TITLE:	LunaRA Security Policies
ABSTRACT:	This document describes the security policies implemented by the LunaRA PC Card (LunaRA token) and how the design of its firmware enforces these policies.
DOCUMENT NUMBER:	CR-0729
ORIGINATOR:	Bob Woodard
DEPARTMENT:	Systems Engineering
LOCATION OF ISSUE:	Ottawa
DATE ORIGINATED:	February 12, 2001
CHANGE LEVEL:	7
CHANGE DATE:	September 7, 2001
SECURITY LEVEL:	None
SUPERSESSION DATA:	CR-0729, 6

Document Approvals			
Name	**Position**	**Date**	**Signature**
Robert Woodard	Manager, PKI Solutions Software Development		
Mike Haines	Director, Software Development		

TABLE OF CONTENTS

1. INTRODUCTION **4**

1.1. PURPOSE 4
1.2. SCOPE 4
1.3. INTENDED AUDIENCE 4
1.4. HISTORY OF REVISION 4
1.5. REFERENCES 4
1.6. GLOSSARY OF ACRONYMS / ABBREVIATIONS 4

2. LUNARA OVERVIEW **5**

3. SECURITY POLICY TOOLS **5**

3.1. FIXED POLICY VECTOR (FPV) 5
3.1.1. Number of SO Login Fails Allowed 6
3.1.2. Secret Key Policy 6
3.1.3. Private Key Policy 6
3.1.4. Token Security Policy 7
3.2. TOKEN POLICY VECTOR (TPV) 8
3.2.1. Number of User Login Fails Allowed 8
3.2.2. Minimum/Maximum PIN Length 8
3.2.3. Local Policies 8

4. IDENTIFICATION AND AUTHENTICATION (I&A) **9**

5. DISCRETIONARY ACCESS CONTROL (DAC) **10**

6. OBJECT REUSE **10**

APPENDIX A. CRYPTOGRAPHIC ALGORITHMS SUPPORT **11**

APPENDIX B. POLICY VECTOR SETTINGS **13**

APPENDIX C. SESSION AND LOGIN STATES REQUIRED FOR LUNA COMMANDS **14**

1. INTRODUCTION

1.1. Purpose

This document describes the security policies implemented by the LunaRA PC Card (*LunaRA token*) and how the design of its firmware and hardware enforces these policies.

1.2. Scope

This document addresses the LunaRA token's security policies.

1.3. Intended Audience

The intended audience for this document is the LunaRA Engineering and Product Management Team, external agencies for validation or endorsement of the LunaRA token; selected industry partners; and potential users of LunaRA tokens who want to understand the security policies of the product for FIPS-compliant operations.

The reader of this document should be familiar with the PKCS#11 standard defined by RSA Laboratories.

1.4. History of Revision

Revision	**Date**	**Description**
Draft	February 12, 2001	Creation of document.
1	March 6, 2001	Formatting changes.
2	March 16, 2001	Incorporation of review comments from T. Casar.
3	March 27, 2001	Change of Security Level to "None" and new proprietary statement.
4	April 11, 2001	Incorporation of B. Woodard's comments for FPV_USE_CAV.
5	July 17, 2001	Corrected FPV/TPV force RSA blinding information (resolves Razor Issue #484).
6	July 20, 2001	Revision required for CorpDocs system.
7	September 7, 2001	Updated Appendix A

1.5. References

Document Number	**Revision**	**Author**	**Title**
PKCS#11	v2.10	RSA Laboratories	PKCS#11: Cryptographic Token Interface Standard, December 1999

1.6. Glossary of Acronyms / Abbreviations

Shortforms	***Longform Explanation***
CAV	Cryptographic Algorithm Vector
CCM	Custom Command Module
CRC	Cyclic Redundancy Check
DAC	Discretionary Access Control
FPV	Fixed Policy Vector
I&A	Identification and Authentication
OEM	Original Equipment Manufacturer
PIN	Personal Identification Number
SO	Security Officer

Shortforms	***Longform Explanation***
SRAM	Static Random Access Memory
TPV	Token Policy Vector
UAV	User Authorization Vector

2. LUNARA OVERVIEW

The LunaRA token securely stores data and keying material inside its cryptographic boundary. The LunaRA token also performs cryptographic operations on data provided by external applications using the keying material stored in the token. These abilities are defined as key management, object management, and cryptographic capability.

