d'Cryptor™ QE
Cryptographic Module

Security Policy

Hardware Version: 3.0L, 3.0S
3.1L, 3.1S

Firmware Version: 2.0

Document Version 1.7
24 May 2005

Configuration Control

Document details

File Name:	QE Security Policy-NonProprietary.doc
Document Title:	*d'Cryptor* QE Cryptographic Module – Security Policy
Document Number:	DC/QE-SP-002
Document Revision No.:	1.7
Author:	Quek Gim Chye
Approved By:	Antony Ng
Number of pages:	28
Revision Date:	24 May 2005
Remarks	Non-proprietary version

Revision History

Revision	Date	Author	Comments on Revision
1.0	15 Mar 2004	Quek GC	Initial version
1.1	24 Mar 2004	Quek GC	Added two conditional tests
1.2	4 Apr 2004	Quek GC	Added security level table.
1.3	7 Apr 2004	Quek GC	Updated self-test description
1.4	12 Apr 2004	Quek GC	Updated Table 3
1.5	1 Sep 2004	Quek GC	Added remark for DES in Table 3. Added Table 8 in §11.3.
1.6	19 May 2005	Quek GC	Updated hardware version number
1.7	24 May 2005	Quek GC	Updated hardware version numbers

Contents

1 Scope

This document defines the non-proprietary security policy for the *d'Cryptor*® QE cryptographic module. This information is required in order to satisfy in part the requirements for the validation of the *d'Cryptor* QE at level 3 of the FIPS 140-2 standard.

This document applies to hardware versions 3.0L, 3.0S, 3.1L and 3.1S, and firmware version 2.0.

2 Introduction

The *d'Cryptor* QE cryptographic module ("QE") is a multi-chip embedded module that accepts the secure loading of an application and executes such an installed application after it has completed its bootstrap and other system initialization processes.

The QE is designed as a secure cryptographic coprocessor for the *d'Cryptor* line of products where it provides a general-purpose computation environment and high-performance cryptographic support in the form of a bootstrap and system firmware that permits the secure loading of an application while in the field. An application to be loaded is signed using the ANSI X9.31 digital signature scheme. Upon loading a new application, the QE verifies its digital signature and marks the new application as operational.

The factory Certificate Authority generates the private and public key pair that is used to sign and verify the application. The public key certificate is loaded into the QE as part of the final stage in manufacturing whereas the private key is maintained outside the QE and held securely by the individual (entity) who is responsible for generating and signing the application.

The bootstrap provides low-level system initialization and memory self-tests, and passes control to the system firmware, which in turn initializes higher-level sub-systems and performs cryptographic self-tests.

If an application is present in the module, the system firmware checks its trustworthiness by verifying its digital signature, and thereafter passes control to the application. If the verification fails, the application is instantly erased and control returns to the system firmware where it enters a command mode that allows the loading of a new application. The module also enters this command mode when there is no application loaded.

For physical security, the QE implements an hard, opaque epoxy potting and a tamper detection and response mesh to protect the hardware and software components as well as CSPs and other cryptographic keys.

A host of services is implemented in the module, including both cryptographic and non-cryptographic, and made available to an installed application through API calls.

The QE is designed to be a FIPS 140-2 Level 3 module for the secure loading of applications. Once an application has been loaded, the QE may continue to function as a Level 3 module only if the loaded application has been validated at FIPS 140-2 Level 3. If a non-FIPS validated application is loaded, the QE will be in a non-Approved mode of operation.

The term *QE* and *d'Cryptor QE* shall be used synonymously throughout this document.

3 Security Level

The *d'Cryptor* QE cryptrographic module meets the overall requirements applicable to Level 3 security of FIPS 140-2. Table 1 below shows the individual security level requirement achieved by the module:

Security Requirement Area	Level Achieved
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	N.A.

Table 1. Security Levels

4 The *d'Cryptor* QE

The *d'Cryptor* QE is a multi-chip embedded security module designed for high security assurance applications. It is made up of the following three components:

- Base hardware (hardware versions 3.0L, 3.0S, 3.1L and 3.1S)
- *d'Cryptor* kernel (firmware version 2.0)
- Field application layer

The hardware is built around a high-performance StrongARM SA-1110 processor and generous memory in the form of Flash and SRAM. Critical security parameters such as cryptographic keys are stored in battery-backed SRAM.

Two hardware configurations for *d'Cryptor* QE are available:

Configuration	Part Number	Version Number
8 MB Flash 1 MB RAM	DC/QE-L.8.1024	3.0L
	DC/QE31-L.8.1024	3.1L
4 MB Flash 512 KB RAM	DC/QE-S.4.512	3.0S
	DC/QE31-S.4.512	3.1S

The *d'Cryptor* kernel contains the base software of the QE, and performs the entire boot-up and initialization processes in the QE before handing control over to the application, if one exists in the field application layer. The QE allows an external application to be loaded in the field (i.e. outside the factory), as long as the application had been cryptographically signed with the correct RSA private signing key. It also provides a variety of cryptographic services through a library and an API that resides in the kernel. These services are available to the application but will not be relevant to the current validation of the QE as there is no application loaded.

4.1 Cryptographic Module Diagram

Figure 1 and Figure 2 show plan views of the *d'Cryptor* QE.

