Cryptosec Dekaton Non-Proprietary Security Policy

Realia Technologies, S.L.

Revision: 12.11.3642
Date: 14/11/2018

Revision History

Revision	Date	Change Description
12.11.3481	09/04/2018	First Release
12.11.3642	14/11/2018	Coordination changes

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Purpose

Federal Information Processing Standards Publication 140-2 - Security Requirements for Cryptographic Modules (NIST FIPS PUB 140-2) details the U.S. Government requirements for cryptographic modules. More information about the NIST FIPS PUB 140-2 Standard and validation program is available on the NIST website.

This is a Cryptosec Dekaton Non-Proprietary Security Policy for the Cryptosec Dekaton Hardware Security Module (HSM) from Realia Technologies, S.L. (Realsec). This Non-Proprietary Security Policy describes how the Cryptosec Dekaton meets the security requirements of NIST FIPS PUB 140-2 , and how to run the Cryptosec Dekaton in secure NIST FIPS PUB 140-2 mode of operation. This Non-Proprietary Security Policy was prepared as part of the Level 3 NIST FIPS PUB 140-2 validation of the module.

1.2 Approved Product Identification

The following table shows the product name, hardware version, and firmware version information:

Table 1.1: Approved Product Identification

Part Name	Hardware Version	Firmware Version
Cryptosec Dekaton	1.1	12.11.3642

Chapter 2

Cryptosec Dekaton

2.1 Overview

The Cryptosec Dekaton is a high-end PCIe card that provides cryptographic services and secure storage of cryptographic keys. It is built to perform cryptographic processing and features a tamper-protective case to physically protect sensitive information contained within the card.

Figure 2.1: Cryptosec Dekaton top view

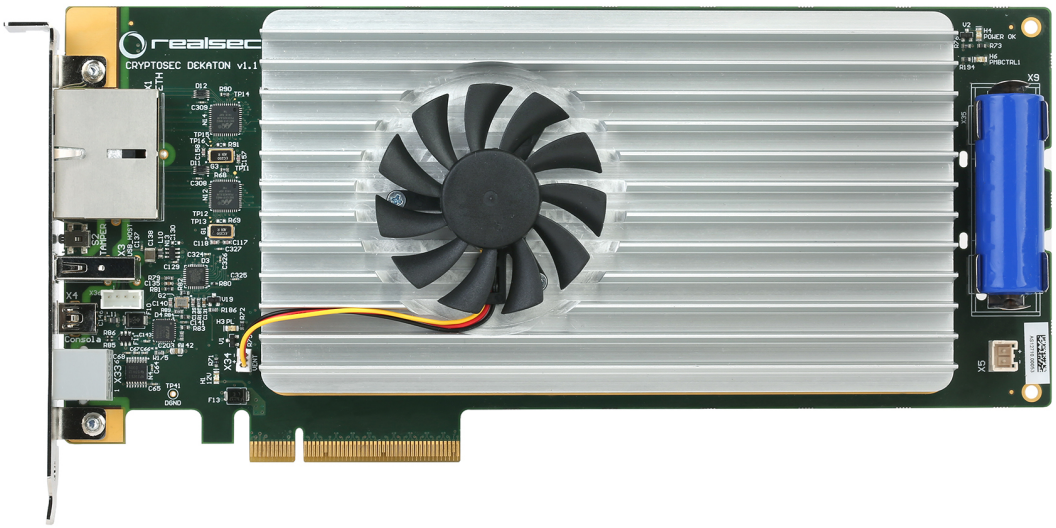


Figure 2.2: Cryptosec Dekaton lateral view

The Cryptosec Dekaton is designed to provide the highest level of security; security has been incorporated into the Cryptosec Dekaton since product inception. The Cryptosec Dekaton product design, development, test and production has satisfied the requirements to ensure a secure product. Security has been the focus of the development team, and the Cryptosec Dekaton product has been designed from the ground up to incorporate security in all design and development steps.

The Cryptosec Dekaton is certified to meet the NIST FIPS PUB 140-2 security requirements for the levels shown in the following table. The overall module is certified NIST FIPS PUB 140-2 Level 3.

Table 2.1: Security requirements levels

Security Requirement	Level
1. Cryptographic Module Specification	3
2. Module Ports and Interfaces	3
3. Roles, Services, and Authentication	3
4. Finite State Model	3
5. Physical Security	3
6. Operational Environment	N/A
7. Cryptographic Key Management	3
8. EMI / EMC	3
9. Self Tests	3

Table 2.1: Security requirements levels (continued)

Security Requirement	Level
10. Design Assurance	3
11. Mitigation of Other Attacks	3

The Cryptosec Dekaton is comprised of the module itself and the supplied drivers to access the functionality of the product. A special USB cable is also included.

2.2 Module Specification

2.2.1 Approved Cryptographic Algorithms in the NIST FIPS PUB 140-2-Approved mode of operation

Table 2.2: Approved Cryptographic algorithms in the NIST FIPS PUB 140-2-Approved mode of operation

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#5326	AES	[FIPS197]	ECB, CBC, CTR	128, 192, 256	Encryption Decryption
		[SP800-38B]	CMAC		Message authentication
		[SP800-38D]	GCM/GMAC		Encryption Decryption
#2688	TDES	[SP800-67r1]	ECB, CBC	192	Encryption Decryption
		[SP800-38B]	CMAC		Message authentication
#1376 #1384	DSA	[FIPS186-4]	KeyGen	L=2048, N=224 L=2048, N=256 L=3072, N=256	Key pair generation
			PQGen	L=2048, N=224, SHA-224 L=2048, N=256, SHA-256 L=3072, N=256, SHA-256	Domain parameters generation
			SigGen		Signature generation
			PQVer	L=1024, P=160, SHA-1 L=2048, N=224, SHA-224 L=2048, N=256, SHA-256 L=3072, N=256, SHA-256	Domain parameters verification
			SigVer		Signature verification

Table 2.2: Approved Cryptographic algorithms in the NIST FIPS PUB 140-2-Approved mode of operation (continued)

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#2854 #2865	RSA	[FIPS186-4]	KeyGen	e=fermat4, M=2048 e=fermat4, M=3072	Key pair generation
			SigGen X9.31	e=fermat4, M=2048, SHA-2 e=fermat4, M=3072, SHA-2 e=fermat4, M=4096, SHA-2	Signature generation
			SigVer X9.31		Signature verification
			SigGen PKCS1v1.5		Signature generation
			SigVer PKCS1v1.5		Signature verification
			SigGen PSS	e=fermat4, M=2048, SHA-2, SLen=160 e=fermat4, M=3072, SHA-2, SLen=160 e=fermat4, M=4096, SHA-2, SLen=160	Signature generation
			SigVer PSS		Signature verification
#1398 #1793 (CVL)	ECDSA	[FIPS186-4]	KeyGen	P-224, -256, -384, -521 K-233, 283, -409, -571 B-233, -283, -409, -571	Key pair generation
			SigVer		Signature verification
			SigGen		Signature generation
			PubKeyVal	P-192, -224, -256, -384, -521 K-163, -233, 283, -409, -571 B-163, -233, -283, -409, -571	Public key validation
#1410 #1830 (CVL)	ECDSA	[FIPS186-4]	KeyGen	P-224, -256, -384, -521	Key pair generation
			SigVer		Signature verification
			SigGen		Signature generation

Table 2.2: Approved Cryptographic algorithms in the NIST FIPS PUB 140-2-Approved mode of operation (continued)

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#1410 #1830 (CVL)	ECDSA	[FIPS186-4]	PubKeyVal	P-192, -224, -256, -384, -521	Public key validation
#4278	SHS	[FIPS180]	-	SHA-1, SHA-224, SHA-256, SHA-384 SHA-512, SHA-512/224, SHA-512/256	Data digest
#3524	HMAC	[FIPS198-1]	KS<BS	SHA-1, SHA-224, SHA-256, SHA-384 SHA-512, SHA-512/224, SHA-512/256	MAC generation
#2056	CTR_DRBG	[SP800-90A]	AES-256 based	-	deterministic random bit generation
#189	KBKDF	[SP800-108]	Counter before fixed data	r=8 HMAC-SHA-1, HMAC-SHA-224 HMAC-SHA-256, HMAC-SHA-384 HMAC-SHA-512, HMAC-SHA-512/224 HMAC-SHA-512/256, CMAC-AES-128 CMAC-AES-192, CMAC-AES-256 CMAC-TDES (3 key)	Key-Based Key Derivation Using Pseudorandom Functions
#1806 (CVL)	KDF	[SP800-135]	SSH	SHA-1, SHA-256, SHA-384, SHA-512	Key Derivation
#V/A	KTS-OAEP-Party V-confirmation	[SP800-56Br1]	-	e=fermat4, M=2048 e=fermat4, M=3072 e=fermat4, M=4096	Key encapsulation
#V/A	CKG (Cryptographic Key Generation)	[SP800-133]	-	-	Key generation
#N/A	KTS (AES #5326 HMAC #3524)	[SP800-38F]	AES CBC HMAC	AES-256 256	Key wrapping

Where #N/A means Not Apply and #V/A means Vendor Affirmed.

Note: AES GCM/GMAC Key/IV pair uniqueness requirements from [SP800-38D]: the IV is generated internally at its entirety randomly.

Note: [SP800-133]: the resulting symmetric key or generated seed is an unmodified output from a DRBG.

Note: The minimum number of bits of entropy generated by the module for use in key generation is 256 bits.

Note: No parts of the SSH protocol, other than the KDF, have been tested by the CAVP and CMVP.

Note: AES key wrapping provides 256 bits of security strength.

Note: RSA-based key encapsulation provides between 112 and 150 bits of security strength.

2.2.2 Allowed Cryptographic Algorithms in the NIST FIPS PUB 140-2-Approved mode of operation

Table 2.3: Allowed Cryptographic algorithms in the NIST FIPS PUB 140-2-Approved mode of operation

Algorithm	Standard	Use
NDRNG	[SP800-90B]	Seed/Reseed CTR_DRBG

2.2.3 Cryptographic Algorithms in the Non-NIST FIPS PUB 140-2-Approved modes of operation

Note: In the context of the security requirements of NIST FIPS PUB 140-2 and specifically in this NIST FIPS PUB 140-2 Non-Proprietary Security Policy, the only Approved mode of operation is the NIST FIPS PUB 140-2-Approved mode of operation. The PCI PTS HSM v2.0-Approved mode of operation is to be considered as a Non-Approved mode of operation, and therefore not allowed in the NIST FIPS PUB 140-2-Approved mode of operation. This is due to not only Approved security functions are used in such mode of operation.

Table 2.4: Cryptographic algorithms in the PCI PTS HSM v2.0-Approved mode of operation

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#N/A	AES	[FIPS197]	ECB, CBC, CTR	128, 192, 256	Encryption Decryption
		[SP800-38B]	CMAC		Message authentication
		-	GCM/GMAC		Encryption Decryption
#N/A	TDES	[SP800-67r1]	ECB, CBC	128, 192	Encryption Decryption
		[SP800-38B]	CMAC		Message authentication
#N/A	MD5	[RFC1321]	-	-	Data digest
#N/A	SHS	[FIPS180]		SHA-1, SHA-224, SHA-256, SHA-384 SHA-512, SHA-512/224, SHA-512/256	
#N/A	HMAC	[FIPS198-1]	KS<BS	SHA-1, SHA-224, SHA-256, SHA-384 SHA-512, SHA-512/224, SHA-512/256	MAC generation
#N/A	DSA	[FIPS186-2]	PQGen	L=1024, N=160, SHA-1 L=2048, N=224, SHA-224 L=2048, N=256, SHA-256 L=3072, N=256, SHA-256	Domain parameters generation
			PQVer		Domain parameters verification
			KeyGen		Key pair generation
			SigGen		Signature generation
			SigVer		Signature verification

Table 2.4: Cryptographic algorithms in the PCI PTS HSM v2.0-Approved mode of operation (continued)

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#N/A	RSA	-	KeyGen	e=3, fermat4, M=512..4096	Key pair generation
			SigGen X9.31	e=3, fermat4, M=512..4096 MD5, SHA1, SHA-2	Signature generation
			SigVer X9.31		Signature verification
			SigGen PKCS1v1.5		Signature generation
			SigVer PKCS1v1.5		Signature verification
			EncDec PKCS1v1.5		Encryption Decryption
			SigGen PSS	e=3, fermat4, M=512..4096 MD5, SHA-1, SHA-2, SLen=160	Signature generation
			SigVer PSS		Signature verification
			SigGen OAEP	e=3, fermat4, M=512..4096 MD5, SHA1, SHA-2 MGF-SHA-1	Signature generation
			SigVer OAEP		Signature verification
			EncDec OAEP		Encryption Decryption
#N/A	KBKDF	[SP800-108]	Counter before fixed data	r=8 HMAC-SHA-1, HMAC-SHA-224 HMAC-SHA-256, HMAC-SHA-384 HMAC-SHA-512, HMAC-SHA-512/224 HMAC-SHA-512/256, CMAC-AES-128 CMAC-AES-192, CMAC-AES-256 CMAC-TDES (3 key)	Key-Based Key Derivation Using Pseudorandom Functions

Table 2.4: Cryptographic algorithms in the PCI PTS HSM v2.0-Approved mode of operation (continued)

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#N/A	ECDSA	[FIPS186-4] [SP800-133]	KeyGen	P-192, -224, -256, -384, -521 K-163, -233, 283, -409, -571 B-163, -233, -283, -409, -571	Key pair generation
			PubKeyVal		Public key validation
		[FIPS186-4]	SigGen		Signature generation
			SigVer		Signature verification
#N/A	CTR_DRBG	[SP800-90A]	AES-256 based	-	deterministic random bit generation
#N/A	KDF	[SP800-135]	SSH	SHA-1, SHA-256, SHA-384, SHA-512	Key Derivation
#N/A	KTS-OAEP-Party V-confirmation	[SP800-56Br1]	-	e=3, fermat4, M=512..4096	Key encapsulation
#N/A	RSAES-PKCS1-OAEP encryption with RSA PSS signature	-			
#N/A	CKG (Cryptographic Key Generation)	[SP800-133]	-	-	Key generation
#N/A	KTS (AES #N/A HMAC #N/A)	[SP800-38F]	AES CBC HMAC	AES-256 256	Key wrapping
#N/A	Derived wrapping key methods	-	DES TDES AES	64 128, 192 128, 192, 256	Key wrapping

Where #N/A means Not Apply.

