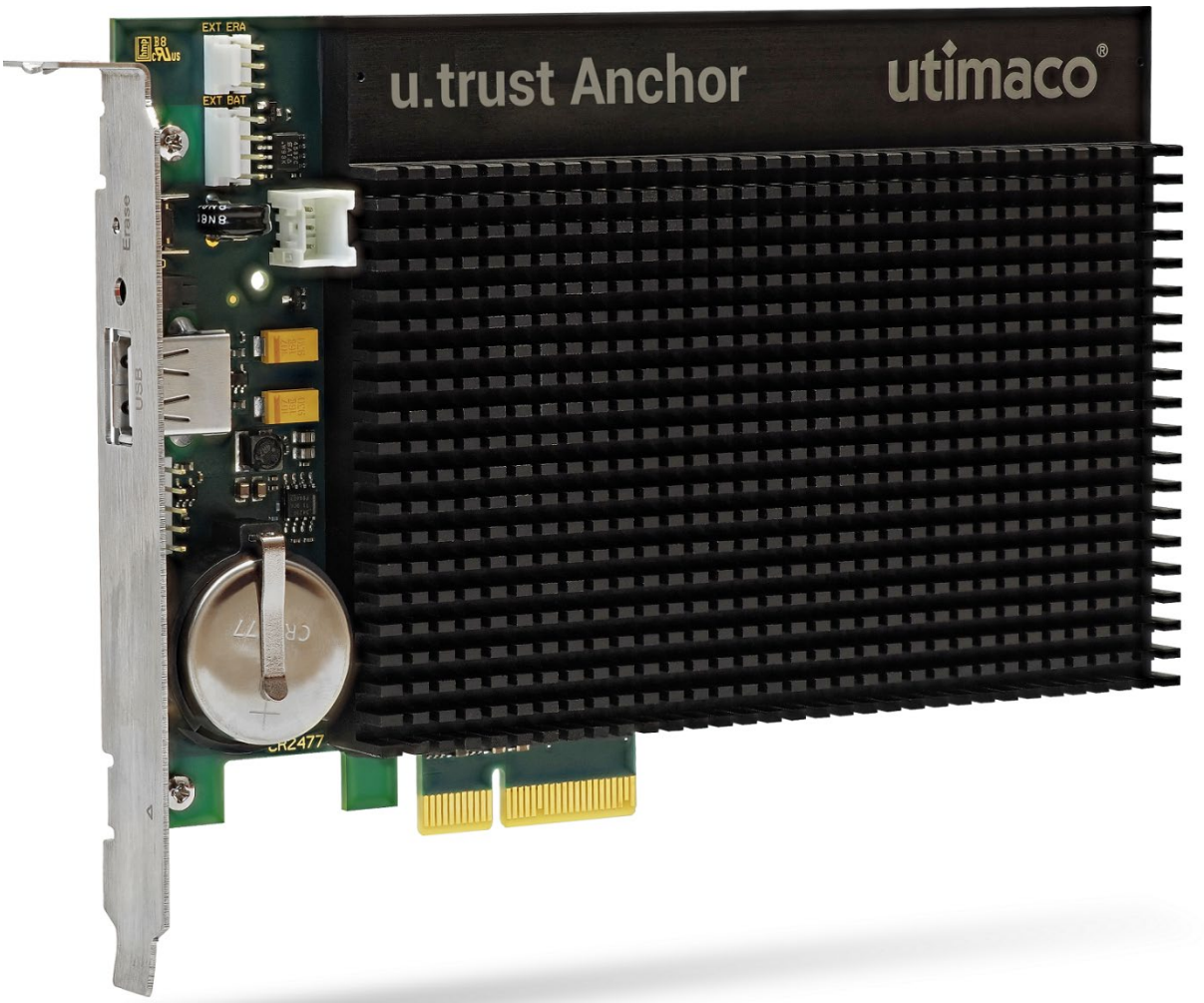

u.trust Anchor

Non-Proprietary
Security Policy

utimaco®

Imprint

Utimaco IS GmbH
Germanusstr. 4
52080 Aachen
Germany

Phone +49 (0)241 / 1696-200

Fax +49 (0)241 / 1696-199

Internet http://hsm.utimaco.com

E-mail hsm@utimaco.com

Document number 2020-0031

Document version 1.0.6

Date 2024-07-29

Status Released

Table of Contents

1 Introduction

This document defines the security policy for Utimaco's u.trust Anchor. u.trust Anchor is a hardware security module made by Utimaco IS GmbH (referred to below also as Utimaco).

u.trust Anchor is suitable for use in a multi-tenant environment where multiple users run their own applications on different containers deployed on the same hardware, as long as the hardware is deployed in a controlled environment.

Table 1 – u.trust Anchor Configuration

Model	HW P/N and Version	FW Version
u.trust Anchor v1.17.4, v4.47.1 and v4.47.2	u.trust Anchor Version 7.03.00.03	Device System v1.17.4, v4.47.1 and v4.47.2 Sensory Controller v3.02.0.7and v3.02.0.8

1.1 Security Level

If run in FIPS mode, the u.trust Anchor meets the overall requirements of FIPS 140-2 Level 3.

Table 2 – Security Level of Security Requirements

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

1.2 Module Description

u.trust Anchor is an encapsulated, protected hardware security module (HSM) realized as a multi-chip embedded cryptographic module as defined in [FIPS140-2].

In general terms, u.trust Anchor is a new generation version of a traditional HSM, comprising all of the traditional hardware security features normally applicable to such a device - but introducing the concept of containerized HSMs (cHSMs).

This new-generation of HSM is developed to improve scalability, both in single, and in multi-tenanted environments (such as data centers or cloud providers) and to deliver any Service Providers with a highly elastic HSM architecture, one that can rapidly scale on demand, but also an architecture that enables the service providers to deliver HSM as a Service (HSMaaS) in multiple use cases.

The goal of the u.trust Anchor platform is to virtualize and allocate shared resources such that each cHSM has visibility only of a resource set that appears to be entirely its own.

Each single cHSM provides secure cryptographic services such as signing and verification of data (like ECDSA and RSA), encryption or decryption (for various cryptographic algorithms like AES and RSA), hashing, on-board random number generation and secure key generation, key storage and further key management functions in a tamper-protected multi-tenant environment. In short, u.trust Anchor allows for the concurrent use multiple types of HSM Firmware within a single hardware that will solve specific problems that address various use cases.

The u.trust Anchor platform consists of the following subsystems:

- The u.trust Anchor hardware
- The u.trust Anchor platform firmware COSMOS: including Boot Loader, Linux kernel, container management firmware and Global Administration service firmware glad
- cHSM (containerized HSM) firmware which can be loaded into containers as provided by COSMOS (e.g. FIPS Approved cHSM firmware)

1.2.1 u.trust Anchor Hardware

The u.trust Anchor hardware is provided in form of a physically protected cryptographic module provided in the form of a PCI Express (PCIe) plug-in card.

The main components of the hardware include a multi-core ARM processor, an internal RAM; 2 GBs of DDR4 RAM (of which, a few MBs are set aside for secure storage of keys)[1]; a non-volatile RAM (NV-RAM) and a flash memory as secondary storage; a cryptographic accelerator with support for RSA and ECC operations; a soft cryptographic accelerator IP block in the FPGA used for acceleration of certain ECC curve operations, and a random number generator (RNG). Secret keys and sensitive data will never be stored unencrypted on NV-RAM and FLASH devices.

[1] Keys stored in Secure RAM are stored unencrypted but are erased if certain extraordinary physical circumstances are detected by internal sensors.

All hardware components of the cryptographic module, including the Central Processing Unit (CPU), all memory chips, Real Time Clock (RTC), and hardware noise generator for random number generation, are located on a printed circuit board (PCI express board). These hardware components are completely covered with potting material (epoxy resin) and heat sink. This hard, opaque enclosure protects the sensitive u.trust Anchor hardware components from physical attacks.

1.2.2 u.trust Anchor Firmware

The u.trust Anchor platform firmware (see Figure 1) consists of:

- A bootloader
- A bespoke Linux kernel compiled with the minimum features necessary to allow the platform to function, and including security components such as mandatory access control, resource control and other sandboxing techniques.
- Custom drivers and services as part of the platform firmware image, to enable communication, for instance, with the random number generator and the cryptographic accelerators;
- The platform operator instance (glad) and the container management middleware
- Individual cHSM firmware templates (there are two templates: FIPS Approved cHSM firmware and general purpose cHSM firmware). Each cHSM template can be loaded into one or more containers as provided by COSMOS. Each container provides the crypto functionality of an HSM in Approved or in non-Approved mode.

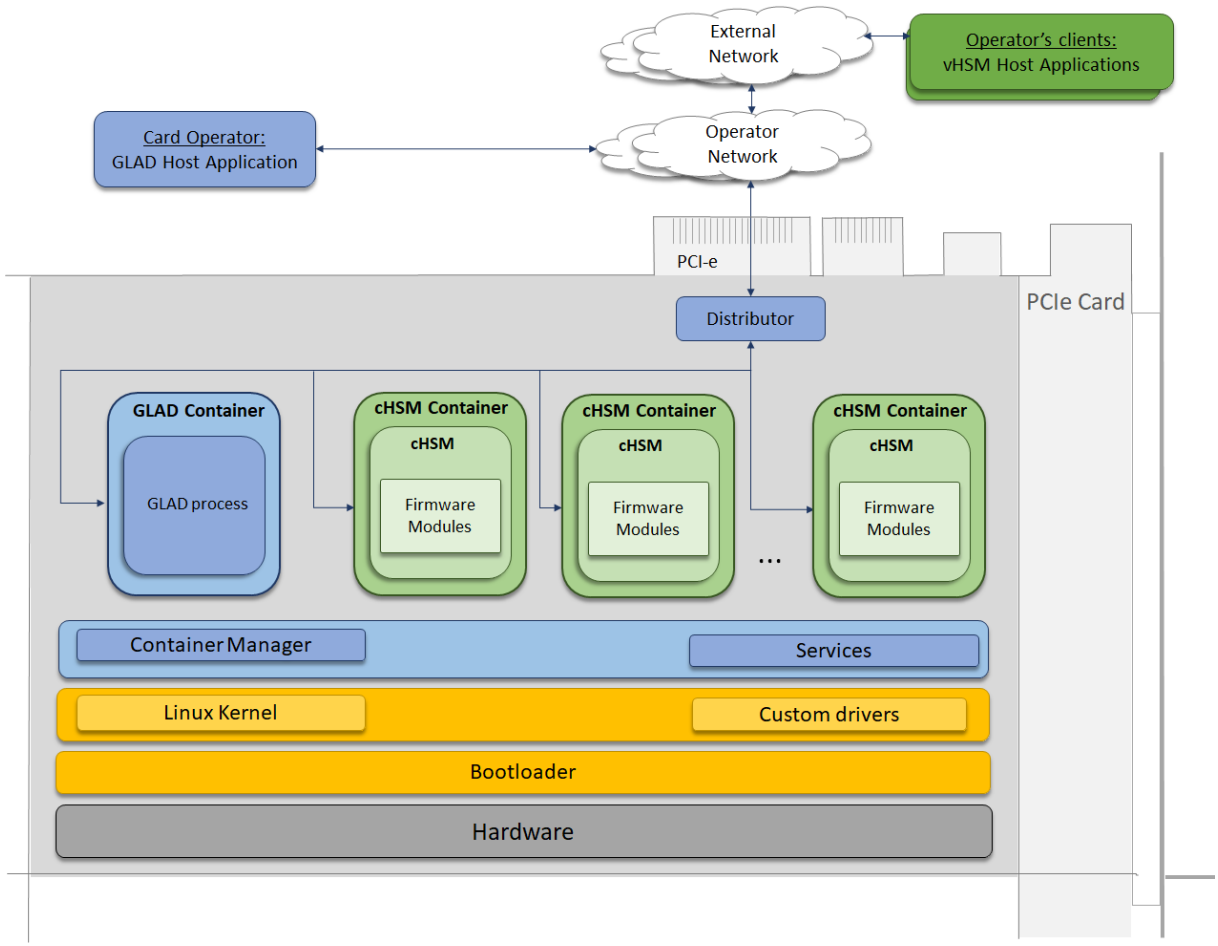

Figure 1: u.trust Anchor Firmware Overview

The u.trust Anchor platform firmware constitutes a limited operational environment. Loaded container firmware cannot be modified and must pass a firmware integrity test on every container start-up.

The u.trust Anchor platform firmware is responsible for the segregation of processes running on different containers: A process running in a container cannot detect, access or modify data belonging to a process running in a different container, or the base operating system.

The containers are isolated from each other and the base operating system by a multi-layered set of technologies (comprising namespaces, mandatory access control and resource controls), allowing multiple cHSM instances to run on a single system.

Management of the containers, including creation, deletion, start, stop, backup and restore of the containers is part of the platform operator role (Global Administrator), and it is expected that this platform operator is a third party. The platform roles and authentication mechanisms are completely separate from the cHSM roles and authentication mechanisms. The platform operator has - by design of the operator roles - no mechanism to access unencrypted data from individual cHSMs.

The cHSM firmware is a collection of firmware components (called modules) instantiated from a cHSM template that provides the required cryptographic functionality like AES, RSA,

ECC, and hashing as well as supporting functionality like key storage and communication with external devices/host applications.

Any process running in a container cannot detect, access or modify any data belonging to a process running in a different container.
Therefore, it is possible to run some cHSMs in FIPS mode and others in non-FIPS mode:

- cHSMs in FIPS mode offer a general-purpose cryptographic API with FIPS Approved algorithms for the above-mentioned cryptographic services, as well as an administrative interface.

- cHSMs in non-FIPS mode can be used in almost all proprietary environments in which cryptographic services and highest security are required, such as archiving systems and payment systems. They can serve as a signature server, time stamp, and generator for PINs, cryptographic keys, or random numbers.

u.trust Anchor offers hardware-based random bit generation (entropy) as well as Approved deterministic random bit generators (DRBG) for glad, for cHSMs in FIPS mode and for cHSMs in non-FIPS mode. The hardware based random bit generation is used to seed and re-seed these DRBGs.

1.2.3 u.trust Anchor Interfaces

For the communication with a host, the PCIe board offers a PCIe interface and a serial log interface. The picture below shows the u.trust Anchor with its PCIe interface:

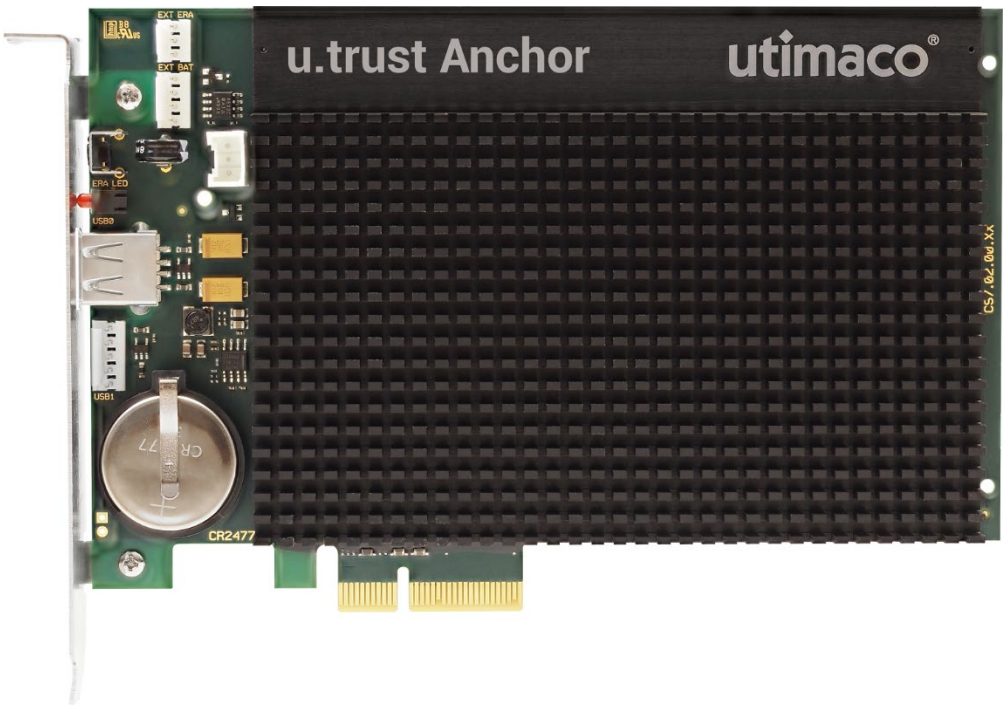

Figure 2 – u.trust Anchor

Together with Utimaco's appropriate host application software the cHSMs also provide cryptographic standard interfaces like PKCS#11, JCE, OpenSSL, CSP/CNG and EKM.

