Amazon Web Services

AWS Nitro Card Security Engine SSMAE
(Firmware version: SCL Module 1.0; Hardware version: SSMAE CAE 1.0 & SSMAE CAE 2.0)

FIPS 140-3 Non-Proprietary Security Policy

Document Version 1.00.00

Table of Contents

Document Version 1.00.00

Document Version 1.00.00

List of Tables

List of Figures

Document Version 1.00.00

1 General

1.1 Overview

This non-proprietary Security Policy for the AWS Nitro Card Security Engine SSMAE, hereafter also referred to as “the module” in this document, from Amazon Web Services (AWS) provides an overview of the Security Engine and a high-level description of how it meets the security requirements of FIPS 140-3. This document contains details on the module’s cryptographic keys and critical security parameters. This Security Policy concludes with instructions and guidance on running the module in the approved mode of operation.

This document may be freely reproduced and distributed in its entirety without modification.

1.2 Security Levels

Section	Title	Security Level
1	General	1
2	Cryptographic module specification	1
3	Cryptographic module interfaces	1
4	Roles, services, and authentication	1
5	Software/Firmware security	1
6	Operational environment	1
7	Physical security	1
8	Non-invasive security	N/A
9	Sensitive security parameter management	1
10	Self-tests	1
11	Life-cycle assurance	1
12	Mitigation of other attacks	N/A
	Overall Level	1

Table 1: Security Levels

Document Version 1.00.00

2 Cryptographic Module Specification

2.1 Description

Purpose and Use:

The Cryptographic Module (CM) is a single-chip hybrid firmware module. The cryptographic services available in the module's Approved mode of operation are as follows:

- Data encryption / decryption utilizing symmetric ciphers, i.e. AES algorithms.
- Computation of hash values, i.e. SHA2 and SHA3 algorithms.

Module Type: Firmware-hybrid

Module Embodiment: Single Chip

Cryptographic Boundary:

The disjoint hardware component of the cryptographic module i.e., SSMAE CAE consists of the cryptographic engine and SRAM in the AWS Nitro Card Security Engine SSMAE System-on-a-chip (SoC), which provides AES, and SHA algorithm implementations.

The disjoint firmware component of the cryptographic module consists of the C-based HAL (Hardware Abstraction Layer) firmware component SCL Module rev 1.0, executed by the Cortex ARMv8 on Nitro devices AL7 and AL8.

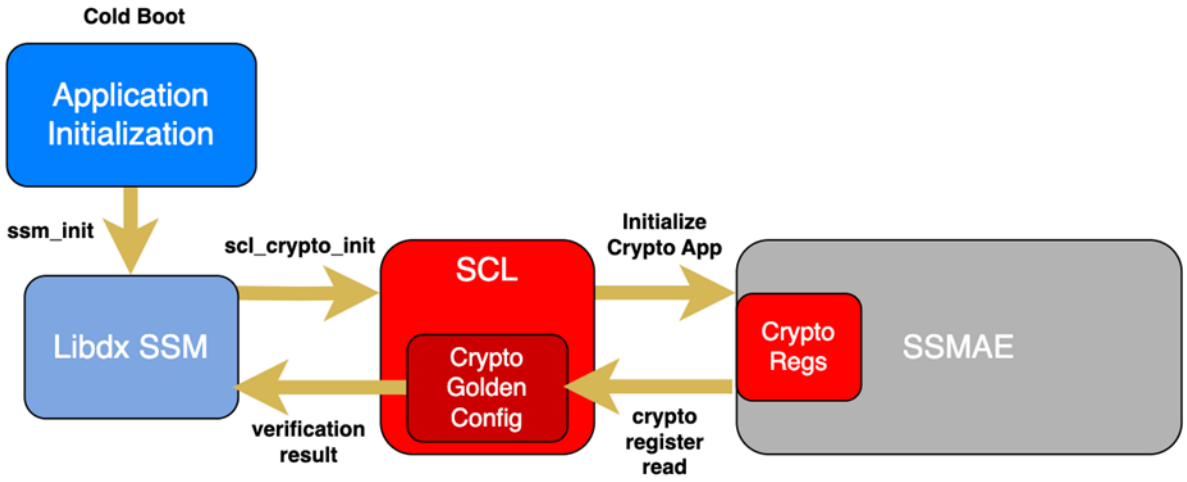

Figure 1: Cryptographic modules depicting the driver modules to access the AL7 and AL8 hardware

The physical perimeter of the module is the AL7 or the AL8 chip. Consequently, the embodiment of the module is a single-chip cryptographic module.