Table 2.5: Cryptographic algorithms in the Non-Approved mode of operation

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#N/A	AES	[FIPS197]	ECB, CBC, CTR	128, 192, 256	Encryption Decryption
		[SP800-38B]	CMAC		Message authentication
		-	GCM/GMAC		Encryption Decryption
#N/A	DES	[SP800-67r1]	ECB, CBC	64	Encryption Decryption
		[SP800-38B]	CMAC		Message authentication
#N/A	TDES	[SP800-67r1]	ECB, CBC	128, 192	Encryption Decryption
		[SP800-38B]	CMAC		Message authentication
#N/A	DSA	[FIPS186-2]	PQGen	L=1024, N=160, SHA-1 L=2048, N=224, SHA-224 L=2048, N=256, SHA-256 L=3072, N=256, SHA-256	Domain parameters generation
			PQVer		Domain parameters verification
			KeyGen		Key pair generation
			SigGen		Signature generation
			SigVer		Signature verification
#N/A	MD5	[RFC1321]	-	-	Data digest
#N/A	SHS	[FIPS180]		SHA-1, SHA-224, SHA-256, SHA-384 SHA-512, SHA-512/224, SHA-512/256	

Table 2.5: Cryptographic algorithms in the Non-Approved mode of operation (continued)

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#N/A	RSA	-	SigGen X9.31	e=3, fermat4, M=512..4096 MD5, SHA1, SHA-2	Signature generation
			SigVer X9.31		Signature verification
			SigGen PKCS1v1.5		Signature generation
			SigVer PKCS1v1.5		Signature verification
			EncDec PKCS1v1.5		Encryption Decryption
			SigGen PSS	e=3, fermat4, M=512..4096 MD5, SHA-1, SHA-2, SLen=160	Signature generation
			SigVer PSS		Signature verification
			SigGen OAEP	e=3, fermat4, M=512..4096 MD5, SHA1, SHA-2 MGF-SHA-1	Signature generation
			SigVer OAEP		Signature verification
			EncDec OAEP		Encryption Decryption
			KeyGen	e=3, fermat4, M=512..4096	Key pair generation
#N/A	HMAC	[FIPS198-1]	KS<BS	SHA-1, SHA-224, SHA-256, SHA-384 SHA-512, SHA-512/224, SHA-512/256	MAC generation
#N/A	CTR_DRBG	[SP800-90A]	AES-256 based	-	deterministic random bit generation
#N/A	KDF	[SP800-135]	SSH	SHA-1, SHA-256, SHA-384, SHA-512	Key Derivation

Table 2.5: Cryptographic algorithms in the Non-Approved mode of operation (continued)

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#N/A	ECDSA	-	KeyGen	P-192, -224, -256, -384, -521 K-163, -233, 283, -409, -571 B-163, -233, -283, -409, -571 Brainpool: P160r1, P192r1, P224r1, P256r1, P320r1, P384r1, P512r1 P160t1, P192t1, P224t1, P256t1, P320t1, P384t1, P512t1 Prime field arbitrary curves from 160 to 521 Binary field arbitrary curves from 160 to 571	Key pair generation
			PubKeyVal		Public key validation
			SigGen		Signature generation
			SigVer		Signature verification
#N/A	KBKDF	[SP800-108]	Counter before fixed data	r=8 HMAC-SHA-1, HMAC-SHA-224 HMAC-SHA-256, HMAC-SHA-384 HMAC-SHA-512, HMAC-SHA-512/224 HMAC-SHA-512/256, CMAC-AES-128 CMAC-AES-192, CMAC-AES-256 CMAC-TDES (3 key)	Key-Based Key Derivation Using Pseudorandom Functions
#N/A	PBKDF2	[SP800-132]	-	112 <kLen <$(2^{32}$-1).160 saltLen >= 128 passwordLen <= 512	Key derivation
#N/A	KTS-OAEP- Party V-confirmation	[SP800-56Br1]	-	e=3, fermat4, M=512..4096	Key encapsulation
#N/A	RSAES-PKCS1-OAEP encryption with RSA PSS signature	-			

Table 2.5: Cryptographic algorithms in the Non-Approved mode of operation (continued)

Certs.	Algorithm	Standard	Mode Method	Key Lengths Curves Module	Use
#N/A	CKG (Cryptographic Key Generation)	[SP800-133]	-	-	Key generation
#N/A	KTS (AES #N/A HMAC #N/A)	[SP800-38F]	AES CBC HMAC	AES-256 256	Key wrapping
#N/A	Derived and Non-derived wrapping key methods	-	DES TDES AES	64 128, 192 128, 192, 256	Key wrapping

Where #N/A means Not Apply.

2.2.4 CSPs and other Protected Information

Table 2.6: CSPs and other Protected Information

CSP	Type	Generation	Input	Output	Storage	Zeroization	Use	Mode
SCCK Key	AES-256 key	Internally generated [SP800-133]	API call: Input in plaintext shares	API call: Output in plaintext shares	Plaintext in secure memory	-API call -Module zeroization	Set up the SCP03 secure channel	All
Old SCCK Key		Former SCCK Key	Never input	Never output			Translation	

Table 2.6: CSPs and other Protected Information (continued)

CSP	Type	Generation	Input	Output	Storage	Zeroization	Use	Mode
AES Encryption Key	AES-128 AES-192 AES-256 keys	API call: Internally generated [SP800-133]	API calls: -Input in plaintext shares -Unencapsulated [SP800-56Br1] -Unwrapped [SP800-38F]	API calls: -Output in plaintext shares -Encapsulated [SP800-56Br1] -Wrapped [SP800-38F]	If stored: in NAND Flash encrypted with the System Key	-API call if stored -Remove power	Encryption Decryption	All
AES GCM/ GMAC Key								
AES CMAC Key							MAC: generation verification	
AES HMAC Key							HMAC: generation verification	
AES Wrap Key	AES-256 key						Wrapping Unwrapping	
TDES (3 key) Encryption Key	TDES (3 key) key						Encryption Decryption	
TDES (3 key) CMAC Key							MAC: generation verification	
TDES (3 key) HMAC Key							HMAC: generation verification	

Table 2.6: CSPs and other Protected Information (continued)

CSP	Type	Generation	Input	Output	Storage	Zeroization	Use	Mode
SSH Key	ECDSA P-521key	Internally generated on boot if not exists [FIPS186-4]	Never input	Never output	In NAND Flash encrypted with the System Key	Not zeroized Unusable when the System Key is zeroized	Set up the SSH secure channel	FIPS PUB 140-2-Approved
System Key	AES-256 Key	Internally generated [SP800-133]	API call: Input in plaintext shares		Plaintext in secure memory	Module zeroization	Confidentiality of CSPs on internal HSM's memory	All
Batch Key		Never generated			Not persistently stored	-API call -Remove power	Set up the SCP03 secure channel in a new smart cards' batch	
AES -256 Master Key		Internally generated [SP800-133]			Plaintext in secure memory	Module zeroization	Derive CMM_AES_ENCRYPTION_KEY Key and CMM_AES_MAC_KEY Key	
Old AES -256 Master Key		Former AES -256 Master Key	Never input			-API call -Module zeroization	Translation	

Table 2.6: CSPs and other Protected Information (continued)

CSP	Type	Generation	Input	Output	Storage	Zeroization	Use	Mode
CMM_ AES_ ENCRYP- TION_ KEY Key	AES-256 Key	Derived from AES -256 Master Key [SP800-108]	Never input	Never output	Not persistently stored	-API call -Remove power	Confidentiality of CSPs outside of HSM	All
CMM_ AES_ MAC_ KEY Key							Authenticity of CSPs outside of HSM	
Old CMM_ AES_ ENCRYP- TION_ KEY Key		Derived from Old AES -256 Master Key [SP800-108]				Remove power	Translation	
Old CMM_ AES_ MAC_ KEY Key								
AES Enc Key		Derived from AES-256 Wrap Key [SP800-108]					Confidentiality of wrapped keys	
AES Auth Key							Authenticity of wrapped keys	

Table 2.6: CSPs and other Protected Information (continued)

CSP	Type	Generation	Input	Output	Storage	Zeroization	Use	Mode
DSA Domain Parameters	DSA L=2048, N=224 L=2048, N=256 L=3072, N=256 domain parameters	API call: Generated [FIPS186-4]	API call: Unwrapped [SP800-38F]	API call: Wrapped [SP800-38F]	Not persistently stored	Remove power	DSA Key Pair generation	All
DSA Key Pair	DSA L=2048, N=224 L=2048, N=256 L=3072, N=256 keys						Signature: generation verification	
ECDSA Key Pair	ECDSA P-224, -256, -384, -521 K-233, 283, -409, -571 B-233, -283, -409, -571 keys						Signature: generation verification	
RSA Key Pair	RSA e=fermat4, M=2048 e=fermat4, M=3072 e=fermat4, M=4096 keys						Signature: generation verification	
RSA Encapsulation Key Pair							Encapsulation Unencapsulation	
Smart Card PIN	8-byte string	Never generated	Input in plaintext	Never output	Not persistently stored	-API call -Remove power	Smart card access	
CO secret	33-byte string	Internal generation	Input in plaintext shares	Output in plaintext shares	Plaintext in secure memory	Module zeroization	Crypto Officer authentication	

Table 2.6: CSPs and other Protected Information (continued)

CSP	Type	Generation	Input	Output	Storage	Zeroization	Use	Mode
User password	8-byte string	Never generated	Input in plaintext	Never output	Plaintext in secure memory	-API call -Module zeroization	User authentication	All
Custodian password	24-byte string			Output in plaintext			Custodian authentication	
CTR_ DRBG Entropy Input	48-byte array	Internal generation	Never input	Never output	Not persistently stored	-When used -Remove power	Random data generation	
CTR_ DRBG Seed								
CTR_ DRBG V Value	16-byte array					Remove power		
CTR_ DRBG Key	AES-256 Key							

2.3 Module Interfaces

The Cryptosec Dekaton is classified as a **multi-chip embedded module** for NIST FIPS PUB 140-2 purposes. The NIST FIPS PUB 140-2 cryptographic boundary is defined by the perimeter of the protection covers. The battery system is excluded from the security requirements of NIST FIPS PUB 140-2. The Cryptosec Dekaton is accessible only through well-defined interfaces.

Note: In the following enumeration, the texts between brackets refer a component on the Cryptosec Dekaton PCB.

Note: In the following figures, the yellow squares identify the elements that give access to each physical port.

The Cryptosec Dekaton has several physical ports:

1. External battery port:

   The purpose of this port is to provide power to the Cryptosec Dekaton in absence of power through the PCIe 8x standard fingers. This power is used for for the retention and protection of the internally stored data.

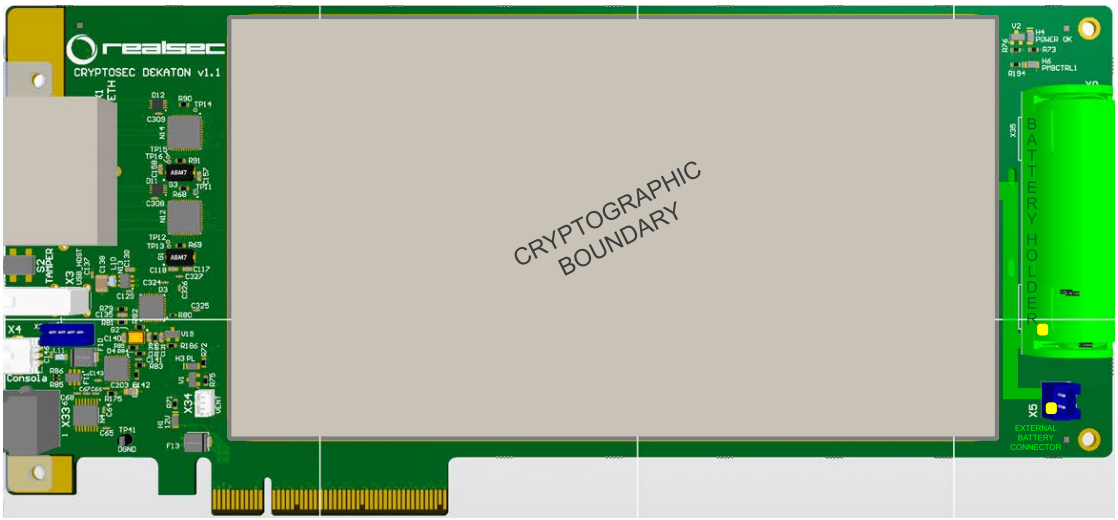
   This port is accessible to the User by means of:

   - AA Battery holder (X9).
   - External battery connector (X5).

   Physically it is conformed by the following nets:

   - VBat.
   - DGND.