A Secure Messaging concept uses message encryption and MAC authentication to protect communication to and from the cryptographic module – towards the Global Administrator command interface and to all cHSM command interface in FIPS mode and in non-FIPS mode.

1.2.4 u.trust Anchor Environment

The PCIe card should be hosted by a card operator whose operational environment (see figure below)) is assumed trustworthy and secure. The operator may provide remote access to cHSMs to its clients via a network. The external environment (see figure below) is not under the control of the operator.

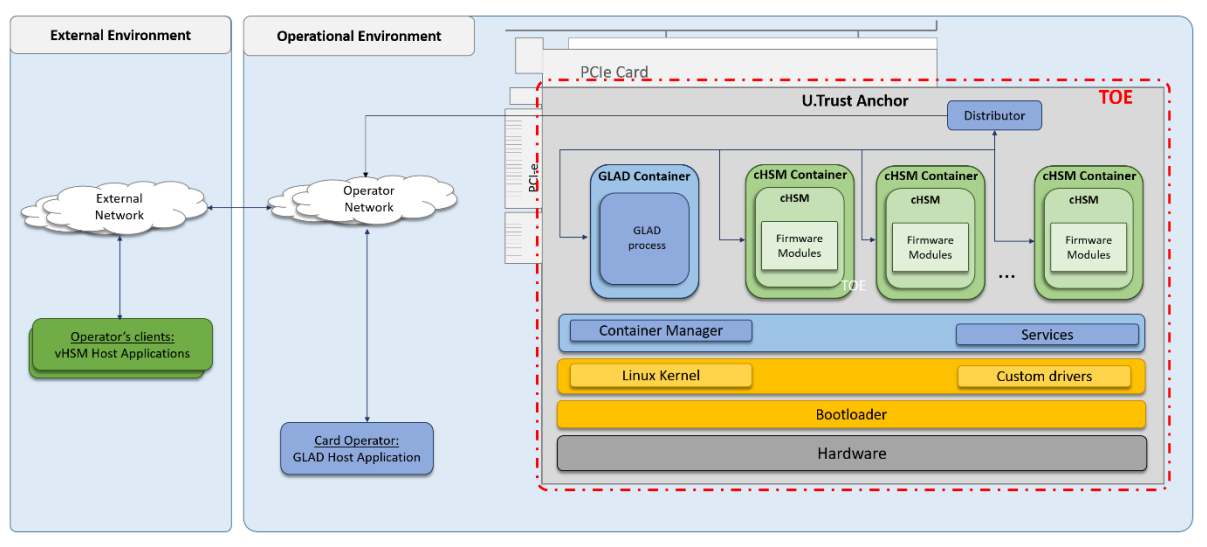

Figure 3: Target of Evaluation (TOE) Boundary and Environment

1.2.5 Cryptographic Boundary

The module's cryptographic boundary is defined as the outer perimeter of the heat sink on the top side and the epoxy surface on the bottom side of the module.

Figure 4 and Figure 5 below show views of the cryptographic boundary from the side and the top, and from the bottom. The red dashed line indicates the cryptographic boundary.

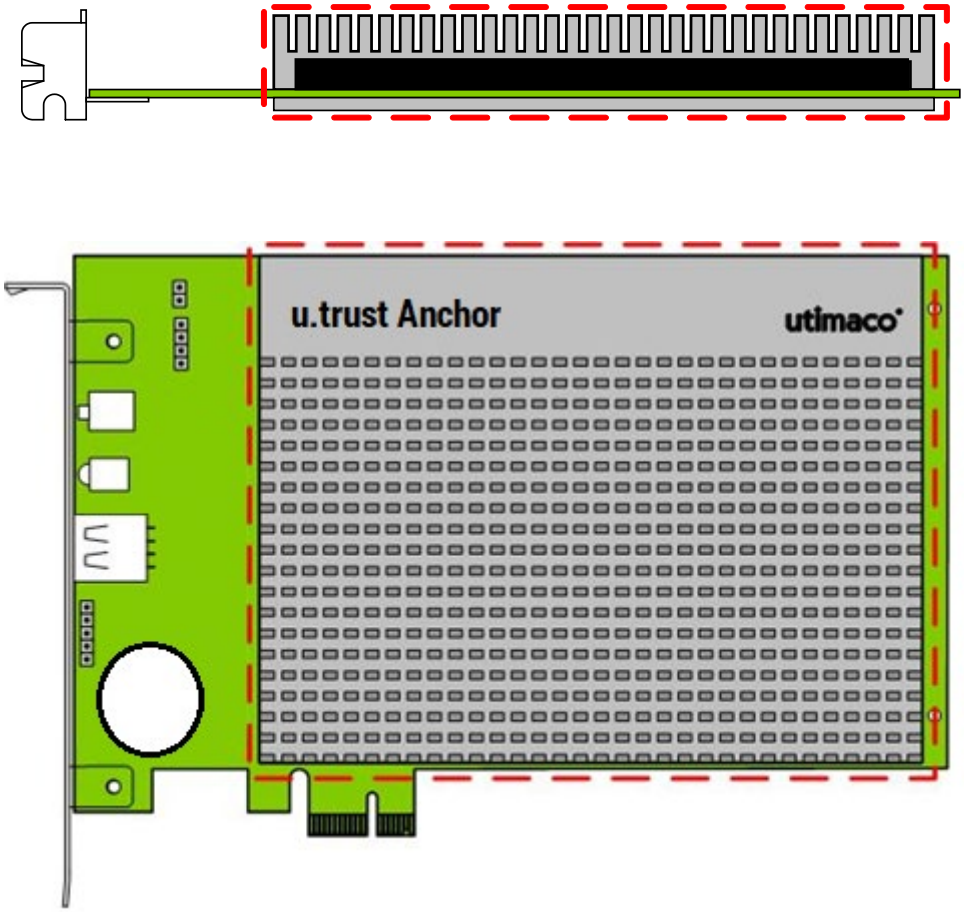

Figure 4 – u.trust Anchor – side view and top view

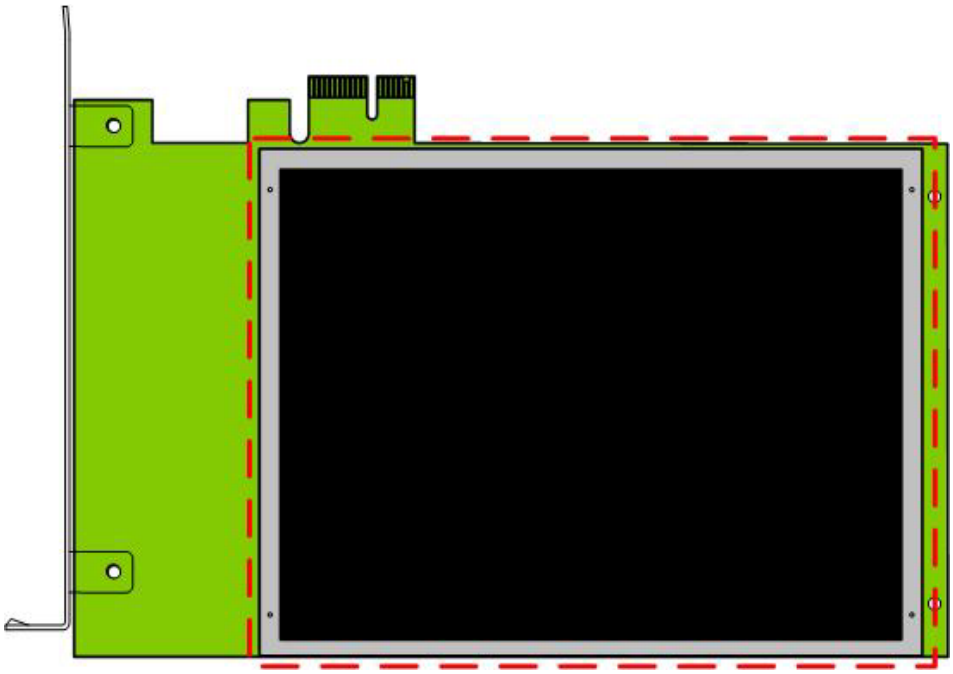

Figure 5 – u.trust Anchor – bottom view

2 Modes of Operation

u.trust Anchor implements an Approved and a non-Approved mode of operation. Each cHSM container can run either the Approved mode cHSM or the non-Approved mode cHSM.

2.1 Approved Mode of Operation

u.trust Anchor implements the FIPS Approved and Non-Approved but Allowed cryptographic algorithms listed in the tables below.

- CAVP Cert. #A1561 is used by the glad services,
- Cert. #A1560 is used by the cHSMs,
- Cert. #A1564 is used for the cHSM start-up integrity test and DRBG implementation,
- and Cert. #A1563 and Cert. #A1562 are used for the module's start-up integrity tests.

Table 3 – Approved and CAVP Validated Cryptographic Algorithms

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
#A1561	AES	FIPS 197; SP 800-38A	CBC CTR	256 128, 256	Data Encryption/ Data Decryption
	AES	SP 800-38B	CMAC	256	Message Authentication (Generation and Verification)
	AES	SP 800-38D	GCM[2]	256	Authenticated Encryption/ Decryption
#A1560	AES	FIPS 197; SP 800-38A	CBC, CFB8, CTR, ECB, OFB	128, 192, 256	Data Encryption/ Data Decryption
	AES	SP 800-38C	CCM	128, 192, 256	Authenticated Encryption and Decryption, Key Transport Scheme
	AES	SP 800-38B	CMAC	128, 192, 256	Message Authentication (Generation and Verification)

[2] The 96 bit IV is randomly generated internally per IG A.5, option 2

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
	AES	SP 800-38D	GCM[2], GMAC	128, 192, 256	Authenticated Encryption/ Decryption, Key Transport Scheme (GCM only)
	AES	SP 800-38F	KW, KWP	128, 192, 256	Key Transport Scheme (Encryption and Decryption)
#A1560	CVL KDF ANS 9.63	SP 800-135; ANSI X9.63	Concatenation SHA-224, SHA-256, SHA-384, SHA-512	128 - 4096	Key Derivation
#A1560	CVL KDF TLS	SP 800-135; TLS 1.2	SHA-256, SHA-384, SHA-512		Key Derivation
#A1561	DRBG	SP 800-90A	Hash DRBG: SHA-512-based		Random Bit Generation
#A1564	DRBG	SP 800-90A	Hash DRBG: SHA-512-based		Random Bit Generation
#A1560	DSA	FIPS 186-4	-	2048/224, 2048/256 or 3072/256	Key Generation
			SHA-224[3], SHA-256, SHA-384, SHA-512	2048/224, 2048/256 or 3072/256	Digital Signature Generation and Domain Parameter Generation
		FIPS 186-4; FIPS 186-2	SHA-1, SHA-224[4] SHA-256, SHA-384, SHA-512	1024/160, 2048/224, 2048/256 or 3072/256	Digital Signature Verification and Parameter Verification
#A1561	ECDSA	FIPS 186-4	-	P-521	Key Generation
	ECDSA	FIPS 186-4	SHA-256	P-256[5], P-521	Digital Signature Generation

[3] Domain Parameter Generation with SHA-224 is only possible for key length 2048/224.

[4] Domain Parameter Verification with SHA-224 is only possible for key lengths 2048/224 and 1024/160, and SHA-1 is only possible with 1024/160. Signature Verification with SHA-1 is only possible with 1024/160.

[5] Called only by the Initiate Self Tests service, per IG G.13 #9

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
	ECDSA	FIPS 186-4	SHA-256	P-256, P-521 Non-NIST (per IG A.2)[6]: brainpoolP320t1	Digital Signature Verification
	ECDSA	FIPS 186-4	-	P-521	Public Key Validation
#A1560	ECDSA	FIPS 186-4		NIST Recommended: See below Non-NIST (per IG A.2)[7]: See below, and curve25519	Key Generation
	ECDSA	FIPS 186-4	SHA-224 SHA-256 SHA-384 SHA-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512	NIST Recommended: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 Non-NIST (per IG A.2)[8]: brainpoolP224r1/ 224t1/ 256r1/ 256t1/ 320r1/ 320t1/ 384r1/ 384t1/ 512r1/ 512t1, secp256k1, FRP256v1	Digital Signature Generation
	ECDSA	FIPS 186-4; FIPS 186-2	SHA-1[9] SHA-224 SHA-256 SHA-384 SHA-512 SHA3-224 SHA3-256 SHA3-384 SHA3-512	NIST Recommended: See above, and P-192, K-163, B-163 Non-NIST (per IG A.2): See above	Digital Signature Verification

[6] Non-NIST-Recommended elliptic curves implemented per IG A.2 are approved per IG A.14, but are not ACVP-testable. Refer to Table 5 for associated security strengths

[7] Non-NIST-Recommended elliptic curves implemented per IG A.2 are approved per IG A.14, but are not ACVP-testable. Refer to Table 5 for associated security strengths

[8] Non-NIST-Recommended elliptic curves implemented per IG A.2 are approved per IG A.14, but are not ACVP-testable. Refer to Table 5 for associated security strengths

[9] SHA-1 is only possible with B-163, K-163, and P-192

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
	ECDSA	FIPS 186-4; FIPS 186-2	-	NIST Recommended: See above, and P-192, K-163, B-163 Non-NIST (per IG A.2): See above	Public Key Validation
#E108	ESV	SP 800-90B	-	Generated entropy: 417 bits Entropy per source output bit: 0.815	Used to generate the seed material for the Approved DRBG. The module does not combine entropy from multiple entropy sources.
#A1561	HMAC	FIPS 198-1	SHA-256	key size >= 256 bits	Message Authentication Generation
				key size >= 256 bits	Message Authentication Verification
#A1560	HMAC	FIPS 198-1	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384, SHA3-512	key size >= 112 bits	Message Authentication Generation
				key size >= 80 bits	Message Authentication Verification
KAS-SSC Cert. #A1561, KDA Cert. #A1561	KAS	SP 800-56Ar3	(Cofactor) Ephemeral Unified Model C(2e, 0s, ECC CDH)	P-521	glad Secure Messaging KAS: used to derive AES CBC and AES CMAC keys for KTS
KAS-SSC Cert. #A1560, KDA Cert. #A1560	KAS	SP 800-56Ar3	(Cofactor) Ephemeral Unified Model C(2e, 0s, ECC CDH)	P-521	cHSM Secure Messaging KAS: used to derive AES CBC and AES CMAC keys for KTS

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
KAS-SSC Cert. #A1560, KDA Cert. #A1560 *or* CVL Cert. #A1560	KAS	SP 800-56Ar3	-	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 (Provides between 112 and 256 bits of encryption strength)	Derive Key KAS (includes different KDFs)
#A1561	KAS-SSC	SP 800-56Ar3	ECC DH primitive (5.7.1.2)	P-256[10] P-521 (Provides 256 bits of encryption strength)	Shared Secret Computation[11]
#A1560	KAS-SSC	SP 800-56Ar3	ECC DH primitive (5.7.1.2)	P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571 (Provides between 112 and 256 bits of encryption strength)	Shared Secret Computation[12]
#A1561	KBKDF	SP 800-108	SHA-256; feedback mode	L=256	Key Derivation[13]
#A1560	KBKDF	SP 800-108	SHA-256; feedback mode	L=256	Key Derivation[13]
#A1561	KDA: NIST One-Step KDF	SP 800-56Cr1	One-step concatenation KDF	HMAC-SHA-256	Key Derivation
#A1560	KDA: NIST One-Step KDF	SP 800-56Cr1	One-step concatenation KDF	HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, HMAC-SHA3-512	Key Derivation

[10] Called only by the Initiate Self Tests service, per IG G.13 #9

[11] With the SP 800-56Cr1 One-Step KDF

[12] Primitive alone or used with the SP 800-56Cr1 One-Step KDF or the SP 800-135 ANSI X9.63 KDF

[13] Used to derive session keys and backup keys.