In the following diagram, the bidirectional arrows depict the flow of status, control and data. The cryptographic boundary of the cryptographic module contains only the user space library i.e., SCL Module and the AWS Nitro Card Security Engine SSMAE's Crypto Engine hardware i.e.,

SSMAE CAE. The operations within the security boundary use the ciphers from the AWS Nitro Card Security Engine SSMAE hardware that is included in the cryptographic boundary.

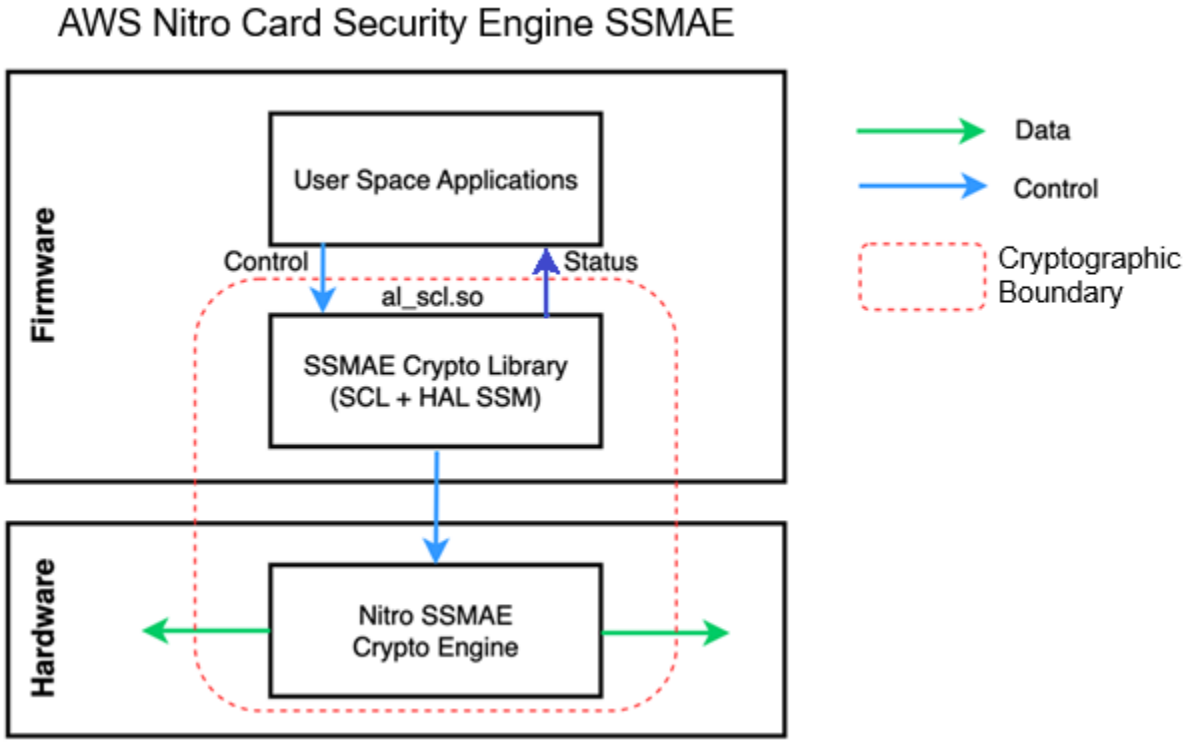

Figure 2: Cryptographic Boundary

Tested Operational Environment's Physical Perimeter (TOEPP):

Figure 3 below depicts a hardware block diagram, where the crypto module (CM) boundary is shown in relation to the hardware on the Nitro card, and Figure 5 and Figure 6 are hi-resolution photos of the AL7 and AL8. In the block diagram, the bidirectional arrows depict the flow of status, control and data. Firmware drivers pass the parameters to the hardware and receive the results from the hardware.