Figure 2.3: External battery port Access



2. PCIe 8x port:

   The purpose of this port is to provide communication with a host system through a PCI Express interface.

   This port is accessible by means of the PCIe 8x standard fingers (X2).

   Physically it is conformed by the following nets:

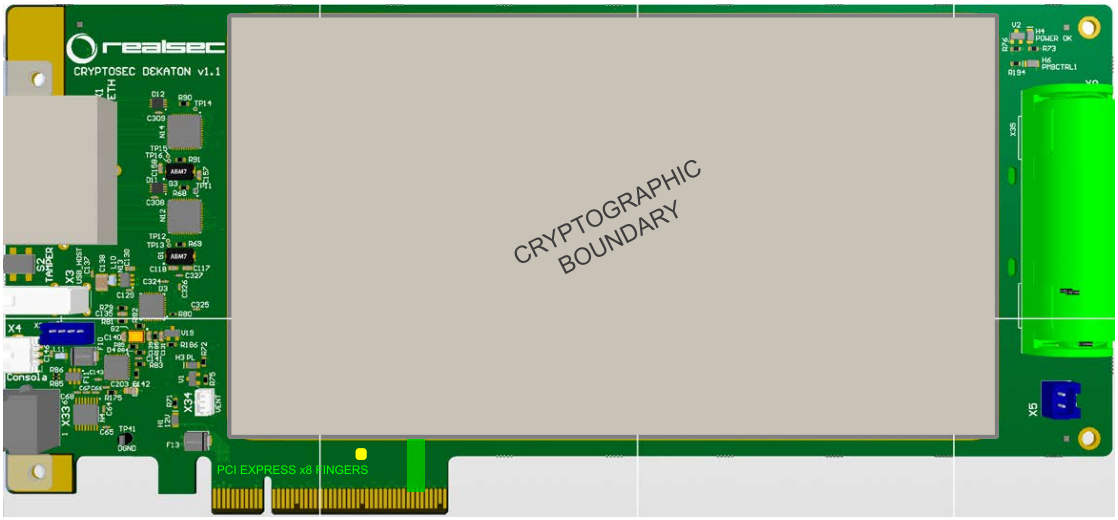
Table 2.7: PCIe 8x standard fingers Pin Assignment

Pin	Side B	Net	Note	Side A	Net	Note
1	+12V	PCIe_+12V		/PRSNT1	PCIe.PRSNT1	
2	+12V	PCIe_+12V		+12V	PCIe_+12V	
3	+12V	PCIe_+12V		+12V	PCIe_+12V	
4	GND	DGND		GND	DGND	
5	SMCLK		N/C	JTAG2		N/C
6	SMDAT		N/C	JTAG3		N/C
7	GND		N/C	JTAG4		N/C
8	+3V3		N/C	JTAG5		N/C
9	JTAG1		N/C	+3V3		N/C
10	3V3aux		N/U	+3V3		N/C
11	/WAKE		N/C	/PERST	PCIe./PERST	
			Key			Key
12	RSVD		N/C	GND	DGND	
13	GND	DGND		REFCLK+	PCIe.CLK_P	
14	PET0_P	PCIe.RX0_P		REFCLK-	PCIe.CLK_N	
15	PET0_N	PCIe.RX0_N		GND	DGND	
16	GND	DGND		PER0_P	PCIe.TX0_P	
17	/PRSNT2		N/C	PER0_N	PCIe.TX0_N	
18	GND	DGND		GND	DGND	
19	PET1_P	PCIe.RX1_P		RSVD		
20	PET1_N	PCIe.RX1_N		GND	DGND	
21	GND	DGND		PER1_P	PCIe.TX1_P	
22	GND	DGND		PER1_N	PCIe.TX1_N	
23	PET2_P	PCIe.RX2_P		GND	DGND	
24	PET2_N	PCIe.RX2_N		GND	DGND	
25	GND	DGND		PER2_P	PCIe.TX2_P	
26	GND	DGND		PER2_N	PCIe.TX2_N	
27	PET3_P	PCIe.RX3_P		GND	DGND	
28	PET3_N	PCIe.RX3_N		GND	DGND	
29	GND	DGND		PER3_P	PCIe.TX3_P	
30	RSVD		N/C	PER3_N	PCIe.TX3_N	
31	/PRSNT2		N/C	GND	DGND	
32	GND	DGND		RSVD		N/C
33	PET4_P	PCIe.RX4_P		RSVD		N/C
34	PET4_N	PCIe.RX4_N		GND	DGND	
35	GND	DGND		PER4_P	PCIe.TX4_P	
36	GND	DGND		PER4_N	PCIe.TX4_N	
37	PET5_P	PCIe.RX5_P		GND	DGND	

Table 2.7: PCIe 8x standard fingers Pin Assignment (continued)

Pin	Side B	Net	Note	Side A	Net	Note
38	PET5_N	PCIe.RX5_N		GND	DGND	
39	GND	DGND		PER5_P	PCIe.TX5_P	
40	GND	DGND		PER5_N	PCIe.TX5_N	
41	PET6_P	PCIe.RX6_P		GND	DGND	
42	PET6_N	PCIe.RX6_N		GND	DGND	
43	GND	DGND		PER6_P	PCIe.TX6_P	
44	GND	DGND		PER6_N	PCIe.TX6_N	
45	PET7_P	PCIe.RX7_P		GND	DGND	
46	PET7_N	PCIe.RX7_N		GND	DGND	
47	GND	DGND		PER7_P	PCIe.TX7_P	
48	/PRSNT2	PCIe./PRSNT1		PER7_N	PCIe.TX7_N	
49	GND	DGND		GND	DGND	

Figure 2.4: PCIe 8x standard fingers Access



3. Gigabit Ethernet ports:

   The purpose of these ports is to provide communication with remote systems through two LAN interfaces.

   These ports are accessible by means of the Ethernet connectors (X1).

   The Gigabit Ethernet 0 port is physically conformed by the following nets or buses:

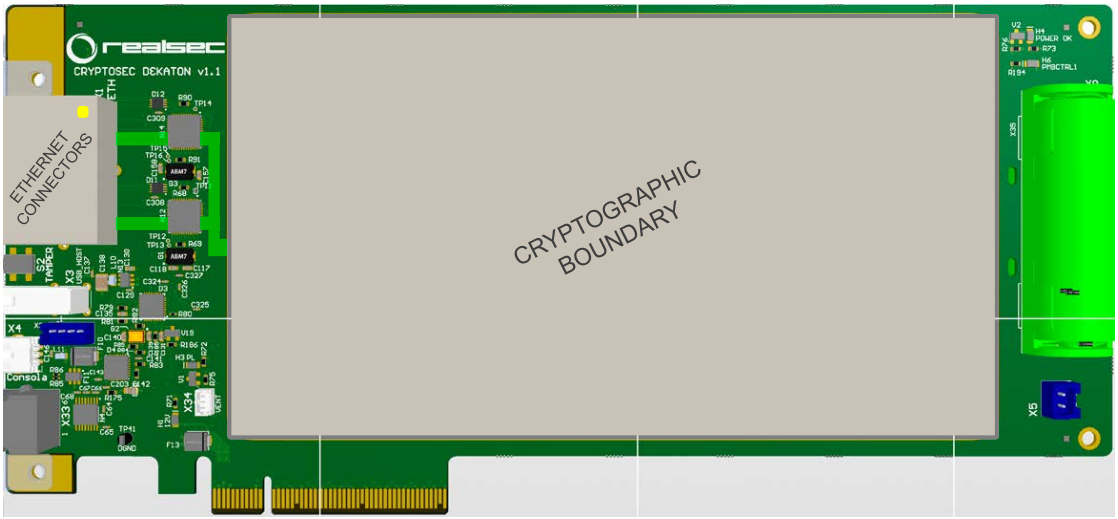
   - +3V3_PS.
   - +1V8_PS.
   - +1V0_PS.
   - RGMII_0.
   - PS_MDC

- PS_MDIO
- /ETH_RST
- RST_PHY_CTRL

The Gigabit Ethernet 1 port is physically conformed by the following nets or buses:

- +3V3_PS.
- +1V8_PS.
- +1V0_PS.
- RGMII_1.
- PS_MDC
- PS_MDIO
- /ETH_RST
- RST_PHY_CTRL

Figure 2.5: Gigabit Ethernet ports Access



4. UART ports:

   The purpose of these ports is to provide communication with close systems through two UART interfaces.

   These ports are accessible by means of the Serial connector (X33) that embeds two RS-232 null modem ports and the USB Target connector (X4) that holds a USB target port.

   Note: The UART 1 port is only accessible through the USB Target connector.

   The UART 0 port is physically conformed by the following nets:

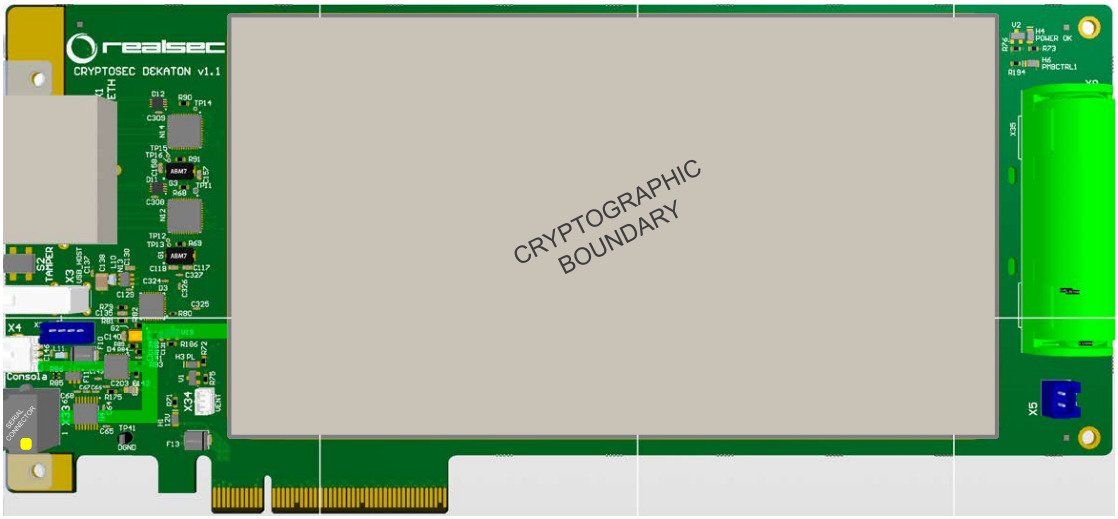
   - +3V3_PL.
   - DGND.
   - UART0_232_TX.

- UART0_232_RX.

The UART 1 port is physically conformed by the following nets:

- +3V3_PL.
- +5V0_PS.
- +1V8_PS.
- DGND.
- UART1_232_TX.
- UART1_232_RX.
- UART1_USB_TX.
- UART1_USB_RX.

Figure 2.6: UART ports Access



5. USB Host port:

   The purpose of this port is to provide communication with USB-target systems.

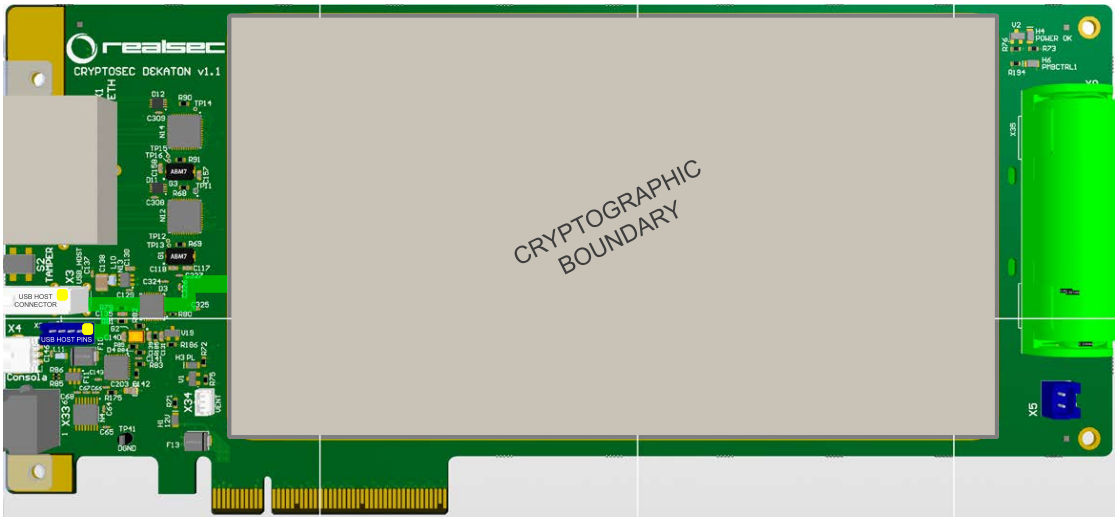
   This port is accessible by means of the USB Host connector (X3) or the USB Host pins (X36).

   Note: The USB Host port is only used during product installation and upgrade.

   Physically it is conformed by the following nets or buses:

   - +5V0_PS.
   - +1V8_PS.
   - ULPI.
   - RST_USB_CTRL.
   - /USB_RST.