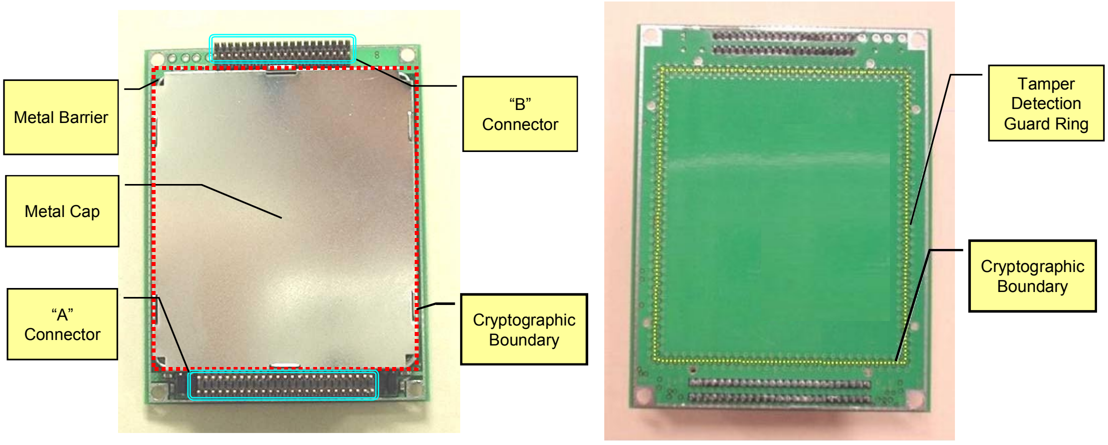

Figure 1. Front view (left) and back view (right) of QE

4.2 Hardware Architecture

Figure 2 shows the architectural block diagram of the QE. The cryptographic boundary of the QE is indicated by the contiguous solid red line. The QE communicates with the external world via the interfaces drawn on the right side of the figure.

Details on the hardware sub-systems can be obtained from DC/QE-MS-001, "*d'Cryptor QE Cryptographic Module – Module Specifications*".

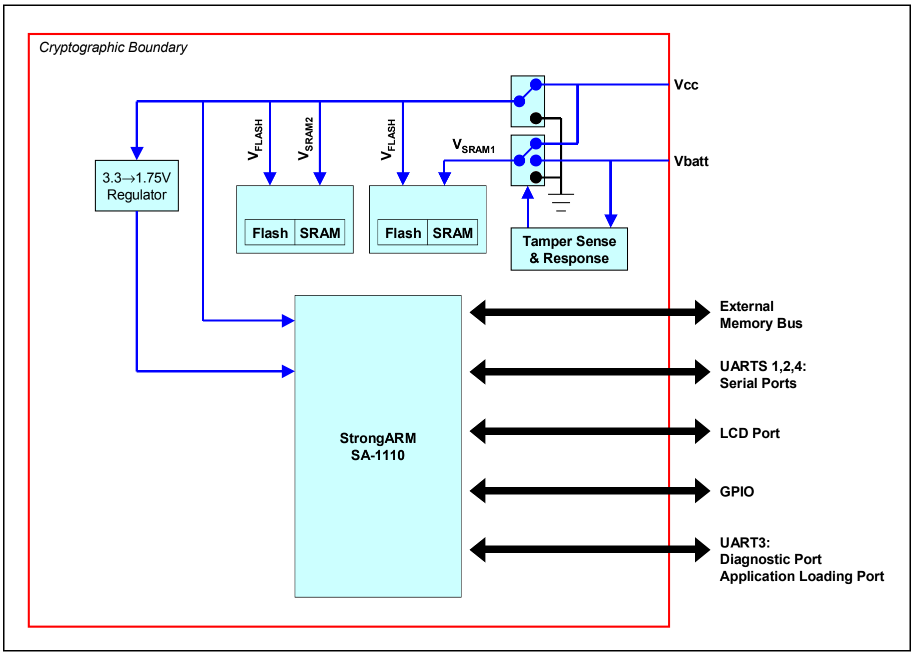

Figure 2. Block Diagram of QE

4.3 Software Architecture

The memory components of the QE consist of FLASH, battery-backed RAM (NVRAM) and volatile static RAM (SRAM).

FLASH is a 4 or 8 MB flash memory that is divided into several layers:

- Layer 0 contains the bootstrap and the operating system. This is the *d'Crypt Secure Micro O/S*, sometimes referred to as the *micro-kernel*. The current micro-kernel version used is version 3.0.
- Layer 1 contains system firmware. Together with Layer 0, they form the *d'Cryptor kernel,* or simply the *kernel*, of the QE. The system firmware version used is V2.0.
- Layer 2 contains the field application (none loaded for this validation)

The FLASH also maintains a copy of the Layer 0 bootstrap as well as a basic version of the Layer 1 system firmware for emergency recovery.

The upper region of FLASH is used to host the Log File System and the Flash File System. These are used by the kernel and the application to respectively store diagnostic log messages and non-volatile data.

NVRAM is used to store CSPs and other cryptographic keys.

Figure 3 shows the memory map of the *d'Cryptor* QE.

Layer	Description	Size
–	Flash File System	2 MB (v3.0S, v3.1S) 6 MB (v3.0L, v3.1L)
–	Log File System	256 KB
2	Field Application	1 MB
1	QE System Firmware	512 KB
0	Bootstrap (Micro-Kernel)	

Figure 3. *d'Cryptor* QE Flash Memory Map

4.4 QE Interfaces

The QE is designed with a host of interfaces listed in Table 2. The interfaces are intended for use by future applications to be loaded into the QE. For the purpose of this validation, only the UART3 port is used to support all the communications with the outside world that it performs via service calls.

The UART3 interface doubles up as the following logical interfaces:

- *Diagnostic Port* – Outputs diagnostic status messages and accepts control commands.
- *Application Loading Port* – Used for loading application firmware images.

The physical access points of the QE are restricted to a connector interface comprising a "A" connector and a "B" connector located at opposite sides of the cryptographic boundary (see Figure 1). Table 2 shows the standard logical interfaces (as mandated in FIPS 140-2) and their mappings to the actual physical access points on the QE.