Document Version 1.00.00

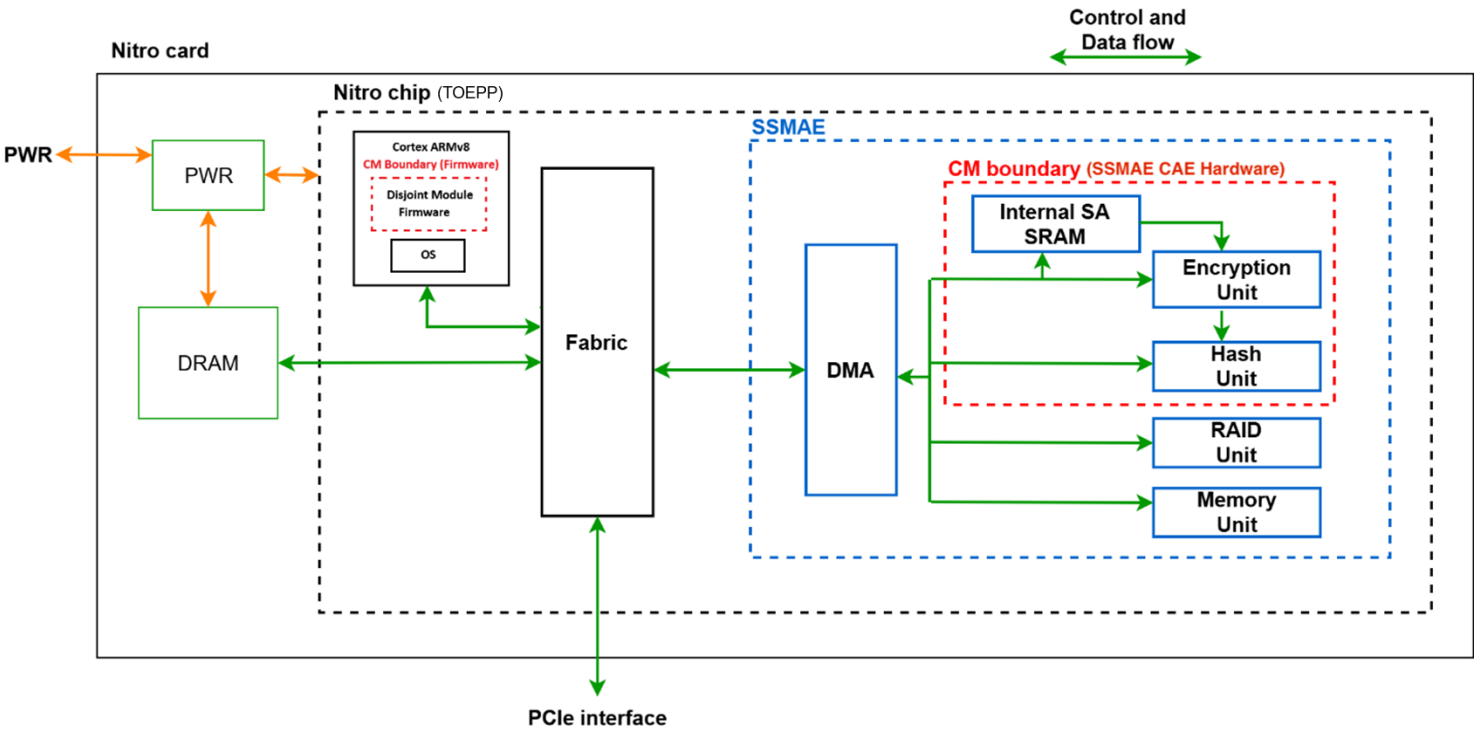

Figure 3: Block Diagram

AL5+:

Figure 4: AL5+ photo

AL7:

Figure 5: AL7 photo

AL8:

Figure 6: AL8 photo

AL10:

Figure 7: AL10 photo

AL11:

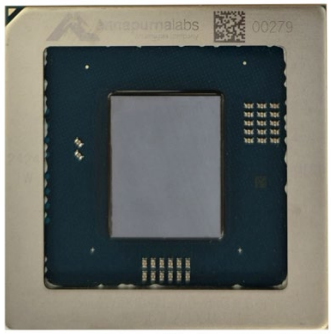

Figure 8: AL11 photo

2.2 Tested and Vendor Affirmed Module Version and Identification

Tested Module Identification – Hardware:

N/A for this module.

Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets):

Package or File Name	Software/ Firmware Version	Features	Integrity Test
SCL Module	1.0		SHA3-256

Table 2: Tested Module Identification – Software, Firmware, Hybrid (Executable Code Sets)

Tested Module Identification – Hybrid Disjoint Hardware:

Model and/or Part Number	Hardware Version	Firmware Version	Processors	Features
SSMAE CAE	1.0			
SSMAE CAE	2.0			

Table 3: Tested Module Identification – Hybrid Disjoint Hardware

Tested Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform	Processors	PAA/PAI	Hypervisor or Host OS	Version(s)
Carbon Linux v5.10	Nitro Device AL7 (Contains Hardware Version 1.0)	ARM Cortex v8	No		1.0
Carbon Linux v5.10	Nitro Device AL8 (Contains Hardware Version 2.0)	ARM Cortex v8	No		1.0

Table 4: Tested Operational Environments - Software, Firmware, Hybrid

Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid:

Operating System	Hardware Platform
Carbon Linux v5.10	Nitro Device AL5+ (Contains Hardware Version 1.0)
Carbon Linux v5.10	Nitro Device AL10 (Contains Hardware Version 2.0)
Carbon Linux v5.10	Nitro Device AL11 (Contains Hardware Version 2.0)
Carbon Linux 6	Nitro Device AL7 (Contains Hardware Version 1.0)
Carbon Linux 6	Nitro Device AL8 (Contains Hardware Version 2.0)
Carbon Linux 6	Nitro Device AL5+ (Contains Hardware Version 1.0)
Carbon Linux 6	Nitro Device AL10 (Contains Hardware Version 2.0)
Carbon Linux 6	Nitro Device AL11 (Contains Hardware Version 2.0)
Carbon Linux 6	Nitro Device AL12 (Contains Hardware Version 2.0)

Table 5: Vendor-Affirmed Operational Environments - Software, Firmware, Hybrid

CMVP makes no statement as to the correct operation of the module when so ported if the specific operational environment is not listed on the validation certificate.