Figure 2.7: USB Host port Access



6. Fan port:

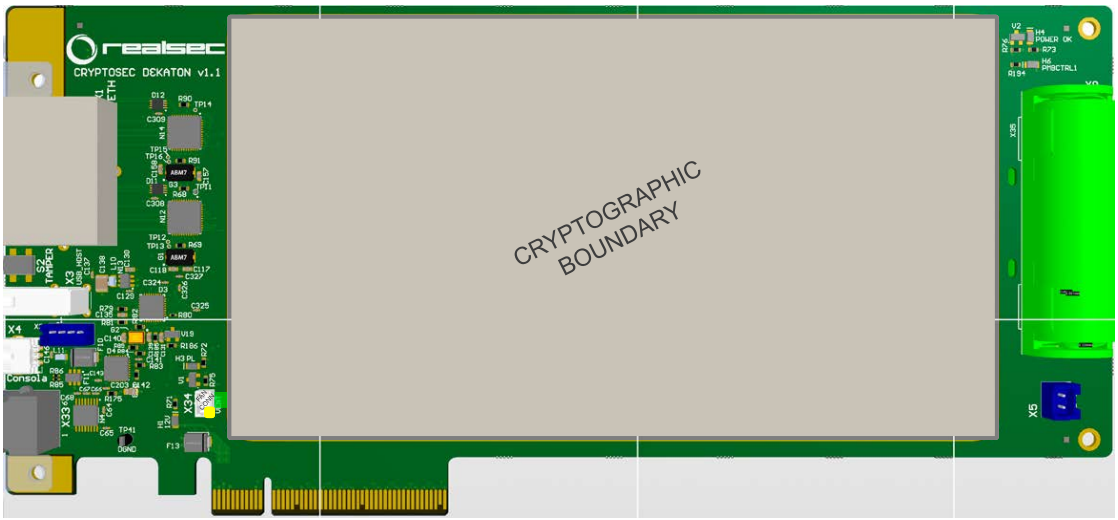
   The purpose of this port is to connect a fan to provide forced heat evacuation if needed.

   This port is accessible by means of the Fan connector (X34).

   Physically it is conformed by the following nets:

   - PCIe_+12V.
   - PWM_CTRL.

Figure 2.8: Fan port Access



7. Tamper port:

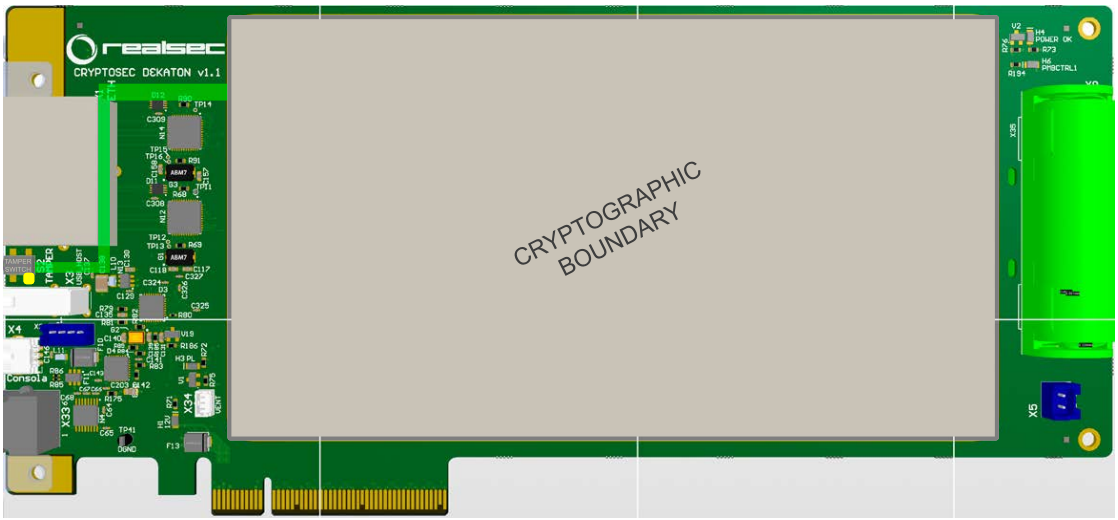
The purpose of this port is to force the activation of the active tamper penetration mechanism.

This port is accessible by means of the Tamper Switch (S2).

Physically it is conformed by the following net:

- MESH3_6.

Figure 2.9: Tamper port Access



8. Power Present port:

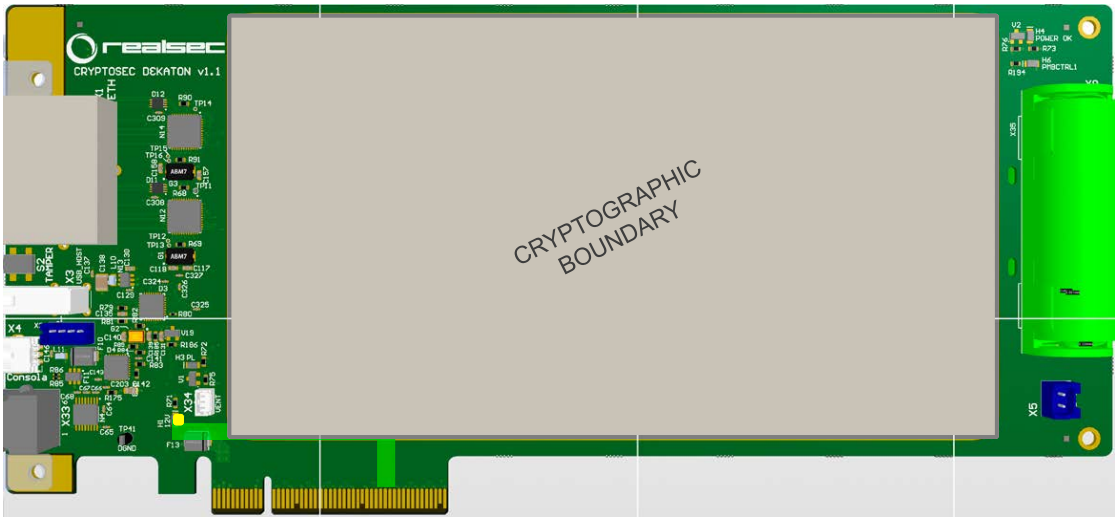
   The purpose of this port is to evidence the presence of power through the PCIe_+12V signal on the PCIe 8x port.

   This port is accessible by means of the Power Present LED (H1).

   Physically it is conformed by the following nets:

   - PCIe_+12V.
   - DGND.

Figure 2.10: Power Present port Access



9. Currently Unsupported port:

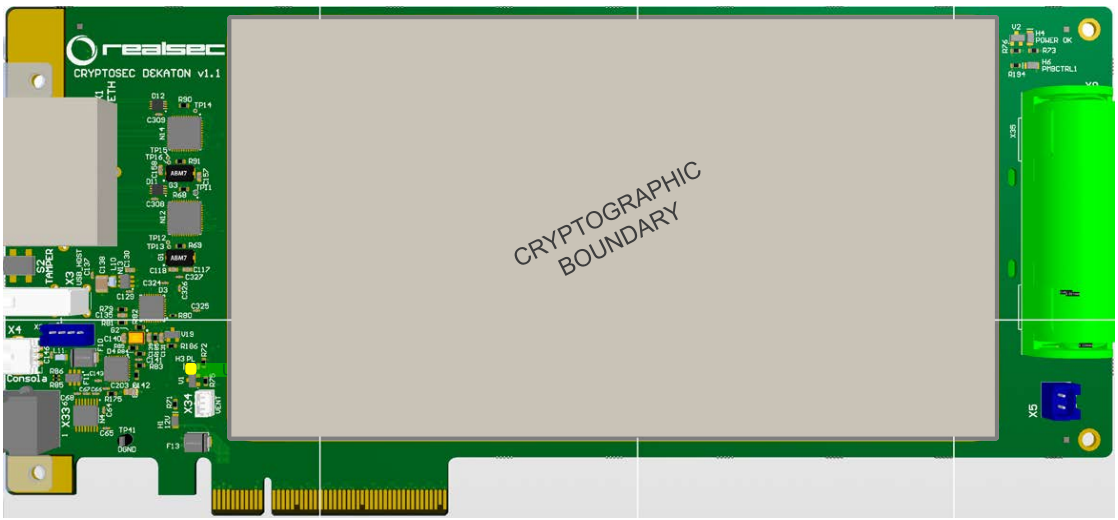
   The purpose of this port is reserved for future use.

   This port is accessible by means of the Currently Unsupported LED (H3).

   Physically it is conformed by the following nets:

   - +3V3_PS.
   - PL_LED.
   - DGND.

Figure 2.11: Currently Unsupported port Access



10. Power OK port:

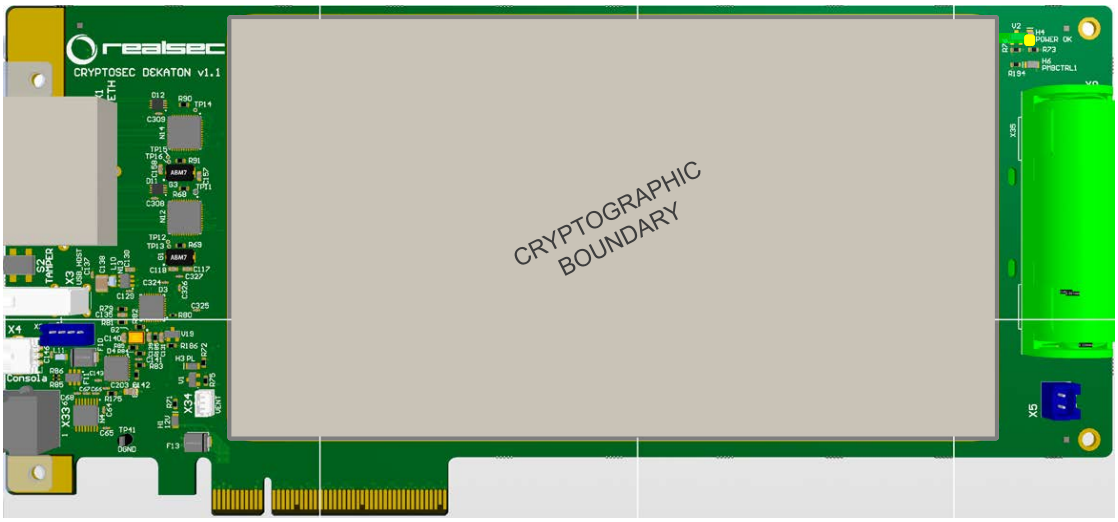
    The purpose of this port is to evidence that the Cryptosec Dekaton is properly powered from the PCIe_+12V signal on the PCIe 8x port.

    This port is accessible by means of the Power OK LED (H4).

    Physically it is conformed by the following nets:

    - +3V3_PS.
    - POWER_OK.
    - DGND.

Figure 2.12: Power OK port Access



11. DDR3 Zeroization Done port:

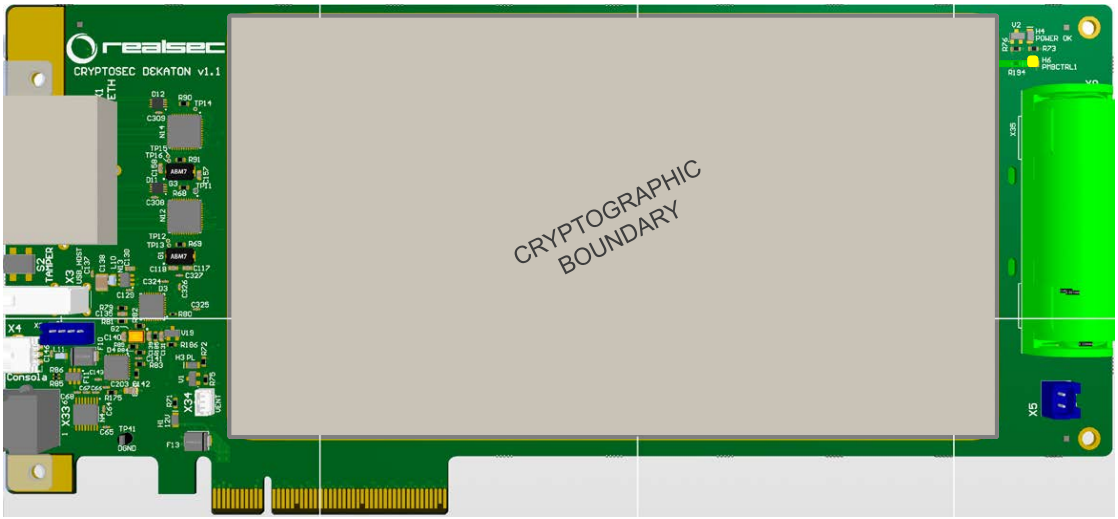
    The purpose of this port is to evidence that the DDR3 RAM of the Cryptosec Dekaton is properly zeroized.

    This port is accessible by means of the DDR3 Zeroization Done LED (H6).

    Physically it is conformed by the following nets:

    - +3V3_UCD9090.
    - PMBCTRL1.
    - DGND.

Figure 2.13: DDR3 Zeroization Done port Access



The Cryptosec Dekaton has several high-level logical interfaces:

1. Machine Interface:

   The Machine Interface is intended to communicate with other systems. The User services and some of the Unauthenticated services are provided through this interface.

   The API for this interface is described in 2.4.2 and in 2.4.3.

   The mapping of this interface to the NIST FIPS PUB 140-2 interfaces is as follows:

   - NIST FIPS PUB 140-2 Data Input Interface:
     Each of the inputs of the given service.
   - NIST FIPS PUB 140-2 Data Output Interface:
     Each of the outputs of the given service.
   - NIST FIPS PUB 140-2 Control Interface:
     The name of the service.
   - NIST FIPS PUB 140-2 Status Interface:
     The return code of the service.

   The Machine Interface is mapped to the physical ports as follows:

   - PCIe 8x port with the exception of the PCIe_+12V node.
   - Gigabit Ethernet ports.