Before a LunaRA token can be used to perform any cryptographic or key/object management functions, the token must receive a valid operator identity (also known as a user number), as well as a valid authentication code (a PIN). These two inputs are processed by the token during a "LOGIN" command. When this command has completed successfully, the token allows the user to perform operations based on the policy settings defined for that token.

The token has the ability to distinguish two categories of users: super-users and normal users. The super-user category is referred to as the Security Officer (SO) and the normal user category is referred to as the user. A token can have only one SO. The SO is allowed to perform all of the cryptographic, key and object management functions provided by the token, as well as a set of functions called the SO functions. These SO functions are available only to the SO, and allow the SO to manage the token users and security policies.

For a LunaRA token, there is no limit on the number of users that can be created by the SO. All users are subjected to the same policy settings as those established by the SO. However, each user has his or her own authentication code (or PIN) initially assigned under control of the SO, which is used internally to protect the data the user owns.

The LunaRA token meets and is validated against FIPS 140-1 level 2 security requirements. For example, one aspect of FIPS 140-1 level 2 physical security is through tamper evidence provided by the case: an attacker cannot get into the LunaRA token and access plain-text keys in an operational state without leaving evidence of tampering. Contact Chrysalis-ITS for more details of the physical security used to protect the LunaRA token.

The LunaRA is a special Luna token variant that was developed for use in a Registration Authorization environment. As such, the capability of being able to extract RSA private keys generated on the token is a requirement of LunaRA. In addition, non-private keys may be cloned between LunaRAs within the same cloning domain. In most other respects, the LunaRA resembles the Luna2 token.

3. SECURITY POLICY TOOLS

The LunaRA token provides two levels of security policy enforcement. A vector that is loaded on the token during manufacturing dictates the first level of security. This vector, called the Fixed Policy Vector (FPV), establishes security policies that cannot be modified after manufacturing.

The second level of security is provided by a policy vector that can be modified by the token's SO. This vector is called the Token Policy Vector (TPV), and consists of a series of policy settings that can be established and modified by the token's legitimate SO.

3.1. Fixed Policy Vector (FPV)

The FPV contains the settings necessary to enforce policy rules that apply across a wide range of token users. For example, one bit in the FPV defines whether the token can be exported. In an exportable version, the token provides a reduced set of algorithms and imposes limitations on maximum key lengths as required by export regulations.

The FPV cannot be modified by the SO or any of the users. The FPV is put on the token during manufacturing and remains in place until the token is destroyed or the firmware is erased. The integrity of the FPV is maintained through the same mechanism used to protect the executable code from being modified – a 32-bit Cyclic Redundancy Check (CRC) computation.

The format of the FPV is a 32-bit vector that is divided into four fields of eight bits. These fields, and their contents, are defined in the following sections.

3.1.1. Number of SO Login Fails Allowed

This field defines the number of consecutive failed login attempts that can be made by the SO before the token erases the flash memory to prevent illegal access to its contents.

This security measure prevents an impostor from cracking the SO's password on the token.

3.1.2. Secret Key Policy

The following table defines the flags that identify the security policies that are followed for secret key objects.

Name	Description
FPV_SECRET_KEY_SENSITIVE	This bit determines whether a secret key object must always be made sensitive or if it can be determined by the high-level application using the token. When this bit is set, all secret keys stored on the token are sensitive. The keys are encrypted when in the flash memory and they can be extracted only outside of the token in encrypted form using the GESC_WRAP_KEY function. This bit must be set for FIPS-compliant tokens using automated key establishment.
FPV_SECRET_KEY_NO_CREATE	This bit determines whether a secret key object can be created by an external application using the token, instead of being generated by the token. When this bit is set, an external application cannot create a secret key on the token; it is not possible to enter a secret key in plain text form on the token. This bit must be set for FIPS-compliant tokens using automated key establishment.

3.1.3. Private Key Policy

The following table defines the flags that identify the security policies that are followed for private key objects.

Name	Description
FPV_PRIVATE_KEY_SENSITIVE	This bit determines whether a private key object must always be made sensitive or if it can be determined by the high-level application using the token through PKCS#11. When this bit is set, all private keys stored on the token must be flagged as sensitive whether or not the high-level application requested this flag when the keys were created. When this bit is set, all private keys are encrypted while stored in flash memory. **Note:** After a private key is sensitive, it cannot be extracted from the token even in encrypted format. This bit must be set for FIPS-compliant tokens using automated key establishment.
FPV_PRIVATE_KEY_NO_CREATE	This bit determines whether a private key object can be created by an external application using the GESC_CREATE_OBJ call, instead of being generated by the token. When this bit is set, an external application cannot create a private key on the token; it is not possible to enter a private key in plain text form on the token. This bit must be set for FIPS-compliant tokens using automated key establishment.