2.3 Excluded Components

There are no excluded components within the boundary of this cryptographic module.

2.4 Modes of Operation

Modes List and Description:

Mode Name	Description	Type	Status Indicator
Approved Mode	The module is placed into the approved mode automatically after performing the initialization and validation steps in Section 11. If any self-test fails, the module enters an error state, returns an error code to the calling application and inhibits all data output	Approved	fips_approved parameter ==1 returned for approved service
Non-Approved Mode	The module is placed in non-approved mode implicitly if any non-approved services are invoked by the calling application after performing the initialization and validation steps in Section 11	Non-Approved	fips_approved parameter ==0 returned for non-approved service

Table 6: Modes List and Description

Mode Change Instructions and Status:

The instructions to run the module are provided in Section 11.1. When the module starts successfully after passing the pre-operational self-test and the cryptographic algorithms self-tests (CASTs), the module is operating in the approved mode of operation by default and can only be transitioned into the non-approved mode by calling one of the non-approved services

listed in the Non-Approved Services table. The module will transition back to approved mode when an approved service is called. Table 6 ‘Modes List and Description’ provides details on the service indicator implemented by the module. The service indicator identifies when an approved service is called.

2.5 Algorithms

Approved Algorithms:

Algorithm	CAVP Cert	Properties	Reference
AES-CBC	A6241	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-CTR	A6241	Direction - Decrypt, Encrypt Key Length - 128, 192, 256 Payload Length - Payload Length: 8-128 Increment 8 Supports Counter larger than maximum value - Yes Incremental Counter - Yes Conformances - RFC3686 Counter Tests Performed - Yes IV Generation Mode - External	SP 800-38A
AES-ECB	A6241	Direction - Decrypt, Encrypt Key Length - 128, 192, 256	SP 800-38A
AES-XTS Testing Revision 2.0	A6241	Direction - Decrypt, Encrypt Key Length - 128, 256 Payload Length - Payload Length: 128-65536 Increment 128 Tweak Mode - Number Data Unit Length Matches Payload Length - Yes	SP 800-38E
SHA2-256	A6241	Message Length - Message Length: 8-65536 Increment 8	FIPS 180-4
SHA2-384	A6241	Message Length - Message Length: 8-65536 Increment 8	FIPS 180-4
SHA2-512	A6241	Message Length - Message Length: 8-65536 Increment 8	FIPS 180-4
SHA3-224	A6241	Message Length - Message Length: 8-65536 Increment 8	FIPS 202
SHA3-256	A6241	Message Length - Message Length: 8-65536 Increment 8	FIPS 202
SHA3-384	A6241	Message Length - Message Length: 8-52416 Increment 8	FIPS 202
SHA3-512	A6241	Message Length - Message Length: 8-36288 Increment 8	FIPS 202

Table 7: Approved Algorithms

Document Version 1.00.00

Vendor-Affirmed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms:

N/A for this module.

Non-Approved, Allowed Algorithms with No Security Claimed:

N/A for this module.

Non-Approved, Not Allowed Algorithms:

Name	Use and Function
DES	Implemented in the module's disjoint hardware component. Used for encryption/decryption purpose
MD5	Implemented in the module's disjoint hardware component. Used for hashing purpose
SHA-1	Output hash: 160 bits. Implemented in the module's disjoint hardware component. Used for hashing purpose
Triple-DES (ECB)	Keylen: 192 bits. Implemented in the module's disjoint hardware component. Used for encryption/decryption only
Triple-DES (CBC)	Keylen: 192 bits. Implemented in the module's disjoint hardware component. Used for encryption/decryption only

Table 8: Non-Approved, Not Allowed Algorithms

2.6 Security Function Implementations

Name	Type	Description	Properties	Algorithms
Hashing	SHA	The firmware SCL Module exposes API functions al_scl_crypto_sa_update() to enqueue the control descriptor (hash/crypto algorithm). It then uses al_scl_crypto_desc_submit() to submit the data descriptors to the firmware. The firmware uses the hardware SSMAE cryptographic engine to do the hash or hash verification operation to the payload in the data descriptors		SHA2-256: (A6241) SHA2-384: (A6241) SHA2-512: (A6241) SHA3-224: (A6241) SHA3-256: (A6241) SHA3-384: (A6241) SHA3-512: (A6241)
Encrypt	BC-UnAuthEncrypt	The firmware SCL Module exposes API functions al_scl_crypto_sa_update()		AES-CBC: (A6241) AES-ECB:

Name	Type	Description	Properties	Algorithms
		to build and enqueue the data descriptors. It then uses al_scl_crypto_desc_submit() to assign data descriptors to the firmware. The firmware uses the hardware SSMAE cryptographic engine to encrypt the payload in the data descriptors		(A6241) AES-CTR: (A6241) AES-XTS Testing Revision 2.0: (A6241)
Decrypt	BC-UnAuthDecrypt	The firmware SCL Module exposes API functions al_scl_crypto_sa_update() to build and enqueue the data descriptors. It then uses al_scl_crypto_desc_submit() to assign data descriptors to the firmware. The firmware uses the hardware SSMAE cryptographic engine to encrypt the payload in the data descriptors		AES-CBC: (A6241) AES-ECB: (A6241) AES-CTR: (A6241) AES-XTS Testing Revision 2.0: (A6241)

Table 9: Security Function Implementations

2.7 Algorithm Specific Information

2.7.1 AES-XTS

The module checks explicitly that Key_1 ≠ Key_2 before using them, in accordance with IG C.I. AES-XTS keys (i.e., Key_1 and Key_2) entered into the module shall be generated and/or established independently according to NIST SP 800-133rev2, Section 6.3. for an approved use of AES-XTS.

The length of the data unit for any instance of implementation of XTS-AES shall not exceed 2^{20} AES blocks.

2.8 RBG and Entropy

N/A for this module.

2.9 Key Generation

This module does not perform any Key Generation.

Document Version 1.00.00

2.10 Key Establishment

This module does not perform any Key Establishment.

2.11 Industry Protocols

The module does not support any protocols.

Document Version 1.00.00

3 Cryptographic Module Interfaces

3.1 Ports and Interfaces

Physical Port	Logical Interface(s)	Data That Passes
N/A	Data Input	Data is fetched over AXI data port after doorbell is provided using al_scl_crypto_desc_submit.
N/A	Data Output	Data is output over this port, and once done the DMA completion ring is updated and al_scl_crypto_cdesc_fetch will notify data output is ready. The data output interface is not used while performing pre-operational self-tests, zeroization or when the module is in error state.
N/A	Control Input	Hardware configuration is done over this port, via al_scl_crypto_init and al_scl_crypto_q_init functions. In addition, Crypto operation context descriptor is enqueued to the DMA queue when calling al_scl_crypto_sa_update. And the module's keys and IVs are zeroed out when calling al_scl_crypto_sa_zeroization
N/A	Status Output	SCL check the status of hardware via a register read, al_scl_fips_state_get, over this port. This will indicate if hardware is properly initialized and if all KAT self-tests have passed. It will also indicate if hardware is in error state. The firmware state is confirmed with al_scl_get_state. The module version can be checked via al_scl_version
N/A	Power	Accept and provide power to the module. No data passes over this interface

Table 10: Ports and Interfaces

All cryptographic functions are inhibited while the module is in an error state. The data input and output via the data input and output interfaces respectively are inhibited while in an error state. The module does not have control output. If a self-test fails, the module enters an error state, where no cryptographic operations are possible.

4 Roles, Services, and Authentication

4.1 Authentication Methods

The module does not support user authentication and there is only one supported role.

4.2 Roles

Name	Type	Operator Type	Authentication Methods
Crypto-Officer	Role	CO	None

Table 11: Roles

The Crypto Officer role has access to all the module's services. The role is not explicitly authenticated but assumed implicitly on access to any of the module services. An operator is implicitly in the Crypto Officer role based upon the service chosen.

The customer who configures and installs the CM is a Crypto Officer. Additionally, a user powering up and initializing the device is assuming the Crypto Officer role.

4.3 Approved Services

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Set the Crypto Descriptor	SCL offers a secure API, al_scl_crypto_sa_update() to build and enqueue the crypto descriptors that will be used for encrypt, hash, and decrypt operations	fips_approved ==1 for approved	scl device id, udma queue id, ssm udma id, flags, metadata, SA context virtual address, SA context physical address, SA context length	fips_approved boolean value 0 or 1	Hashing Encrypt Decrypt	Crypto-Officer - AES Key: R,E

