Geotab Cryptographic Module

FIPS 140-2 Non-Proprietary Security Policy

FIPS Security Level 1

Firmware Version 1.0

Document Version 1.4

Provided By
Geotab Inc.
Canada (Head Office)
2440 Winston Park Drive
Oakville, Ontario, Canada
L6H 7V2

GEOTAB®
management by measurement

Table of Contents

GEOTAB®
management by measurement

List of Figures

List of Tables

GEOTAB®
management by measurement

Acronyms

AES	Advanced Encryption Standard
API	Application Programming Interface
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CMVP	Cryptographic Module Validation Program
CO	Crypto-Officer
CSP	Critical Security Parameter
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
FIPS	Federal Information Processing Standard
KAT	Known Answer Test
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
OS	Operating System
POST	Power On Self Test
RAM	Random Access Memory

GEOTAB®
management by measurement

1. Introduction

This is the non-proprietary security policy for the Geotab Cryptographic Module from the Geotab Inc. This Security Policy describes how the Geotab Cryptographic Module meets the security requirements of Federal Information Processing Standards (FIPS) Publication 140-2, which details the U.S. and Canadian Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the National Institute of Standards and Technology (NIST) and the Communications Security Establishment (CSE) Cryptographic Module Validation Program (CMVP) website at http://csrc.nist.gov/groups/STM/cmvp. This policy was prepared as part of the Level 1 FIPS 140-2 validation of the module.

2. Geotab Cryptography Module

2.1 Overview

2.1.2 Security Level of Security Requirements

Table 1 — Security Level Per FIPS 140-2 Section

Cryptographic Module Specification	1
Cryptographic Module Ports and Interfaces	1
Roles, Services, and Authentication	1
Finite State Model	1
Physical Security	1
Operational Environment	N/A
Cryptographic Key Management	1
EMI/EMC	3
Self-Tests	1
Design Assurance	1
Mitigation of Other Attacks	N/A

2.2 Module Specification

2.2.1 System Description

The system described by this security policy is a single-chip firmware module, running in a limited operational environment. The operating environment does not have an operating system and contains a single CPU with one core (see section 2.2.3). The environment is single threaded and parallel execution is not possible. Program instructions of the module and volatile CSPs are stored within the cryptographic boundary. The module does not provide non-volatile key storage. Keys are provided externally by the application layer and are not stored by the module in non-volatile memory. Raw entropy is provided externally from an accelerometer and is conditioned within the module.

Figure 1 — Microchip dsPIC33E Platform (BG package left and PT package right)

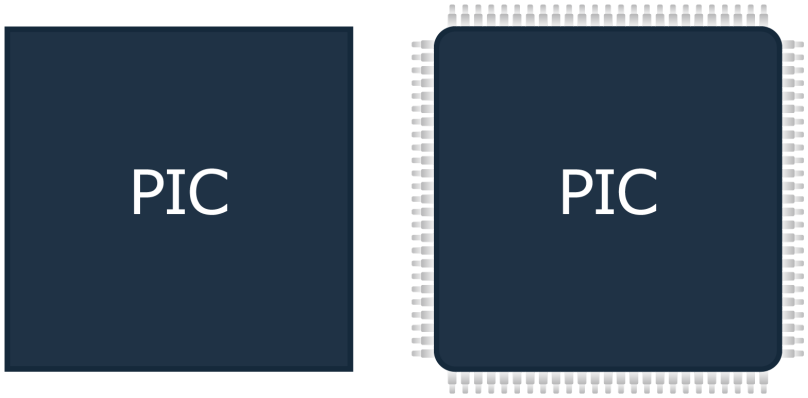
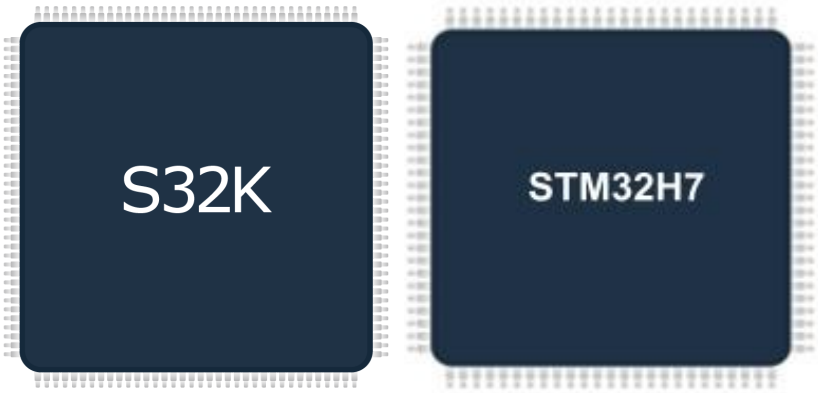


