS
    - NO_FILE_SYSTEM
    - HAVE_FIPS
    - CYASSL_API
    - HAVE_THREAD_LS
    - USE_CYASSL_MEMORY
    - WOLFCRYPT_ONLY
    - NO_RC4
    - NO_RABBIT
    - NO_PWDBASED
    - NO_RSA
    - NO_MD4
    - NO_DES3
    - NO_SIG_WRAPPER
      NO_MD5
    - NO_ASN
    - NO_SHA

- NO_SHA384
- NO_SHA512
- NO_RNG

2. Get the In Core Integrity check value from the target platform by running the application on the target platform and obtaining the “hash” value that is given in the output. The first run should indicate the In Core Integrity check has failed:

```
in my Fips callback, ok = 0, err = 203 message = In Core
Integrity check     FIPS error

hash =
622B4F8714276FF8845DD49DD3AA27FF68A8226C50D5651D320D914A566
0B3F5

In core integrity hash check failure, copy above hash  into
verifyCore[] in fips_test.c and rebuild
```

   The In Core Integrity checksum will vary with compiler versions, runtime library versions, target hardware, and build type.

3. Copy the value given for "hash" in the output, and replace the value of "verifyCore[]" in ctaocrypt/src/fips_test.c with this new value.
4. Rebuild the library.
5. Relink it into the application.

10.2 Kanega Watch FIPS API

Kanega Watch adds the string _fips to all FIPS mode APIs and removes all wc_ from FIPS mode APIs. For example, wc_ShaUpdate() becomes ShaUpdate(), and ShaUpdate() becomes ShaUpdate_fips(). The FIPS mode functions can be called directly, but they can also be used through macros.

HAVE_FIPS is defined when using Kanega Watch in FIPS mode and that creates a macro for each function with FIPS support. For the above example, a User with an application calling ShaUpdate() can recompile with the FIPS module and automatically get ShaUpdate_fips() support without changing their source code. Of course, recompilation is necessary with the correct macros defined.

A new error return code:

FIPS_NOT_ALLOWED_E
May be returned from any of these functions used directly or even indirectly.

The error is returned when the Power-On Self-Tests (POST) are not yet complete or they have failed. POST is done automatically as a default entry point when using the library, no user interaction is required to start the tests. To see the current status including any error code at any time call wolfCrypt_GetStatus_fips(). For example, if the AES Known Answer Test failed during POS GetStatus may return:

AES_KAT_FIPS_E