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
Submit Data Descriptor	SCL offers a secure API, al_scl_crypto_desc_submit, to submit the crypto transactions and process data according to the SA information	fips_approved ==1 for approved	scl device id, udma queue id, ssm udma id, number of transmitted descriptor, number of received descriptor	success of the processing or error if submit fails or device id is invalid	Hashing Encrypt Decrypt	Crypto-Officer - AES Key: R,E
Fetch next completion descriptor	SCL offers a secure API, al_scl_crypto_cdesc_fetch provides the outcome of the data processing - number of data buffers that are consumed by the cryptographic operation, success / error indication, and metadata on the operation	fips_approved ==1 for approved	scl device id, udma queue id, ssm udma id, pointer to the first descriptor	number of descriptors that belong to the packet	Hashing Encrypt Decrypt	Crypto-Officer - AES Key: R,E
Self-Tests	SCL offers a secure API, al_scl_crypto_self_test, to invoke the self-tests on-demand. These tests are also performed during al_scl_crypto_q_init. Reloading the	Module continues to run	scl device id, udma queue id, ssm crypto queue parameters	0 if no error, Err -14 otherwise	None	Crypto-Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	module will also run these self-tests					
Version (Show Version)	SCL offers a secure APIs for giving status information: al_scl_version. It retrieves type of card and the version of the SCL being used	N/A	None	For AL7: "Nitro Device AL7, SCL Module rev v1.0, Crypto HW SSMAE CAE-1.0". For AL8: "Nitro Device AL8, SCL Module rev v1.0, Crypto HW SSMAE CAE-2.0"	None	Crypto-Officer
Firmware Status (Show Status)	SCL offers a secure APIs for giving status information: al_scl_get_state, provides information regarding the current firmware state of the module	Return value - one of State value of SCL_STATE_INACTIVE, SCL_STATE_INIT, SCL_STATE_SELFTEST, SCL_STATE_READY, SCL_STATE_ERROR	scl device id	State value of SCL_STATE_INACTIVE, SCL_STATE_INIT, SCL_STATE_SELFTEST, SCL_STATE_READY, SCL_STATE_ERROR	None	Crypto-Officer
Hardware Status	SCL offers a secure APIs for giving status information: al_scl_fips_state_get, provides information regarding the current hardware state of the module	0 if fips init state not set. 1 if fips init state set	scl device id	State value of 0 or 1	None	Crypto-Officer
Module Initialization	SCL offers a secure API, al_scl_crypto_init, to initialize the SSMAE cryptographic engine and verify	N/A	crypto app address, ssm device revision id,	return scl_dev_id on successful init and verified config, error otherwise	None	Crypto-Officer

Name	Description	Indicator	Inputs	Outputs	Security Functions	SSP Access
	its configuration correctness against expected standards		whether it needs to be initialized			
Zeroization	SCL offers a secure API, scl_crypto_sa_zeroization, to zerioze CSPs.	N/A	N/A	a stream of 0s	None	Crypto-Officer

Table 12: Approved Services

When the module returns this information, for either tested configuration, this is an indication that the operator is running the FIPS 140-3 validated module entitled “AWS Nitro Card Security Engine SSMAE”.

4.4 Non-Approved Services

Name	Description	Algorithms	Role
Set the Crypto Descriptor	SCL offers a secure API, al_scl_crypto_sa_update() to build and enqueue the data descriptors that will be used for encrypt, hash, and decrypt operations. If the API is called with a non-approved algorithm, it is a non-approved service. Approved mode Indicator == 0 for non-approved	DES MD5 SHA-1 Triple-DES (ECB) Triple-DES (CBC)	Crypto-Officer

Table 13: Non-Approved Services

4.5 External Software/Firmware Loaded

The module does not have the capabilities to load External Software/Firmware.

5 Software/Firmware Security

5.1 Integrity Techniques

The module's firmware integrity test is automatically performed during loading. If any of these tests fail, the module will terminate the loading process. The module cannot be used in this state. To recover from the error state, re-initialization is possible by doing a reboot to set it to power on state.

Upon launching, the applications initialize the library, and the library performs the power-on self-test and integrity test using SHA3-256. The user applications shall be aborted and terminated if any power-on self-test (known-answer test or integrity test) fails and the module will stay in DISABLED state. All module APIs first check for the internal state, therefore if the state is DISABLED – no operation will be performed, and the state will be considered an error state.

5.2 Initiate on Demand

The module requires reloading if the operator wishes to test the firmware integrity on demand.

Document Version 1.00.00

6 Operational Environment

6.1 Operational Environment Type and Requirements

Type of Operational Environment: Non-Modifiable

How Requirements are Satisfied:

The module's Operational Environment is non-modifiable and is comprised of Carbon Linux v5.10 running on the AL7 or the AL8 SoC.

The module also runs on Vendor Affirmed Operation Environment of Carbon Linux running on AL5+, AL10, and AL11 SoC.

Each instance of a cryptographic module has control over its own CSPs. The operational environments provide the capability to separate individual application processes from each other in order to prevent uncontrolled access to CSPs. Processes that are spawned by the cryptographic module are owned by the module and are not owned by any external processes/operators.

