Common Criteria for Information Technology Security Evaluation

Department of Defense Public Key Infrastructure and Key Management Infrastructure Token Protection Profile (Medium Robustness)

Version 3.0
22 March 2002

Prepared by Booz Allen Hamilton

Prepared for National Security Agency (NSA)

Foreword

This protection profile (PP) was developed to identify and set forth the security requirements for a Department of Defense (DoD) Public Key Infrastructure (PKI) Token (Extended Protection) based on Version 2.1 of the "Common Criteria," International Standard 15408. The Common Criteria can be found at http://csrc.nist.gov/cc.

Comments on this PP should be e-mailed to Tamara Cleveland at cleveland_tamara@bah.com.

Table of Contents

List of Tables and Figures

Conventions and Terminology

Conventions

The notation, formatting, and conventions used in this protection profile (PP) are consistent with version 2.1 of the Common Criteria for Information Technology Security Evaluation. Font style and clarifying information conventions were developed to aid the reader.

The following label conventions are to aid in the referencing and understanding of assumptions, threats, policies, and objectives used throughout this document.

Labeling Convention	**Reference Category**
A.<*name*>	Assumption
T.<*name*>	Threat
P.<*name*>	Organizational Security Policy (OSP)
SFP.<*name*>	Security Function Policy (SFP)
O.<*name*>	Objective allocated to the TOE
OE.<*name*>	Objective allocated to the non-IT environment of the TOE

The CC permits four functional component operations: assignment, iteration, refinement, and selection to be performed on functional requirements. These operations are defined in Common Criteria, Part 2, paragraph 2.1.4 as:

- assignment: allows the specification of an identified parameter;
- refinement: allows the addition of details or the narrowing of requirements;
- selection: allows the specification of one or more elements from a list; and
- iteration: allows a component to be used more than once with varying operations.

Assignments or selections left to be specified by the developer in subsequent security target documentation are italicized and identified between brackets ("[ ]"). In addition, when an assignment or selection has been left to the discretion of the developer, the text "ST assignment:" or "ST selection:" is indicated within the brackets. CC assignments and selections completed by the PP authors are underlined.

Refinements of requirements are denoted by **bold** text. The **bolded** text alerts the reader that additional text has been added to the CC. They permit the addition of extra detail when the component is used. The underlying notion of a refinement is that of narrowing. There are two types of narrowing possible: narrowing of implementation and narrowing of scope.

Iteration of a component is required when an operation within a component must be completed multiple times with differing values, or for different allocations of functions to portions of the TOE. Iterated functional and/or assurance components are denoted by the word “iteration” appearing in parentheses in **bold** *italics* followed by the number of the iteration, e.g., ***(Iteration 1)***. The notation of the iteration appears after the short family name.

Explicit Requirements are not used in this protection profile.

Application Notes are used to provide the reader with additional requirement understanding or to clarify the intent of the authors. These are italicized and usually appear following the element needing clarification.

Terminology

An extensive glossary of terms appears in Appendix C of this PP. Some terms that appear often in the PP are defined below for the reader.

Common Criteria Terms

Assets	Information or resources to be protected by the countermeasures of a TOE (e.g., user data and cryptographic keys).
Role	A predefined set of rules establishing the allowed interactions between a user and the TOE.
Security objective	A statement of intent to counter identified threats and/or satisfy identified organizational security policies and assumptions.
Security Target (ST)	A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
Target of Evaluation (TOE)	An IT product or system and its associated administrator and user guidance documentation that is the subject of evaluation.
TOE Security Functions (TSF)	A set consisting of all hardware, software, and firmware of the TOE that must be relied on for the correct enforcement of the TSP.
TOE Security Policy (TSP)	A set of rules that regulate how assets are managed, protected, and distributed within a TOE.
TSF data	Data created by and for the TOE that might affect the operations of the TOE.
User	Any entity (human user or external IT entity) outside the TOE that interacts with the TOE.
User data	Data created by and for the user that does not affect the operation of the TSF.

Token Terms

Application	(1) An application may also be called an Executable file, Applet, or Cardlet (for Java Cards). An application is to be run on the token that may be downloaded onto the token during enrollment, or just prior to execution invoked by the host. (2) Intended final use for the token. This may include (but is not limited to) such activities as payment, telephony, identification, secure information storage, or access.
Attack	An attempt to gain unauthorized access to an information system's services, resources, or information or the attempt to compromise an information system's integrity, availability, or confidentiality. There are several forms of attacks including: **Malicious attacks** – virus, worm, Trojan horse, masquerading **Unintentional attacks** – malfunction, human error **Physical attacks** – fire, water, battle damage, power loss
DoD data	All data on the TOE located below the DoD directory. These data are owned by DoD. It includes DoD executables, DoD PINs, DoD cryptographic keys, and DoD user personal information.
Host	Device to which a token authenticates to establish a secure communication path.
Integrated Circuit (IC)	Electronic component(s) designed to perform processing and/or memory functions contained on a single chip.
Private key	A cryptographic key used with a public key cryptographic algorithm, uniquely associated with an entity and not made public.
Public key	A cryptographic key used with a public key cryptographic algorithm, uniquely associated with an entity, and that may be made public.
System Security Officer (SSO)	The role assumed to perform a set of cryptographic initialization or management functions (e.g., cryptographic key and parameter entry, and alarm resetting).
Token	An authentication device carrier that is used to store and carry cryptographic keys and certificates supporting user identity authentication. This technology can consist of (but is not limited to) smart cards, USB tokens, PCMCIA Card, and iButtons® /JavaRing® technology.

1 Introduction

This protection profile (PP) is the result of work done by the National Security Agency (NSA) with guidance from the Department of Defense (DoD) community. It is based on the S*mart Card Security User Group Smart Card Protection Profile*, Draft Version 2.0, May 1, 2000.

The structure for this PP was established through the use of the Common Criteria Toolbox (version 5.0, 28 February 2000). This toolbox was developed by SPARTA, Inc., for the NSA. It is available at http://cctoolbox.sparta.com.