3.1.4. Token Security Policy

The following table defines the flags that identify the security policies that dictate the behavior of the token in general.

Name	Description
FPV_XPPLUS_TOKEN	This bit indicates that the token is built upon XPplus-style hardware. XPplus hardware has: asymmetric math accelerators; extra RAM; non-volatile RAM; tamper detection mechanisms which trigger interrupts and wipe non-volatile RAM. This bit is not set on the LunaRA token.
FPV_XP_TOKEN	This bit determines if the token is used in an XP style functionality, thus allowing the KCV to be set indirectly (allows XP tokens to get a CA^3's domain vector). This allows the token to be initialized as either a FIPS Level 2 or 3 device at InitToken time; all keys must be volatile. This bit is not set on the LunaRA token.
FPV_RA_TOKEN	This bit determines if the token has the RA functionality. In addition to the cloning restriction of private key objects (see FPV_ENABLE_CLONING), special component key wrapping is associated with the LUNA_MECH_3DES_ECB mechanism. This bit is set on the LunaRA.
FPV_DOMESTIC_FLAG	This bit determines whether the token can be exported. When this bit is set, the token is configured for the domestic market and cannot be exported. This bit is verified internally every time a cryptographic function implying an encryption or a decryption is performed. If the bit is set, no restrictions exist on key sizes. If the bit is not set, a limitation of 56 bits is applied to any symmetric keys used for encryption or decryption, and a 512-bit limitation on asymmetric keys used for wrapping and unwrapping operations. Signature and verification operations are not restricted in terms of key lengths. This bit is set on the LunaRA token.
FPV_ENABLE_CLONING	This bit determines whether sensitive objects on the token can be "cloned" to another similarly enabled token. When this bit is set, cloning is enabled. For a LunaRA token, this bit is set, however this bit is tested in conjunction with the FPV_RA_TOKEN bit to restrict the cloning of objects with the private key attribute set (i.e., CKO_PRIVATE_KEY).
FPV_USE_CAV	This bit is used by the firmware to determine if the CAV should be checked to validate the desired algorithm. Normally, this bit is zero, which assumes all algorithms are valid.
FPV_WRAPPING_TOKEN	This bit determines whether RSA private keys can be wrapped. When this bit is set, an RSA private key can be wrapped. Although not normally set on most Luna token variants, this bit is set on the LunaRA.
FPV_USE_M_OF_N	This bit defines whether the token can perform M of N activation. When this bit is set, the token can be configured to perform M of N activation. M of N activation is not a feature ordinarily enabled on a LunaRA token.
FPV_USE_RAW_RSA	This bit determines whether RAW RSA operations can be performed on the token. When this bit is set, RAW RSA operations are allowed. RAW RSA provides access to RSA to perform encrypt and decrypt operations on data without any padding. This bit is set on the LunaRA token.
FPV_SPECIAL_CLONING	This bit determines whether the token allows the factory-default Chrysalis-ITS key cloning certificate to be modified. When this bit is set, customers can create their own key cloning certificate. Key cloning is a feature enabled on a LunaRA token, and this bit is set.
FPV_ENABLE_CCM	This bit determines whether a Custom Command Module (CCM) can be loaded onto the token. When this bit is set, a CCM can be loaded onto the token. This bit must be cleared (i.e., zero) for FIPS-compliant tokens.
FPV_CCM_PRESENT_FWUPDATE	This bit determines whether a CCM must be present before a firmware update operation is allowed to proceed. When this bit is set, a CCM must be loaded on the token to perform a firmware update. Additionally, the CCM must implement the PreModuleUpdate function. This bit is for an OEM version of LunaRA and does not apply to the Chrysalis-ITS labeled product.
FPV_FORCE_RSA_BLINDING	This bit determines whether the token must perform blinding to introduce a random element to the time needed to complete an RSA operation. Blinding defeats timing attacks on an RSA operation. The token firmware performs blinding regardless of this setting.