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
AES Cert. #A1561 and AES Cert. #A1561	KTS	SP 800-38F; FIPS 197; SP 800-38A; SP 800-38B	AES CBC and AES CMAC	Provides 256 bits of encryption strength	Secure Messaging with 256-bit session keys SMEK and SMMK. Key Transport Scheme (keys derived by SP 800-108, key derivation key established by SP 800-56Ar3 and SP 800-56Cr1)
AES Cert. #A1560 and AES Cert. #A1560	KTS	SP 800-38F; FIPS 197; SP 800-38A; SP 800-38B	AES CBC and AES CMAC	Provides 256 bits of encryption strength	Secure Messaging with 256-bit session keys SMEK and SMMK. Key Transport Scheme (keys derived by SP 800-108, key derivation key established by SP 800-56Ar3 and SP 800-56Cr1)
#A1560	KTS	SP 800-38F	AES CCM	Provides between 128 and 256 bits of encryption strength	Key Transport Scheme
#A1560	KTS	SP 800-38F	AES KW, AES KWP	Provides between 128 and 256 bits of encryption strength	Key Transport Scheme
#A1560	KTS	SP 800-38F	AES GCM	Provides between 128 and 256 bits of encryption strength	Key Transport Scheme
#A1561	KTS-RSA	SP 800-56Br2	KTS-OAEP-basic key transport scheme	2048-16384[14] (Provides between 112 and 256[15] bits of encryption strength)	Key Transport Scheme (unencapsulation)

[14] Even key lengths only. ACVP certification covers all testable RSA modulus sizes: 2048, 3072, 4096, 6144, and 8192 per FIPS 186-4, ref. IG A.14.

[15] Per the IG 7.5 key strength calculation

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
#A1560	KTS-RSA	SP 800-56Br2	KTS-OAEP-basic key transport scheme	2048-16384[16] (Provides between 112 and 256[17] bits of encryption strength)	Key Transport Scheme (encapsulation and unencapsulation)
#A1561	RSA	FIPS 186-4	-	2048-16384[18]	Key Generation
			RSASSA- PSS SHA-256	2048-16384[19]	Digital Signature Verification
#A1560	RSA	FIPS 186-4	-	2048-16384[20]	Key Generation
	RSA	FIPS 186-4	ANSI X9.31, PKCS 1.5, PSS SHA-224, SHA-256, SHA-384, SHA-512	2048-16384[21]	Digital Signature Generation
	RSA	FIPS 186-4; FIPS 186-2	ANSI X9.31, PKCS 1.5, PSS SHA-1, SHA-224, SHA-256, SHA-384, SHA-512	1024-16384[22]	Digital Signature Verification
#A1560	SHA-3	FIPS 202	SHA3-224 SHA3-256 SHA3-384 SHA3-512		Message Digest
#A1562	SHA-3	FIPS 202	SHA3-384		Message Digest

[16] Even key lengths only. ACVP certification covers all testable RSA modulus sizes: 2048, 3072, 4096, 6144, and 8192 per FIPS 186-4, ref. IG A.14.

[17] Per the IG 7.5 key strength calculation

[18] Even key lengths only. ACVP certification covers all testable RSA modulus sizes: 2048, 3072, and 4096 per FIPS 186-4, ref IG A.14

[19] Even key lengths only. ACVP certification covers all testable RSA modulus sizes: 2048, 3072 and 4096 per FIPS 186-4, ref. IG A.14.

[20] Even key lengths only. ACVP certification covers all testable RSA modulus sizes: 2048, 3072, and 4096 per FIPS 186-4, ref IG A.14

[21] Even key lengths only. ACVP certification covers all testable RSA modulus sizes: 2048, 3072, and 4096 per FIPS 186-4, ref. IG A.14

[22] Even key lengths only. ACVP certification covers all testable RSA modulus sizes: 1024, 2048, 3072, and 4096 per FIPS 186-4 and 1024, 1536, 2048, 3072, 4096 per FIPS 186-2, ref. IG A.14.

CAVP Cert #	Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
#A1561	SHS	FIPS 180-4	SHA-256, SHA-512		Message Digest
#A1560	SHS	FIPS 180-4	SHA-1, SHA-224, SHA-256, SHA-384, SHA-512		Message Digest
#A1563	SHS	FIPS 180-4	SHA-256		Message Digest
#A1564	SHS	FIPS 180-4	SHA-512		Message Digest (cHSM SMOS)
#A1560	Triple-DES	SP 800-67; SP 800-38A	CBC, ECB	3-key (24 bytes) and 2-key (16 bytes)	Data Decryption

Table 4 - Approved Cryptographic Algorithms: Vendor Affirmed

Algorithm	Standard	Mode/ Method	Key Lengths, Curves or Moduli	Use
CKG	SP 800-133r2		512	Unmodified DRBG output used to derive symmetric keys and seeds for asymmetric private keys. (For glad services and cHSMs)
DSA with SHA-3	FIPS 186-4	SHA3-224 SHA3-256 SHA3-384 SHA3-512	(see DSA entry above)	Digital Signature Generation Digital Signature Verification (for cHSMs)
KDF 135 (X9.63) with SHA-3	SP 800-135; ANSI X9.63	Concatenation KDF	SHA3-224, SHA3-256, SHA3-384, SHA3-512 key lengths: 128 - 4096	Key Derivation
RSA	FIPS 186-4	SHA3-224 SHA3-256 SHA3-384 SHA3-512	(see RSA entry above[23])	Digital Signature Generation Digital Signature Verification (for cHSMs)

The security strength of the Non-NIST Recommended elliptic curves is as follows:

[23] Except for X9.31 padding: RSA sign/verify with X9.31 padding does not support SHA3 hashes.

Table 5 – Security Strength of Non-NIST Elliptic Curves

EC Curve	Security Strength	Reference
brainpoolP224r1 / brainpoolP224t1	112	[ECCBP]
curve25519	128	[RFC 7748]
secp256k1	128	[SEC2]
FRP256v1	128	[ANSSI]
brainpoolP256r1 / brainpoolP256t1	128	[ECCBP]
brainpoolP320r1 / brainpoolP320t1	160	[ECCBP]
brainpoolP384r1 / brainpoolP384t1	192	[ECCBP]
brainpoolP512r1 / brainpoolP512t1	256	[ECCBP]

u.trust Anchor also implements and uses the following non-FIPS Approved but Allowed algorithms:

Table 6 – Non-Approved but Allowed Cryptographic Functions

Algorithm	Standards, Methods	Key Lengths, Curves or Moduli	Caveat	Use
EC Diffie-Hellman (shared secret computation)	IG D.8 scenario X2[24]	Non-NIST (per IG A.2)[25]: brainpoolP224r1/ 224t1/ 256r1/ 256t1/ 320r1/ 320t1/ 384r1/ 384t1/ 512r1/ 512t1, secp256k1, FRP256v1	Provides between 112 and 256 bits of encryption strength	Shared Secret Computation[26] (for cHSMs)
EC Diffie-Hellman (key Agreement)		See above		Key Agreement[27]
RSA (key Wrapping)	SP 800-56Br2 Wrapping with encryption scheme RSAES-PKCS-v1_5 according to [PKCS#1] [28]	2048 - 16384[29]	Provides between 112 and 256 bits of encryption strength	Key establishment methodology (key wrapping and unwrapping) (for cHSMs)
Triple-DES	SP 800-67;	3-key (24 bytes)	--	Legacy Key

[24] As indicated by IG D.8 X2 (b), "the rules of SP 800-56A Rev3 have been followed whenever possible, given that the curves may not be defined in a NIST publication.

[25] Non-NIST-Recommended elliptic curves implemented per IG A.2 are allowed per IG D.8 scenario X2. Refer to Table 5 for associated security strengths

[26] Primitive alone

[27] Shared Secret computation with the SP 800-56Cr1 One-Step KDF or the SP 800-135 ANSI X9.63 KDF.

[28] This key wrapping is considered Allowed per IG D.9.

[29] Even key lengths only

Algorithm	Standards, Methods	Key Lengths, Curves or Moduli	Caveat	Use
(Key Unwrapping)	SP 800-38A	and 2-key (16 bytes)		Unwrapping

u.trust Anchor also implements the following algorithm that may be used in Approved mode of operation, but is not a security function per IG 1.23. In accordance with IG 1.23 example 1, this algorithm is used to obfuscate stored CSPs. This algorithm is not used for security-relevant purposes and no security is claimed from this algorithm.

Table 7 – Non-Approved but Allowed Cryptographic Functions: Non-Security Functions

AES (non-compliant)	AES_CBC-CS3-256, AES_XTS-256 (SP 800-38E)	Obfuscation of encrypted data stored in cHSM file system (by COSMOS)

2.2 Configuration of Approved Mode

Global Administrator operators use the gladm tool to access the Global Administrator Application glad. cHSM administrators use the csadm tool to access their cHSM. The Global Administrator Guide [CsarGladmGuide] and the cHSM Administrator Guide [CSAdmGuide] describe the gladm and csadm commands in more detail.

The Global Administrator Application glad is always operated in FIPS mode. u.trust Anchor cHSMs can be operated in FIPS Approved mode or in non-FIPS Approved mode. The module is initialized and modes of operation can be determined as follows. Commands must be performed by appropriately authenticated operators.

1. Perform the gladm system-info command[30]:

```
$ gladm Dev=<device_address> system-info
Device system version 4.47.2
Sensory Controller software version 3.02.0.8
Hardware revision number 7.03.0.3
UID <device_uuid>
Initial user credentials unchanged
Vendor Secret is present on the device
Vendor DAK Certificate is present on the device
OK
```

The operator should verify that the following line items are included in the output and that the listed version numbers align with these entries:

```
Device system version 4.47.2
Sensory Controller software version 3.02.0.8
Hardware revision number 7.03.0.3
```

Directly after delivery the following line should be included:

[30] The below version numbers belong to u.trust Anchor v4.47.2.

```
Initial user credentials unchanged
```

Especially, the operator should verify that the device system version does not contain 'recovery'. If 'recovery' is indicated, the module is in an error state. If 'recovery' is still indicated after reboot, the device must be returned to Utimaco.

2. List the existing cHSM templates as follows[31]:

```
gladm Dev=<device_address> system-list-templates
SecurityServer 4.47.2
SecurityServer-FIPS 4.47.2
```

Verify that there is

- a template for a FIPS cHSM called SecurityServer-FIPS, and there is
- a template for a non-FIPS cHSM called SecurityServer.

3. Global Administrator users can create cHSMs that operate in Approved mode (FIPS cHSMs), and cHSMs that operate in in non-Approved mode (non-FIPS cHSMs). Log in as Global Administrator user and create cHSMs as described in [CsarGladmGuide] (command create_chsm):

   - In order to create a cHSM in Approved mode, use the FIPS template 'SecurityServer-FIPS'
   - in order to create a cHSM in non-Approved mode, use the non-FIPS template 'SecurityServer'

   The cHSM mode cannot be changed after creation.

4. Each cHSM's mode of operation is indicated by the csadm tool by performing the GetState command:

```
$ csadm Dev=<device address, cHSM slot number> GetState
mode      = Operational Mode
state     = INITIALIZED (0x00140004)
FIPS mode = ON
temp      = ---
alarm     = OFF
…
```

Verify that each cHSM is in Operational mode, in INITIALIZED state, and the alarm state is "OFF".

The addressed cHSM is operated in Approved mode if and only if the following line is contained in the output:

```
FIPS mode = ON
```

2.3 Non-FIPS Modes of Operation

[31] The below version numbers belong to u.trust Anchor v4.47.2.

If a container runs the non-FIPS cHSM firmware 'Security Server', the container provides the following non-FIPS validated algorithms in addition to the Approved cryptographic algorithms as listed above:

Table 8 – Non-FIPS Validated Cryptographic Algorithms in non-FIPS cHSM

Algorithm	Use
RSA (non-compliant)	Key Generation, Sign/Verify, key wrapping (key sizes 512 – 1024)
RSA public key cipher of bulk data (non-compliant)	Encryption/Decryption (key sizes 512-16384)
DSA (non-compliant)	Sign/Verify, DH (|P|/|Q| < 2048/224 and |P|/|Q| >= 2048/224)
EC Cryptography public key cipher of bulk data (ECIES)	Encryption/Decryption (all available curves)
EC Cryptography (non-compliant)	Key Generation, Sign/Verify, Encryption/Decryption, ECDH (key sizes < 224 bits, curve secp224k1, sect239k1, P-192, K-163, B-163)
Curve448	Key Generation, ECDH
Curve25519	ECDH
Edwards25519, Edwards448	Key Generation, Sign/Verify
MD5, MDC-2 or RIPEMD-160	Hashing
Single DES	Encryption/Decryption
TDES, TDES MAC (FIPS 113; SP 800-20) (non-compliant)	Generation, Encryption, Message Authentication
Triple DES ANSI retail MAC (ANSI X9.19: 1986, Financial Institution Retail Message Authentication)	Message Authentication
AES GCM mode (non-compliant to requirements of IG A.5 scenario 3)	Encryption/Decryption and Message Authentication
AES MAC CBC Mode (non-compliant)	Message Authentication
ECC point multiplication according to TR-03111 on P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571, brainpoolP224r1/ 224t1/ 256r1/ 256t1/ 320r1/ 320t1/ 384r1/ 384t1/ 512r1/ 512t1, secp256k1, or FRP256v1	Shared Secret Computation
Several key derivation algorithms as specified in [PKCS#11]: • KDF_ENC_DATA: Derive key using the result of an encryption (chaining mode ECB or CBC) of a given	Key derivation (see [PKCS#11] for details)

Algorithm	Use
text with a base key (DES, AES) (CBC: and given IV) • KDF_HASH: Derive key using the hash value over the key components of a base key (DES or AES or Generic Secret; Hash algorithm must output at least as many bits as are needed for the requested key size within key template). • KDF_ECDH: Derive key according to ANSI X9.63 with ECDH primitive and ANSI KDF on P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571, brainpoolP224r1/ 224t1/ 256r1/ 256t1/ 320r1/ 320t1/ 384r1/ 384t1/ 512r1/ 512t1, secp256k1, FRP256v1, or curve25519 • The hash-based functions (KDF_HASH, KDF_DH, KDF_ECDH_COF and KDF_ECDH) may utilize one of the following hash algorithms: MD5, RipeMD 160, SHA-1. • KDF_XOR_BASE_AND_DATA: Derive key by XORing the key components of a base key (DES, AES, Generic Secret) with given data. • KDF_CAT_BASE_AND_KEY: Concatenate a base key with a second key (both DES, AES, Generic Secret) to derive new key. • KDF_CAT_BASE_AND_DATA: Concatenate a base key (DES, AES, Generic Secret) with given data to derive new key. • KDF_CAT_DATA_AND_BASE: Concatenate given data with a base key (DES, AES, Generic Secret) to derive new key. • KDF_EXTRACT_KEY_FROM_KEY: Extract part of a base key (DES, AES, Generic Secret) to derive new key.	ECDH (key agreement; key establishment; non-compliant below 112 bits of encryption strength) Diffie-Hellman (key agreement; key establishment; non-compliant below 112 bits of encryption strength)

3 Secure Messaging for Secure Communication with u.trust Anchor

u.trust Anchor implements a Secure Messaging concept, which enables any operator to secure their communication with u.trust Anchor glad or with a cHSM over the PCIe interface even from a remote host. With Secure Messaging, commands sent to u.trust Anchor and response data received from u.trust Anchor can be AES CBC encrypted and integrity-protected/signed with an AES CMAC. For glad and for every FIPS mode cHSM, Secure Messaging must be performed for every authenticated command, i.e., for every command that is only available for authenticated users. In a non-FIPS cHSM, Secure Messaging is not mandatory for authenticated commands.

