Humanware Kernel Cryptographic Module

Version No: 1

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level: 1
Document Version: 1
Date: March 31, 2017

Prepared For:

Technologie Humanware
445, du Parc Industriel
Longueuil, (Québec)
Canada
J4H 3V7
www.Humanware.com

Prepared By:

DeltaCrypt Technologies Inc.
261A ch. des Épinettes
Piedmont (Québec)
Canada
J0R 1K0

www.deltacrypt.com

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This non-proprietary Security Policy for the Humanware Kernel Cryptographic Module describes how the module meets the security requirements of FIPS 140-2 and how to run the module in a secure FIPS 140-2 mode.

This document was prepared as part of the Level 1 FIPS 140-2 validation of the Humanware Kernel Cryptographic Module. The following table lists the module's FIPS 140-2 security level for each section.

Section	Section Title	Level
1	Cryptographic Module Specification	1
2	Cryptographic Module Ports and Interfaces	1
3	Roles, Services, and Authentication	1
4	Finite State Model	1
5	Physical Security	N/A
6	Operational Environment	1
7	Cryptographic Key Management	1
8	EMI/EMC	1
9	Self-Tests	1
10	Design Assurance	1
11	Mitigation of Other Attacks	N/A

Table 1 - FIPS 140-2 Section Security Levels

1.2 Background

Federal Information Processing Standards Publication (FIPS PUB) 140-2 – *Security Requirements for Cryptographic Modules* details the requirements for cryptographic modules. More information on the National Institute of Standards and Technology (NIST) and the Communications Security Establishment Canada (CSE) Cryptographic Module Validation Program (CMVP), the FIPS 140-2 validation process, and a list of validated cryptographic modules can be found on the CMVP website:

http://csrc.nist.gov/groups/STM/cmvp/index.html

More information about the vendor, brand and product line can be found at:
www.humanware.com

1.3 Document Organization

This non-proprietary Security Policy is part of the Humanware Kernel Cryptographic Module FIPS 140-2 submission package. Other documentation in the submission package includes:

- Product documentation
- Vendor evidence documents
- Finite state model
- Additional supporting documents

The Humanware Kernel Cryptographic Module is also referred to in this document as the module or the CM. Please refer to the Acronyms table for additional definitions or description of certain terms used in this document.

2 Module Overview

2.1 Cryptographic Module Specification

The CM is a multi-chip standalone software module. The physical boundary of the module is the BrailleNote Touch tablet on which the module is installed. The logical boundary is the Humanware Kernel Cryptographic Module which is loaded in the Linux kernel space. The CM always operates in the approved mode.