A token compliant with this PP may offer security features and functionality beyond that specified in this PP.

1.1 Identification

Title:	Department of Defense Public Key Infrastructure and Key Management Infrastructure Token Protection Profile (Medium Robustness)
Authors:	Tamara Cleveland, Booz Allen Hamilton Michael Alexander, Booz Allen Hamilton Asok Ganguly, Booz Allen Hamilton Brian Green, Booz Allen Hamilton Edward Schneider, Institute for Defense Analyses
Vetting Status:	N/A
CC Version:	2.1 Final
Registration:	N/A
Keywords:	DoD PKI, token, smart card

1.2 Protection Profile Overview

This PP specifies the information technology (IT) security requirements for a token to be used with sensitive but unclassified (SBU) applications (Class 4) in the DoD Public Key Infrastructure (PKI) and the Key Management Infrastructure (KMI). The Key Management Infrastructure's purpose is to unify existing and planned key management systems with the DoD PKI to create a single, integrated whole. Due to the relationship between KMI and the DoD PKI, this protection profile specifies the security requirements for a token that will be used by both infrastructures. Tokens conformant to this PP provide adequate security services, mechanisms, and assurances to

process sensitive information in medium robustness environments, as specified in the "Guidance and Policy for Department of Defense Information Assurance" (GiG). Medium robustness is defined as having a classification of SBU (DoD Unclassified), an Evalution Assurance Level (EAL) of 4+, and an encryption requirement of FIPS 140-2 Level 2 for Subscribers/Level3 for Registration Authorities and Certificate Authorities. The services provided by the DoD PKI and KMI include the generation, distribution, control, tracking, and destruction of public key certificates. The primary goal of the DoD PKI and KMI is to securely transport sensitive but unclassified or unclassified information using unprotected networks. The DoD PKI and KMI token carries public key certificates used to authenticate its user in public key transactions and applications.

The security requirements in this PP apply to the DoD PKI and KMI token as issued to the token holder. These requirements cover the token's integrated circuit, operating software, and specific applications when processing DoD information. This PP does not cover security requirements for token terminals or networks interfacing with them. Throughout the requirements section in this protection profile, references are made to requirements for FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities. If the DoD Common Access Card (CAC) issuing infrastructure is not capable of issuing two different levels of cards, then all CACs will be required to meet FIPS 140-2 Level 3.

Appendix A lists references, and Appendices B and C, respectively, list acronyms and a glossary of terms used in this PP.

1.3 Assurance Level

The assurance level for this PP is EAL4 augmented. Augmentation results from the selection of ALC_TAT.3 and AVA_VLA.3.

1.4 Related Standards and Documents

Additional input was derived from the following security documents furnished by the NSA:

Consideration of Smart Cards as the DoD PKI Authentication Device Carrier, 10 January 2000.

DoD Target Token Requirements Document, Draft, 8 March 2000.

Public Key Infrastructure Target Class 4 Token Security Requirements, Draft version 1.01, April 10, 2000.

Policies and standards that were referenced during the development of this PP include:

Department of Defense Chief Information Officer, Guidance and Policy for Department of Defense Information Assurance Memorandum No. 6-8510, 16 June 2000.

Draft FIPS 140-2, *Security Requirements for Cryptographic Modules*, 1999.

ISO 7816, *Identification Cards - Integrated Circuit Cards with Contacts*.

X.509 Certificate Policy for the U. S. Department of Defense, version 5.2, 13 November 2000.

1.5 Related Protection Profiles

This PP was developed using as a foundation the *Smart Card Security User Group Smart Card Protection Profile* (SCSUG-SCPP) Draft Version 2.0 (May 1, 2000). The SCSUG-SCPP can be found at http://csrc.nist.gov/cc/sc/sclist.htm.

The SCSUG-SCPP was examined for possible use as the PP for the DoD PKI and KMI Token. The SCSUG-SCPP defines security requirements for commercial smart cards used for sensitive applications, such as banking industry financial payment systems. The SCSUG-SCPP allows several of its requirements to be specified in the vendor's security target documentation. The DoD PKI and KMI Token will be used for applications that are unique to the DoD environment. Furthermore, several of the requirements for this token need to be more explicit than those for the smart cards described in the SCSUG-SCPP. Thus, this protection profile was developed in lieu of using the SCSUG-SCPP in order to address the DoD PKI and KMI Token's applications and to add specificity to its security requirements.

1.6 PP Organization

Section 1 provides the introductory material for the protection profile.

Section 2 provides the general purpose of the protection profile and the Target of Evaluation (TOE) description.

Section 3 provides a discussion of the expected environment for the TOE. This section also defines the set of threats that are to be addressed by either the technical countermeasures implemented in the TOE hardware or software or through the environmental controls.

Section 4 defines the security objectives for both the TOE and its environment.

Section 5 contains the functional and assurance requirements derived from the Common Criteria, Parts 2 and 3, respectively, that must be satisfied by the TOE.

Section 6 provides rationale to explicitly demonstrate that the information technology security objectives satisfy the assumptions, policies, and threats. Arguments are provided for the coverage of each assumption, policy, and threat. The section then explains how the set of requirements are complete relative to the objectives, and that each security objective is addressed by one or more component requirements. Arguments are provided for the coverage of each objective. The Evaluation Assurance Level (EAL) is then presented with its supporting assurance requirements. Next, Section 6 addresses dependency analysis and strength of function issues.

The appendices contain references, an acronym list, a glossary, a description of token states, a list of required supported algorithms, references to DoD specifications, comparisons to the SCSUG's PP, and additional threat information.

2 TOE Description

2.1 Token Overview

A token is used to store and carry cryptographic keys and certificates supporting user identity authentication. There are various types of tokens including smart cards, universal serial bus (USB) tokens, Personal Computer Memory Card International Association (PCMCIA) cards, and iButtons®/Java Rings®. The DoD PKI and KMI Token will contain an integrated circuit (IC) and an operating system.