Document Version 1.00.00

7 Physical Security

The Cryptographic Module is a firmware-hybrid module that operates on a single-chip platform which conforms to the Level 1 requirements for physical security. AL7 and AL8 are production grade components with standard passivation (a sealing coat applied over the chip circuitry to protect it against environmental and other physical damage) and a production grade enclosure that surrounds the cryptographic module.

There is no maintenance role or interface that provides physical access to the contents of the module.

8 Non-Invasive Security

The module does not have any non-invasive attack mitigation technique.

Document Version 1.00.00

9 Sensitive Security Parameters Management

9.1 Storage Areas

Storage Area Name	Description	Persistence Type
SRAM	Stored internally in static form while power is applied	Dynamic

Table 14: Storage Areas

9.2 SSP Input-Output Methods

Name	From	To	Format Type	Distribution Type	Entry Type	SFI or Algorithm
AES Key In (Encrypt)	External memory	SRAM	Plaintext	Manual	Electronic	Encrypt
AES Key In (Decrypt)	External memory	SRAM	Encrypted	Manual	Electronic	Decrypt

Table 15: SSP Input-Output Methods

9.3 SSP Zeroization Methods

Zeroization Method	Description	Rationale	Operator Initiation
Call Zeroization service (scl_crypto_sa_zeroization)	Services can be called by the application running in the user space. User level HAL SSM (crypto) driver performs the zeroization operation by accessing hardware. No kernel mode is involved in this process	Overwrite with all zeroes	Operator Initiation Allowed

Table 16: SSP Zeroization Methods

9.4 SSPs

Name	Description	Size - Strength	Type - Category	Generated By	Established By	Used By
AES Key	AES Key used for AES (CBC, CTR, ECB and XTS) encryption and decryption operations by the hardware	128, 192, 256 bits - > 112 bits	Symmetric Key - CSP			Encrypt Decrypt

Table 17: SSP Table 1

Name	Input - Output	Storage	Storage Duration	Zeroization	Related SSPs
AES Key	AES Key In (Encrypt) AES Key In (Decrypt)	SRAM:Plaintext	Stored until replaced by new key, or the module is reset, or there is a power cycle.	Call Zeroization service (scl_crypto_sa_zeroization)	

Table 18: SSP Table 2

10 Self-Tests

10.1 Pre-Operational Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details
SHA3-256 (CAST) (A6241)	256 bits hash output	KAT	SW/FW Integrity	Approved mode Indicator ==1 for success	CAST for the hashing algorithm used for the firmware integrity test
SHA3-256 (Integrity) (A6241)	256 bits	Firmware Integrity Test	SW/FW Integrity	Module is active. In addition, service al_scl_get_state() can be used to find an unambiguous state, either functional or error state	Firmware Integrity Test using SHA3-256

Table 19: Pre-Operational Self-Tests

10.2 Conditional Self-Tests

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-CBC (Encrypt) (A6241)	128 bits	KAT	CAST	Approved mode Indicator ==1 for success	Encrypt	During module initialization after calling al_scl_crypto_q_init()
AES-CBC (Decrypt) (A6241)	128 bits	KAT	CAST	Approved mode Indicator ==1 for success	Decrypt	During module initialization after calling al_scl_crypto_q_init()
AES-XTS Testing Revision 2.0 (Encrypt) (A6241)	128 bits	KAT	CAST	Approved mode Indicator ==1 for success	Encrypt	During module initialization after calling al_scl_crypto_q_init()
AES-XTS Testing Revision 2.0 (Decrypt) (A6241)	128 bits	KAT	CAST	Approved mode Indicator ==1 for success	Decrypt	During module initialization after calling al_scl_crypto_q_init()

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
AES-ECB (Encrypt) (A6241)	128 bits	KAT	CAST	Approved mode Indicator ==1 for success	Encrypt	During module initialization after calling al_scl_crypto_q_init()
AES-ECB (Decrypt) (A6241)	128 bits	KAT	CAST	Approved mode Indicator ==1 for success	Decrypt	During module initialization after calling al_scl_crypto_q_init()
AES-CTR (Encrypt) (A6241)	128 bits	KAT	CAST	Approved mode Indicator ==1 for success	Encrypt	During module initialization after calling al_scl_crypto_q_init()
AES-CTR (Decrypt) (A6241)	128 bits	KAT	CAST	Approved mode Indicator ==1 for success	Decrypt	During module initialization after calling al_scl_crypto_q_init()
SHA2-512 (A6241)	512 bits hash output	KAT	CAST	Approved mode Indicator ==1 for success	Hash	During module initialization after calling al_scl_crypto_q_init()
SHA2-256 (A6241)	256 bits hash output	KAT	CAST	Approved mode Indicator ==1 for success	Hash	During module initialization after calling al_scl_crypto_q_init()
SHA3-512 (A6241)	512 bits hash output	KAT	CAST	Approved mode Indicator ==1 for success	Hash	During module initialization after calling al_scl_crypto_q_init()
AES-XTS Testing Revision 2.0 (Key_1 Key_2 Encrypt) (A6241)	Key Comparison	KAT	CAST	Approved mode Indicator ==1 for success	Encrypt	Runs every time al_scl_crypto_sa_update() is called
AES-XTS Testing Revision 2.0	Key Comparison	KAT	CAST	Approved mode Indicator	Decrypt	Runs every time al_scl_crypto_sa_update() is called