Logical Interface	Physical Port Mapped	Remarks
Data Input Interface	UART1, UART2, UART3, UART4 LCD port External Memory Data Bus GPIOs 1 to 26	All ports except the UART3 can be accessed only through an application that is loaded into the module and as such, are not currently utilized by the module
Data Output Interface	UART1, UART2, UART3, UART4, LCD port External Memory Data Bus External Memory Address Bus External Memory Bus switches: CS3#, WE#, OE# GPIOs 1 to 26	
Control Input Interface	UART3 Reset	Currently used by QE
Status Output Interface	UART3	Currently used by QE
	GPIO 19 and GPIO 26 (These can be connected to LEDs to indicate power status and error condition respectively. LEDs are not included as part of QE)	Requires an application to access and configure these logical ports
Power Interface	Supply voltage port Battery voltage port Ground pins	Currently used by QE

Table 2. Mapping of Logical Interfaces to Processor Ports

5 Approved Mode of Operation

When shipped from the factory, the QE always operates in a FIPS 140-2 Approved mode of operation. This is indicated by the message “`Operating mode = FIPS`” that is displayed via the diagnostic port after powering up (Figure 4):

```
...
Operating mode = FIPS
...
MAIN  : (L)oad (B)oot (S)hutdown (z)eroise (Z)eroise-all (D)ump ?
```

Figure 4. Indication of the Approved Mode of Operation (without an application loaded)

If an application that is not FIPS validated is loaded, the QE loses its FIPS validation and indicates that it is not in an Approved mode of operation by displaying the following:

```
...
Operating mode = Non-FIPS
...
[QE launches the application automatically]
```

Figure 5. Indication of a Non-Approved Mode of Operation (with an application loaded)

5.1 Approved Algorithms

The QE employs six Approved algorithms, four of which have been FIPS validated. These are listed in Table 3.

Security Function	Certificate Number	Remarks
DES	**#205**	For use with legacy systems only
TDEA	**#159**	2-key and 3-key
AES	**#49**	128, 192 and 256 bit keys
SHA-1	**#139**	
HMAC-SHA-1	Vendor Affirmed	HMAC with SHA-1
RSA ▪ **PKCS #1, v2.1** ▪ **ANSI X9.31-1998 (only for digital signature verification)**	Vendor Affirmed	512, 768, 1024, 1536, 2048 bits

Table 3. List of Approved Algorithms

All the above services are available only through the QE internal APIs (see Annex A).

6 Roles and Authentication

This chapter describes the roles supported by the QE followed by the authentication mechanisms that are used to perform identity-based authentication of an operator accessing the module and to verify that the operator is authorized to assume the requested role and perform services within that role.

The QE is a single-user module and does not support concurrent operators or any maintenance role. It always assumes a "no role" state after power up. In this state, no security relevant services can be performed. However, non-security relevant services are available for execution.

6.1 Roles

The QE provides two roles (**crypto-officer** and **user**) and a single identity (**entity**) that assumes either role when operating the module.

6.1.1 Crypto-Officer Role

The **crypto-officer** corresponds to the *Crypto Officer* role as defined in FIPS 140-2. The QE is factory-installed with a 2048-bit RSA public key whose purpose is to allow the QE to verify the signature of an application that it is requested to load. In future, new applications for the QE may be generated and signed using the private key. In the field, the **crypto-officer** would load the signed application using the Application Firmware Load service.

The **crypto-officer** performs no other service.

6.1.2 User Role

The **user** corresponds to a User role as defined in FIPS 140-2. Like the **crypto-officer**, the **user** role is used to perform field loading of an application that has been signed using the private key. The **user** performs no other service.

6.2 Authentication

A pair of 2048-bit RSA keys comprising the EAVK (public key) and the EASK (private key) is generated at the factory by the factory Certificate Authority. The EAVK is loaded into the QE at the factory prior to being fielded. The EASK remains outside the module, and is kept by and known only to the **entity** who is responsible for creating and signing any applications that are to be loaded into the module.

The operator authenticates himself by calling the Application Firmware Load service to load an application that has been digitally signed using the EASK. The module completes the authentication by verifying the application's signature with the EAVK.

The identification and authentication of the **entity** is performed by the fact that the application is authenticated by means of its digital signature, since only the **entity** could have signed the application. Since both the **crypto-officer** and the **user** roles perform the same authenticated service of loading a signed application, the **entity** can assume either role when executing the Application Firmware Load service. It is also observed that both roles are authenticated using the same authentication data, that is, the EAVK.

The QE always erases its current application (if one exists) and reboots after a request to load application has been initiated, regardless of whether the subsequent application loading process completes or is aborted.

6.2.1 Strength of Authentication

The strength of the authentication mechanism is that of the 2048-bit RSA algorithm. This meets the "1 in 1,000,000" requirement in FIPS 140-2 (c.f. **AS03.25** of "*DTR for FIPS 140-2*").

For multiple attempts to use the authentication mechanism during a one-minute period, the probability remains significantly small. The quickest time an authentication attempt can be performed is no less than the time taken to perform a reboot of the module, since the module always has to be rebooted each time an application is loaded.

It takes approximately 6 seconds for the QE to power up and perform self-tests, right up to the point where the kernel verifies the digital signature of the application. Thus, at most ten authentication attempts can be made within a minute. It follows that the probability that a random attempt will succeed within a minute is less than the "one in 100,000" requirement in FIPS 140-2 (c.f. **AS03.26** of "*DTR for FIPS 140-2*").[1]

There is no feedback of any authentication data to the operator during an authentication session.

6.3 Protection of Authentication Data

The only authentication data that is used in the authentication of the **entity** within the cryptographic boundary of the QE is the EAVK. This public key is stored in the *key-bank*, a region in NVRAM that is used to store CSPs and SPs.