Figure 2 — NXP S32K Platform(left) & STM32H7 Platform(right)



2.2.2 Hardware and Physical Cryptographic Boundary

The cryptographic boundary is defined as marked by the red outline on the diagrams below.

Figure 3 — Physical Block Diagram

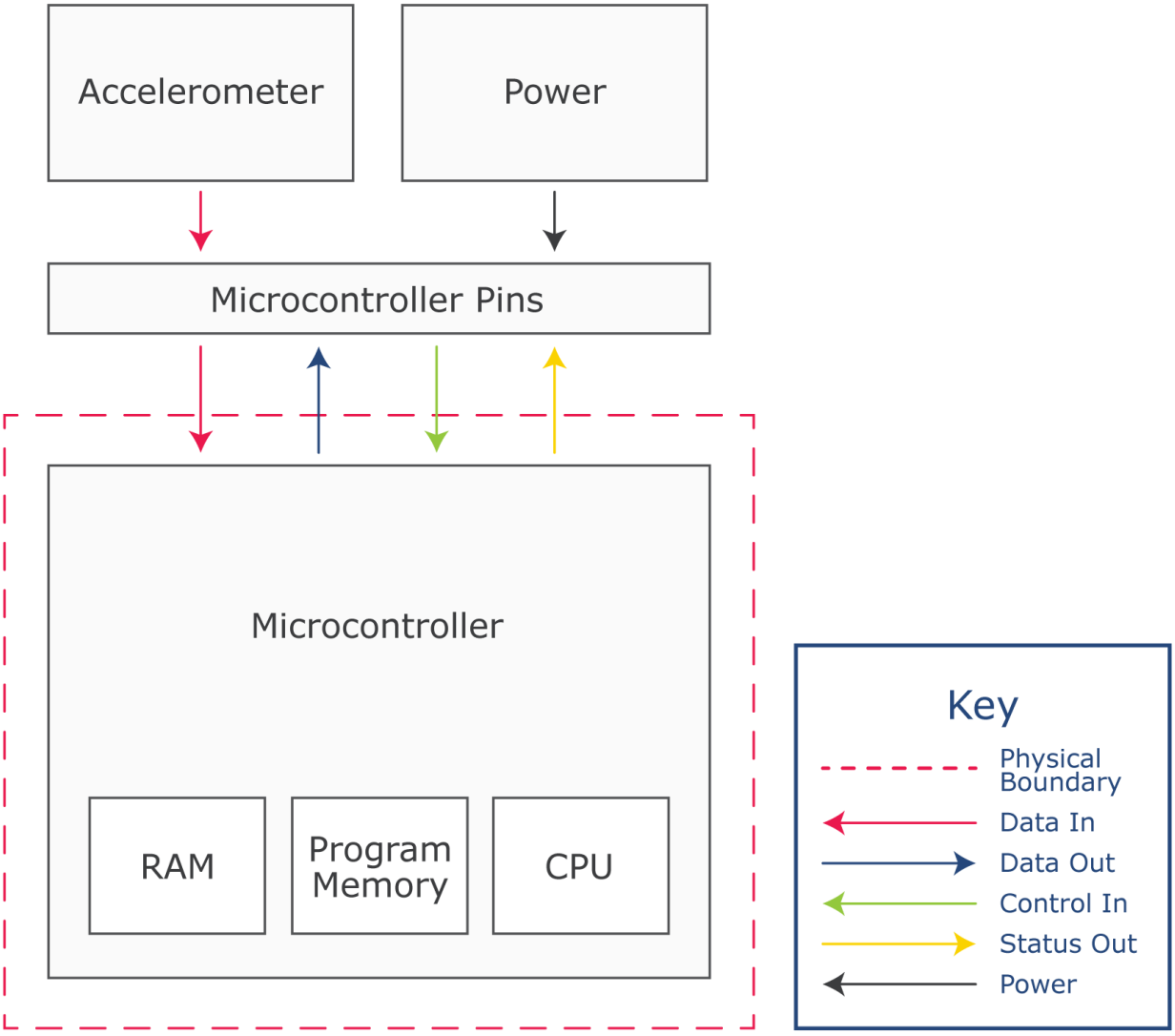
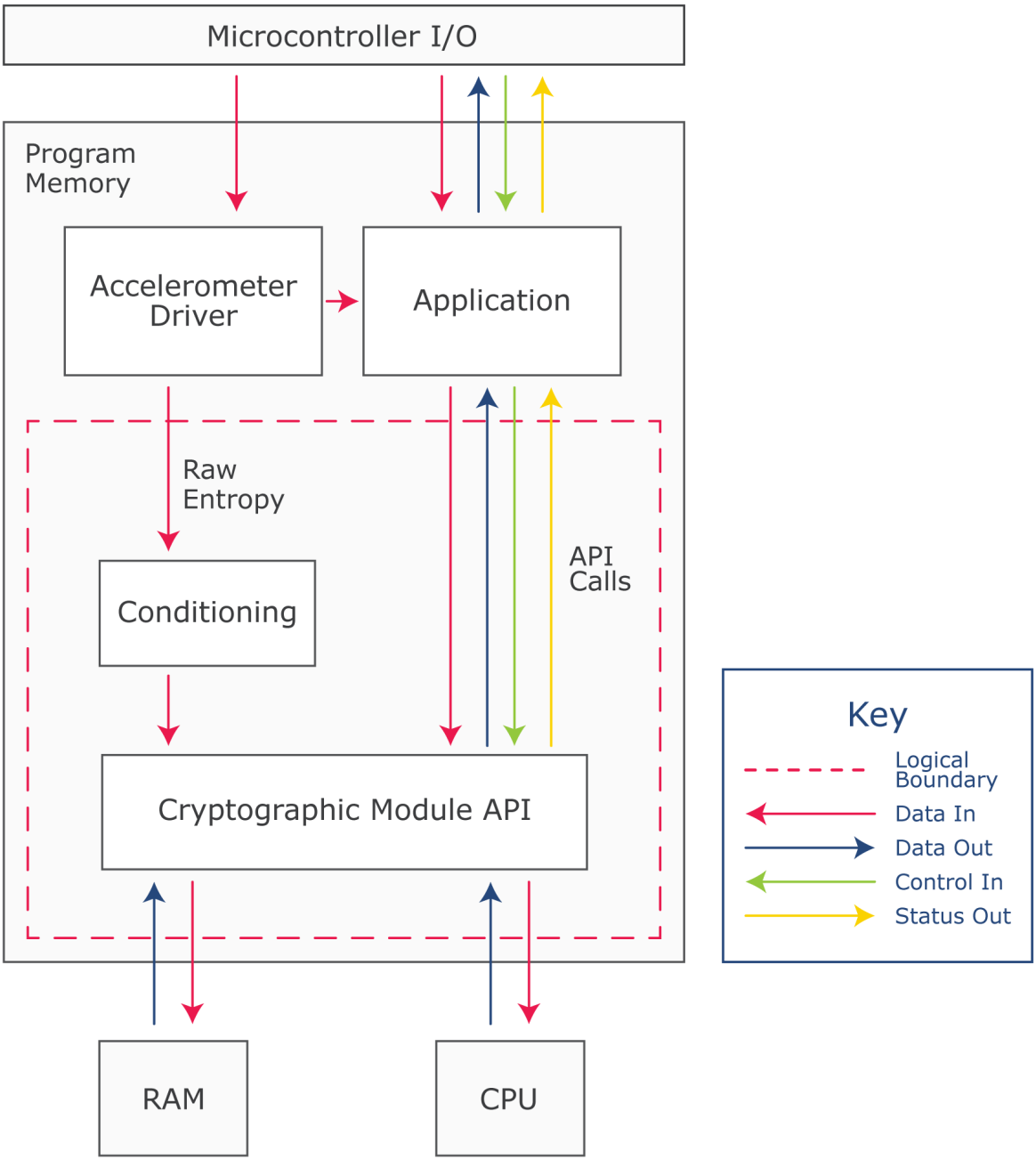