Algorithm or Test	Test Properties	Test Method	Test Type	Indicator	Details	Conditions
(Key_1 Key_2 Decrypt) (A6241)				==1 for success		

Table 20: Conditional Self-Tests

10.3 Periodic Self-Test Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
SHA3-256 (CAST) (A6241)	KAT	SW/FW Integrity	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
SHA3-256 (Integrity) (A6241)	Firmware Integrity Test	SW/FW Integrity	The period is determined by the Crypto Officer	Device Reboot

Table 21: Pre-Operational Periodic Information

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-CBC (Encrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
AES-CBC (Decrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
AES-XTS Testing Revision 2.0 (Encrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
AES-XTS Testing Revision 2.0 (Decrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
AES-ECB (Encrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test

Algorithm or Test	Test Method	Test Type	Period	Periodic Method
AES-ECB (Decrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
AES-CTR (Encrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
AES-CTR (Decrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
SHA2-512 (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
SHA2-256 (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
SHA3-512 (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
AES-XTS Testing Revision 2.0 (Key_1 Key_2 Encrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test
AES-XTS Testing Revision 2.0 (Key_1 Key_2 Decrypt) (A6241)	KAT	CAST	The period is determined by the Crypto Officer	Device Reboot or by calling the self-test API i.e., al_scl_crypto_self_test

Table 22: Conditional Periodic Information

10.4 Error States

Name	Description	Conditions	Recovery Method	Indicator
SCL_STATE_ERROR	Module is aborted with module state error signal;	Pre-operational test failure,	Reinitialization or resetting of the module. Recovery from	Error message is output on the scl_module_state

Name	Description	Conditions	Recovery Method	Indicator
	module is no longer operational. The data output interface is inhibited	conditional test failure	error state is possible by either reloading the library or by calling the reset API	service return and then module fails

Table 23: Error States

10.5 Operator Initiation of Self-Tests

The CASTs for AES and SHS can be invoked by the Crypto Officer on demand by calling the Self-Tests service.

11 Life-Cycle Assurance

11.1 Installation, Initialization, and Startup Procedures

- Carbon Linux v5.10, the applications, al_scl.so, and al_scl_sha256 must be installed - either by flashing the images to SPI flash or downloading them over the network via CoAP.
- Launching the applications will link them to SCL Module, which then copies itself to application memory space.

11.2 Administrator Guidance

In order to install the validated module, the subsequent steps must be followed:

- The Nitro device (AL7 and AL8) must be physically mounted on the Printed Circuit Board (PCB).
- Application shall initialize the module with al_scl_crypto_init, then calls al_scl_crypto_q_init to set up all crypto queues. After the SHA-3 KAT test passes, an integrity check will automatically run to hash al_scl.so with SHA-3 and compares the result to al_scl_sha256.
- Upon crypto operation, the application shall build an SA buffer containing all CSPs. The module validates and enqueues this SA by calling al_scl_crypto_sa_update, which places an SA descriptor into the SSMAE hardware queue.
- Next, data descriptors should be generated by the application and written to the DMA ring. Doorbell to start processing these data descriptors shall be issues using al_scl_crypto_desc_submit.
- Finally, the completion ring should be monitored, and processed data can be used when al_scl_crypto_cdesc_fetch function returns a positive number of completions for Crypto transactions.

The crypto officer can determine the status of the Approved mode by observing that the application initializes successfully. If any of the power on self-tests were to fail, the module will not be operational.

The calling application shall not request any non-Approved services while in the Approved mode. Doing so will result in a transition to a non-Approved state. If the calling application requests an Approved service as specified in Table 12, then the module is considered to be operating in the Approved mode. If the calling application requests a non-Approved service as specified in Table 13, then the module is considered to be operating in the non-Approved mode. SSPs generated in one mode of operation shall not be shared with the other mode of operation.

11.3 Non-Administrator Guidance

The module does not have any non-administrator role.

Document Version 1.00.00

12 Mitigation of Other Attacks

The module does not have any attack mitigation technique.

Document Version 1.00.00