Name	Description
FPV_PIN_MUST_USE_SP	This bit determines if the serial communication port must be used to enter an authentication code. When this bit is set, an authentication code can only be entered through the serial communication port. When this bit is cleared, authentication codes are entered via the host computer. Use of the serial communication port is NOT a feature of a LunaRA token; this bit is clear on a LunaRA token.
FPV_MOFN_MUST_USE_SP	This bit determines if the serial communication port must be used to enter the M of N secret. When this bit is set, the M of N secret can only be entered through the serial communication port. When this bit is cleared, the M of N secret is entered via the host computer. Use of the serial communication port is NOT a feature of a LunaRA token; this bit is clear on a LunaRA token.
FPV_KCV_MUST_USE_SP	This bit determines if the serial communication port must be used to enter the key cloning domain identifier. When this bit is set, the key cloning domain identifier can only be entered through the serial communication port. When this bit is cleared, the key cloning domain identifier is entered via the host computer. Use of the serial communication port is NOT a feature of a LunaRA token; this bit is clear on a LunaRA token.

3.2. Token Policy Vector (TPV)

The TPV contains the settings necessary to enforce policy rules locally in an organization. For example, one bit in the TPV defines whether the token can perform a signature operation using a signing key generated by an outside process or if it must use an internally generated key for this function. The TPV can be modified by the token's SO. The TPV contents are used by the internal code to validate the operations performed by the token's USER.

The format of the TPV is a 32-bit vector that is divided into four fields of eight bits. These fields and their contents are defined in the following sections.

3.2.1. Number of User Login Fails Allowed

This field defines the number of consecutive failed login attempts that can be made by a USER before the USER gets locked out or the USER's data is erased. This security feature prevents illegal access to the USER's data and keys: it prevents an impostor from cracking the USER's password on the token. Whether the user is locked out or the data is erased depends upon the "USER zeroize" bit. If the USER zeroize bit is disabled, too many failed login attempts results in the USER getting locked out. In this case, the USER's identity and private data (including key material) are erased from the token. The SO must create a new user in order to continue. The new user will have no association with the previous (deleted) user.

3.2.2. Minimum/Maximum PIN Length

These two fields define the minimum and maximum length restrictions for a USER's PIN.

3.2.3. Local Policies

The following table defines the flags that identify the security policies that dictate the behavior of the users on the token.

Name	Description
TPV_USER_ZEROIZE	This bit determines whether the token can be zeroized by a normal user or if only the token's SO can zeroize the token. When this bit is set, it indicates that a valid token user can zeroize the token. This bit enables using the token in an environment where the SO is not commonly used. When this bit is set, the SO cannot change a user password, and a user is zeroized after too many unsuccessful login attempts.

Name	Description
TPV_USER_FW_UPDATE	This bit determines whether the firmware can be updated by a normal user or if only the token’s SO can update the firmware. When this bit is set, a normal user can perform the firmware update.
TPV_M_OF_N_ACTIVATION	This bit determines whether M of N activation is required for a user to gain access to the token. When this bit and the FPV_SECURITY_POLICY_USE_M_OF_N bit in the FPV are set, the token is not activated until the required number of parts to a split secret has been entered. Ordinarily, this bit is not set for LunaRA tokens.
TPV_KEY_ATTRIB_LOCK	This bit determines whether the flag attributes of a key can be modified once the key is a valid object on the token. When this bit is set, it indicates that the flag attributes of a key cannot be modified after they have been established. For example, if this bit is set and a DES key is created for encryption and decryption, these attributes cannot be changed to wrap and unwrap once the key exists on the token.
TPV_KEY_SINGLE_FUNCTION	This bit determines whether a key can be used to perform multiple types of operations (i.e., use a key for encrypting, signing, and wrapping). When this bit is set, it indicates that keys can be used only to perform single functions. For symmetric keys, a single function is considered to be a pair of related functions such as encryption/decryption, wrapping/unwrapping, or sign/verify. With the validated release of LunaRA, multiple use of a key is allowed regardless of the value of TPV_KEY_SINGLE_FUNCTION.
TPV_SIGNING_KEY_LOCAL	When performing a signing operation, the private key used may have been generated locally or provided by an external source. In most environments, it is preferable to have the signing/verifying key pair generated by the token and never extracted from it. However, in certain cases the signing keys are generated externally and loaded on the token for subsequent signature operations. When this bit is set, it indicates that externally generated keys cannot be used for signing operations performed by the token.
TPV_FORCE_RSA_BLINDING	This bit determines whether the token must perform blinding, which introduces a random element to the time needed to complete an RSA operation. Blinding defeats timing attacks on an RSA operation. If this bit is set, the token will always use RSA blinding (the TPV_FORCE_RSA_BLINDING bit will have no effect).
TPV_DISABLE_CLONING_BY_USER	This bit determines whether a user or both a user and the token's SO are permitted to clone sensitive objects when the FPV_SECURITY_POLICY_ENABLE_CLONING bit is set. If the FPV_SECURITY_POLICY_ENABLE_CLONING bit is clear, no cloning is permitted and the TPV_DISABLE_CLONING_BY_USER bit has no effect regardless of its value. When TPV_DISABLE_CLONING_BY_USER is clear, both a user and the token's SO are permitted to clone sensitive objects. When the bit is set, only the token's SO is permitted to clone sensitive objects.
TPV_XP_MUST_USE_SP	This bit determines whether the token implements XP-type functionality. When this bit is set: all objects are stored in volatile memory; token KCV may be cloned from a different token; the token is dual mode (i.e., whether the SP is required is defined at token initialization time); and, object headers are always modifiable, even in non-modifiable objects.