Figure 4 — Logical Block Diagram



2.2.3 Intended Platforms of Operation

The product has been tested and is intended to operate on the following platforms:

- Microchip dsPIC33EP512MU810 (PT) with DSPIC33E series processor
- Microchip dsPIC33EP512MU810 (BG) with DSPIC33E series processor
- NXP S32K148 with S32K14X series processor
- STM32H7 single-core line.
    - STM32H7B3 with STM32 series processor is the specific target tested.

2.2.4 Modes of Operation

The module is only able to operate in the FIPS 140-2 approved mode. The module does not support a bypass capability. The module is initialized on power-on by the firmware application layer of the platform. The API returns MODULE_STATE_ERROR status if the module is in an error state or MODULE_STATE_INIT if it is not yet initialized. In these cases either the input is returned as the output or the output is zeroized depending on the function.

2.3 Module Interfaces

Module's external interfaces are limited to the API function calls. All physical interaction with Control In, Data In, Status Out, Data Out interfaces is done via API calls, through internal microcontroller communication. This data may enter/exit the operational environment (the microcontroller) through any of the pins outlined below in the Microcontroller Pin Mapping columns, configurable by the application layer. Power may enter the operational environment through any combination of pins outlined below.

Table 2 — FIPS 140-2 Logical Interface Mappings

Description	Logical Interface	Physical Interface	Microcontroller Pin Mapping (dsPIC33EP512MU810 PT)	Microcontroller Pin Mapping (dsPIC33EP512MU810 BG)	Microcontroller Pin Mapping (S32K148)	Microcontroller Pin Mapping (STM32H7)
Power	N/A	Power pins of the microcontroller	2, 15, 16, 30, 31, 36, 37, 45, 46, 62, 65, 75, 85, 86	A7, B7, B10, C2, F5, F8, F10, G5, G6, G7, H6, H8, H9, J4, K8, L3	11-16, 31, 32, 50, 51, 66, 67, 90, 91, 123, 124	A1,A15, D14-D15, E1-E2, E6,E8,E10, F1-F2, F5-F11, G1-G2,G6-G10,H6-H11, J6-J10, K4-K10,L5, L8, L10, M1, R1, R15
API entry point	Control in	Internal microcontroller communication	1, 3-14, 17-29, 32-35, 38-44, 47-53, 56-61, 63-64, 66-74, 76-84, 87-100	A1-A6, A9-A11, B2-B6, B8-9, B11, C1, C3-C8,C10-C11, D1-3, D7-D9, D11, E1-E4, E6, E8-E11, F1-F4, F9, F11, G1-G3, G9-G11, H1, H2, H10, H11, J1-J3, J5-J7, J10-J11, K1-K4, K6, K7, K9-K11, L1, L2, L4-L11	1-10, 17-30, 33-49, 52-65, 68-89, 92-122, 125-144	A2-A14, B1-B15, C1-C15, D1-D13, E3-E5, E7, E9, E11-E15, F3-F4, F12-F15, G3-G5, G11-G15, H1-H5, H12-H15, J1- J5,J11-J15, K1-K4, K11-K15, L1-L4, L6-L7, L9, L11-L15, M2-M15, N1-N15, P1-P15, R2- R14
API function parameters	Data in	Internal microcontroller communication	1, 3-12, 14, 17-29, 32-35, 38-44, 47-53, 56-61, 63-64, 66-74, 76-84, 87-100	A1-A6, A9-A11, B2-B6, B8-9, B11, C1, C3-C8,C10-C11, D1-3, D7-D9, D11, E1-E4, E6, E8-E11, F2-F4, F9, F11, G1-G3, G9-G11, H1, H2, H10, H11, J1-J3, J5-J7, J10-J11, K1-K4, K6, K7, K9-K11, L1, L2, L4-L11	1-10, 17-30, 33-49, 52-65, 68-89, 92-122, 125-144	A2-A14, B1-B15, C1-C4, C6-C15, D1-D4, D6-D13, E3-E5, E7, E9, E11-E15, F3-F4, F12-F15, G3-G5, G11-G15, H1-H5, H12-H15, J1- J5,J11-J15, K1-K2, K11-K15, L1-L4, L6-L7, L9, L11-L15, M2-M15, N1-N15, P1-P15, R2- R14