The EAVK is protected by both software protection and physical security means. Software means ensure that the EAVK cannot be modified or over-written. This is achieved through the *permission mask* mechanism that is implemented for all keys that are stored in the key-bank. The EAVK is always initialized a permission-mask value that does not allow the key to be *modified* or *written*, only executed. In this way, the QE achieves protection against unauthorized modification or substitution of the EAVK. See DC/QE-KM-001, "*d'Cryptor QE Cryptographic Module – Cryptographic Key Management*" for more details on this aspect of the key management.

Refer to Section 11.2 for the physical security mechanisms.

6.4 Initialization of Authentication Data

During the final stage of manufacturing in the factory, the EAVK is installed into the QE. The RSA key pair comprising the EAVK (public key) and the EASK (private key) is generated by the factory Certificate Authority. The EASK remains outside the module, and is kept by and known only to the **entity** who is responsible for creating and signing the application that is to be loaded into the module.

[1] Let P = Prob(an authentication attempt succeeds). Because the authentication uses 2048-bit RSA security, $P < 10^{-6}$. We then have Prob(at least one of 10 authentication attempts succeeds) = 1–Prob(all 10 authentication attempts fail) $= 1 - (1 - P)^{10} \approx 1 - (1 - 10P)$ [because P is very small] $\approx 10P < 0.00001$.

7 Services

7.1 Operator Services

For the current validation of the QE as a cryptographic coprocessor for secure loading of an application, only the following list of services are applicable to operators of the module:

Application Firmware Load	Authenticated service, available only to **crypto-officer/user**
Module Restart	Un-authenticated service, available to any operator without need for authentication
Module Shutdown	
Module Zeroize-all	
Module Zeroize	
Log Dump	

Services are always initiated from the QE main menu. This main menu is displayed via the diagnostic port after the QE has successfully powered up, as shown in Figure 6.

```
MAIN  : (L)oad (B)oot (S)hutdown (z)eroise (Z)eroise-all (D)ump ?
```

Figure 6. The QE Main Menu

Figure 7 shows a menu tree of the available services.

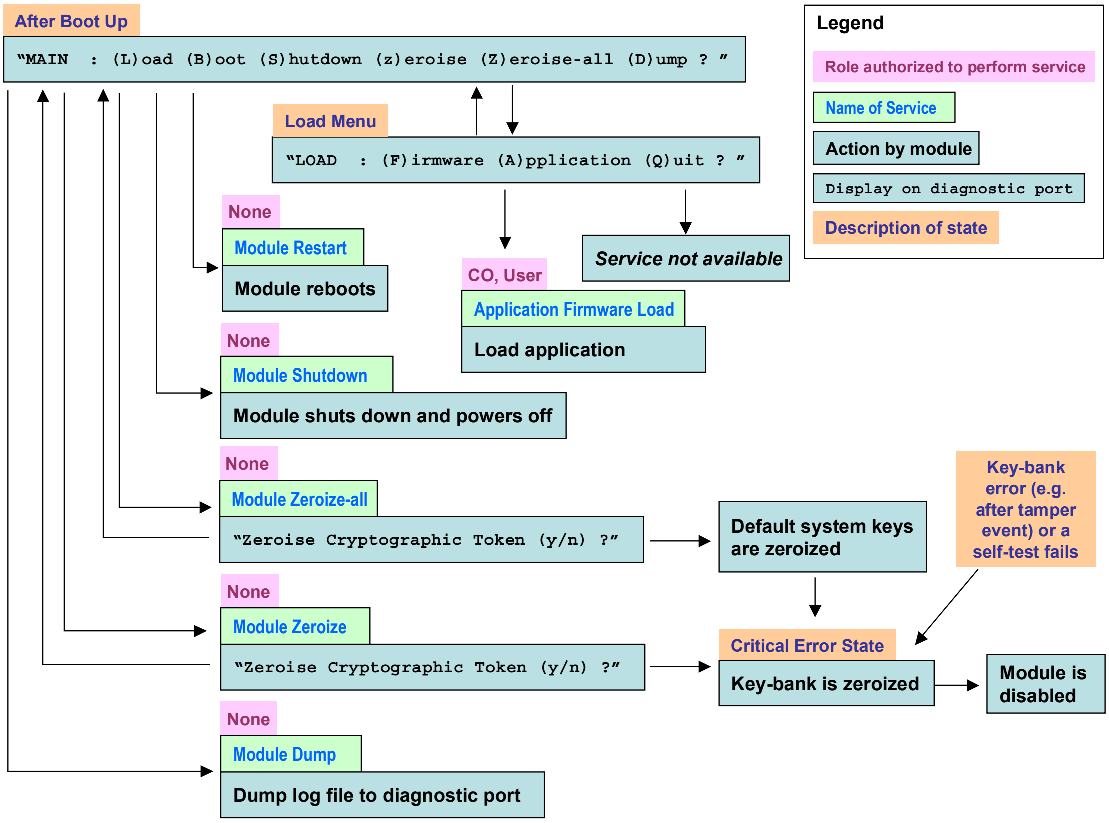
After Boot Up
"MAIN : (L)oad (B)oot (S)hutdown (z)eroise (Z)eroise-all (D)ump ? "
Load Menu
"LOAD : (F)irmware (A)pplication (Q)uit ? "
None
Module Restart
Module reboots
Service not available
CO, User
Application Firmware Load
Load application
None
Module Shutdown
Module shuts down and powers off
None
Module Zeroize-all
"Zeroise Cryptographic Token (y/n) ?"
Default system keys are zeroized
Key-bank error (e.g. after tamper event) or a self-test fails
None
Module Zeroize
"Zeroise Cryptographic Token (y/n) ?"
Critical Error State
Key-bank is zeroized
Module is disabled
None
Module Dump
Dump log file to diagnostic port
Legend
Role authorized to perform service
Name of Service
Action by module
Display on diagnostic port
Description of state

Figure 7. The QE Menu Tree

7.2 Authenticated Service

Application Firmware Load: This service allows an external application to be loaded into the QE through the application loading port using the Xmodem protocol.