Figure 1 – BrailleNote Touch Tablet

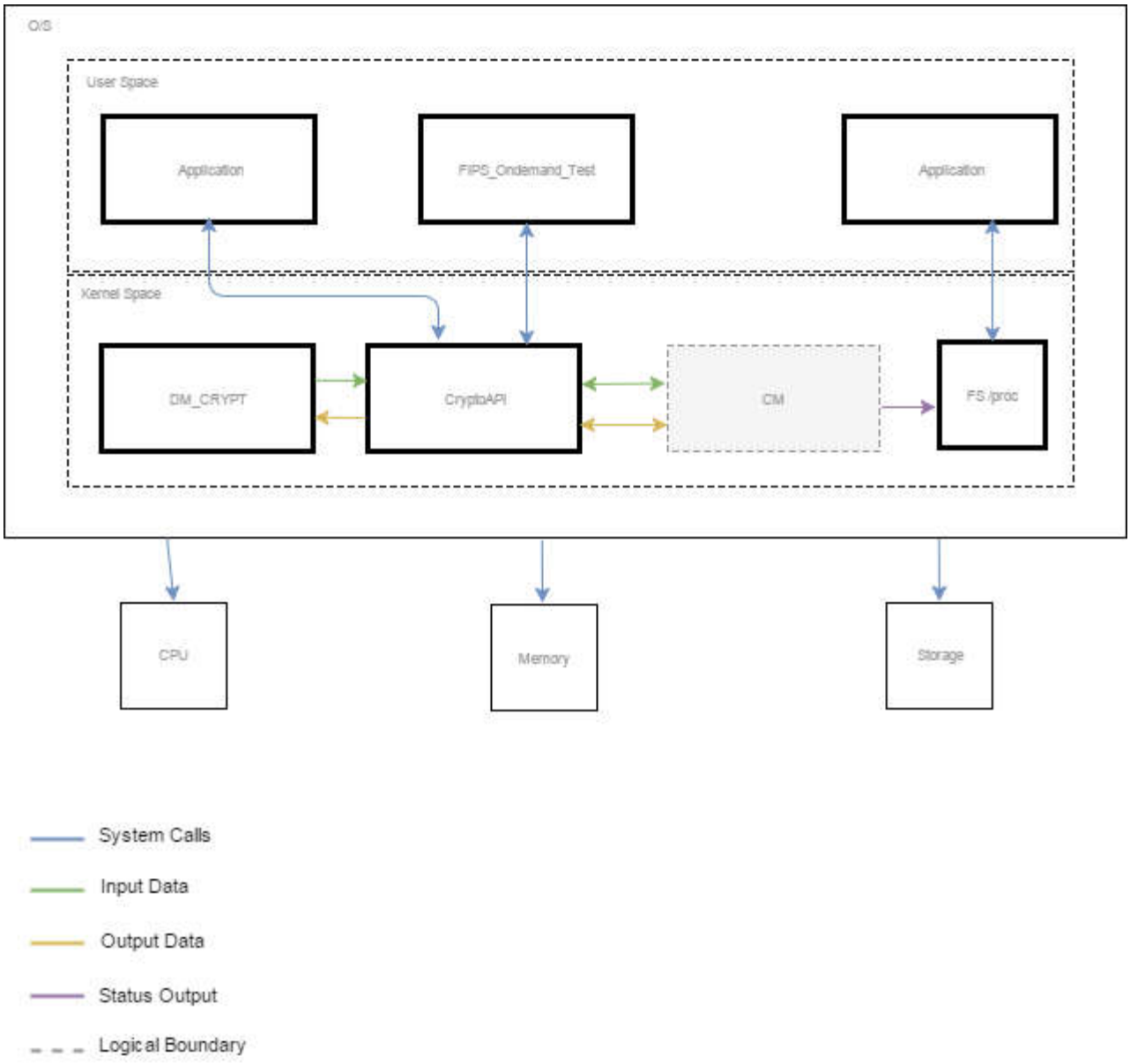

Figure 2 – Logical Boundary

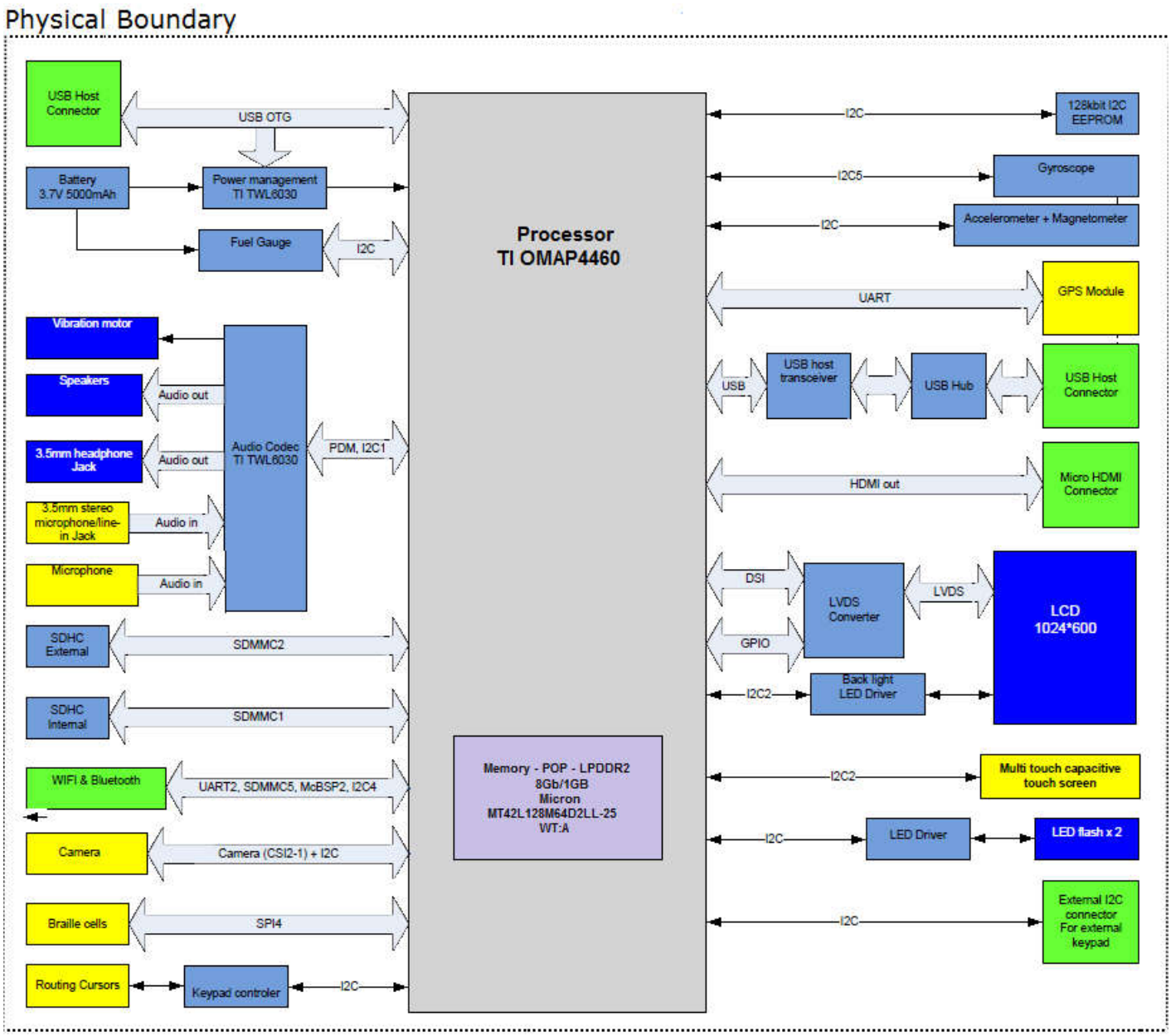

Figure 3 - Physical Block Diagram

2.2 Cryptographic Module Ports and Interfaces

The module's ports and interfaces that are supported when operating in FIPS mode are the same as those of the computer. Please refer to Figure 3 in which interfaces input, output and I/O are represented respectively in yellow, dark blue and green. The tablet has one USB 2.0 physical port

Table 2 shows how the module's physical interfaces map to the logical interfaces defined in FIPS 140-2.

FIPS 140-2 Interface	Logical Interface	Physical Interface
Data Input	Input parameters of API function calls	Braille Cells (18 or 32) Camera GPS Keypad Connector Microphones (2) Routine Cursors (18 or 32) Touch Screen USB Connectors (2) WIFI & Bluetooth
Data Output	Output parameters of API function calls	LCD Screen Headphone Keypad Connector Speakers (2) USB Connectors (2) Vibration WIFI & Bluetooth
Control Input	API function calls	N/A
Status Output	API Return Value and/or Output Kernel Log File(s)	LCD Screen (throught command line)
Power	N/A	Power Management Battery

Table 2 - Module Interface Mappings

2.3 Roles & Services

2.3.1 Roles