4. IDENTIFICATION AND AUTHENTICATION (I&A)

The LunaRA token enforces an identity-based user authentication policy. For normal users, the user number and a valid PIN must be provided to the token before access to private data and token services can be granted. For the SO, only a PIN is required.

Note: Normal users also have a text-based name associated with them. The name corresponding to a particular user number can be queried from the token.

The PINs for the SO and users can be changed at any time by their respective owners. The SO can also reinstate users with lost PINs. Reinstating users does not affect the cryptographic material and data owned by the user and protected under the old PIN.

The LunaRA token implements policy that limits the number of login attempts. This feature prevents an exhaustive search approach for finding the PIN of the SO or user. The implementation of this feature differs from that of an SO PIN search and that of a user PIN search.

For a user PIN search:

- If “*n”* consecutive user logon attempts fail, the token flags the event in the User’s Authorization Vector (UAV). This erases the user’s profile and private data from the token. The SO must create a new user; the new user will have no association with the deleted user. (The value of "n" is defined by the SO in the TPV.)

For an SO PIN search:

- If “*n*” consecutive SO logon attempts fail, the token is zeroized and its operational state goes to ZEROIZED. (The value of "n" is defined in the FPV, which is defined when a LunaRA token is manufactured and cannot be modified without invalidating the CRC value of the software load.)

5. DISCRETIONARY ACCESS CONTROL (DAC)

Every data object on the token can be public or private. Private data objects are labeled with a number that corresponds to the owner and can be accessed only by the legitimate owner. A user cannot create a key or certificate object as a public object. Only data objects can be public or private.

The token does not allow for any granularity of ownership other than that of individual or public (i.e., a data object cannot be owned by two users and restricted from other users). Also, the ownership of an object implies read/write/modify/execute access to the object, which means full access rights to the object.

6. OBJECT REUSE

The token enforces an object reuse policy in that every object is allocated a portion of memory (flash or SRAM). The policy also ensures that no other objects are placed in the same memory location unless all previous memory content is initialized and purged. When cryptographic functions are performed, a cryptographic context is created to hold data required by the function (e.g., a DES key schedule for a DES function or an MD5 chaining vector). The cryptographic context only exists in SRAM memory and is not assigned to any functions except those defined by its owner. The memory assigned to a cryptographic context is always purged of its content before being handed over to a function.

APPENDIX A. Cryptographic Algorithms Support

Encrypt/Decrypt:

- DES-ECB
- DES-CBC
- 3-DES-ECB
- 3-DES-CBC
- RC2-ECB
- RC2-CBC
- RC4
- RC5-ECB
- RC5-CBC
- CAST-ECB
- CAST-CBC
- CAST3-ECB
- CAST3-CBC
- CAST5-ECB
- CAST5-CBC
- RSA X-509

Digest:

- MD2
- MD5
- SHA-1

Sign/Verify:

- RSA-1024
- RSA-2048
- RSA-4096
- DSA
- DES-MAC
- 3-DES-MAC
- RC2-MAC
- RC5-MAC
- CAST-MAC
- CAST3-MAC
- CAST5-MAC
- SSL3-MD5-MAC
- SSL3-SHA1-MAC
- HMAC-SHA1
- HMAC-MD5

Generate Key:

- DES
- double length DES
- triple length DES
- RC2
- RC4
- RC5
- CAST
- CAST3
- CAST5
- PBE-MD2-DES
- PBE-MD5-DES
- PBE-MD5-CAST
- PBE-MD5-CAST3
- PBE-SHA-1-CAST5
- GENERIC-SECRET
- SSL PRE-MASTER

Generate Key Pair:

- RSA-1024

- RSA-2048
- RSA-4096
- DSA-1024
- DH-1024
- DH-2048

Wrap Symmetric Key Using Symmetric Algorithm:
- DES-ECB
- 3-DES-ECB
- RC2-ECB
- CAST-ECB
- CAST3-ECB
- CAST5-ECB

Wrap Symmetric Key Using Asymmetric Algorithm:
- RSA-1024
- RSA-2048
- RSA-4096

Wrap Asymmetric Key Using Symmetric Algorithm:
- 3-DES-ECB (LunaRA only)
- 3-DES-CBC[1]

Unwrap Symmetric Key With Symmetric Algorithm:
- DES-ECB
- 3-DES-ECB
- RC2-ECB
- CAST-ECB
- CAST3-ECB
- CAST5-ECB

Unwrap Symmetric Key With Asymmetric Algorithm:
- RSA-1024
- RSA-2048
- RSA-4096

Unwrap Asymmetric Key With Symmetric Algorithm:
- DES-CBC
- 3-DES-CBC
- CAST-CBC
- CAST3-CBC
- CAST5-CBC

Derive Key Value:
- DH-1024
- DH-2048
- concatenate Base & Key
- concatenate Base & Data
- concatenate Data & Base
- XOR Base and Data
- Extract Key from Key
- MD2 Derivation
- MD5 Derivation
- SHA-1 Derivation
- SSL3-Master
- SSL3-Key & MAC
- 3DES-ECB (LunaRA only)
- 2DES Derivation (LunaRA only)

APPENDIX B. Policy Vector Settings

	Standard LunaRA Domestic	Standard LunaRA Export
Token Policy Vector Settings		
TPV_USER_ZEROIZE	1	1
TPV_USER_FW_UPDATE	0	0
TPV_M_OF_N_ACTIVATION	0	0
TPV_KEY_ATTRIB_LOCK	1	1
TPV_KEY_SINGLE_FUNCTION	0	0
TPV_SIGNING_KEY_LOCAL	0	0
TPV_MAX_PIN_LEN	48	48
TPV_MIN_PIN_LEN	4	4
TPV_LOGIN_FAILS_ALLOWED	10	10
TPV_DISABLE_CLONING_BY_USER	0	0
Fixed Policy Vector Settings		
FPV_SECURITY_POLICY_DOMESTIC	1	0
FPV_SECURITY_POLICY_ENABLE_CLONING	1	1
FPV_SECURITY_POLICY_USE_CAV	0	0
FPV_SECURITY_POLICY_WRAPPING_TOKEN	1	1
FPV_SECURITY_POLICY_USE_M_OF_N	0	0
FPV_SECURITY_POLICY_USE_RAW_RSA	1	1
FPV_SECURITY_POLICY_SPECIAL_CLONING	1	1
FPV_ENABLE_CCM	0	0
FPV_SEC_KEY_POLICY_SENSITIVE	1	1
FPV_SEC_KEY_POLICY_NO_CREATE	1	1
FPV_PRI_KEY_POLICY_SENSITIVE	1	1
FPV_PRI_KEY_POLICY_NO_CREATE	1	1
FPV_SO_LOGIN_FAILS_ALLOWED	3	3
FPV_PIN_MUST_USE_SP	0	0
FPV_MOFN_MUST_USE_SP	0	0
FPV_KCV_MUST_USE_SP	0	0
FPV_XP_TOKEN	0	0
FPV_XPPLUS_TOKEN	0	0
FPV_RA_TOKEN	1	1