7.3 Un-authenticated Service

Module Reboot: This service reboots the QE.

Module Shutdown: This service puts the QE in sleep mode. Power consumption is minimized when the QE is in sleep mode and this is a suitable mode to be in for long-term storage of cryptographic keys in the NVRAM.

Module Zeroize-all: This service zeroizes all keys in the QE, including all the keys in the key-bank and the default system keys in the System Firmware. After performing this service, the module has to be returned to the factory for recovery.

Module Zeroize: This service zeroizes all keys in the key-bank of the QE. After performing this service, the module has to be returned to the factory for recovery.

Log Dump: All status messages that are output through the diagnostic port and printed using `printf` or `lprintf` are logged to the Log File System (see Figure 3). The contents of this log can be read through the diagnostic port via the **Log Dump** service.

7.4 Mandatory Services

This section explains how the services mandated by the FIPS 140-2 requirements are implemented by the QE.

7.4.1 *Show Status*

The current status of the QE can be observed via the diagnostic port and can be displayed on a PC's terminal console using any serial communication program.

The **Log Dump** service allows the viewing of status messages (a maximum of 256KB) that had been output via the diagnostic port.

7.4.2 *Perform Self-Tests*

The power-up self-tests can be initiated by one of the following two methods:

- Power cycle the QE;
- Perform the service **Module Restart**.

The results of the self-tests are sent out via the diagnostic port as they are being executed.

7.4.3 *Perform Approved Security Function*

For the current validation, the QE employs only one Approved security function, i.e. the RSA verification of the signed application that is loaded. This function is performed during the power-up self-tests.

7.5 Other Cryptographic Functions

For future applications that are loaded into the module, the QE provides a host of cryptographic and non-cryptographic services through its internal APIs (see Annex A). A full description of all services provided by the QE can be obtained from DC/QE-SM-001, "*d'Cryptor QE Cryptographic Module – Software Developer Manual*".

7.5.1 Random Number Generator

The QE provides a FIPS-approved DRNG via the **Pseudo-Random Number Generate** service. This is an implementation of the ANSI X9.31 standard, and its usage is described in DC/QE-SM-001, "*d'Cryptor QE Cryptographic Module – Software Developer Manual*".

7.6 Services Authorized for Roles

Table 4 shows what roles can perform which services. An "**x**" signifies that the role in that row is authorized to perform the corresponding service in that column. A "–" implies otherwise.

Role	Module Restart	Module Shutdown	Module Zeroize-all	Module Zeroize	Log Dump	Application Firmware Load
crypto-officer	–	–	–	–	–	x
user	–	–	–	–	–	x
"no role" state	x	x	x	x	x	–

Table 4. Services Authorized for Roles

7.7 Access Rights within Services

Table 5 shows the types of access that a service has to CSPs/SPs in the module.

Service	SPs / CSPs	Access Type(s)
Module Restart	None	–
Module Shutdown	None	–
Module Zeroize-all	All keys in the QE	Write (Erase)
Module Zeroize	All keys in the key-bank	Write (Erase)
Log Dump	None	–
Application Firmware Load	External Application Verification Key (EAVK)	Access (Execute)

Table 5. Access Rights within Services

8 Access Control Policy

8.1 Roles

The QE supports exactly one identity (**entity**) and two roles (**crypto-officer** and **user**). As both roles perform exactly the same function (in loading the application and nothing else), the authentication of the **entity** implicitly implies that the **entity** assumes either the **crypto-officer** or the **user** role. Authentication of the identity of an operator and the authorization of the operator to assume its default assigned role is thus automatically achieved via the authentication mechanism described in Section 6.2.

There are no provisions for an authenticated operator to change roles or to assume a set of roles other than his default assigned role.

8.2 Access to Services

The availability of services to the roles in the QE has been covered in Section 7.5.

8.3 List of CSP and SPs

The CSPs and SPs that are relevant for the current validation of the QE are listed in Table 6. Note that the EAVK is one of the seven system keys in the QE. Note also that the other six system keys are not accessible at all by either the **crypto-officer** or the **user**, and can be used only when an application is loaded in the QE. See also Section 13.1.

Security Parameters	Type	Description	Modes of Access	
			crypto-officer	user
COAK	Symmetric key (CSP)	Authenticates the CRYPTO-OFFICER[2]	None	None
NUAK	Symmetric key (CSP)	Authenticates the NORMAL-USER[3]	None	None
Incoming Public KEK	Public key (SP)	Encrypts keys that exit the QE	None	None
Incoming Private KEK	Private key (CSP)	Decrypts keys that enter the QE	None	None
EAVK	Public key (SP)	Used internally by the kernel to verify the integrity of the field application that is to be loaded into the QE.	Execute	Execute
DRNG Key	Symmetric key (CSP)	Key for DRNG	None	None
DRNG Seed	Secret seed (CSP)	Seed for DRNG	None	None

Table 6. Modes of Access to Security Parameters

[2] Note that this "CRYPTO-OFFICER" is not the same as the **crypto-officer** role described in this security policy. This takes on a different set of roles and responsibilities held by a crypto officer as defined in FIPS 140-2, and will only be relevant when the QE is loaded with an application that makes calls to cryptographic services in this Section.

[3] As in CRYPTO-OFFICER, the NORMAL-USER role that is relevant to the QE only when an application is loaded.

9 Self-Tests

The QE performs a series of self-tests during power-up and on-demand to ensure that all the cryptographic operations it provides are functioning properly. Two types of self-tests are implemented: power-up self-tests, which are performed when the QE is powered up, and conditional self-tests, which are performed whenever a security function is invoked.

If any of the self-tests (other than the memory test) fails, the QE immediately enters a *Critical Error state* and repeatedly zeroizes all keys in the key-bank, thus leaving the QE in a zeroized and unusable state and requiring a return of the module to the factory for recovery.