The module has two operator roles.

Crypto Officer (CO): Performs initialization of the module.
User: perform cryptographic operations.

2.3.2 Services

Below is the list of all CM services.

Service	Operator	Description	CSP	Access
Initialize	CO	CM initialization and self-test launch	N/A	None
Self-Test	User, CO	Perform self-tests when device boots or restarts	Test on algorithms: AES in ECB mode (128, 192 and 256 bits) AES in CBC mode (128, 192 and 256 bits) SHA-256 and SHA-512 HMAC (with SHA-256 and SHA-512) HMAC-SHA-512	None
Show Status	User	Provide CM status information	Read from pseudo files: /proc/fips140_2/fsm /proc/fips140_2/cms	None
Encryption / Decryption	User	Encrypt or decrypt using the AES algorithm	AES in ECB mode (128, 192 and 256 bits) AES in CBC mode (128, 192 and 256 bits)	Can set the key and use it for the life time of the context
Message digest	User	Hash data using algorithm SHA-256 or SHA-512	SHA-256 and SHA-512	No key use
Keyed hash	User	Generated a MAC (Message authentication code) using the HMAC algorithm with the hashing function SHA-256 or SHA-512	HMAC (with SHA-256 and SHA-512)	Can set the key and use it for the life time of the context
Zeroize	User	CSP bit values set to 0	AES in ECB mode (128, 192 and 256 bits) AES in CBC mode (128, 192 and 256 bits)	None