A semiconductor (silicon) IC is fabricated in a complex microelectronic process, which involves repeatedly masking and doping the surface of a silicon substrate to form transistors, followed by patterning metal connections, and applying a protective overcoat. This process eventually yields a design typically comprising several hundred thousand transistors, arranged in an area of less than 25 square millimeters. The design consists of a central processing unit, input and output lines, and volatile and nonvolatile memory.

The IC itself is packaged in a token. The current predominant packaging method is die bonding in a module. A module consists of a carrier board on which the IC is seated. Wire bonds are connected from the IC's input/output (I/O) pads to the carrier, which has contacts on its reverse side.

The token also contains an operating system that may be stored in Read Only Memory (ROM). The DoD PKI and KMI Token's operating system will allow authorized applications to be added to the token. Examples of operating systems are MultOS®, Java Virtual Machine®, and Smart Cards for Windows® with MEL®, Java®, and Visual Basic® as their programming languages, respectively.

2.2 Types of Tokens

A smart card is a credit card-sized token that often has a microprocessor on its IC. In a smart card, the IC is encapsulated in a protective material (usually some type of epoxy), and this module is adhesively embedded into a premilled hole in a plastic card. Two common examples are the familiar payment card-sized smart cards and the smaller postage-stamp-sized subscriber identity module (SIM) frequently used in mobile telephones. Smart cards communicate with the outside world via a reader connected to a standard (e.g., serial, USB, or PCMCIA) interface in a contact environment or via radio frequency (RF) electromagnetic waves in a contact-less environment.

Categories of smart cards include:

• Contact Cards

- Memory only (sometimes with protection features)
- Microprocessor with memory
- Microprocessor with memory and additional coprocessor

• Contact-less Cards

- As above, but with power derived from energy obtained through a contact-less interface

A smart card serving as a DoD PKI and KMI token will have symmetric and asymmetric (i.e., public key) cryptographic algorithm capability. Asymmetric cryptography typically requires a math coprocessor. Additionally, the smart card will be able to hold multiple applications in separate protected areas on the card.

USB is an interface incorporating the high-speed external bus for PCs. A USB token is a device containing an embedded microprocessor IC that interfaces directly with a PC's USB port without any additional hardware, e.g., a card reader. The microprocessors used in USB tokens are just as powerful as those in smart cards.

A PCMCIA card is a hardware device that supports specific dedicated functions. Examples of PCMCIA card functions include memory devices, input/output devices (e.g., modems and fax modems), and portable disk drives. PCMCIA cards are most commonly used to provide additional computing features for portable computers such as laptops. NSA's FORTEZZA® Crypto Card is an example of a PCMCIA card. PCMCIA cards provide the strongest security and largest memory storage capacity of available tokens.

Dallas Semiconductor's iButton® is a computer chip encased in a 16-mm stainless steel case. It can be attached to articles of clothing, wallets, etc. A Java Ring® is a ring with an iButton® attached to it.

2.3 TOE Overview

The target of evaluation is the DoD PKI and KMI token. The DoD is implementing a PKI that will serve as a key and certificate management infrastructure designed to support confidentiality, integrity, availability, and authentication in computer networks. This PKI will require authentication devices (i.e., tokens) to store and carry cryptographic keys supporting user identity authentication. This token will be used for Class 4 applications, which refers to the assurance level intended for applications handling high value UNCLASSIFIED information (i.e., Mission Critical, National Security System Information) in a minimally protected environment.

The word "token," as used in this PP, refers to the DoD PKI and KMI end-entity cryptographic hardware device that houses the DoD PKI and KMI private keys and associated public key certificates and algorithms. The TOE is an operational token platform, consisting of the integrated circuit and on-card operating software, including DoD-provided applications and the mechanisms that allow communication with the outside world. The TOE consists of sufficient hardware and software elements to be capable of establishing a secure channel to a trusted source for application loading or for other potentially privileged commands.

This PP does not include printing on the TOE, including printed security features such as holograms. This PP also does not apply to the terminal, non-DoD applications loaded onto the token, nor to any network with which the token interfaces.

2.4 Applications

The DoD PKI and KMI token allows for multiple applications to support the DoD mass population (i.e., all DoD military, civilian, and contractor personnel operating in the SBU environment). Security functions present will be of appropriate level and protection. An application can only be used by the DoD PKI and KMI token after its signature by a DoD entity has been validated. Typical applications for tokens within the DoD include:

• Financial–Payment schemes may include credit, debit, and stored value functions provided by an electronic commerce (EC) application. The EC application is loaded onto the token and will probably require a public/private key to be used. The specific EC application (Visa/MasterCard, banking, electronic payment, etc.) will specify how that key is to be ordered and loaded.

• Secure Messaging–Secure Messaging, as it pertains to the DoD community, establishes requirements for an integrated common-user, writer-to-reader organizational, and individual messaging service accessible from DoD locations worldwide, tactically deployed users, and other designated Government users with interfaces to Allied users and Defense contractors. The DoD PKI and KMI Token will provide identification, authentication, and encryption functions for secure messaging. Specifically, the token technology combines encryption, digital certificates, and other PKI technologies to authenticate a user's identity and to ensure that data and transactions are not tampered with during transmission.

• Identification–Various public and private schemes provide identification credentials to participants. The identification credentials are typically associated with various rights and duties defined by the identification provider. These can include memberships, driver's licenses, benefit access, security access, passports, national identification, etc. Typically the identification credentials have value because the credential holder cannot easily alter them, and assets (e.g., user data and cryptographic keys) in the credential must be protected against alteration by the cardholder. Digital certificates used in public key systems fit into this category.

• Secure information storage–Information that is useful to store in a secure fashion includes health records, health insurance, medical information, etc.

• Access Control–Tokens can hold access credentials such as passwords, biometrics, and PINs that authenticate and verify a user's access to a building, a sensitive controlled area, and a computing environment and its applications (e.g., computer network, workstation, e-mail, or Web browser) containing personal or mission-critical data (that requires signing or encrypting data), or the right to be issued firearms/weapons.