2. Human Interface:

   The Human Interface is intended to communicate with operators. The Crypto Officer services and some of the Unauthenticated services are provided through this interface. Also, some of the User services provide printed output through this interface.

   This communication can be achieved in different forms. Each of these forms imply a different interface:

- Trusted Path:
  Implements the Cryptosec Dekaton console. It is implemented as an SSH connection, to secure plaintext security-related information flow. A public key is provided to the Cryptosec Dekaton during its initialization in order to setup this connection.
  The API for this interface is described in 2.4.1 and in 2.4.3.
  The Trusted Path goes from the cryptographic boundary of the HSM to a trusted IT element that allows the operator interaction, based on a text console and a keyboard.
  The SSH connection is slightly different from the usual SSH. There is a single user allowed to connect to the HSM and there is a single console available. The user name used for connecting to the console is "remote" and is authenticated using public key cryptography. When the "remote" user is authenticated, the console is detached from the current session (if any) and attached to the newly created session. Before attaching the console to the newly created SSH session, the system is logged out so the console is in an unauthenticated state when the SSH session is created.
  The operator may then authenticate on the console using his smart cards and providing the needed PINs to unblock the smart card.
  From a cryptographic point of view, the SSH server embedded in the HSM has a key pair and the user has another key pair:
  - The SSH server key pair is used to securely identify and authenticate the HSM. This key pair is created during the HSM initialization, when its public key hash is provided to the operator. The private key is stored encrypted on NAND Flash using the System Key.
  - The SSH user key pair is used to identify the user "remote" on the system. During initialization, the Crypto Officer must provide the public key for the "remote" user and securely store the corresponding private key.

  Both key pairs are P-521 ECDSA keys.
  Both plaintext security-related information and non-security-related information enter and leave the Cryptosec Dekaton Trusted Path. In order to distinguish appropriately between them, two logical interfaces are defined to conform the Trusted Path:
  - Non-CSP Interface:
    Implements the non-security-related information flow over the Cryptosec Dekaton console.
    The mapping of this interface to the NIST FIPS PUB 140-2 interfaces is as follows:
    * NIST FIPS PUB 140-2 Data Input Interface:
      Each of the non-security-related inputs of the given service.
    * NIST FIPS PUB 140-2 Data Output Interface:
      Each of the non-security-related outputs of the given service.
    * NIST FIPS PUB 140-2 Control Interface:
      The name of the service.
    * NIST FIPS PUB 140-2 Status Interface:
      The return code of the service.
  - CSP Interface:
    Implements the security-related information flow over the Cryptosec Dekaton console.
    The mapping of this interface to the NIST FIPS PUB 140-2 interfaces is as follows:
    * NIST FIPS PUB 140-2 Data Input Interface:
      Each of the security-related inputs of the given service.

    - NIST FIPS PUB 140-2 Data Output Interface:
      Each of the security-related outputs of the given service.

  The mapping implies that the Non-CSP Interface and the CSP Interface share the Trusted Path.

  The Trusted Path is mapped to the physical ports as follows:

  - PCIe 8x port with the exception of the PCIe_+12V node.
  - Gigabit Ethernet ports.

  Note: To configure the Trusted Path: Connect one of the Ethernet ports of the Cryptosec Dekaton to the Ethernet port of the SSH client device. The device's Ethernet port is to be configured as `169.254.1.5/30`.

  A key is to be used to establish the SSH connection. It can be generated as shown:

  ```
  ssh-keygen -t ecdsa -b 521 -f ssh_client.key -C "Your comment here"
  ```

  The connection is to be opened as follows:

  ```
  ssh -i ssh_client.key remote@169.254.1.2
  ```

  Note: ssh_client.key.pub is to be provided to the Cryptosec Dekaton during initialization through the serial console.

- Printer Interface:

  Implements the printing facilities of the Cryptosec Dekaton.

  In NIST FIPS PUB 140-2-Approved mode, the printing is restricted to CSPs, i. e. symmetric key shares and banking PINs.

  Due to its nature, this interface presents no API. Instead, some of the services associated to other interfaces generate output through this interface.

  This interface is mapped to the NIST FIPS PUB 140-2 Data Output Interface:

  Each of the outputs of the given service that are to be produced through this interface.

  The Printer Interface is mapped to the UART 0 port.

- Serial Interface:

  Implements status indicators during the power-up and power-down of the Cryptosec Dekaton.

  Due to its nature, this interface presents no API.

  This interface is mapped to the NIST FIPS PUB 140-2 Status Interface.

  The Serial Interface is mapped to the UART 1 port.

- Physical Interface:

  Implements the manual controls and physical status indicators.

  Due to its nature, this interface presents no API.

  The mapping of this interface to the NIST FIPS PUB 140-2 interfaces is as follows:

  - NIST FIPS PUB 140-2 Control Interface:
    The attack emulation request.
  - NIST FIPS PUB 140-2 Status Interface:
    - The power supply indicator.
    - The power OK indicator.
    - The DDR3 zeroization indicator.

  The Physical Interface is mapped to the physical ports as follows:

  - Tamper port.

  - Power Present port.
  - Power OK port.
  - DDR3 Zeroization Done port.

3. Power Interface:

   The Power Interface is intended to provide electrical power and to evacuate heat.

   Due to its nature, this interface presents no API.

   The mapping of this interface to the NIST FIPS PUB 140-2 interfaces is as follows:

   - NIST FIPS PUB 140-2 Power Interface:
     The power supplies.
   - NIST FIPS PUB 140-2 Status Interface:
     The fan control.

   The Power Interface is mapped to the physical ports as follows:

   - External battery port.
   - The following nodes of PCIe 8x port:
     - PCIe_+12V.
     - DGND.
   - Fan port.

The physical ports connected to the logical interfaces are under sole control of the logical interfaces, they are logically disconnected from other modules of the Cryptosec Dekaton. The logical interfaces present a well defined API to be accessed from other modules of the Cryptosec Dekaton. This scheme isolates both physically and logically the output data path from the internal processes of the Cryptosec Dekaton, and in particular from the circuitry and processes that perform key generation, manual key entry and key zeroization.

There is not a maintenance interface. To perform physical maintenance, the battery has to be replaced with a new one every 4 years. This operations is to be made by a Crypto Officer.

Every sensitive information that is entered to the Cryptosec Dekaton in plain-text form is entered through the Trusted Path. Every sensitive information that leaves the module in plain-text form is extracted through the Trusted Path or the Printer Interface. A trusted SSH client system must be connected to the Gigabit Ethernet ports. A trusted printer can be connected to the UART 0 port.

2.4 Roles and Services

The Cryptosec Dekaton performs identity-based authentication. The role of an operator is assigned when the operator is created.

The roles supported by the Cryptosec Dekaton are:

- Unauthenticated: for those services available before any authentication.

- User: User role as defined in [FIPS140-2, AS03.03].
- Crypto Officer: Crypto Officer role as defined in [FIPS140-2, AS03.03].

Aside from them, there are two operators that are double authentication factors associated to the Crypto Officer role:

- Custodian: Is responsible of the input, the output and the protection of symmetric keys shares.
- Master Keys Holder: Is responsible of the input, the output and the protection of Master Keys Cards.

The unauthenticated services do not provide any security relevant functionality and they are not only available for the Unauthenticated role, but for the Crypto Officer role through the Human Interface (except the Crypto Officer login service), and for the User role through the Machine Interface.

The custodians do not have any specific services. They only take part in processes started by the Crypto Officer when required.

The Master Key Holders do not have any specific services. They only take part in processes started by the Crypto Officer when required.

Note: All of the services apply to all of the modes of operation, Approved (NIST FIPS PUB 140-2-Approved) and Non-Approved (PCI PTS HSM v2.0-Approved and Non-Approved).

Note: When the services are executed in Non-Approved modes, they are considered "non-compliant".

2.4.1 Crypto Officer Role

Note: Most of the services invoked from the Human Interface do not present an explicit return code when successfully executed. In such case, the absence of error indication is to be interpreted as an indication of its success.

The Crypto Officer services are invoked from the Human Interface:

Table 2.8: Crypto Officer Services from the Human Interface

Service	Description/Access and CSP
Asymmetric key management	Generate, import and export asymmetric keys.
	Read access: SSH Key, System Key Write access: RSA Public Key, RSA Encapsulation Public Key
Authenticated configuration	Provide security-relevant module configuration and management.
	Read access: SSH Key, System Key

Table 2.8: Crypto Officer Services from the Human Interface (continued)

Service	Description/Access and CSP
Custodian management	Custodian creation, modification and deletion.
	Read access: SSH Key, System Key Write access: Custodian password Read/Write access: CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed
Log management	System log management.
	Read access: SSH Key, System Key
Master Keys management	Master Keys generation, import and export.
	Read access: SSH Key, System Key, Smart card PIN Write access: CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, Old CMM_AES_ENCRYPTION_KEY Key, Old CMM_AES_MAC_KEY Key Read/Write access: AES-256 Master Key, Old AES-256 Master Key, CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed
Card PIN management	Card PIN configuration, generation, printing, translation and verification. Decimalization table management.
	Read access: SSH Key, System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key
Printing configuration	Printer port and printer strings configuration. Not accessible in NIST FIPS PUB 140-2-Approved mode.
	Read access: SSH Key, System Key
System smart cards management	System smart cards management.
	Read access: SSH Key, System Key Write access: Old SCCK Key Read/Write access: SCCK Key, Batch Key, CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed
Symmetric key management	Generate, import, export and delete symmetric keys.
	Read access: SSH Key, System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, Custodian password Read/Write access: AES Encryption Key, AES CMAC Key, AES HMAC Key, AES GCM/GMAC Key, AES Wrap Key, TDES (3 key) Encryption Key, TDES (3 key) CMAC Key, TDES (3 key) HMAC Key, CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed
User management	User creation, modification and deletion.
	Read access: SSH Key, System Key Read/Write access: User password

2.4.2 User Role

The User services are invoked from the Machine Interface:

Table 2.9: User Services from the Machine Interface

Service	Description/Access and CSP
Asymmetric key management	Generate, import and export asymmetric keys.
	Read access: CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key Write access: DSA Private Key, DSA Public Key, ECDSA Private Key, ECDSA Public Key Read/Write access: AES Enc Key, AES Auth Key, CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed, RSA Private Key, RSA Encapsulation Private Key, RSA Public Key, RSA Encapsulation Public Key, DSA Domain Parameters
Card validation code	Generate or verify card validation codes.
	Read access: System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key Read/Write access: AES Enc Key, AES Auth Key
Authenticated configuration	Provide security-relevant module configuration and management.
	Read access: CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, RSA Private Key Read/Write access: AES Enc Key, AES Auth Key
EMV	Generate or verify EMV transaction authorization values. Generate EMV security scripts.
	Read access: System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, TDES (3 key) Encryption Key, TDES (3 key) CMAC Key Read/Write access: AES Enc Key, AES Auth Key
Key encapsulation	Perform key transport using asymmetric keys.
	Read access: System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, RSA Encapsulation Public Key, RSA Private Key, RSA Encapsulation Private Key, RSA Public Key Read/Write access: AES Enc Key, AES Auth Key, AES HMAC Key, AES Encryption Key, AES CMAC Key, AES GCM/GMAC Key, AES Wrap Key, CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed, TDES (3 key) Encryption Key, TDES (3 key) CMAC Key, TDES (3 key) HMAC Key
Keyed hash	Generate or verify data integrity with HMAC.
	Read access: System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, TDES (3 key) HMAC Key, AES HMAC Key Read/Write access: AES Enc Key, AES Auth Key
Log management	System log management.
	Read access: CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key Read/Write access: AES Enc Key, AES Auth Key
Symmetric digest	Generate or verify data integrity with CMAC.

Table 2.9: User Services from the Machine Interface (continued)

Service	Description/Access and CSP
	Read access: System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, TDES (3 key) CMAC Key, AES CMAC Key Read/Write access: AES Enc Key, AES Auth Key
Master Keys information	Master Keys information.
Card PIN management	Card PIN configuration, generation, printing, translation and verification. Decimalization table management.
	Read access: System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, TDES (3 key) Encryption Key Read/Write access: AES Enc Key, AES Auth Key, CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed
Printing configuration	Printer port and printer strings configuration. Not accessible in NIST FIPS PUB 140-2-Approved mode.
Random number generation	Provide random data.
	Read/Write access: CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed
Digital signature	Generate or verify digital signatures.
	Read access: CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, RSA Private Key, RSA Public Key, DSA Private Key, DSA Public Key, ECDSA Private Key, ECDSA Public Key Read/Write access: AES Enc Key, AES Auth Key, CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed
Encryption/Decryption	Perform encryption and decryption.
	Read access: System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, TDES (3 key) Encryption Key, AES Encryption Key, AES GCM/GMAC Key Read/Write access: AES Enc Key, AES Auth Key
Symmetric key management	Generate, import, export and delete symmetric keys.
	Read access: System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, Old CMM_AES_ENCRYPTION_KEY Key, Old CMM_AES_MAC_KEY Key Read/Write access: AES Enc Key, AES Auth Key, CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed, TDES (3 key) Encryption Key, TDES (3 key) CMAC Key, TDES (3 key) HMAC Key, AES Encryption Key, AES CMAC Key, AES HMAC Key, AES GCM/GMAC Key, AES Wrap Key
User information	User information.
Wallet and transport certificate management	Generate TIBC or Advantis wallet and transport certificates. Not accessible in NIST FIPS PUB 140-2-Approved mode.
Key wrapping	Perform key transport using symmetric keys.
	Read access: System Key, CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key

Table 2.9: User Services from the Machine Interface (continued)

Service	Description/Access and CSP
	Read/Write access: AES Enc Key, AES Auth Key, AES Wrap Key, TDES (3 key) Encryption Key, TDES (3 key) CMAC Key, TDES (3 key) HMAC Key, CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed, AES Encryption Key, AES CMAC Key, AES HMAC Key, AES GCM/GMAC Key, RSA Private Key, RSA Public Key, DSA Private Key, DSA Public Key, DSA Domain Parameters

2.4.3 Unauthenticated Services

Machine Interface

The following Unauthenticated services are invoked from the Machine Interface:

Table 2.10: Unauthenticated Services from the Machine Interface

Service	Description/Access and CSP
Unauthenticated configuration	Provide non security-relevant module configuration and management.
	Read access: User password Read/Write access: CTR_DBGR Entropy Input, CTR_DBGR V Value, CTR_DBGR Key, CTR_DBGR Seed
Message digest	Generate message digest.
Information	Provide module information.