API return value	Status out	Internal microcontroller communication	1, 3-12, 14, 17-29, 32-35, 38-44, 47-53, 56-61, 63-64, 66-74, 76-84, 87-100	A1-A6, A9-A11, B2-B6, B8-9, B11, C1, C3-C8,C10-C11, D1-3, D7-D9, D11, E1-E4, E6, E8-E11, F2-F4, F9, F11, G1-G3, G9-G11, H1, H2, H10, H11, J1-J3, J5-J7, J10-J11, K1-K4, K6, K7, K9-K11, L1, L2, L4-L11	1-10, 17-30, 33-49, 52-65, 68-89, 92-122, 125-144	A2-A14, B1-B15, C1-C4, C6-C15, D1-D4, D6-D13, E3-E5, E7, E9, E11-E15, F3-F4, F12-F15, G3-G5, G11-G15, H1-H5, H12-H15, J1- J5,J11-J15, K1-K2, K11-K15, L1-L4, L6-L7, L9, L11-L15, M2-M15, N1-N15, P1-P15, R2- R14
API function parameters	Data out	Internal microcontroller communication	1, 3-12, 14, 17-29, 32-35, 38-44, 47-53, 56-61, 63-64, 66-74, 76-84, 87-100	A1-A6, A9-A11, B2-B6, B8-9, B11, C1, C3-C8,C10-C11, D1-3, D7-D9, D11, E1-E4, E6, E8-E11, F2-F4, F9, F11, G1-G3, G9-G11, H1, H2, H10, H11, J1-J3, J5-J7, J10-J11, K1-K4, K6, K7, K9-K11, L1, L2, L4-L11	1-10, 17-30, 33-49, 52-65, 68-89, 92-122, 125-144	A2-A14, B1-B15, C1-C4, C6-C15, D1-D4, D6-D13, E3-E5, E7, E9, E11-E15, F3-F4, F12-F15, G3-G5, G11-G15, H1-H5, H12-H15, J1- J5,J11-J15, K1-K2, K11-K15, L1-L4, L6-L7, L9, L11-L15, M2-M15, N1-N15, P1-P15, R2- R14

2.4. Roles & Services

2.4.1 Assumption of Roles

There are two roles in the module (as required by FIPS 140-2) that operators may assume: a Crypto-Officer (CO) role and a User role. The operator implicitly assumes one of these roles when selecting each command documented in this section. Operator authentication is not supported. All services are blocking and do not support multi-threading. As the microcontroller does not have an OS, multithreading is not possible. Due to this nature of the implementation, concurrent operators & operations are also not possible.