Each of these may have somewhat different security requirements, security features, roles, and environmental considerations (e.g., whether always on-line, always used off-line, usually off-line with the capability of going on-line, etc.). The security requirements for operating software, applications, and procedures for adding or deleting those applications must therefore be clearly identified, and the security functions that are present must be appropriate to the type and intended use of the token.

A DoD PKI and KMI Token must contain an application manager as detailed in Appendix F in Section 2.3 of the *Application Specification for the DoD PKI and KMI Token*. An application manager is a module of code embedded in the OS of a token which controls the load, verification, selection, and execution of a token application.

Application signatures will be used to sign approved applications to control the loading of applications onto the DoD PKI and KMI token. Valid applications must be signed by an approved DoD entity. The requirements for the use of application signatures are detailed in Appendix F in Section 2.4 of the *Application Specification for the DoD PKI and KMI Token*.

2.5 TOE Identification

Through selection of the Configuration Management Class of assurance functions, this PP imposes the requirement that a unique reference be utilized to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labeling the TOE with this reference ensures that users of the TOE are aware of which instance of the TOE they are using. The TOE described herein is, however, a combination of hardware and software, each portion of which may be composed of a further collection of components. This aggregate collection offers the potential for confusion in identifying a unique reference for the TOE.

To further complicate identification, an IC can usually be produced with multiple features, only some of which are enabled. The design layout of the IC (the photomask) determines the functionality; however, as fabrication technology improves, this photomask may be used to produce an otherwise identical chip but with a reduced feature size. Likewise, software features may be selectively employed, depending on hardware functions. However, the presence or absence of specific features may directly contribute to the possible introduction of vulnerabilities. For example, the size of the IC features is directly related to the relative difficulty of probing. A potentially unknown, but present, software feature may allow backdoors or other routes for penetration.

It is therefore essential that the unique reference for a TOE compliant with this PP must allow

the identification of at least:
• the microprocessor specification
• the memory size and allocation (ROM, EEPROM, RAM, etc.)
• the physical instantiation of the IC design regarding layout and feature size
• all hardware security features on the IC, whether they are initially enabled or not
• all enabled hardware security features
• the software specification
• all software security features present, whether they are enabled or not
• all enabled software security features.

2.6 Cryptography

A variety of cryptographic keys are typically used with smart cards, including transport keys, personalization keys, application-specific keys, etc. Handling of these keys must be done in accordance with the DoD key management procedures and policies as specified in the Key Management Specification for the DoD PKI and KMI Token (see Appendix F).

Cryptography may be implemented in hardware or software, with various algorithms and various key lengths. Many tokens have dedicated cryptographic coprocessors that execute DES, triple DES, RSA, and other standard algorithms much faster than software implementations can. Some applications use no cryptography, some use private key, and some use public key systems.

Any TOE claiming compliance with this protection profile must handle cryptographic functions in accordance with applicable international, industrial, or organizational policies. This extends to any applications using cryptography, although there may be additional applications on the token that do not use cryptography at all.

2.7 Key Management

A token is a portable, physical device in which keys and certificates are stored. There are a host of key management operations and procedures that dictate how keys and certificates will be generated, loaded, removed, stored, and otherwise managed on the TOE. The *Key Management Specification for the DoD PKI and KMI Token* in Appendix F establishes the key management requirements for tokens to be used for Class 4 applications within DoD. The requirements section of the appendix (Part II), is correlated to appropriate sections of the protection profile, i.e., requirements stated in the specification are correlated to requirements in the protection profile.

The adequacy of the key management approach is critical to the security of all systems that rely on the token. Weak, ineffective key management approaches could undermine the applications that rely on the token for cryptographic security services. But from the perspective of

operational effectiveness, burdensome key management solutions could render the token unusable. Key management approaches for the token must be able to support diverse operational environments and the size of the DoD community to which tokens will be issued. Given a community of millions of users, automated yet secure key distribution methods must be used that will allow keys to be added or changed without necessitating a return to some central issuing authority. Thus, the key management requirements applicable to a cryptographic token must achieve a comfortable balance between the sometimes competing needs of security and operational effectiveness. These types of issues are explored in the *Key Management Specification for the DoD PKI and KMI Token* in Appendix F.

2.8 Attacker Capabilities

Attackers are assumed to have various levels of expertise, resources, and motivation. Relevant expertise may be in general semiconductor technology, software engineering, hacker techniques, or the specific TOE. Resources may range from personal computers and inexpensive card-reading devices to very expensive and sophisticated engineering test and measurement devices. They may also include software routines, some of which are readily available on the Internet. Motivation may include economic reward, intelligence gathering by hostile nations, or notoriety of defeating high-grade security. Given sufficient time and expertise, any token can be compromised. The strength of function for a TOE based on this PP is medium. In the DoD, this refers to a minimum strength of mechanism level (SML) of 2 as defined in chapter 4 of the *Information Assurance Technical Framework*, which can be found at http://www.iatf.net/.

2.9 Description of Token States

Some states of the DoD PKI and KMI Token need to be defined to effectively describe the conditions under which some of the token security requirements apply. A detailed description of these token states is provided in Appendix D.

3 TOE Security Environment

This section identifies the following:

- Significant assumptions about the TOE's operational environment
- IT-related threats to the organization countered by DoD PKI and KMI Token PP compliant components
- Threats requiring reliance on environmental controls to provide sufficient protection
- Organizational security policies for which DoD PKI and KMI Token compliant TOEs are appropriate.

3.1 Secure Usage Assumptions

The specific conditions listed below are assumed to exist in the DoD PKI and KMI Token environment. Each assumption is stated in bold type font. Some assumptions are followed by an explanation, in normal font, that supplies additional information and interpretation.

A.Dev_Protect: Protection of TOE by Developer

During the development and manufacturing process, the TOE and associated development tools are assumed to be protected by the developer from any kind of unauthorized use, e.g., tampering or theft.

A.Key_Gen: Key Exchange Key Generation

Key exchange keys are assumed to be generated off-TOE in a secure manner in accordance with X.509 Certificate Policy.

The Key Exchange Key is critical for secure communication with the host.