9.1 Power-Up Self-Tests

9.1.1 Memory Test

This test is carried out during the early stages of the bootstrap, and performs a rudimentary read/write test on both Flash and RAM. Upon detecting a critical memory error, the module halts and needs to be reset.

9.1.2 Cryptographic Algorithm Tests

These are *known-answer test* (KAT) and are performed for each of the cryptographic algorithms that are implemented by the QE. See Table 7.

Cryptographic Algorithm	Modes Tested and Keys Used	KAT Test Performed
DES	ECB, CBC, CFB64, OFB64 64-bit key	Encryption, Decryption
TDEA	ECB, CBC, CFB64, OFB64 192-bit key	Encryption, Decryption
AES	ECB, CBC, CFB128, OFB128 128, 192 and 256-bit keys	Encryption, Decryption
RSA	512-bit key	Encryption, Decryption
SHA-1	–	Hash of 1024-byte data
HMAC-SHA-1	160-bit HMAC key	Key-hash of 1024-byte data
DRNG (ANSI X9.31)	128-bit DRNG key (2-key TDEA), 64-bit seed	Two 64-bit blocks of random numbers

Table 7. Cryptographic Algorithm Test

9.1.3 Firmware Integrity Test

The module uses a 16-bit error detection code (EDC) to verify the integrity of the kernel upon power-up. This test is performed before any of the other power-up self-tests, as there is no point in conducting other tests if the firmware itself fails an integrity check, possibly through data corruption in the flash.

9.1.4 Critical Function Test

The module performs two critical functions test:

Key Validity Test: This verifies each key in the key-bank through a computation of its CRC. The test fails if any key cannot be successfully verified.

DRNG Test: As part of the power-up self-tests, the DRNG generates a number of blocks of output, with each block comprising 64 bits (size of output ciphertext in TDEA algorithm). The first block is checked to see if it is all zeros, and each subsequent block is checked against its previous block for equality. The test fails if any check gives an affirmative result.

9.2 Conditional Tests

9.2.1 Pair-wise Consistency Test

The pair-wise consistency test is performed each time a RSA key pair is generated using the service **Generate RSA Key Pair** (see Annex A). It comprises one of the following two tests:

- Perform encryption/decryption test for each key pair generated for encryption purposes.
- Perform signing/verification test for each key pair generated for signing purposes.

9.2.2 Firmware Load Test

An application firmware is loaded using the service **Application Firmware Load**. Upon each power up, the QE uses the EAVK to verify that the application firmware has been correctly signed using the EASK. If verification fails, the QE automatically erases the application and displays the main menu.

9.2.3 Continuous Random Number Generator Test

This test is performed each time the DRNG is called, and test the DRNG for failure to a constant value as described in Section 9.1.4. The first 64 bits generated by the DRNG after power up are not used since the DRNG is already called upon to generate 313 blocks of output as part of the critical function test during power-up self-test.

9.2.4 Building DES Key Schedule

This test is performed when calling the service **DES Context Init**, and verifies that the DES key has odd parity and is not a weak key.

9.2.5 Building TDEA Key Schedule

This test is performed when calling the service **TDEA Context Init**, and verifies that the associated DES keys have odd parity and are not weak keys.

10 Zeroization of CSPs/Cryptographic Keys

The module provides both hardware and software methods to zeroize CSPs and other cryptographic keys in the key-bank. Hardware means for zeroization is described in the last paragraph of Section 11.2. Software means for zeroization is achieved via calling the service **Module Zeroize-all**.

11 Physical Security Policy

11.1 Physical Embodiment

The QE is a multi-chip embedded cryptographic module.

11.2 Physical Security Mechanisms

Tamper detection is provided by a hard, opaque epoxy that covers the RAM and the Flash (as well as other electronic components). In addition, the module's tamper detection and response instantly zeroizes the entire key-bank upon detection of tamper.

11.3 Physical Security Checks

The following physical checks (Table 8) on the QE should be carried out periodically to ensure that physical security is maintained:

Physical Security Mechanism	**Recommended Frequency of Inspection/Test**	**Inspection/Test Guidance Details**
Hard opaque enclosure	Once every 6 months	Examine all exposed surfaces of the epoxy cover and look for any signs of attempted tamper such as scratches.
Tamper detection and response	Once every two years, or 80% of the expected lifetime of the battery, whichever comes first	Power up the QE and connect the diagnostic port to a console terminal. Trigger the tamper detection circuit by probing the tamper detection guard ring (see Figure 1) at the bottom side of the QE, using a measuring probe (e.g. oscilloscope or multi-meter probe operating in DC mode). The QE should instantly reboot and display the following message through the diagnostic port: `The cryptographic token is permanently disabled!` Reboot the QE three more times and verify that each time, the QE displays the above message and remains in a state of infinite loop. This shows that all the cryptographic keys in the module have been zeroized and that the tamper detection and response is functional

Table 8. Inspection/Testing of Physical Security Mechanisms

It is important to note that activating the tamper detection and response test **erases** all keys in the device, including the system keys, and thereafter necessitates the return of the module to the factory for recovery.

It should also be noted that the actual frequency of inspections should depend on the application that the QE is used for, as well as the security threat that the QE is exposed to under its operational environment. For highly sensitive applications, it may be necessary to increase the frequency beyond the recommended frequency.

12 Mitigation of Other Attacks Policy

The QE is not designed to mitigate any specific attacks.

13 Secure Operation of QE

13.1 Factory Defaults

A *d'Cryptor* QE is delivered from the factory in an Approved mode of operation, without any application loaded, and installed with a set of seven default cryptographic system keys (also known as *transport keys*) in the key-bank (see Table 6). Other than the EAVK, the rest of the default keys can only be accessed from using APIs within an application that is subsequently loaded into the QE (see Annex A).