Service	Operator	Description	CSP	Access
			SHA-256 and SHA-512 HMAC (with SHA-256 and SHA-512)	

Table 3 – Services

2.4 Operational Environment

The CM operates in a modifiable environment as defined by FIPS 140-2. The environment consists of Android v. 4.4 interfacing with the Linux kernel v. 3.2.0 and the physical tablet BrailleNote Touch on which it runs.

The operating system is limited to a single operator operation mode. An application needing cryptographic functions performed by the CM must call via API interface the CryptoAPI kernel module which inputs to the CM. The latter outputs to CryptoAPI which in turn outputs to the calling application.

2.5 Cryptographic Key Management

2.5.1 Algorithm Implementations

The list of FIPS-Approved algorithms implemented by the module can be found in Table 5. The module doesn't implement non-Approved algorithms

Algorithm	Modes and Key Sizes	Validation Number
SHS	SHA-256 and SHA-512	3676
HMAC	HMAC-SHA-256 and HMAC-SHA-512	2962
AES	Modes: ECB, CBC, Key Sizes: 128, 192, 256	4464

Table 4 - FIPS-Approved Algorithm Implementations

2.5.2 Key Management

Key or CSP	Type	Size	Usage	Key Storage	Key Generation	Input	Output	Zeroization
AES	CBC mode	128 bits	Encryption Decryption	In a context in memory (beyond lifetime of API call)	Outside CM	Plaintext through an API call to CM function fips_e_aes_set_key	No API function may retrieve the key from the module	The context is destroyed by filling its memory with 0's using the function crypto_destroy_tfm which is found in the module CryptoAPI
AES	CBC mode	192 bits	Encryption Decryption	In a context in memory (beyond lifetime of API call)	Outside CM	Plaintext through an API call to function fips_e_aes_set_key	No API function may retrieve the key from the module	The context is destroyed by filling its memory with 0's using the function crypto_destroy_tfm which is found in the module CryptoAPI
AES	CBC mode	256 bits	Encryption Decryption	In a context in memory (beyond lifetime of API call)	Outside CM	Plaintext through an API call to function fips_e_aes_set_key	No API function may retrieve the key from the module	The context is destroyed by filling its memory with 0's using the function crypto_destroy_tfm which is found in the module CryptoAPI
AES	ECB mode	128 bits	Encryption Decryption	In a context in memory (beyond lifetime of API call)	Outside CM	Plaintext through an API call to function fips_e_aes_set_key	No API function may retrieve the key from the module	The context is destroyed by filling its memory with 0's using the function crypto_destroy_tfm which is found in the module CryptoAPI
AES	ECB mode	192 bits	Encryption Decryption	In a context in memory (beyond lifetime of API call)	Outside CM	Plaintext through an API call to function fips_e_aes_set_key	No API function may retrieve the key from the module	The context is destroyed by filling its memory with 0's using the function crypto_destroy_tfm which is found in the module CryptoAPI

Key or CSP	Type	Size	Usage	Key Storage	Key Generation	Input	Output	Zeroization
AES	ECB mode	256 bits	Encryption Decryption	In a context in memory (beyond lifetime of API call)	Outside CM	Plaintext through an API call to function fips_e_aes_set_key	No API function may retrieve the key from the module	The context is destroyed by filling its memory with 0's using the function crypto_destroy_tfm which is found in the module CryptoAPI
HMAC	Using SHA-256 bits	256 bits	Message Authentication Code	RAM (beyond lifetime of API call)	Outside CM	Plaintext through an API call to the CM function fips_e_hmac_setkey	No API function may retrieve the key from the module	The context is destroyed by filling its memory with 0's using the function crypto_destroy_tfm which is found in the module CryptoAPI
HMAC	Using SHA-512 bits	512 bits	Message Authentication Code	RAM (beyond lifetime of API call)	Outside CM	Plaintext through an API call to the CM function fips_e_hmac_setkey	No API function may retrieve the key from the module	The context is destroyed by filling its memory with 0's using the function crypto_destroy_tfm which is found in the module CryptoAPI

Table 5 - Cryptographic Keys, Key Components, and CSPs

2.5.3 Key Generation & Input

There is no key generation inside the CM. Keys are input in plaintext through an API call to a CM function.