2.4.2 Crypto Officer Services

Table 3 — Crypto Officer Services

Service	Description	Input	Output	CSP and Type of Access
gc_Init	Initialize the cryptographic module instance and verify its integrity. Run POSTs. Proceed into operational state if initiation is successful or into error state if it is not. This is typically run on power-on of the module.	Module HMAC	Init Status	HMAC key - WX

2.4.3 User Services

All cryptographic algorithms are approved cryptographic algorithms. The module does not implement any

non-approved algorithms or have vendor affirmation claims for any cryptographic algorithms beyond those CAVP tested.

Table 4 — User Services

Service	Description	Input	Output	Key/CSP Type & Access
gc_aes_CbcEncrypt	Perform AES 256 CBC encryption on provided plaintext	Plaintext	Ciphertext	AES 256 key - X
gc_aes_CbcDecrypt	Perform AES 256 CBC decryption on provided plaintext	Ciphertext	Plaintext	AES 256 key - X
gc_aes_CbcSetKey	Update the AES key in the specified AES context. This also resets both encryption and decryption CBC chains.	Key	None	AES 256 key - W
gc_aes_CbcResetEncryption	Reset the CBC chain for encryption in the specified AES context context	Injection Vector	None	None
gc_aes_CbcResetDecryption	Reset the CBC chain for decryption in the specified AES context	Injection Vector	None	None
gc_sha_Reset	Reset the SHA calculation for the specified SHA context	None	None	N/A
gc_sha_DataAdd	Add the data input to the SHA calculation for the specified context	Data	None	N/A
gc_sha_Calculate	Finalize the SHA calculation and provide the calculated hash	None	SHA 256 hash	N/A
gc_rsa_Init	Initialize RSA context	None	None	None
gc_rsa_IsSignatureValid	Using the provided modulus and exponent (public key), verify if the signed has was signed with the matching private key	RSA N, RSA E, calculated hash, signed hash	Validation status	RSA 2048 signature verification key - WX
gc_hmac_Reset	Reset the HMAC calculation for the specified HMAC context	Key	None	HMAC key - WX
gc_hmac_Update	Add the data input to the HMAC calculation for the specified context	Data	None	HMAC key - X
gc_hmac_Finish	Finalize the HMAC calculation and provide the calculated hash	None	HMAC	HMAC key - X

gc_drbg_Init	Initialize the DRBG context and use the module defined entropy function to retrieve entropy for both entropy input and additional seed input.	None	None	DRBG Seed, DRBG entropy input string - WX
gc_drbg_AlgInit	Initialize the DRBG algorithm context and provide it with its entropy acquisition function. This is only used for CAVP testing of DRBG.	Entropy, additional input	None	DRBG seed, DRBG entropy input string - WX
gc_drbg_GetNextRandom	Use DRBG to derive the required number of bytes	Requested number of bytes	RBG output	DRBG seed, DRBG entropy input string - X
gc_AddRawEntropy	Add raw entropy bytes to be conditioned	Raw entropy bytes	None	N/A
gc_IsConditionedEntropyReady	Indicates whether a sufficient number of raw entropy bytes has been added and processed by the module to begin performing RBG	None	Status	N/A
gc_GetModuleState	Indicates what state the module is in. (Init, Error or Operational)	None	State	N/A
gc_GetModuleVersion	Indicates what the current module version is	None	Version	N/A

(Access types: R - read, W - write, X - execute/use)

* In addition to the listed inputs, each cryptographic function also requires the specified algorithm context variable as an input.

** In addition the the listed outputs, most cryptographic functions also return the operation state (operation successful or if not, the associated error code).

2.5 Physical Security

The module obtains its physical security from a single chip production grade component with standard passivation as allowed by FIPS 140-2 level 1.

2.6 Operational Environment

N/A. As the module is firmware, the operational environment requirements from FIPS 140-2 do not apply.

2.7 Cryptographic Key Management

Only FIPS approved cryptographic algorithms are implemented by this module (no unapproved cryptographic algorithms exist). Keys are provided externally by the application layer and are not stored by the module in non-volatile memory. The Geotab Cryptographic Module will use the key provided in the API calls to perform requested functions. It will return an error if it detects an issue with the provided input.