13.2 Operating the QE

The QE operates in an Approved mode of operation when it is shipped from the factory. This can be determined by powering up the QE and observing that the printable output displayed via the diagnostic port appears as in Figure 4.

An individual who wishes to assume the identity of the **entity** and to load in an application should ensure the following conditions are satisfied before proceeding:

- The application has been properly signed using the EASK.
- The QE is in an Approved mode of operation.

In addition, if it is desired that the QE continue operating in an Approved mode after loading the application, the application to be loaded needs to be validated to FIPS 140-2 Level 3.

Figure 8 shows the various modes that the QE can transit into depending on whether an application has been loaded and if so, whether it has been FIPS validated.

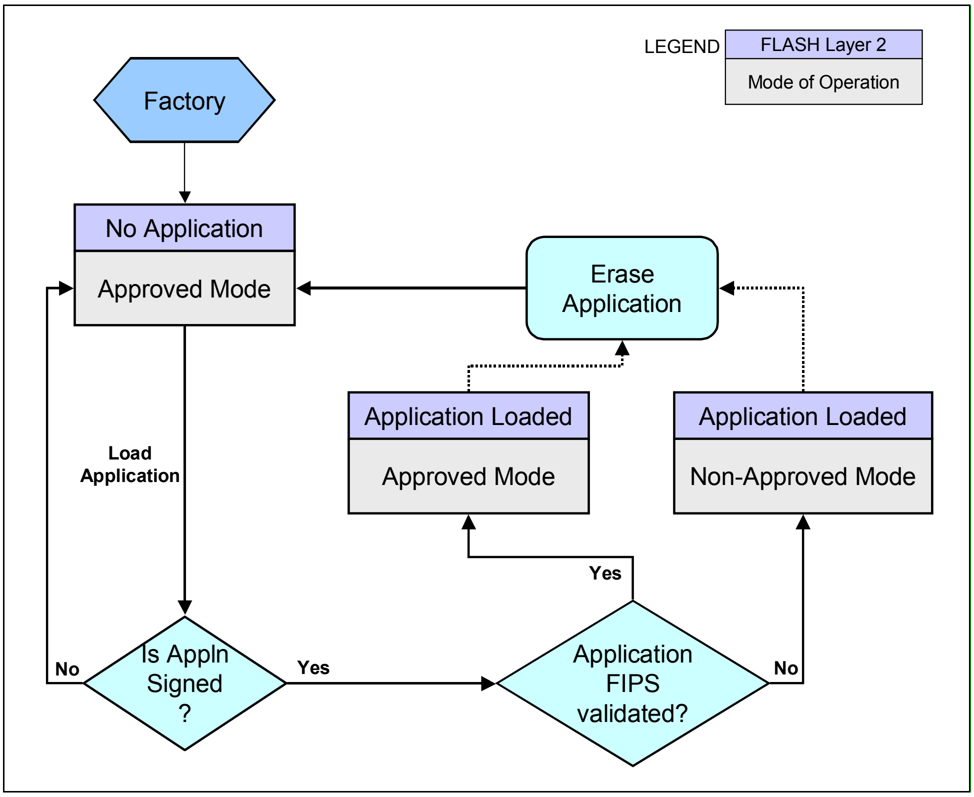

Figure 8. Flow of QE Mode of Operations

13.3 Security Rules

A proper operational security policy should be in place that requires the EASK to be kept under lock and key, and known only to the **entity**. This will ensure the integrity of the **entity** authentication whenever an application is loaded.

Future customers of the QE should replace all the default transport keys (with the exception of the EAVK, which is non-modifiable and non-writeable outside the factory) to ensure that the module is not deployed operationally with default values.

14 Applicable Documents

FIPS Documents:

Name of Document		Date
FIPS 140-2	Security Requirements for Cryptographic Modules (With Change Notices 1, 2, 3, 4)	May 25, 2001
DTR for FIPS 140-2	Derived Test Requirements for FIPS PUB 140-2, *Security Requirements for Cryptographic Modules*	March 24, 2004 (Draft)
ANSI X9.31 – 1998	American Bankers Association, *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*	September 9, 1998

Internal Documents:

Reference Number	Name of Document	Date
DC/QE-FS-001	*d'Cryptor* QE Cryptographic Module – Finite State Model	24 Mar 2004
DC/QE-KM-001	*d'Cryptor* QE Cryptographic Module – Cryptographic Key Management	15 Mar 2004
DC/QE-MS-001	*d'Cryptor* QE Cryptographic Module – Module Specifications	24 Mar 2004
DC/QE-SM-001	*d'Cryptor* QE Cryptographic Module – Software Developer Manual	15 Mar 2004

15 Glossary

15.1 Acronyms

ANSI	American National Standard Institute
CMT	Cryptographic Module Testing
COAK	Crypto-Officer Authentication Key
CSP	Critical security parameter(s)
DRNG	Deterministic Random Number Generator
DTR	Derived Test Requirements
EASK	External application signing key
EAVK	External application verification key
KEK	Key-Encryption Key
FIPS	Federal Information Processing Standards
NUAK	Normal-User Authentication Key
SP	Security Parameter(s)

15.2 Definitions

key-bank An array of keys stored in the non-volatile memory of the QE. Keys in the key-bank are identified by non-negative indices that run from `0` to **`MAX_KEYS`**–1, where **`MAX_KEYS`** is the maximum number of keys that can be stored in the key-bank. **`MAX_KEYS`** is currently set to 256.

key-index A non-negative integer that identifies the key in the key-bank

key pair A pair of keys that are related cryptographically. Two types of key pairs are used in the QE: RSA key pair and DRNG/SEED key pair. A RSA key pair comprises a public key and a private key. A DRNG/SEED key pair comprises a TDEA key, and a seed for seeding the DRNG.

mode of operation The mode in which the QE is operating. This is either *FIPS mode* (which is the *Approved mode of operation*) or *non-FIPS mode* (any mode which is not an *Approved mode of operation*).

security parameter Security-related information (e.g. public cryptographic keys) whose modification can compromise the security of a cryptographic module. Note the distinction between *SP* and *CSP*. The *disclosure* of a SP does not affect the security of the module.

system keys A set of seven cryptographic keys that are installed by default in the QE. These keys occupy key-indices 0 to 6.

trusted application An application that has been FIPS 140-2 validated.