A.Secure_Host_Comms: Secure Host Communications

If the host establishes a secure connection between itself and the TOE that conforms to the requirements imposed by the TOE, the host, including code and security data it contains, is assumed to be trusted.

The host may have the capability to establish a secure communication channel with the TOE. This is typically accomplished through shared private keys, public/private key pairs, and/or generation of local keys derived from other stored keys. When such a secure link is established, the TOE may assume the host to be

adequately secure for trusted communications. The host is considered to be beyond the scope of this PP.

3.2 Threats to Security

DoD PKI and KMI Token PP compliant TOEs are required to counter threats that may be broadly categorized as:

- Threats addressed by the TOE:
  - Threats associated with physical attack on the TOE
  - Threats associated with logical attack on the TOE
  - Threats associated with control of access
  - Threats associated with unanticipated interactions
  - Threats regarding cryptographic functions
  - Threats that monitor information
  - Miscellaneous threats

- Threats addressed by the Operating Environment

Each threat is stated in bold type font. Most threats are followed by an explanation, in normal font, that supplies additional information and interpretation. In parentheses, a code for the source of the threat statement is given. The key for the source codes is as follows:

SCSUG	Smart Card Security User Group's Smart Card Protection Profile
TSRD	Public Key Infrastructure Target Class 4 Token Security Requirements Document
NEW	Created by Token PP Team

3.2.1 Threats Addressed by the TOE

3.2.1.1 Threats Associated with Physical Attack on the TOE

T.E_Manip: Electrical Manipulation of the IC (SCSUG)

An attacker may utilize electrical probing and manipulating of the TOE to modify security-critical data so that the TOE can be used fraudulently.

This modification may include manipulation of debug lockouts, first-use indicators, token use blocking, blocking function configuration, token block indicators, or token disablement indicators. This threat is distinguished by the intent to utilize a modified TOE rather than to derive information from the TOE. The attacker may attempt to introduce faults in the TOE or change TOE assets (PIN or biometric data, user data, certificate information, private keys, etc.) to use the TOE in a fraudulent manner. This threat characterizes active threats. Refer also to T.Power_Clock, T.Forced_State_Change, and T.Env_Strs.

T.P_Modify: Physical Modification of the IC (SCSUG)

An attacker may physically modify the TOE in order to reveal design- or security-related information.

This modification may be achieved through techniques commonly employed in IC failure analysis and IC reverse engineering efforts. The goal is to identify such design details as hardware security mechanisms, access control mechanisms, authentication systems, data protection systems, memory partitioning, or cryptographic programs. Determination of software design, including initialization data, personalization data, passwords, or cryptographic keys, is also a goal.

T.P_Probe: Physical Probing of the IC (SCSUG)

An attacker may perform physical probing of the TOE to reveal design information and operational contents.

Such probing may include electrical functions but is referred to here as physical, since it requires direct contact with the chip internals. Physical probing may entail reading data from the chip through techniques commonly employed in IC failure analysis and IC reverse engineering efforts. The goal of the attacker is to identify such design details as hardware security mechanisms, access control mechanisms, authentication systems, data protection systems, memory partitioning, or cryptographic programs. Determination of software design,

including initialization data, personalization data, passwords, or cryptographic keys is also a goal.

T.Power_Clock: Power and Clock (New [Assumption in SCSUG])

An attacker may interrupt, reset, or alter TOE power or clock to disrupt security-critical functions.

TOE power and clock may be provided by unreliable sources. The TOE is not internally powered, so support must be delivered to the card from the card acceptor device (CAD) or through an alternate connection to the TOE terminals. Both power and clock may be interrupted or reset in the normal course of business. The CAD is independent of the TOE and may belong to a different entity, which may be considered in some way hostile. Power may deviate from the design level (above or below) and may be supplied intermittently. The clock can likewise be manipulated. The intent of such manipulation may be to generate errors in the TOE operation, leading to a compromise of security.

3.2.1.2 Threats Associated with Logical Attack on the TOE

T.Bad_Load: Load Bad Software or Security Data (SCSUGmod of T. Load_Mal)

An attacker, an SSO, or the user may load improper software (operating system, executable files) or security data (authentication information, keys, access control information) onto the TOE that could modify or expose software (e.g., security functions) or data on the TOE.

During the stages of card preparation that involve loading the TOE with special keys, identification of roles, etc., the data itself may be changed from the intended information or may be corrupted. Either event could be an attempt to penetrate the TOE security functions or to expose the security in an unauthorized manner.

T.Component_Fail: Failure of a Critical System Component (TSRD)

An attacker exploits a failure of one or more system components, resulting in the loss of system-critical functionality.

This threat is relevant when there are components that may fail due to hardware and/or software imperfections and when the availability of system functionality is important.

The two basic engineering objectives to provide maximum securities are (a) minimize the probability of a security failure, and (b) minimize the consequences of failure. Usually, in chip cards all hardware components represent single points of failure (ROM, RAM, EEPROM, bus structure, microprocessor, etc.). Most failures occur where components designed by different people interact.

T.Developer_Flawed_Code: Software containing security-related flaws (TSRD)

An attacker exploits code delivered by a system or application developer that does not perform according to specifications, contains security flaws, or is not appropriate for operational use.

An important special case of this threat is when the security flaws prevent the system's security mechanism (TSF) from adequately protecting itself.

Various token OS developers, IC manufacturers, and token suppliers are involved in producing tokens. The number of different entities involved in the token development process creates a greater potential for security-related flaws to be introduced to the token.

T.Flt_Ins: Insertion of Faults (SCSUG)

An attacker may determine security-critical information through observation of the results of repetitive insertion of selected data.

Insertion of selected inputs followed by monitoring the output for changes is a relatively well-known attack method for cryptologic devices that can be applied to this TOE as well. The intent is to determine operational and security-related information based on how the TOE responds to the selected inputs. This threat is distinguished by the deliberate choice and manipulation of input data as opposed to random selections or manipulation of the physical characteristics involved in input/output operations. Manipulation may involve direct control of the I/O, clock, or power lines to generate security-critical information either directly or through inference.