Note: The document CryptosecDekatonUserGuide (Revision: 12.11.3642 Date: 14/11/2018) contains detailed information on the Unauthenticated Services invoked from the Machine Interface. To obtain this document, please contact Realia Technologies, S.L.

Human Interface

Note: Most of the services invoked from the Human Interface do not present an explicit return code when successfully executed. In such case, the absence of error indication is to be interpreted as an indication of its success.

The following Unauthenticated services are invoked from the Human Interface:

Table 2.11: Unauthenticated Services from the Human Interface

Service	Description/Access and CSP
Unauthenticated configuration	Provide non security-relevant module configuration and management.
	Read access: SSH Key, System Key, Smart card PIN, Crypto Officer secret

Table 2.11: Unauthenticated Services from the Human Interface (continued)

Service	Description/Access and CSP
	Write access: CMM_AES_ENCRYPTION_KEY Key, Old CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, Old CMM_AES_MAC_KEY Key
Information	Provide module information.
	Read access: SSH Key, System Key
Self test	Perform On Demand Self Tests.
	Read access: SSH Key, System Key
Zeroization	Perform the module zeroization using the Tamper Switch.
	Write access: AES Encryption Key, AES CMAC Key, AES HMAC Key, AES GCM/GMAC Key, AES Wrap Key, TDES (3 key) Encryption Key, TDES (3 key) CMAC Key, TDES (3 key) HMAC Key, System Key, SCCK Key, Old SCCK Key, Batch Key, AES-256 Master Key, Old AES-256 Master Key, CMM_AES_ENCRYPTION_KEY Key, Old CMM_AES_ENCRYPTION_KEY Key, CMM_AES_MAC_KEY Key, Old CMM_AES_MAC_KEY Key, AES Enc Key, AES Auth Key, DSA Domain Parameters, DSA Private Key, DSA Public Key, ECDSA Private Key, ECDSA Public Key, RSA Private Key, RSA Public Key, RSA Encapsulation Private Key, RSA Encapsulation Public Key, Smart Card PIN, Crypto Officer secret, User password, Custodian password, CTR_DRBG Entropy Input, CTR_DRBG V Value, CTR_DRBG Key, CTR_DRBG Seed

Note: The document CryptosecDekatonAdministratorGuide (Revision: 12.11.3642 Date: 14/11/2018) contains detailed information on the Unauthenticated Services invoked from the Human Interface. To obtain this document, please contact Realia Technologies, S.L.

2.5 Physical Security

The physical security mechanisms consist not only on the metallic covers and the epoxy resin. But it also extends to the module design as a whole: the buses are physically isolated.

The metallic covers are made of aluminum and they cover both sides of the PCB, delimiting perfectly the cryptographic boundary. The space between them is completely filled with epoxy resin, making the module even more protected.

The epoxy resin is opaque to the visible spectrum.

2.6 Self-Tests

Self-Tests are formed by:

- Power-up Self-Tests

– Cryptographic Power-up Self-Tests

AES.ECB.128 AES-128 encryption and decryption in ECB mode KAT.
AES.ECB.192 AES-192 encryption and decryption in ECB mode KAT.
AES.ECB.256 AES-256 encryption and decryption in ECB mode KAT.
AES.CBC.128 AES-128 encryption and decryption in CBC mode KAT.
AES.CBC.192 AES-192 encryption and decryption in CBC mode KAT.
AES.CBC.256 AES-256 encryption and decryption in CBC mode KAT.
AES.CTR.128 AES-128 encryption and decryption in CTR mode KAT.
AES.CTR.192 AES-192 encryption and decryption in CTR mode KAT.
AES.CTR.256 AES-256 encryption and decryption in CTR mode KAT.
AES.CMAC.128.length0 AES-128-CMAC KAT.
AES.CMAC.128.length16 AES-128-CMAC KAT.
AES.CMAC.128.length20 AES-128-CMAC KAT.
AES.CMAC.128.length64 AES-128-CMAC KAT.
AES.CMAC.192.length0 AES-192-CMAC KAT.
AES.CMAC.192.length16 AES-192-CMAC KAT.
AES.CMAC.192.length20 AES-192-CMAC KAT.
AES.CMAC.192.length64 AES-192-CMAC KAT.
AES.CMAC.256.length0 AES-256-CMAC KAT.
AES.CMAC.256.length16 AES-256-CMAC KAT.
AES.CMAC.256.length20 AES-256-CMAC KAT.
AES.CMAC.256.length64 AES-256-CMAC KAT.
AES.CBCMAC.128.length16 AES-128-CBCMAC KAT.
AES.CBCMAC.128.length64 AES-128-CBCMAC KAT.
AES.CBCMAC.192.length16 AES-192-CBCMAC KAT.
AES.CBCMAC.192.length64 AES-192-CBCMAC KAT.
AES.CBCMAC.256.length16 AES-256-CBCMAC KAT.
AES.CBCMAC.256.length64 AES-256-CBCMAC KAT.
AES.GCM.128.aad=0,pt=0,tag=128 AES-128 encryption and decryption in GCM mode KAT.
AES.GCM.128.aad=0,pt=512,tag=128 AES-128 encryption and decryption in GCM mode KAT.
AES.GCM.128.aad=512,pt=0,tag=128 AES-128 encryption and decryption in GCM mode KAT.
AES.GCM.128.aad=512,pt=512,tag=128 AES-128 encryption and decryption in GCM mode KAT.
AES.GCM.128.aad=160,pt=480,tag=128 AES-128 encryption and decryption in GCM mode KAT.
AES.GCM.128.aad=160,pt=480,tag=96 AES-128 encryption and decryption in GCM mode KAT.
AES.GCM.192.aad=0,pt=0,tag=128 AES-192 encryption and decryption in GCM mode KAT.
AES.GCM.192.aad=0,pt=512,tag=128 AES-192 encryption and decryption in GCM mode KAT.

AES.GCM.192.aad=512,pt=0,tag=128 AES-192 encryption and decryption in GCM mode KAT.

AES.GCM.192.aad=512,pt=512,tag=128 AES-192 encryption and decryption in GCM mode KAT.

AES.GCM.192.aad=160,pt=480,tag=128 AES-192 encryption and decryption in GCM mode KAT.

AES.GCM.192.aad=160,pt=480,tag=96 AES-192 encryption and decryption in GCM mode KAT.

AES.GCM.256.aad=0,pt=0,tag=128 AES-256 encryption and decryption in GCM mode KAT.

AES.GCM.256.aad=0,pt=512,tag=128 AES-256 encryption and decryption in GCM mode KAT.

AES.GCM.256.aad=512,pt=0,tag=128 AES-256 encryption and decryption in GCM mode KAT.

AES.GCM.256.aad=512,pt=512,tag=128 AES-256 encryption and decryption in GCM mode KAT.

AES.GCM.256.aad=160,pt=480,tag=128 AES-256 encryption and decryption in GCM mode KAT.

AES.GCM.256.aad=160,pt=480,tag=96 AES-256 encryption and decryption in GCM mode KAT.

DES.ECB.SIMPLE.1 DES encryption and decryption in ECB mode KAT.

DES.ECB.DOUBLE.1 TDES (2 key) encryption and decryption in ECB mode KAT.

DES.ECB.TRIPLE.1 TDES (3 key) encryption and decryption in ECB mode KAT.

DES.CBC.SIMPLE.1 DES encryption and decryption in CBC mode KAT.

DES.CBC.DOUBLE.1 TDES (2 key) encryption and decryption in CBC mode KAT.

DES.CBC.TRIPLE.1 TDES (3 key) encryption and decryption in CBC mode KAT.

DES.ECB.TRIPLE.2 TDES (3 key) encryption and decryption in ECB mode KAT.

DES.ECB.TRIPLE.3 TDES (3 key) encryption and decryption in ECB mode KAT.

DES.CBC.TRIPLE.2 TDES (3 key) encryption and decryption in CBC mode KAT.

DES.CBC.TRIPLE.3 TDES (3 key) encryption and decryption in CBC mode KAT.

DES.CMAC.SIMPLE.1 DES-CMAC, KAT.

DES.CMAC.DOUBLE.1 TDES (2 key)-CMAC, KAT.

DES.CMAC.TRIPLE.1 TDES (3 key)-CMAC, KAT.

DES.CMAC.TRIPLE.2 TDES (3 key)-CMAC, KAT.

DES.CMAC.TRIPLE.3 TDES (3 key)-CMAC, KAT.

DES.CMAC.TRIPLE.4 TDES (3 key)-CMAC, KAT.

DES.CMAC.TRIPLE.5 TDES (3 key)-CMAC, KAT.

DES.CMAC.TRIPLE.6 TDES (3 key)-CMAC, KAT.

DES.CMAC.TRIPLE.7 TDES (3 key)-CMAC, KAT.

DES.CMAC.TRIPLE.8 TDES (3 key)-CMAC, KAT.

DSA.1024@64 DSA L=1024 N=160, KAT. Test over 64 bit cores.
DSA.1024@256 DSA L=1024 N=160, KAT. Test over 256 bit cores.
DSA.2048.224@64 DSA L=2048 N=224, KAT. Test over 64 bit cores.
DSA.2048.224@256 DSA L=2048 N=224, KAT. Test over 256 bit cores.
DSA.2048.256@64 DSA L=2048 N=256, KAT. Test over 64 bit cores.
DSA.2048.256@256 DSA L=2048 N=256, KAT. Test over 256 bit cores.
DSA.3072@64 DSA L=3072 N=256, KAT. Test over 64 bit cores.
DSA.3072@256 DSA L=3072 N=256, KAT. Test over 256 bit cores.
ECDSA.B163 ECDSA B163 KAT.
ECDSA.B233 ECDSA B233 KAT.
ECDSA.B283 ECDSA B283 KAT.
ECDSA.B409 ECDSA B409 KAT.
ECDSA.B571 ECDSA B571 KAT.
ECDSA.P192@64 ECDSA P192 KAT. Test over 64 bit cores.
ECDSA.P192@256 ECDSA P192 KAT. Test over 256 bit cores.
ECDSA.P224@64 ECDSA P224 KAT. Test over 64 bit cores.
ECDSA.P224@256 ECDSA P224 KAT. Test over 256 bit cores.
ECDSA.P256@64 ECDSA P256 KAT. Test over 64 bit cores.
ECDSA.P256@256 ECDSA P256 KAT. Test over 256 bit cores.
ECDSA.P384@64 ECDSA P384 KAT. Test over 64 bit cores.
ECDSA.P384@256 ECDSA P384 KAT. Test over 256 bit cores.
ECDSA.P521@64 ECDSA P521 KAT. Test over 64 bit cores.
ECDSA.P521@256 ECDSA P521 KAT. Test over 256 bit cores.
ECDSA.K163@64 ECDSA K163 KAT. Test over 64 bit cores.
ECDSA.K163@256 ECDSA K163 KAT. Test over 256 bit cores.
ECDSA.K233@64 ECDSA K233 KAT. Test over 64 bit cores.
ECDSA.K233@256 ECDSA K233 KAT. Test over 256 bit cores.
ECDSA.K283@64 ECDSA K283 KAT. Test over 64 bit cores.
ECDSA.K283@256 ECDSA K283 KAT. Test over 256 bit cores.
ECDSA.K409@64 ECDSA K409 KAT. Test over 64 bit cores.
ECDSA.K409@256 ECDSA K409 KAT. Test over 256 bit cores.
ECDSA.K571@64 ECDSA K571 KAT. Test over 64 bit cores.
ECDSA.K571@256 ECDSA K571 KAT. Test over 256 bit cores.
ECDSA.BP160@64 ECDSA BP160 KAT. Test over 64 bit cores.
ECDSA.BP160@256 ECDSA BP160 KAT. Test over 256 bit cores.
ECDSA.BP192@64 ECDSA BP192 KAT. Test over 64 bit cores.
ECDSA.BP192@256 ECDSA BP192 KAT. Test over 256 bit cores.
ECDSA.BP224@64 ECDSA BP224 KAT. Test over 64 bit cores.
ECDSA.BP224@256 ECDSA BP224 KAT. Test over 256 bit cores.
ECDSA.BP256@64 ECDSA BP256 KAT. Test over 64 bit cores.
ECDSA.BP256@256 ECDSA BP256 KAT. Test over 256 bit cores.
ECDSA.BP320@64 ECDSA BP320 KAT. Test over 64 bit cores.
ECDSA.BP320@256 ECDSA BP320 KAT. Test over 256 bit cores.