2.5.4 Key Output

Keys are not output from the module.

2.5.5 Storage

AES and HMAC key memory allocations occur before calling the CM which sets their value. Only the keys address is input to the CM.
Hence, there is no key storage.

2.5.6 Zeroization

Zeroization replaces all CSPs bit value in memory with 0s. The application calling the CM with CSPs initiates zeroization itself right after their use. The calling API functions free the context memory (and thus the CSPs) with function crypto_destroy_tfm from the CryptoAPI module in kernel space. The latter performs zeroization.

2.6 Electromagnetic Interference / Electromagnetic Compatibility

The test device which runs the module conforms to the EMI/EMC requirements specified by 47 Code of Federal Regulations, Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class B (i.e., for home use)

2.7 Self Tests

2.7.1 Power Up Self Tests

The module performs the following tests upon power up:

- CM integrity test using HMAC-SHA 512 bits
- KAT Self-tests for all validated algorithms:
  - AES in ECB mode (128, 192 and 256 bits)
  - SHA-256 and SHA-512
  - HMAC (with SHA-256 and SHA-512)

During the self tests, all data output to the data output interface is inhibited.
If a self test fails, a kernel panic occurs and the CM prevents data input and output. In order to clear the error state, the operator must power cycle the module

2.8 Design Assurance

All source code is stored and maintained in a private server. Releases and revisions will be based on the change log between the changed files and the current files.

Configuration management for the module is provided by Visual Source Safe which uniquely identifies each configuration item and the version of each configuration item.

Documentation version control is performed manually by updating the document date as well as the major and minor version numbers in order to uniquely identify each version of a document.

2.9 Mitigation of Other Attacks

The module does not claim to mitigate any attacks outside the requirements of FIPS 140-2.

3 Secure Operation

The CM is initialized in FIPS mode. No other actions are required on the part of the end user in order to ensure that the CM is operating in a FIPS compliant mode

At module load-time, KAT tests and module integrity tests are performed. If an error is detected, the kernel panic state is enabled and the module enters into an error state. The module integrity test is done with HMAC-SHA-512.

3.1 Initialization

Under no circumstances can the end-user modify the initialization mode of the CM.

3.2 Crypto Officer Guidance

The role is performed by the *init* application of the tablet. It loads the module at initialization and unloads it at the end.
The end user has no control on this role.

3.3 User Guidance

The user is any FIPS mode user, whether a physical user or a user program (Ex. fips_ondemand_test) or another kernel module (Ex. dm-crypt). The encryption keys are not created by the CM but are input to it as plaintext.

4 Acronyms

Acronym	Definition
AES	Advanced Encryption Standard
CA	Certificate Authority
CBC	Cipher Block Chaining
CM	Cryptographic Module
CMVP	Cryptographic Module Validation Program
CO	Crypto Officer
CSE	Communications Security Establishment Canada
CSP	Critical Security Parameter
CVS	Concurrent Versions System
DRBG	Deterministic Random Bit Generator
ECC	Elliptic Curve Cryptography
EFP	Environmental Failure Protection
EMI/EMC	Electromagnetic Interference / Electromagnetic Compatibility
FCC	Federal Communications Commission
FIPS	Federal Information Processing Standards
HMAC	(Keyed-) Hash Message Authentication Code
KAS	Key Agreement Scheme
KAT	Known Answer Test
LED	Light Emitting Diode
NIST	National Institute of Standards and Technology
NRBG	Non-Deterministic Random Bit Generator
NVM	Non-Volatile Memory
QVGA	Quarter Video Graphics Array
ROM	Read Only Memory
RSA	Rivest, Shamir, and Adleman
SHA	Secure Hash Algorithm
SHS	Secure Hash Standard
TDES	Triple Data Encryption Standard
USB	Universal Serial Bus

Table 6 - Acronym Definitions