To perform Secure Messaging, the operator must open a Secure Messaging Session. For a Session, two 32-byte AES session keys (Session Encryption key SMEK, Session MAC Key SMMK) are negotiated between u.trust Anchor and host, using (Cofactor) Ephemeral Unified Model EC Diffie-Hellman (P-521) and the SP 800-56Cr1 One-Step KDF as the key establishment technique, with additional key derivation per SP 800-108. For generating its local EC Diffie-Hellman key agreement key SMLK that is needed for the key agreement, u.trust Anchor uses its deterministic random bit generator. Optionally, a Secure Messaging Session with mutual authentication may be requested. In this case u.trust Anchor returns additionally a signature over the answer data which on the host side can be used for authentication of the HSM (glad signs with GAK, and cHSM signs with its CAK) towards the host.

u.trust Anchor glad as well as each cHSM can simultaneously manage multiple sessions (each session authenticated by one or more operators): Each session manages its own session key, which is identified by a session ID. All commands using the same session ID and the same session key are said to belong to one session. In this way, a secure channel is established between the u.trust Anchor and the host application.

4 Ports and Interfaces

The physical interface of the u.trust Anchor consists of 38 printed circuit board tracks, embedded inside the printed circuit board (PCB) and passing the cryptographic boundary to the outer world (see Figure 2). The device provides the following physical ports on these tracks:

- Power input (including operational power input and backup power input).
- Battery Measuring Inputs (Data Input)
- An External Erase button, which acts as a control input and can be used to zeroize all security relevant information inside the module. (Control Input)
- An LED indicating that the External Erase Button is pressed (Status Output)
- A serial status output
- External communication port (PCIe) that is used for data input, data output, control input and status output.
- Two USB interfaces and an SMBUS interface (all unused)

To enable communication with a host, u.trust Anchor supports a PCIe interface and a serial status output interface. All requests for services are sent over the PCIe interface. The serial status output interface is used for status output only. The USB interfaces and the SMBUS interface are not used. All Critical Security Parameters (CSPs) are input and output over the services that are offered over the PCIe interface. In particular, CSPs are entered and output only in a wrapped form: All command and response data (except for status requests) to and from the u.trust Anchor are AES CBC encrypted and AES CMAC authenticated by the Secure Messaging layer. For details, see previous subsection 3 “Secure Messaging for Secure Communication with u.trust Anchor“.

Additionally, all secret or private keys may optionally be exported encrypted with a Key Encryption Key (via e. g. the *Export Key* or *Wrap* services, see section 6.1 “ Roles and Authenticated Services“*)*.

5 Identification and Authentication Policy

5.1 Assumption of Roles

The u.trust Anchor supports the following operator roles:

The Global Administrator is allowed to perform global management services for the whole device (e.g. cHSM instantiation or glad user management). It can be split in arbitrary sub roles on a per-service basis.

The *Cryptographic User* is allowed to perform key management and cryptographic services within a single cHSM.

The *Security Officer* is allowed to perform key group specific administration functions like key group specific user management or key group specific configuration management within a single cHSM.

The *Administrator* is allowed to perform global configuration and user management within a single cHSM.

The *NTP Manager* is allowed to perform time synchronization functions on a single cHSM by using an NTP server over a network.

The *Cryptographic User* role can optionally be split into two different user roles:

A *User* who is allowed to perform cryptographic services like encryption or signing within a cHSM,

A *Key Manager* who is allowed to perform key management services like key generation or key backup/restore within a cHSM.

Additionally, any user is allowed to perform non-sensitive services such as requesting status information without prior authentication.

The cryptographic module uses identity-based operator authentication to enforce the separation of roles. Three authentication methods are supported by the module: Password authentication, ECDSA signature authentication and RSA signature authentication.

For *password-based authentication (cHSM only)* the operator must enter its user name and its password to log in. The user name is an alphanumeric string. The password is a binary string of a minimum of four (4) characters. To prevent the password from being eavesdropped, an HMAC is calculated including authentication data, command data, and a random challenge. The hash algorithm for the HMAC calculation is SHA-256. This HMAC value is sent to the cHSM instead of the password. The cHSM recalculates and checks the HMAC value using the operator's password that is stored inside the cHSM user database.

For *ECDSA signature-based authentication* the user sends an ECDSA signed command containing its user name to authenticate to the cryptographic module (glad as well as cHSM).

For *RSA signature-based authentication* the user sends an RSA signed command containing its user name to authenticate to the cryptographic module (glad as well as cHSM).

Upon correct authentication the role is selected based on the operator's user name. During authentication, session keys SMEK and SMMK are negotiated which is used to secure subsequent service requests by the operator (see the description of the Secure Messaging

concept in section 3). Since the session keys (and session ID) are stored in volatile memory, all information about the authentication and session is lost if the module is powered down.

glad and each cHSM support multiple simultaneous authenticated sessions, each using their own session key for message authentication for the service requests. This ensures the separation of the authorized roles and services performed by each operator.

At the end of a session, the operator can log out, or, after 15 minutes of inactivity, the session key is invalidated inside the cryptographic module.

Table 9 – Roles and Required Identification and Authentication

Role	Type of Authentication	Authentication Data
Cryptographic User (called *User* in [FIPS140-2])	Identity-based operator authentication	User Name and Password or User Name and RSA Signature (1024...16384) or User Name and ECDSA[32] Signature
Key Manager (sub-role of Cryptographic User)		
User (sub-role of Cryptographic User)		
Administrator (called *Crypto Officer* in [FIPS140-2])		
Security Officer		
NTP Manager		
Global Administrator		User Name and RSA Signature (2048 ...16384 bits) or User Name and ECDSA Signature (P-256 or P-521 or brainpoolP320t1)

[32] *NIST Recommended: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571; Non-NIST (per IG A.2): brainpoolP224r1/ 224t1/ 256r1/ 256t1/ 320r1/ 320t1/ 384r1/ 384t1/ 512r1/ 512t1, secp256k1, FRP256v1*

Table 10 – Strength of Authentication Mechanisms

Authentication Mechanism	Strength of Mechanism
Username and Password (minimum 4 characters password chosen from 94 printable ASCII characters)	The probability that a random attempt will succeed or a false acceptance will occur is 1/(94^4), which is less than 1/1,000,000. Due to a correctional delay of 120 milliseconds for every non-successful authentication on a cHSM, there is a maximum limit of 60 * 1000 / 120 = 500 non-successful authentications per minute. This can be stated as allowing only 500 non-successful authentication attempts per minute based on a rate of 120 ms per attempt. Therefore the probability of successfully authenticating to the module within one minute is (less than) 500 * 1/(94^4), which is less than 1/100,000.
RSA Signature (minimum 1024 bit key)	The probability that a random attempt will succeed or a false acceptance will occur is less than or equal to approximately [1/(2^{80})] (according to SP 800-57-Part1 Table 2) which is less than 1/1,000,000. Assuming a duration of at least 1 µs for each signature verification, there is a maximum limit of 60 * 1,000,000 authentication attempts per minute. Therefore, the probability of successfully authenticating to the module within one minute is less than 60 * 1,000,000 * [1/(2^{80})] which is less than 1/100,000.
ECDSA Signature (key size >= 224)	The probability that a random attempt will succeed or a false acceptance will occur is less than or equal to approximately [1/(2^{112})] (according to SP 800-57-Part1 Table 2) which is less than 1/1,000,000. Assuming a duration of at least 1 µs for each signature verification, there is a maximum limit of 60 * 1,000,000 authentication attempts per minute. Therefore, the probability of successfully authenticating to the module within one minute is less than 60 * 1,000,000 * [1/(2^{112})] which is less than 1/100,000.

6 Access Control Policy

u.trust Anchor offers different administration and cryptographic services, some of them require operator authentication (denoted below as authenticated services) and other can be used by all users without any authentication (denoted below as unauthenticated services).

This chapter describes all services provided by u.trust Anchor and identifies the operator roles allowed to access those services (see sections 6.1 and 6.2). Furthermore, it specifies the Critical Security Parameters (CSPs) of u.trust Anchor (see section 6.4) and the public keys stored on it (see section 6.5). Section 6.6 specifies for each user role, the services an operator is authorized to perform within that role and for each service within each role, the type(s) of access to the cryptographic keys and CSPs.

6.1 Roles and Authenticated Services

General definitions:

An *Operator* may be a Global Administrator or a cHSM Operator.

A cHSM Operator may be an Administrator, an NTP Manager, a Security Officer or a Cryptographic User, User or Key Manager.

An *Object* may be a (cryptographic) key, a storage object or a configuration object within a cHSM.

A *Backup Blob* contains an Object. Secret keys (incl. Generic Secrets) and private key parts are always encrypted with the Master Backup Key (back-up key) within a Backup Blob.

Each Object and each cHSM Operator may be assigned to a *Key Group*.

An Object is *Local* if it is assigned to a Key Group; an Object which is assigned to no Key Group is called *Global*.

An Object is *Assigned* to an Operator if both are assigned to the same Key Group, or if the Object is Global.

6.1.1 Authenticated Services on glad

Table 11 – Authenticated Services on glad

glad Role	Authenticated Services
Global Administrator: This role provides all services necessary for cHSM	start_session: start an authenticated Secure Messaging session end_session: Terminate the current Secure Messaging session. This command will fail outside a Secure Messaging session. system_update: Update the device firmware including the Operational Boot Loader, glad firmware, and the cHSM templates. Device firmware must be signed with the Image Signing Key.[33] system_update_status: Display status of system update.

[33] Any firmware loaded into this module that is not shown on the module's certificate, is out of the scope of this validation and requires a separate FIPS 140-2 validation

glad Role	Authenticated Services
management and global user management.	system_set_time: Set the device system time. As a cHSM's system time is based on the device's system time, changing the device's system time also changes the system time of all cHSMs running on the device. system_metrics: Retrieve detailed device system metrics with information on current system activity. system_fetch_log: Fetch the device system log. The system log contains status output that may be used for maintenance and debugging purposes. The parts of the system log that are fetched are deleted. system_set_quorum_requirements: Set the authentication requirements for a set of commands. The authentication requirement must at least be the minimum authentication requirement enforced by the device. system_clear: Clear all system data. This erases the Device Master Key and all DMK protected keys from the device. In addition, all cHSMs are deleted, but not the manufacturer secrets DAK, DAK vendor certificate, vendor base secret and SDMK. system_get_audit_log: Get the system audit log with all available entries. system_trim_audit_log: Delete the system audit log up to and including a specified log entry identified by its hash value. system_list_templates: Retrieve a list of all cHSM templates available on the device. system_reset_alarm: Reset the alarm state if the cause for the alarm is no longer present. Regenerates SDMK, DMK and DAK keypair if they were missing. system_get_quota: Get allowed resources assigned to system services. system_set_quota: Configure allowed resources for system services. slot_get_quota: read configuration on allowed resources assigned to a cHSM slot. slot_set_quota: Configure allowed resources for a cHSM slot. key_get_csr: Get a certificate signing request (CSR) for the Device Authentication Key. key_import_cert: Import an Operator DAK Certificate into the device. Key_get_wrapping_key: Generate (and store) RSA OAEP key, return public part. key_import_operator_secret: Import a new wrapped Operator Secret and mark it as active. key_list_operator_secret: List all stored Operator Secrets, marking the active one as such. Only fingerprints are returned. key_delete_operator_secret: Deletes a stored Operator Secret. user_add: Add a global admin.

glad Role	Authenticated Services
	user_change_credentials: Change authentication credentials (public key) of authenticating global admin. user_delete: Delete a global admin. user_backup_create: Backup the User storage data of the device in an encrypted blob. user_backup_restore: Restore the User storage data from a blob back to the device. First deletes all users and then restores users from the backup. chsm_fetch_log: Fetch the boot log of a cHSM. The boot log is written by the cHSM itself. chsm_snapshot: Take and return a snapshot („Backup“) of a cHSM. The snapshot will be encrypted with a key derived from the Manufacturer Secret and the device's active Operator Secret. chsm_clone: Clone a cHSM from a snapshot. Restore a cHSM from the given snapshot and clone it to all slots specified. The clones will run in cluster mode. This effectively either creates a cHSM cluster or adds instances to an already existing cHSM cluster. chsm_restore: Restore a cHSM from a snapshot to a slot, starting it in regular mode. The snapshot can only be decrypted if both the Manufacturer Secret and Operator Secret used for deriving the encryption key are present on the device. chsm_create: Create a new cHSM from a template. Certificates, keys, and additional configuration to personalize the created cHSM instance may be included in a blob containing cHSM initialization data (init data for short). chsm_halt: Halt the cHSM in the specified slot. chsm_free_slot: Remove a cHSM from the device and clear its associated data. cHSM must have been halted in order for this command to be executed successfully, otherwise this command fails.