ECDSA.BP384@64 ECDSA BP384 KAT. Test over 64 bit cores.
ECDSA.BP384@256 ECDSA BP384 KAT. Test over 256 bit cores.
ECDSA.BP512@64 ECDSA BP512 KAT. Test over 64 bit cores.
ECDSA.BP512@256 ECDSA BP512 KAT. Test over 256 bit cores.
ECDSA.BPT160@64 ECDSA BPT160 KAT. Test over 64 bit cores.
ECDSA.BPT160@256 ECDSA BPT160 KAT. Test over 256 bit cores.
ECDSA.BPT192@64 ECDSA BPT192 KAT. Test over 64 bit cores.
ECDSA.BPT192@256 ECDSA BPT192 KAT. Test over 256 bit cores.
ECDSA.BPT224@64 ECDSA BPT224 KAT. Test over 64 bit cores.
ECDSA.BPT224@256 ECDSA BPT224 KAT. Test over 256 bit cores.
ECDSA.BPT256@64 ECDSA BPT256 KAT. Test over 64 bit cores.
ECDSA.BPT256@256 ECDSA BPT256 KAT. Test over 256 bit cores.
ECDSA.BPT320@64 ECDSA BPT320 KAT. Test over 64 bit cores.
ECDSA.BPT320@256 ECDSA BPT320 KAT. Test over 256 bit cores.
ECDSA.BPT384@64 ECDSA BPT384 KAT. Test over 64 bit cores.
ECDSA.BPT384@256 ECDSA BPT384 KAT. Test over 256 bit cores.
ECDSA.BPT512@64 ECDSA BPT512 KAT. Test over 64 bit cores.
ECDSA.BPT512@256 ECDSA BPT512 KAT. Test over 256 bit cores.
MD5 MD5 KAT.
RSA.SSA-PKCS1.2048.1@64 RSASSA-PKCS1-v1.5 with 2048-bit key KAT 1. Test over 64 bit cores.
RSA.SSA-PKCS1.2048.1@256 RSASSA-PKCS1-v1.5 with 2048-bit key KAT 1. Test over 256 bit cores.
RSA.SSA-PKCS1.2048.2@64 RSASSA-PKCS1-v1.5 with 2048-bit key KAT 2. Test over 64 bit cores.
RSA.SSA-PKCS1.2048.2@256 RSASSA-PKCS1-v1.5 with 2048-bit key KAT 2. Test over 256 bit cores.
RSA.SSA-PKCS1.3072.1@64 RSASSA-PKCS1-v1.5 with 3072-bit key KAT 1. Test over 64 bit cores.
RSA.SSA-PKCS1.3072.1@256 RSASSA-PKCS1-v1.5 with 3072-bit key KAT 1. Test over 256 bit cores.
RSA.SSA-PKCS1.3072.2@64 RSASSA-PKCS1-v1.5 with 3072-bit key KAT 2. Test over 64 bit cores.
RSA.SSA-PKCS1.3072.2@256 RSASSA-PKCS1-v1.5 with 3072-bit key KAT 2. Test over 256 bit cores.
RSA.SSA-PKCS1.4096.1@64 RSASSA-PKCS1-v1.5 with 4096-bit key KAT 1. Test over 64 bit cores.
RSA.SSA-PKCS1.4096.1@256 RSASSA-PKCS1-v1.5 with 4096-bit key KAT 1. Test over 256 bit cores.
RSA.SSA-PKCS1.4096.2@64 RSASSA-PKCS1-v1.5 with 4096-bit key KAT 2. Test over 64 bit cores.
RSA.SSA-PKCS1.4096.2@256 RSASSA-PKCS1-v1.5 with 4096-bit key KAT 2. Test over 256 bit cores.
RSA.SSA-PSS.2048.1@64 RSASSA-PSS with 2048-bit key KAT 1. Test over 64 bit cores.

RSA.SSA-PSS.2048.1@256 RSASSA-PSS with 2048-bit key KAT 1. Test over 256 bit cores.

RSA.SSA-PSS.2048.2@64 RSASSA-PSS with 2048-bit key KAT 2. Test over 64 bit cores.

RSA.SSA-PSS.2048.2@256 RSASSA-PSS with 2048-bit key KAT 2. Test over 256 bit cores.

RSA.SSA-PSS.3072.1@64 RSASSA-PSS with 3072-bit key KAT 1. Test over 64 bit cores.

RSA.SSA-PSS.3072.1@256 RSASSA-PSS with 3072-bit key KAT 1. Test over 256 bit cores.

RSA.SSA-PSS.3072.2@64 RSASSA-PSS with 3072-bit key KAT 2. Test over 64 bit cores.

RSA.SSA-PSS.3072.2@256 RSASSA-PSS with 3072-bit key KAT 2. Test over 256 bit cores.

RSA.SSA-PSS.4096.1@64 RSASSA-PSS with 4096-bit key KAT 1. Test over 64 bit cores.

RSA.SSA-PSS.4096.1@256 RSASSA-PSS with 4096-bit key KAT 1. Test over 256 bit cores.

RSA.SSA-PSS.4096.2@64 RSASSA-PSS with 4096-bit key KAT 2. Test over 64 bit cores.

RSA.SSA-PSS.4096.2@256 RSASSA-PSS with 4096-bit key KAT 2. Test over 256 bit cores.

RSA.ES-OAEP.2048.1@64 RSAES-OAEP with 2048-bit key KAT 1. Test over 64 bit cores.

RSA.ES-OAEP.2048.1@256 RSAES-OAEP with 2048-bit key KAT 1. Test over 256 bit cores.

RSA.ES-OAEP.2048.2@64 RSAES-OAEP with 2048-bit key KAT 2. Test over 64 bit cores.

RSA.ES-OAEP.2048.2@256 RSAES-OAEP with 2048-bit key KAT 2. Test over 256 bit cores.

RSA.ES-OAEP.3072.1@64 RSAES-OAEP with 3072-bit key KAT 1. Test over 64 bit cores.

RSA.ES-OAEP.3072.1@256 RSAES-OAEP with 3072-bit key KAT 1. Test over 256 bit cores.

RSA.ES-OAEP.3072.2@64 RSAES-OAEP with 3072-bit key KAT 2. Test over 64 bit cores.

RSA.ES-OAEP.3072.2@256 RSAES-OAEP with 3072-bit key KAT 2. Test over 256 bit cores.

RSA.ES-OAEP.4096.1@64 RSAES-OAEP with 4096-bit key KAT 1. Test over 64 bit cores.

RSA.ES-OAEP.4096.1@256 RSAES-OAEP with 4096-bit key KAT 1. Test over 256 bit cores.

RSA.ES-OAEP.4096.2@64 RSAES-OAEP with 4096-bit key KAT 2. Test over 64 bit cores.

RSA.ES-OAEP.4096.2@256 RSAES-OAEP with 4096-bit key KAT 2. Test over 256 bit cores.

RSA.generate.512@64 RSA 512-bit private key generation test. Test over 64 bit cores.

RSA.generate.512@256 RSA 512-bit private key generation test. Test over 256 bit cores.

RSA.generate.1024@64 RSA 1024-bit private key generation test. Test over 64 bit cores.

RSA.generate.1024@256 RSA 1024-bit private key generation test. Test over 256 bit cores.

RSA.generate.1536@64 RSA 1536-bit private key generation test. Test over 64 bit cores.

RSA.generate.1536@256 RSA 1536-bit private key generation test. Test over 256 bit cores.

RSA.generate.2048@64 RSA 2048-bit private key generation test. Test over 64 bit cores.

RSA.generate.2048@256 RSA 2048-bit private key generation test. Test over 256 bit cores.

RSA.generate.2560@64 RSA 2560-bit private key generation test. Test over 64 bit cores.

RSA.generate.2560@256 RSA 2560-bit private key generation test. Test over 256 bit cores.

RSA.generate.3072@64 RSA 3072-bit private key generation test. Test over 64 bit cores.

RSA.generate.3072@256 RSA 3072-bit private key generation test. Test over 256 bit cores.

RSA.generate.3584@64 RSA 3584-bit private key generation test. Test over 64 bit cores.

RSA.generate.3584@256 RSA 3584-bit private key generation test. Test over 256 bit cores.

RSA.generate.4096@64 RSA 4096-bit private key generation test. Test over 64 bit cores.

RSA.generate.4096@256 RSA 4096-bit private key generation test. Test over 256 bit cores.

SHA1 SHA1 KAT.

SHA2.SHA224 SHA224 KAT.

SHA2.SHA256 SHA256 KAT.

SHA2.SHA384 SHA384 KAT.

SHA2.SHA512 SHA512 KAT.

SHA2.SHA512/224 SHA512/224 KAT.

SHA2.SHA512/256 SHA512/256 KAT.

Entropy source Entropy source test.

PRNG CTRAES CTR_DRBG health test.

HMAC.SHA1.KS<BS HMAC with SHA1 (KS<BS) KAT.

HMAC.SHA224.KS<BS HMAC with SHA224 (KS<BS) KAT.

HMAC.SHA256.KS<BS HMAC with SHA256 (KS<BS) KAT.

HMAC.SHA384.KS<BS HMAC with SHA384 (KS<BS) KAT.

HMAC.SHA512.KS<BS HMAC with SHA512 (KS<BS) KAT.
HMAC.SHA512/224.KS<BS HMAC with SHA512/224 (KS<BS) KAT.
HMAC.SHA512/256.KS<BS HMAC with SHA512/256 (KS<BS) KAT.

- Integrity Power-up Self-Tests

  Userland integrity Userland integrity test.
  FPGA integrity FPGA BRAM integrity test.
  Non-volatile storage integrity Non-volatile storage integrity test.

Note: The operator self-tests are formed by exact same test suite than the power-up self-tests.

- Conditional Tests

  - Pair wise consistence tests:

    RSA encryption RSA encryption and decryption comparison test.
    RSA signature RSA signature generation and verification test.
    DSA signature DSA signature generation and verification test.
    ECDSA signature ECDSA signature generation and verification test.

  - Manual entry tests:

    Manual key entry Error detection code.
    Password modification Duplicate entries.
    Smart card PIN modification Duplicate entries.

    Information on these tests can be requested to Realsec.

  - Continuous tests:

    Entropy source continuous Implements testing as described in [FIPS140-2IG, Section 9.8].

2.7 Mitigation of other Attacks

The Cryptosec Dekaton presents the following mechanisms for the mitigation of other Attacks:

- Protection against the generation of PINs that the Crypto Officer has determined to be non-secure.

  A list of non-secure PINs can be loaded into the HSM. The randomly generated PIN candidate is compared against those listed. If it matches any PIN on the list, a new PIN candidate is randomly generated and compared. Otherwise, the PIN candidate is accepted as PIN.

- Tamper detection.

  A number of wires run inside both sides of the module. In case one of them is cut or broken in any way, the tamper mechanism is launched. The tamper mechanism actively erases the data memory.

2.8 Rules of Operation

2.8.1 Secure Administration

Initialization

When the Cryptosec Dekaton is received, the Crypto Officer shall check its cases for evidence of physical tampering. Such indications include prying, bending, or cutting of the metal casing.

After checking the Cryptosec Dekaton for evidence of tampering, the Crypto Officer shall connect the module to the PCIe port on the computer to be used. The console connection shall also be established. The installation files contain all the setup files needed to access the Cryptosec Dekaton.

The Cryptosec Dekaton is delivered initialized is such a way that a challenge mechanism is used to assure that the Cryptosec Dekaton has not been tampered with throughout the delivery process.

Once the challenge has been solved, the module is to be initialized according to the customer's preferences. The initialization process is performed through the terminal console. It is guided by a wizard:

- In order to include the correct time-stamp in the logging information, set the correct date, time, country and timezone.
- When a network interface is not used, leave the IP as 0.0.0.0, so it is not configured.
- Be careful when setting the Operation Mode. It can only be changed when re-initializing.
- The Smart Cards Communication Key is a key factor in the Cryptosec Dekaton security. If a Cryptosec Dekaton is being replicated, use its key. Otherwise, letting the Cryptosec Dekaton generate one instead of loading your own "random" shares is strongly recommended.
- In order to enroll new smart cards, the batch key provided by Realsec is needed. The smart cards come with the default password. It is not possible to change the card's password when enrolling, but it shall be changed before writing sensitive information in the card.
- The administrator card set is a key factor in the Cryptosec Dekaton security. If a Cryptosec Dekaton is being replicated, use its set.
- At least a User is needed in order to access Machine Interface services. If a Cryptosec Dekaton is being replicated, use an existing User. Otherwise, select its credentials with care and keep the password secret.
- The Master Keys are a crucial factor in the Cryptosec Dekaton security. If a Cryptosec Dekaton is being replicated, use the Master Keys from its Master Keys Smart Card Set. Otherwise, let the Cryptosec Dekaton generate the Master Keys and create a Master Keys Smart Cards Set from them.

  Note: The test Master Keys are intended to be used in development or test environments. They shall not be used in production environment.

Management

The Cryptosec Dekaton can be administered using the console and the supplied drivers. This software allows the User to access all the functions supported by the Cryptosec Dekaton, and check its status. Crypto Officer and User guides are available from Realsec.

Once the initialization process has finished, the module can be operated in its normal way.

The Crypto Officer is responsible for:

- Keep track of the Cryptosec Dekaton.
- Routinely check the Cryptosec Dekaton for signs of physical tampering. Such indications include prying, bending, or cutting of the metal casing. The fan is to be removed to check for tamper evidence behind it. If strange activity or damage to the cases is shown, the Crypto Officer shall take the module off-line and investigate.
- Keep up to date the Custodian and User lists.
- Encourage the back-up of the smart card sets and individual cards, and that the back-ups are up to date.

The battery shall be changed every 4 years. It can be done, with care, while the module is working. If it is done in absence of PCIe power supply, the operation should not last more than a minute.