T.Forced_State_Change: Forced State Change (SCSUGmod: T.Forcd_Rst)

An attacker may force the TOE into a nonsecure state through inappropriate termination of selected operations.

Attempts to generate a nonsecure state in the TOE may be through premature termination of transactions or communications between the TOE and the card-

reading device, insertion of interrupts, or by selecting related applications that may leave files open.

T.Inv_Inp: Invalid Input (SCSUG)

An attacker or authorized user of the TOE may compromise the security features of the TOE through the introduction of invalid inputs.

Invalid input may take the form of operations, which are not formatted correctly, requests for information beyond register limits, or attempts to find and execute undocumented commands. The result of such an attack may be a compromise of the security functions, generation of exploitable errors in operations, or release of protected data.

T.Spoof: Spoofing Legitimate System Services (TSRD)

An attacker tricks users into interacting with spurious system services, e.g., an unauthorized (bogus) terminal, that request sensitive information from the TOE.

The attack method may involve writing software to spoof users or modifying message protocol information in transit.

T.UA_Use: Unauthorized Program Use (SCSUGmod: UA_Load)

An attacker may utilize unauthorized programs to penetrate or modify the security functions of the TOE.

Some commands and functions may be built into the TOE that are never utilized in the intended application(s). Use of these existing but unauthorized operations may be attempted to create a compromise in TOE security. This threat is distinguished by the use of commands that exist but are not used in any of the authorized operational modes of the TOE.

3.2.1.3 Threats Associated with Control of Access

T.First_Use: Fraud on First Use (SCSUG)

An attacker may gain access to TOE information by unauthorized use of a new, previously unissued TOE.

The process of issuance may involve setting of indicators in the TOE or notification by the TOE to the (external) issuing bodies that this specific TOE is

now in operation. Attempts to use an unissued TOE without such mandated approval could result in fraudulent use.

T.Impers: Impersonation (SCSUG)

An attacker may gain access to TOE information by impersonating an authorized user of the TOE.

Impersonation may be accomplished through using a stolen TOE and spoofing the authentication mechanism(s). The TOE is required to allow certain roles be granted certain privileges. Impersonation of a user with such privileges could expose security functions or information that is to be protected by the TOE from unauthorized release.

3.2.1.4 Threats Associated with Unanticipated Interactions

T.App_Ftn: Use of Unallowed Application Functions (SCSUG)

An attacker may exploit interactions between applications to expose sensitive TOE or user data.

Interactions may include execution of commands that are not required or allowed in the specific application being performed. Examples include use of native Token OS functions that are unnecessary or that could compromise security. Inappropriate interactions could also include passing secure information such as PINs or cryptographic data between applications, or transferring value or information into applications that have been exited.

T.Fail_Secure: Failing in a Nonsecure State (TSRD)

An attacker may cause failure of the TOE security functions by exposing the TOE to conditions outside of its normal operating range, causing the TOE to enter a nonsecure state.

T.LC_Ftn: Use of Unallowed Life-Cycle Functions (SCSUG)

An attacker may exploit interactions between life-cycle functions to expose sensitive TOE or user data.

Interactions may include execution of commands that are not required or allowed in the specific phase of operation being executed. Examples include use of test,

debug, or native token OS functions that are unnecessary or that could compromise security.

T.Res_Con: Resource Contention (SCSUG)

A user or attacker may willfully, or through negligence, monopolize resources of the TOE, denying service to another user or function.

If the limited resources of the TOE are allocated to a user or attacker without the authorization of the owner of the resource, then another user or function that requires the same resource may not be able to operate normally.

3.2.1.5 Threats Regarding Cryptographic Functions

T.Crypt_Attk: Cryptographic Attack (SCSUGmod: T.Crypt_Atk, T.Reuse)

An attacker may defeat security functions through a cryptographic attack against the algorithm, through cryptanalysis on encrypted data, or through a brute-force attack.

There is no protection against inherent flaws in algorithms. However, given any algorithm, there is a list of countermeasures that the implementer should follow.

3.2.1.6 Threats that Monitor Information

T.Hacker_Comm_Eavesdrop: Hacker Eavesdrops on User Data Communications (TSRD & SCSUG: T.Reuse)

Hacker obtains user data by eavesdropping on communications lines.

This threat is relevant when the system must exchange user data with a remote system, and the confidentiality of that data is important.

T.I_Leak: Information Leak (SCSUG)

An attacker may exploit information that is leaked from the TOE during normal usage.

Leakage may occur through emanations, variations in power consumption, I/O characteristics, clock frequency, or by changes in processing time requirements. This may be interpreted as a covert channel transmission but is more closely

related to measurement of operating parameters. These may be derived either from direct (contact) measurements or measurement of emanations and can then be related to the specific operation being performed. An attacker may use differential power analysis or make electrical observations to exploit leaked information from the TOE.

Passive analysis security attacks on tokens may be timing attacks or waveform attacks. Analysis techniques such as Simple Power Analysis (SPA), Differential Fault Analysis (DFA), and Differential Power Analysis (DPA) may be used to perform security attacks. Recent attack developments include voltage manipulation, glitching, and combination attacks. Although external attacks are under constant development and improvement, token vendors are also making rapid progress in combating these attacks.

DPA can be used to break implementations of some symmetric or asymmetric algorithms. This technique is being used to reverse-engineer unknown algorithms and protocols by using DPA data to test hypotheses about a device's computational process. DPA and SPA can use power consumption measurements to extract secret keys from tamper-resistant devices. [4]

T.Link: Linkage of Multiple Observations (SCSUG)

An attacker may observe multiple uses of resources or services and, by linking these observations, deduce information that would reveal critical security information.

The combination of observations over a period of many uses of the TOE or the integration of knowledge gained from observing different operations may reveal information that allows an attacker to either learn information directly or to formulate an attack that could further reveal information that the TOE is required to keep secret.

3.2.1.7 Miscellaneous Threats

T.Clon: Cloning (SCSUG)

An attacker may clone part or all of a functional TOE to develop further attacks.