6.1.2 Authenticated Services on cHSM

Table 12 – Authenticated Services on cHSM

cHSM Role	Authenticated Services
All cHSM user roles:	Change Operator’s Password or Key: This service changes the password or RSA or ECDSA public key which is used for the operator’s authentication and resets the operator’s counter for consecutive failed authentication attempts. Get Session Key: This service generates a new Secure Messaging session key for secure communication to the module.

cHSM Role	Authenticated Services
	End Session: Terminate a Secure Messaging session by invalidating the relevant session key.
Cryptographic User:	This role provides all cryptographic services, i.e., services for management and use of Assigned private, public and secret keys, hashing services and random number generation. It comprises all services authorized for *Key Managers* and all services authorized for *Users*.
Key Manager: This role provides all key management services.	List Keys: This service outputs the key properties (such as the algorithm, key name, key size, etc.) of all Assigned cryptographic keys and storage objects stored inside the cHSM. Open Key: This service opens an Assigned Object which is stored inside the cHSM and returns a key handle or a Backup Blob containing the Object itself. Get Key Property: This service returns one or more properties (attributes) of an Assigned Object. It can export the public part of a cryptographic key but no secret or private key parts. Set Key Property: This service sets one or more properties (attributes) for an Assigned key or storage object (but no key parts). Backup Key: This service outputs a Backup Blob containing an Assigned key or storage object for back-up purposes. The Backup Blob additionally AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. Restore Key: This service imports a Backup Blob containing the back-up of an Assigned key or storage object into the cHSM. Optionally the key or storage object can also be exported within a Backup Blob. All Backup Blobs are additionally AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. Delete Key: This service deletes an Assigned key or storage object from the module. Generate DSA Parameters: This service generates a DSA Domain Parameter set P, Q and G using the DRBG. Generate DSA Parameters PQ: This service generates a DSA Domain Parameter set P and Q using the DRBG. Generate DSA Parameters G: This service generates a DSA Domain Parameter G by given P and Q (optionally) using the DRBG. Compute Hash: This service calculates a SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 or SHA-3 hash or HMAC value for given data or for the components of an Assigned key. Generate Key: This service generates a cryptographic key (AES, RSA, DSA, EC or Generic Secrets) using the DRBG. On request, the generated key is not stored but exported within a Backup Blob.

cHSM Role	Authenticated Services
	Export Key: This service outputs an Assigned cryptographic key. The exported key is AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. Import Key: This service imports a cryptographic key into the cHSM. The key must be AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. On request the imported key can be exported again. Generate Key Pair: This service generates a cryptographic key pair (RSA, DSA or EC) using the DRBG and stores the two key parts in different key objects. On request the generated key parts are not stored but exported within two Backup Blobs. Derive Key: This function derives an AES key or a Generic Secret from an Assigned base key (DSA or EC). The derived key or secret is stored in the cHSM, or exported within a Backup Blob. Split Key: This function cuts keying material (stored as a Generic Secret) in non-overlapping AES keys and/or Generic Secrets. The original key is deleted from the database, the derived keys are stored in the cHSM, or exported within a Backup Blob. Wrap Key: This function exports an Assigned key in form of a key blob, which is formatted as required by PKCS#11 (see [PKCS#11]). The key blob is additionally AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. Unwrap Key: This function imports an Assigned key from an encrypted key blob. The key is encoded as specified by PKCS#11 (see [PKCS#11]). The key blob is additionally AES CBC encrypted and authenticated with an AES CMAC by the current Secure Messaging session. Create Object: This function creates an Assigned cryptographic key or storage object according to the given property list. The created object is either stored within the cHSM or exported within a Backup Blob. Copy Object: This function copies an Assigned key or storage object. A template may be given that contains an additional list of properties which should be added to the original properties or replace existing properties. The copied object is either stored within the cHSM or exported within a Backup Blob.
User:	List Keys: This service outputs the key properties (such as the algorithm, key name, key size, etc.) of all Assigned keys and storage objects stored inside the cHSM. Open Key: This service opens an Assigned Object which is stored inside the cHSM and returns a key handle or a Backup Blob containing the Object itself. Get Key Property: This service returns one or more properties (attributes) of an Assigned Object. It can export the public part of a key but no secret or private key parts.

cHSM Role	Authenticated Services
This role provides all cryptographic services, i.e., services for use of private, public and secret keys, hashing services and random number generation.	Generate DSA Parameters: This service generates a DSA Domain Parameter set P, Q and G using the DRBG. Generate DSA Parameters PQ: This service generates a DSA Domain Parameter set P and Q using the DRBG. Generate DSA Parameters G: This service generates a DSA Domain Parameter G by given P and Q (optionally) using the DRBG. Generate Random Number: This service generates a random number using the DRBG. Crypt Data: This service encrypts or decrypts data using an Assigned Triple-DES or AES key in CBC or ECB mode (Triple-DES, decryption only) or in ECB, CBC, OFB, CTR, GCM, CCM mode (AES). Sign Data: This service generates an RSA, DSA or ECDSA signature or calculates an AES CMAC, AES GMAC or HMAC for given data with an Assigned signing key. Verify Signature: This service verifies an RSA, DSA or ECDSA signature or a Triple-DES MAC, AES CMAC, AES GMAC or HMAC using an Assigned verification key. Compute Hash: This service calculates a SHA-1, SHA-2 or SHA-3 hash or HMAC value for given data or for the components of an Assigned key.
Administrator: This role provides all services necessary for firmware and user management.	Add User: This service adds an cHSM operator (any role) to the cHSM. Delete User: This service deletes an cHSM operator (any role) from the cHSM. Add Group User (for Security Officer): This service adds a *Security Officer* to the cHSM. Delete Group User (for Security Officer): This service deletes a *Security Officer* from the cHSM. Backup User: This service exports all user account data for a given cHSM operator for backup purposes. All secrets (passwords) are encrypted in the exported data with the Master Backup Key and additionally AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. Restore User: This service creates a new cHSM operator in the user database. All information about the user (name, permission, authentication token, etc.) is taken from a backup data block that was output by the *Backup User* service and which is additionally AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. List Master Backup Keys: This service outputs information (key type, key size, key check value, etc.) about all Master Backup Keys (back-up keys) that are stored inside the cHSM. Generate Master Backup Key: This service generates and outputs a Master Backup Key (back-up key). The key is only exported in a wrapped form, AES CBC encrypted and authenticated with an AES CMAC by the

cHSM Role	Authenticated Services
	current Secure Messaging session. The generated key is not stored inside the cHSM. Import Master Backup Key: This service imports a Master Backup Key (back-up key). The key is only imported if AES CBC encrypted and authenticated with an AES CMAC by the current Secure Messaging session. Load File: This service loads files. If a file with the same file name is currently loaded, that current file will be replaced. This command cannot be used to load firmware modules. Delete File: This service is used to delete files. Clear Audit Log: This service deletes the audit log file except for the first 'k' parts. Clear Audit Log Files: This service deletes audit log files up to the given file number 'n'. Optionally it can be checked before, if the youngest file to be deleted has not changed compared to the latest audit log file that was read out. Generate Audit Log Key: This service generates and stores an (RSA or ECDSA) Audit Log Signature Key which may be used for signing audit log files with function 'Get Signed Audit Log'. Get Signed Audit Log: This service returns the requested audit log file, signed with the Audit Log Signature Key. List DB Search Keys: This service returns all search keys of a given database. Export DB Entry: This service exports a given database entry encrypted by the cHSM's Master Backup Key. Import DB Entry: This service imports an encrypted database entry created by the function Export DB Entry. Set Maximum Failure Counter: This service sets the maximum number of allowed consecutive failed authentication attempts before a user is blocked. Set Administration-Only Mode: This service switches the cHSM into Administration-Only Mode (all cryptographic services are blocked, only administrative services are available) or back to the Operational Mode. Set Startup Mode: This function configures the startup mode of the cHSM. Set Time, Set Time Rel: These services are used to set the internal clock on the cHSM. List Keys (for the Global configuration object): This service lists the Global configuration objects.

cHSM Role	Authenticated Services
	Open Key (for configuration objects): This service opens a configuration object and returns a reference, or the configuration object itself is exported. Get Key Property (for configuration objects): This service returns one or more configuration properties. Set Key Property (for the Global configuration object): This service sets one or more Global configuration properties. Backup Key (for the Global configuration object): This service outputs the Global configuration object for back-up purposes. The backup blob is AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. Set Config Param: sets the configuration for auditing. Restore Key (for the Global configuration object): This service imports the back-up of the Global configuration object into cHSM. The backup blob is AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. Delete Key (for the Global configuration object): This service deletes all Global configuration values by setting them to their default values. Load Certificate: import a customer certificate for the CAK.
Security Officer: This role provides all services necessary for Key Group specific user and configuration management.	Add Group User (for a *Cryptographic User, Key Manager* or *User*): This service adds a *Cryptographic User, Key Manager* or *User* to the cHSM. The added operator and the authorizing *Security Officer* must be assigned to the same Key Group. Delete Group User (for a *Cryptographic User, Key Manager* or *User*): This service deletes a *Cryptographic User, Key Manager* or *User* from the cHSM. The deleted operator and the authorizing *Security Officer* must be assigned to the same Key Group. List Keys (for Local configuration objects): This service lists all Assigned Local configuration objects. Open Key: This service opens an Assigned Object and returns a reference or a Backup Blob containing the Object itself. Get Key Property: This service returns one or more properties (attributes) of an Assigned Object. It can export the public part of a key but no secret or private key parts. Set Key Property: This service allows the Security Officer to set a Local configuration value, or to set the TRUSTED attribute of an Assigned key encryption key. Backup Key (for Local configuration objects): Output an Assigned Local configuration object for backup purposes. The backup blob is AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging.

cHSM Role	Authenticated Services
	Restore Key (for Local configuration objects): Import the backup copy of a Local configuration object into the cHSM. The backup blob is AES CBC encrypted and authenticated with an AES CMAC by Secure Messaging. Delete Key (for Local configuration objects): Delete an Assigned Local configuration object by setting all configuration attributes to their default values. Init Key Group: Delete all Local Objects belonging to a given Key Group.
NTP Manager: This role provides all services necessary for NTP time synchronization on the cHSM by using an NTP server over a network	Change Activation State: Change the state of the NTP firmware module from deactivated to activated and vice versa. Set NTP Settings: Allow setting the NTP attributes `MaxAdjustPerOperation` and `MaxAdjustPerDay` for the maximum time adjustment that can be performed with the 'Set Time Delay' function. Set Time NTP: This service sets the time of the cHSM.

6.2 Unauthenticated Services

In addition to the services requiring operator authentication, the u.trust Anchor supports the following unauthenticated services available to any operator without any authentication required.

Unauthenticated glad services:

auth_init: Get challenges and mechanisms for user authentication.

system_info: Retrieve device status information and FIPS mode indicator.

system_monitor: Retrieve global device system metrics.

system_get_time: Get the device system time. As each cHSM may specify its own time offset, the device system time does not necessarily equal the system time of a cHSM currently running on the device.

user_list: Get information about all global admins.

chsm_list_slots: Get basic information about all cHSM slots and their cHSMs. This includes cHSM slots that are currently free, i.e. not occupied by a cHSM.

system_get_quorum_requirements: Get the authentication requirements for command calls enforced by the device.

Initiate Self Tests: At any time, the execution of all self-tests required by FIPS 140-2 can be forced by performing a reset of the module. During self-test execution, no further command processing is possible.

External Erase: Zeroize the Device Master Key and all CSPs in volatile storage (Secure RAM) by pressing the External erase button. As a result, invalidate all data encrypted with the Device Master Key or the Container Master Key (which is stored in Secure RAM).

system_clear_to_factory_defaults: like external_erase, but additionally deletes the user credentials and reinstalls the default global admin.

Zeroize (Alarm): Zeroize all CSPs in volatile storage, the Device Master Key, and the Sticky Device Master Key. As a result, invalidate all data encrypted with the Device Master Key, the Sticky Device Master Key, and the Container Master Key (which is stored in Secure RAM). Called by removing power from the module.

If the device is in FIPS error state, the only services that are available are a small subset of these unauthenticated services. These services only output status information and do not perform any cryptographic services.

Unauthenticated cHSM services:

Restart_cHSM: restart the cHSM and initiate all cHSM self-tests. During self-test execution, no further command processing within this cHSM is possible.

Get Boot Log: Retrieve a log file containing log messages written by the operating system and other firmware modules (or by the boot loader if the command is called in bootloader mode) during the boot process.

Show Status (or "GetState"): View the current status of the cHSM, including the FIPS mode indicator.

Get Time: Read out the current time of the internal Real Time Clock of the cHSM.

Get Maximum Fail Count: Output the maximum number of allowed consecutive failed authentication attempts before a user is blocked.

Get Startup Mode: Show the startup mode of the cHSM.

Get HSM Auth Key: Retrieve the public part of the cHSM-individual HSM Authentication Key. This key may be used for mutual authentication. On first execution of the service, the HSM Authentication Key is generated.

Get Audit Log Key: Retrieve the public part of the Audit Log Signature Key.

List Files: Retrieve a list of all files stored in the cHSM.

List Active Modules: List all currently active firmware modules.

List Users: Read a list of all Security Officers, Cryptographic Users, Key Managers, Users, NTP Managers and Administrators.

Get User Info: Retrieve all non-sensitive information about the specified operator.

Get Audit Log: Read an audit log file.

Get Config Param: Read the configuration for auditing.

Get Memory RAM Info: Return statistical information about the cHSM memory usage.

Echo: Communication test (echo input data).

Get Challenge: Generate and output a challenge (16 bytes random value generated by the cHSM's deterministic random bit generator) for using the challenge/response mechanism in the next authenticated command.

Get Authentication State: Return the current authentication state and an optional list of all operators that are authenticated within the current session.

Get CXI Info: Return some status information about the CXI firmware module, for example, module version number or the fill level of the database.

Set Time Delay: Adjust the cHSM time (delta to u.trust Anchor system time) by a given number of seconds and milliseconds. The relative time change cannot exceed the limits given by the `MaxAdjustPerOperation` and `MaxAdjustPerDay` NTP attributes.

Get NTP Settings: Return the current settings of the `MaxAdjustPerOperation` and `MaxAdjustPerDay` NTP attributes.

P11 Permissions: Return information about the roles regarding users who are currently logged in to the cHSM (defined according to [PKCS#11]: Cryptographic User, Security Officer and Key Manager), restricted to users matching the specified Key Group.

If the cHSM is in FIPS error state, the only services that are available are a small subset of these unauthenticated services. These services only output status information and do not perform any cryptographic function.

6.3 Services in Non-FIPS Modes

The non-FIPS cHSM provides the following additional services compared to the FIPS cHSM:

- 'List Registered Functions' (unauthenticated cHSM service)
- Authenticated commands can be executed without Secure Messaging
- Additional authentication mechanisms for cHSM operators:
    - ECDSA signature authentication (all available curves are allowed)
    - All implemented hashing algorithms are allowed
- cHSM Command Get Session Key can be used without authentication.
    - As a result, end_session may also be executed without authentication.

6.4 Definition of Critical Security Parameters (CSPs)

The following CSPs are used in the module:

Name	*Abbr.*	*Type*	*Usage*
Directly protected by the sensory:			
Sticky Device Master Key	SDMK	AES CBC 32 bytes	Encryption of basic device secrets
Device Master Key	DMK	AES CBC 32 bytes	Encryption of all other device secrets

Name	*Abbr.*	*Type*	*Usage*
Encrypted with the Sticky Device Master Key SDMK:[34]			
Vendor Base Secret	VBS	256 bit generic secret	Derivation of container backup keys
Device Authentication Key	DAK	NIST P-521 based ECDSA	Generation of device individual certificates
Encrypted with the Device Master Key DMK:[34]			
glad Authentication Key	GAK	NIST P-521 based ECDSA	Device Authentication for Secure Messaging
Operator Base Secret	OBS	256 bit generic secret	Derivation of container backup keys
Operator Base Secret Encryption Key	OBSEK	RSA-OAEP, min. 2048 bits	Wrapped import of Operator Base Secret
Container Base Key	CBK	AES 256	Key derivation of Container Master Key
Volatile Storage only:			
Container Master Key	CMK	AES CBC 256	Encryption of secrets within cHSM container
Snapshot (Container Backup) Keys:			
Container Base Key Encryption Key	CBKEK	AES-CTR- 256 with random IV	key wrapping of CBK in snapshot
Snapshot Encryption Key	SEK	AES-CTR 128 with random IV	Encryption of the cHSM container in snapshot
Snapshot MAC Key	SMK	AES 256 CMAC	Integrity protection of cHSM snapshot
User Snapshot Encryption Key	USEK	AES-CTR 128 with random IV	Encryption of user backup
User Snapshot MAC key	USMK	AES 256 CMAC	Integrity protection of user backup
Secure Messaging Keys:			
Secure Messaging Local Diffie Hellman Key	SMLK	ECDSA for curve NIST-P521	generated by the glad as well as each cHSM and used to establish a shared session key derivation key via EC Diffie Hellman for Secure Messaging, see section 3

[34] Note: These non-volatile CSPs are not subject to the zeroization requirement since they are stored in encrypted form (using the AES algorithm).