Termination

When the usage of the Cryptosec Dekaton has been completed, it should be zeroized by the Crypto Officer in order to wipe all data. This zeroization should be done by:

- using the appropriate command from recovery mode;

or

- pushing the tamper switch on the PCIe bracket.

Once the data is zeroized, take apart the battery. The module should then be stored in a secure location. The Cryptosec Dekaton's smart cards should be destroyed or reset.

2.8.2 Secure Operation

The User (User, Custodian or card holder) behavior with respect to the secure operation of the Cryptosec Dekaton is mainly related to the secret of the smart card password and the keys or keys components that the User guards. The User should be careful not to provide private keys and secret keys to other parties, nor provide the smart card password to anyone. The User should change regularly the card password.

2.8.3 General Rules

- User and Crypto Officer shall logout their sessions when they finish using the Cryptosec Dekaton.
- The Cryptosec Dekaton enforces that the operator secrets (User passwords and smart card PINs) are formed by eight printable characters.
- It is recommended that for increased security the secret is formed by at least one character of each the following types:
  - Lower case letters.
  - Upper case letters.
  - Numbers.
  - Symbols.
- It shall be checked that the number of the Triple-DES encryptions with the same key is fewer than 2^{16}.

2.9 Modes of Operation Management

2.9.1 Differences between Modes of Operation

Note that the Cryptosec Dekaton has three modes of operation:

- PCI PTS HSM v2.0-Approved mode: PCI PTS HSM v2.0 requirements are enforced.
- NIST FIPS PUB 140-2-Approved mode: NIST FIPS PUB 140-2 requirements are enforced.
- Non-Approved: none of the above requirements are enforced.

Note: In the context of the security requirements of NIST FIPS PUB 140-2 and specifically in this NIST FIPS PUB 140-2 Non-Proprietary Security Policy, the only Approved mode of operation is the NIST FIPS PUB 140-2-Approved mode of operation. The PCI PTS HSM v2.0-Approved mode of operation is to be considered as a Non-Approved mode of operation, and therefore not allowed in the NIST FIPS PUB 140-2-Approved mode of operation. This is due to not only Approved security functions are used in such mode of operation.

The differences between the modes are as follows:

- Commands
  - Change RSA Private Key LMK command is not allowed in PCI PTS HSM v2.0-Approved mode.
  - RSA Encryption and decryption commands are not allowed in NIST FIPS PUB 140-2-Approved mode.

  - DES Key Generation and Printing command is not allowed in NIST FIPS PUB 140-2-Approved mode.
  - DES Key Printing commands are not allowed in NIST FIPS PUB 140-2-Approved mode.
  - AES Key Generation and Printing command is not allowed in NIST FIPS PUB 140-2-Approved mode.
  - AES Key Printing command is not allowed in NIST FIPS PUB 140-2-Approved mode.
  - AES GCM/GMAC encryption/decryption initialization with user provided initialization vector command is not allowed in NIST FIPS PUB 140-2-Approved mode for encryption.

- Algorithms
  - DES is only allowed in non-Approved mode.
  - TDES (2 key) is not allowed in NIST FIPS PUB 140-2-Approved mode.
  - TDES (3 key) where the three sub-keys are unique is enforced in NIST FIPS PUB 140-2-Approved mode.
  - RSA RSAES-PKCS1-v1_5 is not allowed in NIST FIPS PUB 140-2-Approved mode.
  - [FIPS186-4] restrictions on DSA parameters size are not enforced in non-Approved mode.
  - [FIPS186-4] restrictions on RSA modulus size are not enforced in non-Approved mode.
  - [FIPS186-4] restrictions on ECDSA parameters size are not enforced in non-Approved mode.
  - MD5 is not allowed in NIST FIPS PUB 140-2-Approved mode.
  - SHA-1 is not allowed for signature generation in NIST FIPS PUB 140-2-Approved mode.

- PIN translations
  - Encrypted PIN Translation only allows the translation of ISO PIN blocks to ISO PIN blocks in PCI PTS HSM v2.0-Approved mode.
  - Encrypted PIN Translation does not allow translation of ISO0 or ISO3 PIN blocks to ISO1 PIN blocks.
  - Encrypted PIN Translation does not allow the translation of ISO2 PIN blocks to PIN blocks different than ISO2 in PCI PTS HSM v2.0-Approved mode.

- PIN input or output
  - PIN Change Script service does not allow ISO2 input PIN blocks in PCI PTS HSM v2.0-Approved mode.
  - PIN Offset Generation service only allows the generation of ISO0 and ISO3 PIN blocks when using the IBM 3624 PIN Offset or the VISA PVV generation algorithms in PCI PTS HSM v2.0-Approved mode.
  - Encrypted PIN Verification service only allows the verification of ISO0 and ISO3 PIN blocks when using the IBM 3624 PIN Offset or the VISA PVV generation algorithms in PCI PTS HSM v2.0-Approved mode.

  - Encrypted PIN Generation service only allows the generation of ISO0 and ISO3 PIN blocks when using the IBM 3624 PIN Offset or the VISA PVV generation algorithms in PCI PTS HSM v2.0-Approved mode.
  - Clear PIN encryption service only allows for random PIN encryption in PCI PTS HSM v2.0-Approved mode.

- Key input or output
  - Container Version 0 is not allowed in NIST FIPS PUB 140-2-Approved mode.
  - Container Version 0 for AES keys is only allowed in non-Approved mode.
  - Container Version 0 for asymmetric keys for PKCS#11 services is not allowed in PCI PTS HSM v2.0-Approved mode.
  - DES/TDES (2 key and 3 key) wrapping of AES keys is only allowed in non-Approved mode.
  - [SP800-57pt1r4] restrictions on RSA encapsulation of AES keys are not enforced in non-Approved mode.
  - [SP800-57pt1r4] restrictions on DES/TDES (2 key and 3 key) wrapping of RSA keys are not enforced in non-Approved mode.
  - Key binding to its purpose when wrapped or encapsulated is only enforced in PCI PTS HSM v2.0-Approved mode.
  - RSA encapsulation of DES/TDES (2 key and 3 key) for PKCS#11 services is only allowed in non-Approved mode.
  - AES ECB and CBC wrapping of AES keys for PKCS#11 services is not allowed in PCI PTS HSM v2.0-Approved mode.
  - CMM_TRANSPORT_SIGNATURE_KEYS public keys cannot be imported in PCI PTS HSM v2.0-Approved mode.
  - RSA transport of CMM_CUSTODIAN_KEYS TDES (3 key) is not allowed in NIST FIPS PUB 140-2-Approved mode.
  - TDES (2 key and 3 key) and AES wrapping of CMM_TRANSPORT_SIGNATURE_KEYS and CMM_TRANSPORT_KEYS RSA keys is not allowed in PCI PTS HSM v2.0-Approved mode.
  - Only FIPS Approved key transport methods are allowed in NIST FIPS PUB 140-2-Approved mode.

- Key deletion
  - Deletion of CMM_CUSTODIAN_KEYS TDES (3 key) keys from Machine Interface is not allowed in NIST FIPS PUB 140-2-Approved mode.

- LMK-related
  - Generic PKCS#11 LMK (CMM_PKCS11_ENCRYPTION_KEYS) is replaced in PCI PTS HSM v2.0-Approved mode by:
    * CMM_NEW_PKCS11_ENCRYPTION_KEYS for encryption keys,
    * CMM_NEW_PKCS11_SIGNATURE_KEYS for signature keys,
    * CMM_NEW_PKCS11_WRAPPING_KEYS for wrapping keys.

  - CMM_THALES_1 and CMM_RSA_ENCRYPT_SIGN_KEYS are not valid in PCI PTS HSM v2.0-Approved mode.
  - Symmetric keys diversification output LMK shall be the same an the input LMK in PCI PTS HSM v2.0-Approved mode.
  - RSA transport with encryption under CMM_RSA_ENCRYPT_KEYS is not allowed in PCI PTS HSM v2.0-Approved mode, CMM_TRANSPORT_KEYS is used instead.
  - RSA transport with signature under CMM_TRANSPORT_SIGNATURE_KEYS is enforced in PCI PTS HSM v2.0-Approved mode.
  - LMK Keys are not used and not generated in NIST FIPS PUB 140-2-Approved mode.

- Other
  - ARQC and ARPC Calculation and Verification service returns the Cryptosec Dekaton state and the computed ARQC in non-Approved mode.
  - Padding character is not checked against the PIN characters when building an IBM 3624 PIN block in PCI PTS HSM v2.0-Approved mode.
  - Error code returned when extracting an IBM 3624 PIN block and the padding character is not an hex value as expected changes from WRONG_PAD_CHARACTER_ERROR_CODE to WRONG_PINBLOCK_ERROR_CODE in PCI PTS HSM v2.0-Approved mode.
  - When extracting an IBM 3624 PIN block the PIN length computation method takes into account the possibility of the padding character being part of the PIN in PCI PTS HSM v2.0-Approved mode.
  - When extracting a PIN block the PIN digits are not checked in PCI PTS HSM v2.0-Approved mode.
  - User sessions are not enforced in non-Approved mode.
  - Decimalization table encryption and checking are enforced in PCI PTS HSM v2.0-Approved mode.

2.9.2 Selecting the Mode of Operation

The HSM mode of operation is selected as part of the initialization process. Therefore it is necessary to re-initialize the HSM to change its mode of operation.

The HSM is zeroized as a result of the required re-initialization.

Once the initialization process ends, the power-up self-tests are forced.

Every time the HSM starts the selected mode of operation is automatically enforced from the very beginning. It implies that the power-up self-tests are passed in the selected mode of operation. Once the self-tests successfully pass, the selected mode of operation is reached.

Therefore for an HSM whose selected mode of operation as part of the initialization process is NIST FIPS PUB 140-2-Approved, the NIST FIPS PUB 140-2-Approved mode of operation is automatically reached once the power-up self-tests successfully pass.

2.9.3 Reporting the Current Mode of Operation

The Information service and the Authenticated Configuration service inform of the HSM mode.

Appendix A

Glossary of Terms

The following table lists the terms discussed in this security policy and their respective definitions:

Table A.1: Glossary of terms

Term	Definition
AA	Standard size single cell cylindrical dry battery.
AES	Advanced Encryption Standard.
ARPC	Authorization ResPonse Cryptogram.
ARQC	Authorization ReQuest Cryptogram.
CBC	Cipher Block Chaining.
CMM	See LMK.
CSP	Critical Security Parameter.
CTR_DRBG	CounTeR-mode Deterministic Random Bit Generator.
DES	Data Encryption Standard.
DSA	Digital Signature Algorithm.
ECB	Electronic CodeBook.
ECDSA	Elliptic Curve Digital Signature Algorithm.
EMC	ElectroMagnetic Compatibility.
EMI	ElectroMagnetic Interference
EMV	Europay, Mastercard, Visa.
FIPS	Federal Information Processing Standards.
HMAC	keyed-Hash Message Authentication Code.
HSM	Hardware Security Module.
IP	Internet Protocol.
ISO	International Organization for Standardization.
KAT	Known-Answer Test.
LED	Light-Emitting Diode.
LMK	Local Master Key.
MAC	Message Authentication Code.
MD5	Message-Digest algorithm 5.
NIST	National Institute of Standards and Technology.

Table A.1: Glossary of terms (continued)

Term	Definition
PCI	Payment Card Industry security standards.
PCIe	Peripheral Component Interconnect Express.
PIN	Personal Identification Number.
RSA	Rivest, Shamir and Adleman.
SCCK	SmartCards Communication Key.
SCP03	globalplatform Secure Channel Protocol 03.
SHA	Secure Hash Algorithm.
SHS	Secure Hash Standard.
SSH	Secure SHell.
TDES	Triple DES.
USB	Universal Serial Bus.

Bibliography

[FIPS140-2] *FIPS PUB 140-2,* Initial Release: May 2001 - Last Update: December 2002

[FIPS140-2IG] *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program.*

[FIPS180] *FIPS PUB 180,* August 2015

[FIPS186-2] *FIPS PUB 186-2,* january 2000

[FIPS186-4] *FIPS PUB 186-4,* July 2013

[FIPS197] *FIPS PUB 197,* November 2001

[FIPS198-1] *FIPS PUB 198-1,* July 2008

[RFC1321] R. Rivest, *Request for Comments: 1321,* April 1992

[SP800-108] Lily Chen, *NIST Special Publication 800-108,* October 2009

[SP800-132] Meltem Sönmez Turan, Elaine Barker, William Burr, and Lily Chen, *NIST Special Publication 800-132,* December 2012

[SP800-133] Elaine Barker and Allen Roginsky, *NIST Special Publication 800-133,* December 2012

[SP800-135] Quynh Dang, *NIST Special Publication 800-135,* December 2011

[SP800-38B] Morris Dworkin, *NIST Special Publication 800-38B,* October 2016

[SP800-38D] Morris Dworkin, *NIST Special Publication 800-38D,* November 2007

[SP800-38F] Morris Dworkin, *NIST Special Publication 800-38F,* December 2012

[SP800-56Br1] Elaine Barker, Lily Chen and Dustin Moody, *NIST Special Publication 800-56B Revision 1,* September 2014

[SP800-57pt1r4] Elaine Barker, *NIST Special Publication 800-57 Part 1 Revision 4,* January 2016

[SP800-67r1] William C. Barker and Elaine Barker, *NIST Special Publication 800-67 Revision 1,* January 2012

[SP800-90A] Elaine Barker and John Kelsey, *NIST Special Publication 800-90A,* January 2012

[SP800-90B] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry A. McKay, Mary L. Baish and Mike Boyle, *NIST Special Publication 800-90B,* January 2018