The information necessary to successfully clone part or all of the IC may be derived from detailed inspection of the IC itself or from illicit appropriation of design information.

Counterfeit smart cards can be mass produced using a thermal dye printer, an embosser, and an encoder.

T.Env_Strs: Environmental Stress (SCSUG)

An attacker may exploit failures in the TOE induced by environmental stress.

Exposure of the integrated circuit to conditions outside its specified operating range may result in malfunction or failure of security-critical components, allowing manipulation of programs or data. These conditions could either be extremes (high or low) in normal parameters such as temperature, voltage, or clock frequency, or could be the introduction of abnormal conditions such as external energy fields. The goal may be to generate an immediate failure, leading to unauthorized exposure of secure information, or to stimulate premature aging, thereby generating an end-of-life failure.

T.Lnk_Att: Linked Attacks (SCSUG)

An attacker may perform successive attacks with the result that the TOE becomes unstable or some aspect of the security functionality is degraded. A following attack may then be successfully executed.

Monitoring outputs while manipulating inputs in the presence of environmental stress is an example of a linked attack.

T.Rep_Atk: Repetitive Attack (SCSUG)

An attacker may utilize repetitive, undetected attempts at penetration to expose memory contents or to change security-critical elements in the TOE.

Repetitive attempts related to some or all of the other threats discussed herein may be used to iteratively develop an effective penetration of the TOE security. Monitoring outputs while manipulating inputs in the presence of environmental stress is also an example of repetitive penetration.

3.2.2 Threats Addressed by the Operating Environment

T.Hacker_Social_Engineer: Social Engineering (TSRD)

A hacker uses social engineering techniques to gain information about system entry, use, design, or operation.

This threat always exploits non-IT vulnerabilities, possibly in conjunction with IT vulnerabilities.

T.Privilege: Abuse by Privileged Users (SCSUGmod)

A careless, willfully negligent, or hostile administrator or other privileged user may create a compromise of the TOE assets through execution of actions that expose, change, or destroy the security functions or the protected/security-critical data.

A privileged user or administrator could directly implement or facilitate attacks based on any of the threats described here. TOE assets are defined as information or resources to be protected by countermeasures of the TOE (e.g., user data and cryptographic keys).

3.3 Organizational Security Policies

The organizational security policies discussed below are addressed by DoD PKI and KMI Token compliant TOEs. Each policy is stated in bold type font, which may be followed by an explanation, in normal font, that supplies additional information and interpretation.

P.Control_of_Applications: Control of Applications

1) A DoD PKI and Token must contain an application manager as detailed in Appendix F in Section 2.3 of the *Application Specification for the DoD PKI and KMI Token.*
2) Application signatures must be used to sign approved applications to control the loading of applications onto the DoD PKI and KMI token. Valid applications must be signed by an approved DoD entity as detailed in Appendix F in Section 2.4 of the *Application Specification for the DoD PKI and KMI Token.*
3) Any proposed DoD PKI and KMI Token application signature algorithm and key length must comply with Appendix E of this Protection Profile.
4) The token platform developer must provide guidance to develop secure applications on the developer's operating system platform.

P.Key_Length: Cryptographic Key Length

X.509 Certificate Policy for the U. S. Department of Defense. Digital Signature Standard keys shall use at least 160 bit private key and at least 1024 bit prime modulus. Minimum public key size shall be 1024 bits for Key Exchange Algorithm (KEA). Minimum public key size shall be 2048 bits for RSA. For Class 4, Elliptic Curve Digital Signature Algorithm key prime field (//p//) shall be not less than 384 bits.

P.Protection_Mechanisms: Application of Protection Mechanisms

DoD Information Assurance Guidance and Policy Memorandum 6-8510. Protection mechanisms shall be applied such that the TOE maintains the appropriate level of confidentiality, integrity, authentication, and nonrepudiation based on mission criticality, sensitivity of information handled by the system, and need to know.

Each authorized role has certain specified privileges that allow access only to selected portions of the TOE and its contained information. Access beyond those specified privileges could result in exposure of security-related information.

4 Security Objectives

4.1 Security Objectives for the TOE

This section defines the security objectives of the TOE. These security objectives reflect the stated intent to counter identified threats and/or comply with any organizational security policies identified. Each objective is stated in bold type font. Most objectives are followed by an explanation, in normal font, that supplies additional information and interpretation.

O.Auth_Protect: Protection of Authentication Data

Authentication data maintained by the TOE will be protected from disclosure and modification.

Authentication data are used to verify the claimed identity of the user. Normally, the protection will be provided by encrypting the authentication data.

O.Authenticate: Authentication of Users and SSOs

Before cryptographic or other DoD data[1] are accessed, either the user's identity or an administrative role will be authenticated by the TOE.

O.Control_of_Applications: Control of Applications

A DoD PKI and Token must contain an application manager that will control the loading and use of applications on the TOE. The loading of applications onto the TOE will be controlled by using application signatures from an approved DoD entity. Any proposed DoD PKI and KMI Token application signature algorithm and key length must comply with Appendix E of this Protection Profile. Lastly, the token platform developer must provide guidance to develop secure applications on the developer's operating system platform.

O.Crypt: Cryptography

[1] DoD data are all data on the TOE located below the DoD directory. These data are owned by the DoD. It includes DoD executables, PINs, cryptographic keys, and user personal information.

The TOE must perform cryptographic functions with sufficient strength for Sensitive But Unclassified (SBU) data.

The TOE must perform any cryptographic operations consistent with established cryptographic usage polices and standards for SBU data.

O.DAC: Data Access Control

The TOE must provide each authorized user with the means of controlling and limiting access to the objects and resources it owns or for which it is responsible, on the basis of user identity or role and in accordance with the P.Protection_Mechanisms Security Policy.

The TOE may have a variety of users (DoD and Non-DoD), administrators, card issuers, associations, etc., each requiring some control over the assets being handled. Some rules will apply in all cases. These are represented in security functional requirement FDP_ACF.1. The remainder must be explicitly stated as required by the needs of the owners of the data.