Name	*Abbr.*	*Type*	*Usage*
Secure Messaging Session Key Derivation Key	SMKDK	32 bytes generic secret	established according to [NIST SP 800-56A r3] using the EC Diffie Hellman algorithm and used to derive Session Keys for Secure Messaging, see section 3
Session Keys	SMEK *and* SMMK	32 bytes AES CBC and CMAC	derived from the *Session Key Derivation Key* SMKDK and used for Secure Messaging, see section 3
DRBG Secrets S_{DRBG} used by the Deterministic Random Bit Generator (DRBG) of glad and of each cHSM as specified in [NIST 800-90A]:			
Entropy input	S_{DRBG_EI}	raw	generated by Entropy source
Seed	S_{DRBG_SEED}	raw	calculated from Entropy input S_{DRBG_EI}
Working state constant	S_{DRBG_C}	raw	calculated from the Seed S_{DRBG_SEED}
Working state value	S_{DRBG_V}	raw	initially calculated from the Seed S_{DRBG_SEED} and updated each time the DRBG is called
encrypted with the Container Master Key CMK[35] :			
cHSM Operator Password	PSW_{AUTH}	Password (minimum 4 characters)	For cHSM operator authentication
Container Authentication Key	CAK	ECDSA P521	Container Authentication for Secure Messaging
HSM Authentication Key	HAK	RSA 3072 bit	HSM Authentication for Secure Messaging; alternative to CAK
Master Backup Key	MBK	AES CBC 16, 24 or 32 bytes	key for back-up purposes of certain cHSM data
HSM Audit Log Signature Key	Private K_{AL_PRIV}	NIST P-256 based ECDSA or 3072-bit RSA	Key for signing audit log
cHSM User Keys:			
RSA User Key	$K_{USR\_RSA\_PRIV}$	RSA	Signature Generation, Key Decryption
DSA User Key	$K_{USR\_DSA\_PRIV}$	DSA	Signature Generation, Key Agreement[36]

[35] Note: These non-volatile CSPs are not subject to the zeroization requirement since they are stored in encrypted form (using the AES algorithm).

[36] Key Agreement is not available in FIPS approved mode

Name	*Abbr.*	*Type*	*Usage*
EC User Key	$K_{USR\_EC\_PRIV}$	EC	Signature Generation, Key Agreement
AES User Key	K_{USR_AES}	AES	Key Encryption, Data Encryption or MAC
Triple-DES User Key	K_{USR_TDES}	Triple-DES	for Key Decryption, Data Decryption
User Generic Secret	K_{USR_GS}	Generic secret	to be used as keying material or as a HMAC key; at least 112 bits for HMAC generation

The functionality of cHSM user keys is dependent on their attributes, as indicated by the vendor-imposed security rules in Chapter 7. Keys with “CRYPT” or “DECRYPT” attribute can be used for encryption, keys with the “SIGN” or “VERIFY” attribute can used for digital signatures or MACs or HMACs, keys with the "WRAP" or "UNWRAP" attribute can be used for key wrapping, and keys with the “DERIVE” attribute can be used for key establishment.

6.5 Definition of Public Keys

Public Keys:			
Image Signing Key	ISK+	Public RSA 4096	Authenticates new boot images on download
Public Device Authentication Key	DAK+	Public ECDSA P521	Certificate verification, exportable
Public glad Authentication Key	GAK+	Public ECDSA P521	Device Authentication verification for Secure Messaging, exportable
Public Container Authentication Key	CAK+	Public ECDSA P521	Container Authentication verification for Secure Messaging, exportable
Public HSM Authentication Key	HAK+	Public 3072-bit RSA	Container Authentication verification for Secure Messaging, exportable
Public Operator Base Secret Encryption Key	OBSEK+	Public RSA-OAEP, min. 2048 bits	Wrapped import of Operator Base Secret
Public cHSM *Audit Log Signature Key*	K_{AL_PUB}	Public ECDSA P-256 or 3072-bit RSA	Audit Log Signature Verification, exportable
Initial glad Admin Authentication Key	GIAK+	Public ECDSA P-256	initial glad operator authentication
glad Admin Authentication Key	GAAK+	Public ECDSA (P-256 or P-521 or brainpoolP320t1)	glad operator authentication

		or RSA (2048...16384 bits)	
Container Initial Admin Key	CIAK+	Public ECDSA (P-256 or P-521) or RSA (1024…16384 bits)	initial cHSM operator authentication
cHSM Operator's Public Authentication Key	CAAK+	Public ECDSA[37] or RSA (1024...16384 bits)	cHSM operator authentication (may alternately occur with PSW_{AUTH} CSP)
Public cHSM User Keys:			
Public EC User Key	$K_{USR\_EC\_PUB}$	EC	Signature Verification, Key Agreement
Public DSA User Key	$K_{USR\_DSA\_PUB}$	DSA	Signature Verification, Key Agreement[38]
Public RSA User Key	$K_{USR\_RSA\_PUB}$	RSA	Signature Verification, Key Agreement
Volatile only			
Remote Public ECDH Key for Secure Messaging	SMRK+	Public ECDSA P-521	generated by the host and used to establish a Session Key Derivation Key via EC Diffie Hellman for Secure Messaging (glad or cHSM)
Local Public ECDH Key for Secure Messaging	SMLK+	Public ECDSA P-521	generated by the module (glad or cHSM) and used to establish a Session Key Derivation Key via EC Diffie Hellman for Secure Messaging

6.6 Modes of Access to CSPs

The tables in this section define the relationship between the different services provided by the cryptographic module and access to CSPs. The types of access (for example, Use/Write/Update) are given in the right-hand column.

6.6.1 Definitions

The following types of access are possible:

Write: the CSP is created (newly written) and stored.

Update: replaces the current value of the CSP with a new value.

Use: the value of the CSP is used for some cryptographic calculation.

[37] *NIST Recommended: P-224, P-256, P-384, P-521, K-233, K-283, K-409, K-571, B-233, B-283, B-409, B-571; Non-NIST (per IG A.2): brainpoolP224r1/ 224t1/ 256r1/ 256t1/ 320r1/ 320t1/ 384r1/ 384t1/ 512r1/ 512t1, secp256k1, FRP256v1*

[38] In validated FIPS mode, DSA User Keys cannot be used for Key Agreement

Wrapped Export: the CSP is wrapped by some wrapping key and exported from the cryptographic module.

Export: the CSP is exported from the cryptographic module (only possible for public RSA, DSA or EC keys $K_{USR\_RSA\_PUB}$, $K_{USR\_DSA\_PUB}$ and $K_{USR\_EC\_PUB}$).

Delete: invalidates the CSP

(xxx): The access type (one of the access types listed above) is set in brackets if this access type is conditional.

The following definitions are used in all tables mentioned above:

Any *User Key* can be a *Secret User Key* (K_{USR_AES}, K_{USR_TDES} or K_{USR_GS}) or a *Private* and/or *Public User Key* ($K_{USR\_RSA\_PRIV}$, $K_{USR\_RSA\_PUB}$, $K_{USR\_DSA\_PRIV}$, $K_{USR\_DSA\_PUB}$, $K_{USR\_EC\_PRIV}$, $K_{USR\_EC\_PUB}$) [39]

A *Secret Data Encryption Key* is a *Secret AES or DES User Key* (K_{USR_AES} or K_{USR_TDES}) with attribute[40] "CRYPT"/"DECRYPT". [41]

A *Secret Key Encryption Key* can be a *Secret AES or Triple-DES User Key* (K_{USR_AES} or K_{USR_TDES}) with attribute[42] "WRAP"/"UNWRAP".[41]

A *Secret MAC Key* can be a *Secret User Key* (K_{USR_AES} or K_{USR_GS}) with attribute[43] "SIGN"/"VERIFY".

A *Key Derivation Key* can be a *Private or Public EC or DSA*[44] *User Key* ($K_{USR\_EC\_PRIV}$, $K_{USR\_EC\_PUB}$, $K_{USR\_DSA\_PRIV}$, $K_{USR\_DSA\_PUB}$) with attribute[45] "DERIVE".

A *Private Sign Key* can be a *Private RSA, DSA or EC User Key* ($K_{USR\_RSA\_PRIV}$, $K_{USR\_DSA\_PRIV}$ or $K_{USR\_EC\_PRIV}$) with attribute[43] "SIGN".

A *Public Verify Key* can be a *Public RSA, DSA or EC User Key* ($K_{USR\_RSA\_PUB}$, $K_{USR\_DSA\_PUB}$ or $K_{USR\_EC\_PUB}$) with attribute[43] "VERIFY".

* General remark concerning the access to internal or external keys: If a key is marked with an asterisk, the key may be an internal[46] or an external[47] key. In case that such a key is accessed, the following CSPs must additionally be used:

When an internal *Secret or Private User Key* is to be accessed, the *Container Master Key* CMK must be used to decrypt or encrypt the internal key.

When an external key is to be accessed, the MBK must be used to verify or update the MAC and/or to decrypt or encrypt the secret or private key part.

---

[39] In validated FIPS mode, TDES keys cannot be generated.

[40] See chapter 7, vendor imposed security rule 19.

[41] In validated FIPS mode, TDES keys can only be used for decryption and unwrapping.

[42] See chapter 7, vendor imposed security rule 22.

[43] See chapter 7, vendor imposed security rule 20.

[44] In validated FIPS mode, DSA User Keys cannot be used as Key Derivation Keys

[45] See chapter 7, vendor imposed security rule 21.

[46] An "internal key" is any User Key that is stored inside the cryptographic module.

[47] An "external key" is any User Key that is stored outside the cryptographic module in the form of a secured *Backup Blob* (e.g. as result of the *Backup Key* service). A *Backup Blob is integrity protected with a MAC;* secret and private key parts are always encrypted with the Master Backup Key MBK.

** General remark concerning *DRBG Secrets* $S_{DRBG:}$

glad and each cHSM container has its own DRBG instantiation and its own set of *DRBG secrets*.

- If a new block of random values must be generated but no reseeding is required, the *DRBG Secrets* S_{DRBG_C} and S_{DRBG_V} are used and S_{DRBG_V} is updated.
- If a new block of random values must be generated and reseeding is required, all *DRBG Secrets* S_{DRBG_EI}, S_{DRBG_SEED}, S_{DRBG_C} and S_{DRBG_V} are updated and used.

Below, the four left-hand columns indicate the *Roles* for which each service is available.

An asterisk in brackets (*) indicates that the service can be executed by the user but no keys or CSPs are accessed by the service.

6.6.2 glad Services

glad services are only accessible to the Global Administrator role.

Table 13 – CSP and Key Access Rights within glad Services

Auth	Service	Cryptographic Keys and CSPs Access Operation	Type of Access
x	any command authentication	*glad Admin Authentication Key **GAAK+*** of respective Global Administrator	*Use*
x	any command using *Secure Messaging*	*Session Keys **SMEK** and **SMMK***	*Use*[48]
-	Auth_init	-	
x	Start_session	*DRBG Secrets* S_{DRBG}**	*Use, Update*
		*Remote Public ECDH Key **SMRK+***	*Use*
		*Local Private ECDH Key **SMLK***	*Use*
		*Local Public ECDH Key **SMLK+***	*Export*
		*Session Key derivation key **SMKDK***	*Use*
		*Session Keys **SMEK** and **SMMK***	*Write*
		*Device Authentication key **DAK** (glad)*	*(Use*[49]*)*
		*Public Device Auth.key **DAK+** (glad)*	*Export*
		*glad Authentication Key **GAK***	*Use (write*[50]*)*

[48] KTS with AES CBC + CMAC

[49] Signs GAK after generation

[50] Keypair is generated on first usage

Auth	Service	Cryptographic Keys and CSPs Access Operation	Type of Access
		*Public glad Authentication key **GAK+** (glad)*	*Export (Write[50])*
(x)	End_Session	*Session Keys **SMEK** and **SMMK***	*Delete[51]*
x	system_update	*Initial Global Admin Authentication key **GIAK+*** *Image Signature Key **ISK+*** *Container Base key **CBK***	*Update* *Use, Update* *Erase*
x	system_update_status	-	
x	key_get_wrapping_key	*Operator Base Secret Encryption Key **OBSEK*** *Public Operator Base Secret Encryption Key **OBSEK+***	*Write (Erase[52])* *Write (Erase[52]), Export*
x	key_import_operator_secret	*Operator Base secret **OBS*** *Device Master Key **DMK*** *Operator Base Secret Encryption Key **OBSEK***	*Write or Update* *Use* *Use, Erase*
x	key_list_operator_secret	-	-
x	key_delete_operator_secret	*Operator Base secret **OBS***	*Delete*
x	slot_get_quota	-	
x	slot_set_quota	-	
-	system_get_quorum_requirements	-	
x	system_set_quorum_requirements	-	
x	system_get_quota	-	
x	system_set_quota	-	
-	system_get_time	-	
x	system_set_time	-	
-	system_info	-	
-	system_monitor	-	
x	system_list_templates	-	
x	system_metrics	-	

[51] Invalidated within Secure RAM; Secure RAM is zeroized on power cycle and in case of an alarm.

[52] If there are already 32 wrapping keys stored, the oldest wrapping key is erased.

Auth	Service	Cryptographic Keys and CSPs Access Operation	Type of Access
x	System_Reset_Alarm	*DMK* *(DAK, DAK+, SDMK)*	*Generate, write* *(generate, write)*
x	system_fetch_log	-	
x	System_get_audit_log	-	
x	System_trim_audit_log	-	
x	user_add	*Global Admin Authentication Key GAAK+* *Device Master Key DMK*	*Write* *use*
x	user_delete	*Global Admin Authentication Key GAAK+*	*delete*
-	user_list	-	
x	User_backup_create	*Operator Base Secret OBS, User Snapshot MAC key USMK, User Snapshot Encryption Key USEK* *GAAK+*	*Use* *Use, wrapped Export*
x	User_backup_restore	*Operator Base Secret OBS, User Snapshot MAC key USMK, User Snapshot Encryption Key USEK* *GAAK+*	*Use* *Use, Write/update*
x	chsm_fetch_log	-	
x	chsm_snapshot	*All cHSM keys protected by the Container Master Key CMK as listed in section 6.4 (and corresponding public keys)* *Container Base Key CBK*	*Wrapped Export*
		Vendor Base Secret VBS, Operator Base secret OBS Container Base Key Encryption Key CBKEK Snapshot Encryption Key SEK, Snapshot MAC Key SMK	*use*
x	chsm_clone	*All cHSM keys protected by the CMK as listed in section 6.4 (and corresponding public keys)* *Container Base Key CBK* *Container Master Key CMK*	*write*
		DRBG Secrets S_{DRBG}**	*Write, use, update*
		Vendor Base Secret, Operator Base secret *Container Base Key Encryption Key, Snapshot Encryption Key, Snapshot MAC Key* *CBK, CMK, CAK, DAK*	*use*

Auth	Service	Cryptographic Keys and CSPs Access Operation	Type of Access
x	chsm_restore	*All cHSM keys protected by the CMK as listed in section 6.4 (and corresponding public keys)* *Container Base Key CBK* *Container Master Key CMK*	*write*
		Vendor Base Secret, Operator Base secret *Container Base Key Encryption Key, Snapshot Encryption Key, Snapshot MAC Key*	*use*
		*DRBG Secrets S_{DRBG}***	*Write, use, update*
		CBK, CMK, CAK, DAK	*use*
x	chsm_create	*Container Base Key CBK, Container Authentication Key CAK*	*Write, use*
		Master Backup Key MBK	*write*
		Container Master Key CMK	*Write, use*
		Glad Authentication Key GAK	*use*
		CAK+, DAK+, GAK+	*export*
		Container Initial Admin Key CIAK+ (public)	*Write, export*
		*DRBG Secrets S_{DRBG}***	*Write, use, update*
x	chsm_halt	-	
x	chsm_free_slot	*Container Base Key **CBK**,* *all keys derived from the CBK (like Master Key **CMK**)*	*Delete*
		All CSPs that are stored temporarily in the cHSM's Secure RAM (volatile storage)	*Delete*
		All CSPs that are stored wrapped with the Container Master Key	*Delete*[53]
-	chsm_list_slots	-	
x	key_get_csr	*DAK+* *DAK*	*Export* *Use*
x	key_import_cert	*DAK+*	*use*
x	User_change_credentials	*GIAK+* *GAAK+*	*Delete* *Update*

[53] CSPs are invalidated by zeroizing the Container Master Key **CMK** because they are encrypted with the Container Master Key.