Annex A. Services Available in the QE

This Annex briefly describes the full list of services that are available to future applications that may be loaded in the QE. The list provided here is for informational purposes only and is not relevant to the current validation of the QE.

Access to any cryptographic service in the QE is governed by a simple yet elegant mechanism of assigning permission bits to keys. See DC/QE-KM-001, "*d'Cryptor QE Cryptographic Module – Cryptographic Key Management*" for details on this mechanism.

A.1 List of Services

The services in the cryptographic module are grouped under the following service modules:

- Key Management Service Module
- Crypto Service Module
- Operator Management Service Module
- System Management Service Module
- Utilities Service Module

All cryptographic services in the QE are contained in the first three service modules. These are listed in Table 9 to Table 11. Each of these modules is called a *cryptographic service module*. The services in the other two service modules listed in Table 12 and Table 13 are non-cryptographic. For a full description of all the services supported by the QE, see DC/QE-SM-001, *"d'Cryptor QE Cryptographic Module – Software Developer Manual*".

Key Management Service Module	
Get Key Type	Returns key type
Get Key Mask	Returns key mask
Get Key Size	Returns size of keying material
Get Key Link	Returns link of key
Get Key Type Mnemonic	Returns type of key in the form of 4-char mnemonic
Read Key	Exports a key from the key-bank (in encrypted form)
Modify Key	Updates the keying material for a key
Write Key	Installs a (encrypted) key into the key-bank

Table 9. Services in the Key Management Service Module

Crypto Service Module	
Generate Symmetric Key	Generates a random symmetric key
Generate RSA Key Pair	Generates a random RSA key pair
DES Context Init*	Initializes DES context in preparation for a DES operation
DES Context Execute*	Performs DES encryption/decryption using a DES context
DES Context Quit*	Frees a DES context.
DES Execute*	Performs DES encryption/decryption in ECB/CBC/CFB/OFB modes
TDEA Context Init	Initializes TDEA context in preparation for a TDEA operation
TDEA Context Execute	Performs TDEA encryption/decryption using a TDEA context

Crypto Service Module	
TDEA Context Quit	Frees a TDEA context.
TDEA Execute	Performs TDEA encryption/decryption in ECB/CBC/CFB/OFB modes
AES Context Init	Initializes AES context in preparation for an AES operation
AES Context Execute	Performs AES encryption/decryption using an AES context
AES Context Quit	Frees an AES context.
AES Execute	Performs AES encryption/decryption in ECB/CBC/CFB/OFB modes
RSA Context Init	Initializes RSA context in preparation for an RSA operation
RSA Context Execute	Performs RSA encryption/decryption or sign/verify using PKCS#1/ANSI X9.31/ANSI X.509 mechanisms and an RSA context
RSA Context Quit	Frees an RSA context.
RSA Execute	Performs RSA encryption/decryption or sign/verify using PKCS#1/ANSI X9.31/ANSI X.509 mechanisms
SHA-1 Context Init	Provide SHA-1 services
SHA-1 Context Execute	
SHA-1 Context Quit	
SHA-1 Execute	
HMAC Context Init	Provide HMAC services
HMAC Context Execute	
HMAC Context Quit	
HMAC Execute	
Deterministic-Random Number Generate	Generates pseudo-random bytes

* For legacy use only

Table 10. Services in the Crypto Service Module

Operator Management Service Module	
Get Current User Role	Returns the current role of the QE
CO Auth Challenge	Provides identification & authentication services for the CO[4]
CO Auth Response	
CO Auth Exit	
Normal-User Auth Challenge	Provides identification & authentication services for the Normal-User
Normal-User Auth Response	
Normal-User Auth Exit	

Table 11. Services in the Operator Management Service Module

[4] Note that this "CO" is not the same as **crypto-officer** role described in this security policy. The CO takes on a different set of roles and responsibilities held by a crypto officer as defined in FIPS 140-2, and will only be relevant when the QE is loaded with an application that make calls to cryptographic services in this Section.

System Management Service Module	
Application Firmware Load	Loads external application program
Application Firmware Install	Installs application program
Module Shutdown	Shutdowns the QE after a specified delay
Module Reboot	Reboots the QE after a specified delay
Get Operating Mode	Returns the security mode the QE is currently in
Set Non-FIPS Operating Mode	Sets the QE to the non-FIPS mode of operation

Table 12. Services in System Management Service Module

Utilities Service Module	
NVRAM Write	Writes data to the non-volatile RAM
NVRAM Size	Returns size of the NVRAM in bytes
NVRAM Read	Reads data from the non-volatile NVRAM
Flash Memory Write	Writes data to the flash memory
Flash Memory Erase	Erases data from the flash memory
Flash Memory Size	Returns size of the flash memory in bytes
Get Factory ID	Returns the factory ID of the QE
Get User ID	Returns the unique identifier of the QE
Get Serial Number	Returns the serial number of the QE
Get Firmware Version	Returns the firmware version of the QE
Get Hardware Version	Returns the hardware version of the QE
DMA Channel Open	Opens a channel for DMA transfer
DMA Transfer Start	Starts a DMA transfer
DMA Transfer Continue	Continues a DMA transfer that has been stopped
DMA Transfer Stop	Stops a DMA transfer

Table 13. Services in Utilities Service Module