2.7.1 FIPS Approved Algorithm Implementations

Table 5 — FIPS-Approved Algorithm Implementations

Algorithm	Functions	Cert #
AES CBC 256	Encryption, Decryption	5542, A3251
RSA 2048	Signature Verification	2974, A3251
SHA 256	Hash Calculation	4448, A3251
HMAC (SHA 256 based)	Calculate, Verify	3693, A3251
DRBG (HMAC SHA 256 based)	Random Generation	2197, A3251
CKG (Using DRBG)	Key Generation	Vendor Affirmed

2.7.2 Cryptographic Keys and CSPs

Keys (including the seed entropy) are entered and (where applicable) output from the module via API calls (function calls). DRBG HMAC SHA 256 is used for key generation. It is the only key generation method supported by this module. Asymmetric key generation is not implemented as part of this module (asymmetric keys are provided externally and stored in the program memory or RAM at the application level - see SP 2.2).

Table 6 — List of Cryptographic Keys and CSPs

Key Type	Generation & Input	Storage	Output	Zeroization	CSP Type	Use
AES 256 key	Generated externally, or internally by DRBG, input via API function call	Resides in plaintext, in volatile memory while in use by the module. The key is not stored in non volatile memory by the module.	Never	Power cycle clears volatile memory	256 bit symmetric AES key	Used an an input for AES cipher operations
HMAC key	Generated externally, or internally by DRBG, input via API function call	Resides in plaintext, in volatile memory while in use by the module. The key is not stored in non volatile memory by the module.	Never	Power cycle clears volatile memory	256 bit symmetric HMAC SHA-256 key	Used as an input for the HMAC operation
RSA 2048 signature verification key	Generated externally, input via API function call	Resides in plaintext, in volatile memory while in use by the module. The key is not stored in non volatile memory by the module.	Never	Power cycle clears volatile memory	2048 bit asymmetric public key	Used as an input for the RSA verify operation
DRBG Entropy Input String	Generated externally for CAVP or internally from entropy conditioning output, input via API function pointer	Resides in plaintext, in volatile memory while in use by the module. The key is not stored in non volatile memory by the module.	Never	Power cycle clears volatile memory	128 bit nonce	Used as an input to initialize DRBG
DRBG Seed	Generated externally for CAVP or internally from entropy conditioning output, input via API function call	Resides in plaintext, in volatile memory while in use by the module. The key is not stored in non volatile memory by the module.	Never	Power cycle clears volatile memory	256 bit random value	Used as an input to initialize DRBG
DRBG HMAC Key (secret value of internal state	Static, defined by DRBG algorithm	Resides in plaintext, in volatile memory while in use by the module. The key is not stored in non volatile memory by the	Never	Power cycle clears volatile memory	256 bit	Used by HMAC during DRBG

IG 14.5)		module.				
DRBG HMAC V (secret value of internal state IG 14.5)	Static, defined by DRBG algorithm	Resides in plaintext, in volatile memory while in use by the module. The key is not stored in non volatile memory by the module.	Never	Power cycle clears volatile memory	256 bit	Used by HMAC during DRBG

2.7.3 Random Number Generation

DRBG (SHA 256 HMAC based) is the only algorithm implemented for random number generation. See NIST SP 800-90A for details. The seed entropy (key) is provided internally from conditioned entropy. In order for DRBG to initialize successfully, it must get at least 384 bits of conditioned entropy. The keys generated for the use in this module are 256 bits and are always less than the amount of available entropy. Raw entropy input to the conditioning function is provided from the accelerometer driver which is located outside of the logical boundary but within the physical boundary. Since the source is in the physical boundary and outside of the logical boundary, it falls under IG 7.14 scenario 1(b). No post processing is performed on DRBG output (Geotab affirms compliance with SP 800-133, IG D.12).