Auth	Service	Cryptographic Keys and CSPs Access Operation	Type of Access
-	Initiate Self Tests (device restart)	*DRBG Secrets S_{DRBG}*** *DMK, CBK, SDMK, DAK, GAK+, CAK+*	*Write, Use, Update* *Use*
X	System_clear	*Device Master Key **DMK***	*Delete*
		All CSPs that are stored wrapped with the Device Master Key, or with a key which is stored wrapped with the Device Master Key (All except for SDMK, DAK and VBS)	*Delete*[54]
		All cHSMs together with all its CSPs and public user keys (CAAK+, CAK+, …) *public system keys belonging to deleted CSPs (GAK+, …)*	*Delete* *Delete*
-	external erase	*Device Master Key **DMK***	*Delete*[55]
		All CSPs that are stored wrapped with the Device Master Key, or with a key which is stored wrapped with the Device Master Key (All except for SDMK, DAK and VBS)	*Delete*[56]
		All cHSMs together with all their CSPs and public user keys (CAAK+, CAK+, …) *public system keys belonging to deleted CSPs (GAK+, …)*	*Delete* *Delete*
-	system_clear_to_factory_defaults	*As external_erase, but additionally all user authentication keys GAAK+*	*Delete*
-	Zeroize (Alarm)	*All CSPs in module*	*Delete*[57]

[54] CSPs are invalidated by zeroizing the Device Master Key DMK because they are encrypted with the Device Master Key or with a DMK encrypted key.

[55] Zeroized by overwriting the Key-RAM five times, alternately with 00_h and FF_h patterns.

[56] CSPs are invalidated by zeroizing the Device Master Key DMK because they are encrypted with the Device Master Key or with a DMK encrypted key.

[57] Zeroization zeroizes SDMK and DMK and Secure RAM. All CSPs which are not stored in Secure RAM are encrypted with SDMK or DMK or CMK (which is stored in Secure RAM).

6.6.3 cHSM Services - General

Table 14 – CSP and Key Access Rights within cHSM Roles & Services – General Services

cHSM User Role				Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Ad-minis trator	Security Officer	CU, KM, U[58]	NTP Manager			
X	X	X	X	any command authentication	*Public Authentication Key **CAAK+*** or *Password **PSW_{AUTH}*** of respective operator	*Use*
X	X	X	X	any command using *Secure Messaging*	*Session Keys **SMEK** and **SMMK***	*Use*[59]
X	X	X	X	Get Session Key	*DRBG Secrets **S_{DRBG}*** **	*Use, Update*
					*Remote Public ECDH Key **SMRK+***	*Use*
					*Local Private ECDH Key **SMLK***	*Use*
					*Local Public ECDH Key **SMLK+***	*Export*
					*Session Key derivation key **SMKDK***	*Use*
					*Session Keys **SMMK** and **SMEK***	*Write*
					*Public Device Auth.Key **DAK+***	*Export*[60]
					*Container Authentication Key **CAK***	*Use*[60]
					*Public Container Auth. Key **CAK+***	*Export*[60]
					*HSM Authentication key **HAK***	*Use*[61] *(write*[62]*)*
(within authenticated session)				End Session	*Session Keys **SMMK** and **SMEK***	*Delete*[63]

[58] Cryptographic User, Key Manager, User

[59] KTS with AES CBC + CMAC

[60] Optional; only if mutual authentication with Container Authentication Key is requested

[61] Optional; only if mutual authentication with HSM Authentication Key is requested

[62] Keypair is generated on first usage

[63] Invalidated within Secure RAM; Secure RAM is zeroized on power cycle and in case of an alarm.

cHSM User Role				Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Ad-minis trator	Security Officer	CU, KM, U[58]	NTP Manager			
				Get HSM Auth Key	*Public HSM Authentication Key **HAK+***	*Export (write[62])*
					*Private HSM Authentication Key **HAK***	*Use (write[65])*
				Get Audit Log Key	*Public Audit Log Signature Key* K_{AL_PUB}	*Export*
X	X	X	X	Change Operator's Key or Password	*Public Authentication Key **CAAK+** or Password* PSW_{AUTH} *of Operator*	*Update*
					If operator uses a password: *Container Master Key **CMK***	*(Use)*
				Restart_cHSM: and initiate all cHSM self-tests.	*DRBG Secrets* S_{DRBG}** *DMK, CBK, SDMK, DAK, CAK+*	*Write, Use, Update Use*

6.6.4 cHSM Services - Administration

Table 15 – CSP and Key Access Rights within Roles & Services – Administration

Role				Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Adminis-trator	Security Officer	CU, KM, U[64]	NTP Manager			
X				Add User	*Public Authentication Key **CAAK+*** or *Password* PSW_{AUTH} *of Operator*	*Write*
					If operator uses password: *Container Master Key **CMK***	*(Use)*
X				Delete User	Public Authentication Key ***CAAK+*** or Password PSW_{AUTH} of Operator	Delete[65]
X	X			Add Group User	Public Authentication Key ***CAAK+*** or Password PSW_{AUTH} of Operator	*Write*
					If operator uses password: *Container Master Key **CMK***	*(Use)*
X	X			Delete Group User	*Public Authentication Key **CAAK+*** or *Password* PSW_{AUTH} *of Operator*	*Delete[65]*

[64] Cryptographic User, Key Manager, User

[65] Invalidated within database; no zeroization needed because it is stored encrypted with the Container Master Key CMK.

Role				Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Adminis-trator	Security Officer	CU, KM, U[64]	NTP Manager			
X				Backup User	*Public Authentication Key* ***CAAK+*** or *Password* PSW_{AUTH} *of Operator*	*Wrapped Export*
					Master Backup Key ***MBK***	*Use*
					Container Master Key ***CMK***	*Use*
X				Restore User	*Public Authentication Key* ***CAAK+*** or *Password* PSW_{AUTH} *of Operator*	*Write* or *Update*
					Master Backup Key ***MBK***	*Use*
					Container Master Key ***CMK***	*Use*
X				Load File	If file to be loaded is a firmware module: *Public Module Signature Key* MS***K+***	(Use)
X				Delete File	---	---
X				Clear Audit Log	---	---
X				Set Maximum Failure Counter	---	---
X				Set Config Param	---	---
X				Load Certificate	*Container Authentication Key* ***CAK+***	*Use*
X				Set Time	---	---
X				Set Time Rel	---	---
			X	Set Time NTP	---	---
			X	Change Activation State	---	---
			X	Set NTP Settings	---	---
X				List Master Backup Keys	---	---
X				Clear Audit Log Files	---	---
X				Generate Audit Log Key	*Public Audit Log Signature Key* K_{AL_PUB}	*Write, Export*
					Private Audit Log Signature Key K_{AL_PRIV}	*Write, Use*
X				Get Signed Audit Log	*Private Audit Log Signature Key* K_{AL_PRIV}	Use
X				List DB Search Key	---	---
X				Export DB	*Master Backup Key* ***MBK***	*Use*

Role				Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Adminis-trator	Security Officer	CU, KM, U[64]	NTP Manager			
				Entry	If database entry whose back-up copy will be exported contains a *User Key* or the *Audit Log Signature Key*: *Any User Key* or *Private* and *Public Audit Log Signature Key* K_{AL_PRIV} and K_{AL_PUB}	*(Wrapped Export)*
					If database entry whose back-up copy will be exported is a user database entry: *Public Authentication Key* ***CAAK+*** or *Password* PSW_{AUTH} *of Operator*	*(Wrapped Export)*
					If database entry whose back-up copy will be exported contains a secret part (private/secret key or password): *Container Master Key* ***CMK***	*(Use)*
X				Import DB Entry	*Master Backup Key MBK*	*Use*
					If database entry whose back-up copy will be imported contains a *User Key* or the *Audit Log Signature Key*: *Any User Key* or *Private* and *Public Audit Log Signature Key* K_{AL_PRIV} and K_{AL_PUB}	*(Write* or *Update)*
					If database entry whose back-up copy will be imported is a user database entry: *Public Authentication Key* ***CAAK+*** or *Password* PSW_{AUTH} *of Operator*	*(Write* or *Update)*
					If database entry whose back-up copy will be imported contains a secret part (private/secret key or password): *Container Master Key* ***CMK***	*(Use)*
X				Set Administration-Only Mode	---	---
X				Set Startup Mode	---	---
X				Generate Master Backup Key	Master Backup Key MBK	*Wrapped Export*
					Session Keys ***SMEK*** *and* ***SMMK***	*Use*
					cHSM's DRBG Secrets S_{DRBG}**	*Use* and *Update*
X				Import Master Backup Key	Master Backup Key MBK	*Write* or *Update*
					Session Keys ***SMEK*** *and* ***SMMK***	*Use*

Role				Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Adminis-trator	Security Officer	CU, KM, U[64]	NTP Manager			
					*Container Master Key **CMK***	*Use*

6.6.5 cHSM Services – Key Management

Table 16 – CSP and Key Access Rights within Roles & Services – Key Management

Role					Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Ad-minis-trator	Security Officer	Cryptographic User		NTP Manager			
		User	Key Mgr				
	X				Init Key Group	*Any User Key*	*Delete*[66]
(*)	X	X	X		Open Key	If requested key is to be exported: *Any User Key**	*(Wrapped Export)*
(*)	(*)	(*)	(*)		List Keys	---	---
(*)	(*)		X		Delete Key	*Any User Key*	*Delete*[66]
X	X	X	X		Get Key Property*	If Public User Key is requested: *Any Public User Key** (***K**USR_RSA_PUB*, ***K**USR_DSA_PUB* or ***K**USR_EC_PUB*)	*(Export)*
(*)	(*)		(*)		Set Key Property*	--- (if an external key is addressed, the ***MBK*** is used to verify and update the MAC)	---
(*)	(*)		X		Backup Key	*Any User Key*	*Wrapped Export*
						*Master Backup Key **MBK***	*Use*
						If key whose back-up copy will be exported is *Private* or *Secret User Key*: *Container Master Key **CMK***	*(Use)*
(*)	(*)		X		Restore Key	*Any User Key*	*Write* or *Update* or *Wrapped export*
						*Master Backup Key **MBK***	*Use*
						If key which will be restored is *Private* or *Secret User Key* and shall be stored internally: *Container Master Key **CMK***	*(Use)*

[66] Invalidated within database; no zeroization needed because it is only stored encrypted with the Container Master Key CMK.

Role					Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Ad-minis-trator	Security Officer	Cryptographic User		NTP Manager			
		User	Key Mgr				
			X		Generate Key, Generate Key Pair	*cHSM's DRBG Secrets* S_{DRBG}**	*Use* and *Update*
						*Any User Key**	*Write* or *Update*[67] or *Wrapped Export*[68]
			X		Export Key	*Any User Key**	*Wrapped Export*
						Optional: *Secret Key Encryption Key** or *Public RSA User Key* $K_{USR\_RSA\_PUB}$*	*(Use)* [69]
						Only if random padding is required: *cHSM's DRBG Secrets* S_{DRBG}**	*(Use* and *Update)*
			X		Import Key	*Any User Key**	*Write* or *Update* or *Wrapped Export*
						Optional: *Secret Key Encryption Key** or *Private RSA User Key* $K_{USR\_RSA\_PRIV}$*	*(Use)*[69]
			X		Derive Key (option ECDH_COF or TLS12_PRF)	*Key Derivation Key(s)**	*Use*
						*Secret User Key**	*Write* or *Update or Wrapped Export*
			X		Split Key	*Generic Secret* K_{USR_GS}*	*Use* and *Delete*[70]
						*Secret User Key**	*Write* or *Update* or *Wrapped Export*
			X		Wrap	*Any User Key**	*Wrapped Export*

[67] if the generated key shall be stored in the cHSM

[68] if the generated key shall be exported outside the cHSM

[69] Key (un)wrapping: AES KW(P), AES CCM, AES GCM, KTS-RSA or allowed RSA key (un)wrapping

[70] Invalidated within database; no zeroization needed because it is only stored encrypted with the Container Master Key CMK.

Role					Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Ad-minis-trator	Security Officer	Cryptographic User		NTP Manager			
		User	Key Mgr				
						Secret Key Encryption Key or Public RSA User Key* $K_{USR\_RSA\_PUB}$*	*Use*[69]
						Only if random padding is required: *DRBG Secrets* S_{DRBG}**	*(Use* and *Update)*
			X		Unwrap	*Any User Key**	*Write* or *Update* or *Wrapped Export*
						Secret Key Encryption Key or Private RSA User Key* $K_{USR\_RSA\_PRIV}$*	*Use*[69]
			X		Create Object	*Any User Key**	*Write* or *Update* or *Wrapped Export*
			X		Copy Object	*Any User Key**	*Write* or *Wrapped Export*

6.6.6 cHSM Services – Cryptographic Services

Table 17 – CSP and Key Access Rights within Roles & Services – Cryptographic Services

Role					Service	Cryptographic Keys and CSPs Access Operation	Type of Access
Ad-minis-trator	Security Officer	Cryptographic User		NTP Manager			
		User	Key Mgr				
		X			Crypt Data	*Secret Data Encryption Key**	*Use* [69]
						If random padding is required: *cHSM's DRBG Secrets* S_{DRBG}**	*(Use* and *Update)*
		X			Sign Data	*Private Sign Key** or *Secret MAC Key**	*Use*
						If random padding is required: *cHSM's DRBG Secrets* S_{DRBG}**	*(Use* and *Update)*
		X			Verify Signature	*Public Verify Key** or *Secret MAC Key**	*Use*
		X			Generate Random Number	*cHSM's DRBG Secrets* S_{DRBG}**	*Use* and *Update*
		X	X		Compute Hash	optional*: Generic Secret* K_{USR_GS}*	*(Use)*
		X	X		Generate DSA Parameters (PQ/G)	*cHSM's DRBG Secrets* S_{DRBG}**	*Use* and *Update*

7 Security Rules

The cryptographic module's design complies with the cryptographic module's security rules.

This section documents the security rules enforced by the cryptographic module to implement the security requirements of a FIPS 140-2 Level 3 module.

1. The cryptographic module provides at least two distinct operator roles. These are the *User* role (Cryptographic User) and the *Crypto Officer* role (Global Administrator and cHSM Administrator).
2. The cryptographic module provides identity-based authentication.
3. No access to any cryptographic services is permitted until the operator has been authenticated into the *User* role and the *Crypto Officer* role (or respective sub roles) by the module.
4. The cryptographic module performs the following tests:
    a) Device Power-up Self-Tests:
        i) Firmware Integrity Tests
            (1) SHA3-384 (Cert. #A1562) verification of bootROM and BOOT.bin (incl. FSBL, PMU, FPGA, ATF and U-Boot)
            (2) SHA-256 (Cert. #A1563) verification of boot image (containing linux kernel, fipscheck, init and openrc)
            (3) HMAC-SHA256 (Cert. #A1561) verification of ROOTFS (containing glad, cmgr, vBL) and SMOS
            (4) SHA-512 (Cert. #A1564) hash value verification for the module program code for every cHSM firmware module (i.e. firmware component)
        ii) Entropy source Power-Up tests, implemented for each ESV
            (1) According to SP 800-90B:
                (a) Repetition Count Test according to SP 800-90B §4.4.1
                (b) Adaptive Proportion Test according to SP 800-90B §4.4.2
            (2) According to [AIS 20/31] (RNG class PTG.2):
                (a) Continuous Chi-Squared Test according to AIS 20/31 §5.5.3
                (b) Start-up Chi-Squared Test according to AIS 20/31 §5.5.2
        iii) COSMOS Cryptographic Algorithm Power-up Tests:
        COSMOS includes all algorithm implementations used by glad (Cert. #A1561), and it implements the entropy sources used by glad and the cHSMs.
            (1) AES Known Answer Tests (encrypt and decrypt: CBC, CTR)
            (2) AES-CMAC Known Answer Test
            (3) AES GCM encrypt and GCM decrypt Known Answer Tests
            (4) DRBG Known Answer Tests according to [NIST 800-90A] (testing the Instantiate Function, the Generate Function and the Reseed Function)

(5) ECDSA Pair-wise Consistency Test[71] (sign/verify)

(6) ECC DH Known Answer test (meeting IG D.8[71])

(7) HMAC Known Answer Test[72]

(8) KBKDF SP 800-108 Known Answer Test

(9) NIST 56C KDF: Known Answer Test

(10) RSA Known Answer Tests (sign/verify)

(a) Per IG D.9, these KATs fulfill the SP 800-56Br2 power-up tests

(11) SHA-256, SHA-512 Known Answer Tests

iv) cHSM (Cert. #A1560) Cryptographic Algorithm Tests
(FIPS cHSM as well as non-FIPS cHSM):

(1) AES Known Answer Tests (encrypt and decrypt: CBC, CFB8, CTR, ECB, OFB)

(2) AES-CMAC Known Answer Test

(3) AES GMAC, GCM encrypt and GCM decrypt Known Answer Tests

(4) DRBG Known Answer Tests according to [NIST 800-90A] (testing the Instantiate Function, the Generate Function and the Reseed Function; Cert #A1564)

(5) DSA Pair-wise Consistency Test (sign/verify)

(6) ECDSA Pair-wise Consistency Tests[73] (sign/verify for specified curves)

(7) ECC DH Known Answer tests[74] (meeting IG D.8)

(8) HMAC Known Answer Tests[75]

(9) KBKDF SP 800-108 Known Answer Test

(10) KDF Known Answer Tests for:
(a) ANSI X9.63 KDF
(b) NIST 56C KDF
(c) TLS 1.2 KDF
All KDFs are tested with SHA-2 and SHA-3 (where applicable).