APPENDIX C.   Session And Login States Required For Luna Commands

Command To Module	No Session Open	Session Open, No Login	SO Logged On	User Logged On
Token Main Module Commands				
LUNA_ZEROIZE	√			
LUNA_INIT_TOKEN			√	
LUNA_GET	√			
LUNA_GET_USV			√	
LUNA_SET_TPV			√	
LUNA_FW_UPDATE			√	
LUNA_CONFIGURE_SP	√			
Session Manager Commands				
LUNA_OPEN_ACCESS	√			
LUNA_CLEAN_ACCESS	√			
LUNA_CLOSE_ACCESS	√			
LUNA_GET_ALL_ACCESSES	√			
LUNA_OPEN_SESSION	√			
LUNA_CLOSE_SESSION		√		
LUNA_CLOSE_ALL_SESSIONS	√			
LUNA_GET_SESSION_INFO		√		
LUNA_EXTRACT_CONTEXTS		√		
LUNA_INSERT_CONTEXTS		√		
User Module Commands				
LUNA_GET_USER_LIST		√		
LUNA_GET_USER_NAME		√		
LUNA_LOGIN		√		
LUNA_LOGOUT				√
LUNA_SET_PIN				√
LUNA_INIT_PIN			√	
LUNA_CREATE_USER			√	
LUNA_DELETE_USER			√	
Object Management Module				
LUNA_CREATE_OBJECT		√		
LUNA_COPY_OBJECT		√		
LUNA_DESTROY_OBJECT		√		
LUNA_GET_OBJECT_SIZE		√		
LUNA_GET_ATTRIBUTE_VALUE		√		
LUNA_GET_ATTRIBUTE_SIZE		√		
LUNA_MODIFY_OBJECT		√		
LUNA_FIND_OBJECTS		√		
Random Number Generator Module				
LUNA_GET_RANDOM		√		
LUNA_SEED_RANDOM		√		
Key Management Module				
LUNA_GENERATE_KEY				√
LUNA_GENERATE_KEY_W_VALUE				√
LUNA_GENERATE_KEY_PAIR				√
LUNA_WRAP_KEY				√
LUNA_UNWRAP_KEY				√
LUNA_UNWRAP_KEY_W_VALUE				√
LUNA_DERIVE_KEY				√
LUNA_DERIVE_KEY_W_VALUE				√
LUNA_MFG_LOAD				√
Cryptographic Algorithm Module				
LUNA_ENCRYPT_INIT				√
LUNA_ENCRYPT_INIT_W_VALUE				√
LUNA_ENCRYPT_INIT				√

Command To Module	No Session Open	Session Open, No Login	SO Logged On	User Logged On
LUNA_ENCRYPT_INIT_W_VALUE				√
LUNA_ENCRYPT				√
LUNA_ENCRYPT_FIFO				√
LUNA_ENCRYPT_END				√
LUNA_DECRYPT_INIT				√
LUNA_DECRYPT_INIT_W_VALUE				√
LUNA_DECRYPT				√
LUNA_DECRYPT_FIFO				√
LUNA_DECRYPT_END				√
LUNA_DECRYPT_RAW_RSA				√
LUNA_DIGEST_INIT		√		
LUNA_DIGEST		√		
LUNA_DIGEST_FIFO		√		
LUNA_DIGEST_KEY				√
LUNA_DIGEST_KEY_VALUE				√
LUNA_DIGEST_END		√		
LUNA_SIGN_INIT				√
LUNA_SIGN_INIT_W_VALUE				√
LUNA_SIGN				√
LUNA_SIGN_FIFO				√
LUNA_SIGN_END				√
LUNA_SIGN_SINGLEPART				√
LUNA_SIGN_UPDATE_KEY				√
LUNA_SIGN_FINAL_DERIVE_KEY				√
LUNA_VERIFY_INIT				√
LUNA_VERIFY_INIT_W_VALUE				√
LUNA_VERIFY				√
LUNA_VERIFY_FIFO				√
LUNA_VERIFY_END				√
LUNA_VERIFY_SINGLEPART				√
LUNA_GET_MECH_LIST	√			
LUNA_GET_MECH_INFO	√			
LUNA_SELF_TEST	√			
LUNA_SET_UP_MASKING_KEY	√			
LUNA_CLONE_AS_SOURCE				√
LUNA_CLONE_AS_TARGET_INIT				√
LUNA_CLONE_AS_TARGET				√
LUNA_GEN_TKN_KEYS			√	
LUNA_LOAD_CERT			√	
LUNA_GEN_KCV				√
LUNA_LOAD_CUSTOMER_VERIFICATION_KEY			√	
LUNA_M_OF_N_GENERATE			√	
LUNA_M_OF_N_ACTIVATE				√
LUNA_M_OF_N_MODIFY			√	
Special Packet Processing Commands				
LUNA_IPSEC_INIT_NO_USER	√			
LUNA_IPSEC_PROCESS_PACKET	√			
LUNA_IPSEC_END	√			
LUNA_GEN_CRC32	√			
LUNA_SCP_TEST	√			