2.8 EMI/EMC

The Geotab Cryptographic Module was tested and found to be conformant to the Electromagnetic Interference/Electromagnetic Compatibility (EMI/EMC) requirements specified by Federal Communications Commission 47 Code of Federal Regulations (CFR), Part 15, Subpart B, Unintentional Radiators, Digital Devices, Class B.

2.9 Self-Tests

If any self-test fails, module enters error state. To attempt to clear the error, re-initialize the module. A power cycle is required in order to run power one self-tests on demand.

2.9.1 Power On Self-Tests

Table 7 — List of Power-On Self-Tests

Power-On Test	Description
Firmware Integrity Test	Integrity test performed on the module image. Integrity verified using HMAC SHA-256.
AES-CBC-256 Encryption KAT	Known answer test for AES-CBC-256, confirms that a known plaintext encrypts to a known ciphertext.
AES-CBC-256 Decryption KAT	Known answer test for AES-CBC-256, confirms that a known ciphertext decrypts to a known plaintext.
RSA-2048 Verify KAT	Known answer test for RSA-2048 confirms that a known signed message is successfully verified by a known key.
DRBG KAT	Known answer test for DRBG confirms that a known seed & input string input generates the expected entropy value

As per IG 9.3 HMAC-SHA-256 KAT is not required because HMAC SHA-256 is used for the Firmware Integrity Test. Additionally SHA-256 KAT is performed as part of the HMAC-SHA-256 Firmware Integrity Test.

2.9.2 Conditional Self-Tests

Table 8 — List of Conditional Self-Tests

Conditional Self Test	Description
NDRNG Continuous Test	Continuous test performed whenever a new seed is requested by the DRBG from the conditioned entropy source (NDRNG). Confirms that the conditioned entropy source is not stuck.
DRBG Continuous test	Continuous test performed whenever a new random value is requested from DRBG. Confirms that the DRBG output is not stuck.

2.9.3 Critical Function Tests

Table 9 — List of Critical Function Self-Tests

These tests are always performed as part of DRBG KAT. The reseed function is not available as the module is never re-seeded during a power-on cycle. The DRBG module will go into an error state if more than 10000 bytes are requested during a single power cycle.

Critical Function Test	Description	Operational Frequency
DRBG Test Instantiate	Known answer test to confirm that the instantiation completes as expected. The known answer is compared against the Generate output which is always called after.	On Demand as part of DRBG KAT, each Instantiation, as part of DRBG KAT POST
DRBG Test Generate	Known answer test to confirm that generation completes as expected.	On Demand as part of DRBG KAT, each Instantiation, as part of DRBG KAT POST
DRBG Test Uninstantiate	Tests to ensure that error handling is performed correctly, and the internal state has been zeroized	On Demand as part of DRBG KAT, each Instantiation, as part of DRBG KAT POST

2.10 Mitigation of Other Attacks

N/A. The module does not claim to mitigate any additional attacks in an approved FIPS mode of operation.

3. Secure Operation

3.1 Security Rules and Guidance

3.1.1 Initialization

The module initializes automatically when the device is powered on. The module begins to perform power on self tests upon initialization. The module is not operational until the self tests are completed successfully. The services listed in section 3.1 will return an error state output if the self tests are not yet completed or if the module is in an error state. Upon successful completion of the self tests, the module enters its approved mode of operation. If initialization is unsuccessful and returns an error state, it is up to the application layer to report this error to the management server to be acted upon.

Special note for DRBG usage: gc_drbg_Init must be used in order to initialize DRBG (gc_drbg_AlgInit is available for CAVP testing only).

3.1.2 Management

If the module is operating correctly, it will return the “MODULE_STATE_OPERATIONAL” status. If an error occurs during the module operation, an error code will be provided to the application layer. The application layer will then report these errors to the management server to be acted upon.

3.1.3 Zeroization

All CSPs stored within the cryptographic boundary of the module are stored in volatile memory. The keys are zeroized upon power reset of the system, setting all variable instances to 0.