(11) RSA Known Answer Tests (sign and verify)

(a) Per IG D.9, these KATs fulfill the SP 800-56Br2 power-up tests

(12) SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 Known Answer Tests

(13) SHA3-224, SHA3-256, SHA3-384, and SHA3-512 Known Answer Tests

(14) Triple-DES ECB and CBC encrypt and decrypt Known Answer Tests

[71] NIST P-256

[72] kat_hmac_sha256

[73] NIST P-256 and NIST B-283 according to IG 9.4 Note 10.

[74] NIST P-256 and NIST B-283 (meeting IG D.8)

[75] kat_hmac_sha1, kat_hmac_sha224, kat_hmac_sha256, kat_hmac_sha384, kat_hmac_sha512, kat_hmac_sha3_224, kat_hmac_sha3_256, kat_hmac_sha3_384, kat_hmac_sha3_512

b) Conditional Self-Tests:

i) *COSMOS and cHSM: Continuous Random Number Generator (RNG) Test* performed on DRBG: Prior to each use, each DRBG instantiation is tested using the conditional test specified in FIPS 140-2 §4.9.2.

ii) *COSMOS: Entropy source Continuous tests* implemented for each ESV:

(1) According to SP 800-90B:

(a) Repetition Count Test according to SP 800-90B §4.4.1

(b) Adaptive Proportion Test according to SP 800-90B §4.4.2

(2) According to [AIS 20/31] (RNG class PTG.2):

(a) Continuous Chi-Squared Test according to AIS 20/31 §5.5.3

iii) *cHSM: DSA* Key *Pair-wise Consistency Test* (sign/verify) for DSA key generation

iv) *COSMOS and cHSM:* ECDSA Key *Pair-wise Consistency Test* (sign/verify) for EC key generation

v) *COSMOS and cHSM: RSA* Key *Pair-wise Consistency Test* (both sign/verify[76] and encrypt/decrypt) for RSA key generation

vi) *COSMOS Firmware Load Test* (via RSA 4096 signature verification, Cert. #A1561)

vii) *COSMOS and cHSM Public Key Validation* as required by SP 800-56Ar3 (Cofactor) Ephemeral Unified Model (full public key validation according to SP 800-56Ar3 section 5.6.2.3.3)

5. At any time, the Global Administrator operator can force the module to perform all power-up self-tests (for COSMOS and all cHSMs).

5a. Each cHSM operator can force his cHSM to perform all cHSM power-up self-tests.

6. Data output is inhibited during key generation, self-tests, zeroization, and error states.

7. Status information does not contain CSPs or sensitive data that if misused could lead to the compromising of the module.

8. The module supports concurrent operators.

9. The successful completion of the glad power-up self-tests is indicated by executing the gladm “system-info” command which returns device_system_version which does not contain ‘recovery’.

10. The successful completion of each FIPS cHSM power-up self-tests is indicated by executing the csadm “GetState” command which returns state = INITIALIZED and FIPS mode = ON.

The following security rules are imposed by the vendor:

11. The module zeroizes all plaintext CSPs within a maximum of 7 ms after any attack or alarm (see chapter 8 below).

12. If the cryptographic module remains inactive in any valid role for a maximum period of 15 minutes, the module automatically logs off the operator.

[76] For COSMOS, RSA key pairs are only used for key transport and not digital signatures

13. The module provides functionality for protecting command and response data on their way to and from the module via a *Secure Messaging* mechanism. This mechanism encrypts and integrity protects the data with the AES encrypting algorithm and CMAC. In FIPS mode, the use of Secure Messaging is mandatory for every command that has to be authenticated.
14. The module implements a Challenge-Response mechanism to prevent the replay of older authenticated messages.
15. The module prohibits the export of plaintext secret or private cryptographic keys or other CSPs.
16. The cHSM supports an “Exportable” attribute for every stored private or secret cryptographic user key. The module only permits the (wrapped) export of a key if this attribute is set.
17. The module supports a “Deny_backup” attribute for every stored private or secret cryptographic key. The module only permits the MBK encrypted export (export for backup purposes) of a key if this attribute is NOT set.
18. The module supports an (optional) “Key Group” attribute for every stored key and for every registered operator. Access to a key can be restricted by assigning this key to a specific key group. Operators who are not assigned to the same key group are forbidden to access or even ‘see’ the key.
    A key is assigned to a key group by setting its key group attribute value to the desired key group name. An operator is assigned to a key group by setting their operator key group attribute value to the desired key group name.
19. The module supports the “CRYPT” (”DECRYPT”) attribute for every stored secret cryptographic AES or Triple-DES key. The module only permits encryption (decryption) with a secret user key if this attribute is set. In FIPS mode this attribute cannot be set for private or public user keys. In particular, RSA and EC keys cannot be used for bulk data encryption or decryption. In FIPS mode, Triple-DES keys cannot be used for encryption and cannot be generated.
20. The module supports the “SIGN” (”VERIFY”) attribute for every private, public or secret cryptographic key. The module only permits the generation (verification) of a signature with a private (public) user key only if this attribute is set. The module allows the generation (verification) of a MAC or HMAC with a secret user key only if this attribute is set. In FIPS mode, Triple-DES keys cannot be used for TDES MAC calculation and verification. This attribute can only be set if attributes DERIVE and WRAP/UNWRAP are not set.
21. The module supports a “DERIVE” attribute for private and public cryptographic EC or DSA keys. The module only permits key derivation with a private or public user key if this attribute is set. In FIPS mode, DSA keys cannot be used for key derivation.
    This attribute cannot be set for RSA keys or secret user keys. This attribute can only be set if attributes SIGN and VERIFY are not set.
22. The module supports the “WRAP” (”UNWRAP”) attribute for every stored secret AES, Triple-DES or public (private) RSA key. The module only permits the key to be used to encrypt (decrypt) other keys for export (import) if, and only if, this attribute is set.
    This attribute cannot be set for EC or DSA keys. In FIPS mode, Triple-DES keys cannot be used for key wrapping. This attribute can only be set if attributes SIGN and VERIFY are not set.

23. The module supports the attribute “TRUSTED” (default: false) for every stored wrapping key (attribute “WRAP” = TRUE), which can only be set to TRUE by a *Security Officer*. It also supports the “WRAP WITH TRUSTED” attribute (default: false) for any key. If set to TRUE, the key can only be wrapped with a wrapping key that has the attribute “TRUSTED” set to TRUE.

8 Physical Security Policy

The u.trust Anchor is a multi-chip embedded cryptographic module encapsulated in a hard, opaque, tamper-evident coating.

On the top side of the module a (hollow) metal heat sink is directly mounted on the printed circuit board on three edges, and the space between the PCB and the heat sink is completely filled with potting material (epoxy resin) (see Figure 4). On the bottom side of the PCB, a metal frame is stuck directly onto the printed circuit board, and the space inside the metal frame is completely filled with potting material (see Figure 5). Epoxy hardness testing was performed over the module's operating temperature range from -10 ˚C to +60 ˚C.

The heat sink and potting material together define the top and bottom sides of the module and deliver a hard, opaque coating. All the cryptographic module's hardware components (which are all mounted on the PCB) are entirely covered by this coating.

Each u.trust Anchor also has two sensor patch wires. If either of these are disrupted (i.e. in the case of physical attack), it activates the tamper response. This feature falls under FIPS 140-2 Area 11 (Mitigation of Other Attacks).

The u.trust Anchor module with its tamper-evident enclosure (the heat sink and the potting material) implements the following physical security mechanisms:

Active tamper response and zeroization circuitry.

The cryptographic module's hardware components are covered by hard, opaque potting material or the heat sink which show evidence of tampering on the enclosure when a physical attack is attempted.

The potting material is hard and opaque enough to prevent direct observation and easy penetration to the depth of the underlying hardware components. It is highly probable that anyone attempting to penetrate to the depth of the circuitry will break off large pieces of potting material and tear important hardware components off the module, causing serious damage to the module.

Temperature sensors that activate a tamper response if the module is outside of the defined temperature range of –18°C to 81°C (-0.4°F to 177.8°F).

Voltage sensors that monitor the power supply of the module and activate a tamper response if the power input is outside of the defined range (including low or removed battery).

Tamper response and zeroization circuitry is active while module is in standby mode (powered down).

Zeroization is performed within less than 7 milliseconds after tamper detection (temperature or voltage outside of defined range).

The module regularly inverts all bits of the plaintext master keys (DMK and SDMK) to avoid "burn in" of information into SRAM cells.

To ensure security of the cryptographic module, the module must be periodically inspected for evidence of tampering. The recommended inspection schedule depends on the customer's application area. This may vary between inspecting the module once a week and once a year.

The physical security mechanisms listed above function autonomously and under all circumstances.

9 Operational Environment

The FIPS 140-2 Area 6 Operational Environment requirements are not applicable because the cryptographic module does not contain a modifiable operational environment. The operational environment is defined as limited.

10 Mitigation of Other Attacks Policy

The cryptographic module has been designed to mitigate several physical attacks, Simple and Differential Power Analysis (SPA/DPA) and timing analysis.

Table 18 - Mitigation of Other Attacks

Other Attacks	Mitigation Mechanism
Timing Analysis	Triple-DES and AES operations are executed in fixed time, so that it is not feasible to determine the value of an algorithm's keys by measuring the execution time of a cryptographic operation. If blinding is switched on for RSA and ECDSA on a cHSM, the input data for a single RSA and ECDSA signature generation is randomized by use of a blinding technique so that the input parameters of the algorithm are not known by the operator. In this case it is not possible to gain knowledge about the private key by the amount of time required by the signature operation. Blinding is not possible for bulk signing.
Mechanical attack	Each u.trust Anchor with hardware version 7.03.00.03 has two sensor patch wires installed in the epoxy. If one of these is disrupted (i.e. in the case of physical attack), it activates the tamper response (zeroization).
Temperature and voltage	The module provides physical EFP temperature and voltage protections that are outside the scope of FIPS 140-2 physical security Level 3. A tamper response (zeroization) is activated if the module is outside the defined temperature range (–18°C to 81°C) or voltage range

11 References

Reference	Title/Company
[ANSSI]	ANSSI: "Avis relatif aux paramètres de courbes elliptiques définis par l'Etat français" in: Journal Officiel de la République Française (JORF), n° 0241 du 16 octobre 2011 page 17533 text n° 30 (Announcement about elliptic curve parameters set by the French government). NOR: PRMD1123151V. Available: https://www.legifrance.gouv.fr/affichTexte.do?cidTexte=JORFTEXT000024668816
[CsarGladmGuide]	u.trust ANCHOR CSAR- Administrator Guide for u.trust ANCHOR CSAR in FIPS Mode, doc. no 2020-0035 / Utimaco IS GmbH
[CSAdmGuide]	u.trust ANCHOR CSAR - Administrator Guide for u.trust ANCHOR CSAR cHSM in FIPS Mode, Doc. no 2020-0040 / Utimaco IS GmbH
[ECCBP]	RFC 5639: Elliptic Curve Cryptography ECC Brainpool Standard - Curves and Curve Generation, March 2010, including Errata, http://tools.ietf.org/html/rfc5639
[FIPS140-2]	FIPS PUB 140-2, Security Requirements for Cryptographic Modules / National Institute of Standards and Technology (NIST), May 2001
[FIPS186-2]	FIPS PUB 186-2: Digital Signature Standard (DSS) / National Institute of Standards and Technology (NIST), January 2000
[FIPS186-4]	FIPS PUB 186-4: Digital Signature Standard (DSS) / National Institute of Standards and Technology (NIST), July 2013
[NIST 800-90A]	NIST Special Publication 800-90A, Recommendation for Random Number Generation Using Deterministic Random Bit Generators / National Institute of Standards and Technology (NIST), January 2012
[NIST SP 800-56A r3]	NIST Special Publication 800-56A Revision 3, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography
[PKCS#1]	PKCS#1: RSA Encryption Standard v2.1, 14th June 2002 / RSA Laboratories, http://www.rsa.com/rsalabs/node.asp?id=2125
[PKCS#3]	PKCS#3: Diffie-Hellman Key Agreement Standard v1.4, 1st November 1993 / RSA Laboratories, http://www.rsa.com/rsalabs/node.asp?id=2126
[PKCS#11]	PKCS#11: Cryptographic Token Interface Standard v2.20, 28th June 2004 / RSA Laboratories, http://www.rsa.com/rsalabs/node.asp?id=2133
[RFC 7748]	RFC 7748: Elliptic Curves for Security / Internet Research Task Force (IRTF), January 2016, ISSN 2070-1721, including Errata ID 4730 reported and verified on 2016-07-05
[SEC2]	SEC2: Recommended Elliptic Curve Domain Parameters – Certicom Research – September 20, 2000, Version 1.0
[AIS 20/31]	Application Notes and Interpretation of the Scheme (AIS): AIS 20/AIS 31: A proposal for: Functionality classes for random number generators, Version

Reference	Title/Company
	2.0 / Wolfgang Killmann (T-Systems GEI GmbH, Bonn), Werner Schindler (Bundesamt für Sicherheit in der Informationstechnik/BSI, Bonn), 18. September 2011

12 Definitions and Acronyms

AES	Advanced Encryption Standard
CSP	Critical Security Parameter
cHSM	Containerized HSM
COSMOS	u.trust Anchor platform firmware containing u.trust Anchor boot loader, Linux kernel, container manager and glad
DES	Data Encryption Standard
DH	Diffie-Hellman
DPA	Differential Power Analysis
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
EC	Elliptic Curve
ECDH	Elliptic Curve Diffie-Hellman Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
glad	Global Administration Service for u.trust Anchor
gladm	Global Administration Tool for u.trust Anchor
HSM	Hardware Security Module
KDF	Key Derivation Function
MAC	Message Authentication Code
MBK	Master Backup Key
PCB	Printed Circuit Board
PCI	Payment Card Industry
PTS	PIN Transaction Security
RNG	Random Number Generator
SHA	Secure Hash Algorithm
SPA	Simple Power Analysis
Triple-DES	Triple-DES with key size 16 or 24 bytes