O.D_Read: Data Read Format

The TOE shall format data passing between modules on the IC such that information (user and TSF data) is not exposed.

The TOE must act in a fashion that does not expose information being transferred between processing and storage modules inside the IC to any greater risk of compromise than that derived from long-term storage. Bus scrambling is a technique that reduces the risk of exposing information passing between modules.

O.Data_Exchange_Conf: Enforce data exchange confidentiality

Protect user data confidentiality when exchanging data with a host.

This objective can support several types of data exchange policies, including those that do not allow communications between the local system and specific remote sites, as well as those that constrain the types of information that can be sent over communications lines.

O.Env_Strs: Environmental Stress

The TOE must protect itself against compromise by having a structure that neither reveals security information nor operates in an insecure fashion when exposed to

out-of-standard conditions (high or low) in the environment, including such factors as temperature, voltage, clock frequency, and external energy fields.

The basic TOE must be designed and fabricated so that it continues to provide security to its critical information, including user assets and internal security information, even when exposed to environmental stress. Environmental stress may be a result of the normal environment in which the TOE is used, but it may also be representative of an attack against it. In the event of attack, stress may be the only driving force or it may be used in conjunction with one or several other attacks. This objective should work to prevent disclosure of security-related information.

O.Fail_Secure: Preservation of secure state for failures in critical components

Preserve the secure state of the system in the event of a secure component failure.

This objective is relevant if the TOE needs to continue some form of operation in the presence of failures. The scope of the identified failures for which the system will fail secure will, in general, directly impact the feasibility and cost of implementing this protection feature.

O.I_Leak: Information Leak

The TOE must provide the means of controlling and limiting the leakage of information in the TOE so that no useful information is unintentionally revealed over the power, ground, clock, reset, or I/O lines.

The TOE must be designed and programmed so that analysis of such elements as power consumption does not reveal information about processing operations or compromise secure information.

O.Init: Initialization

An initialized TOE not in the totally locked state must assume the nonauthenticated state immediately upon power-up, reset, or after other restart conditions.

The TOE must always start in a defined and controlled state regardless of how it was reset. This objective works to prevent attacks that attempt to upset the operation and leave the TOE in an undefined state.

O.Input_Probe: Probing by Selected Inputs

The TOE must be resistant to repeated probing through insertion of erroneous data.

If possible, the TOE must prevent the release of information though the analysis of responses to repetitive probing. This objective could also work through the detection of such attacks and the initiation of corrective actions to counter such attempts.

O.Key_Encrypt: Encryption of Stored Keys

Keys stored in nonvolatile memory on the TOE must be encrypted.

O.Life_Cycle: Life-Cycle Functions

The TOE must provide means of controlling and limiting the use of life-cycle-specific commands to the life-cycle stages in which they are intended.

The design and implementation of the TOE must be such that the only commands available to a specific operation are related to the TOE life cycle appropriate to that application. Thus, elements such as debug or one-time loading of identification registers should never be available during operational TOE use.

O.Log_Prot: Logical Protection

The TOE must protect itself against logical compromise by having a structure that is resistant to logical manipulation or modification.

The TOE must be designed and programmed so that it resists attempts to compromise its security features through attacks on its logical operation. The TOE must prevent the release of security-related information while it is operating properly in the presence of logic probes and command modifications.

Updated versions of the TOE should counter vulnerabilities discovered in previous TOE versions.

O.Mult_App: Multiple Applications

The TOE must support an application (or applications) while providing and maintaining security between and among the various resident elements.

The design and implementation of the TOE must be such that each application or major operational unit cannot affect the secure operation of other such applications. This separation must be maintained such that information that is

restricted to a single application is not accessible elsewhere, nor can it be changed except from within that application.

O.Phys_Prot: Physical Protection

The TOE must be resistant to physical attack or be able to create difficulties in understanding the information derived from such an attack.

Techniques used to achieve physical protection for the TOE include: protective layering, special rules regarding IC layout, removal of test pads after completion of initial (wafer) testing, custom-made cell families, hidden logic functionality, bus scrambling, serpentine patterns, coatings, and active and passive tamper techniques.

O.Res_Access: Resource Access

The TOE shall protect its resources against monopolization by a user or attacker to the detriment of other users of the TOE.

The TOE should be designed and implemented so that resource allocation is controlled in a manner that supports all intended users.

O.Role_Man: Role Management

Management of roles for the TOE is performed in a secure manner.

The various roles involved in working with the TOE are established in the development and user community through the TOE manufacturers, card issuing bodies, etc. These roles will be managed off-card by these or other bodies. However, the TOE is required to recognize the defined roles, which is often done through the use of special (transport, personalization, initialization, etc.) keys. The TOE is not required to support their management.

Table 4-1 Required Roles

Role	Role Description
SSO	System Security Officer (SSO) who performs a set of cryptographic intialization and management functions.
DoD User	User of the TOE who is allowed access to DoD data
Non-DoD User	User of the TOE who is not allowed access to DoD data

O.User_Data_Control: User Data Control

The setting or modification of authentication, initial security, and personalization data must be controlled.

O.Secure_Host_Comms: Secure Host Communications

The TOE and the host shall establish a secure channel, using a session key composed of components created by the TOE and the host, before exchanging cryptographic or other DoD data.

O.Self_Test: Self-Test

Self-tests shall ensure the TOE is functioning properly. Integrity of all code on the TOE shall be checked. Cryptographic and other security-critical functions shall be tested. These tests shall be performed during power-up and under certain conditions.

O.Set_Up: Set up Sequence

The TOE shall require that the SSO updates the preset (default) SSO authentication data prior to entering the Nonauthenticated state.

The TOE must be placed into operation in a controlled and defined manner. This acts to prevent use of the TOE before all of the protective measures may be enabled or protective codes entered.

O.Tamper_Response: Respond to Tamper

The TOE shall respond to physical tampering against specified system devices and components.

O.Tamper_Response provides capabilities to automatically respond to physical attacks against specified parts of the TOE, thereby resisting such attacks. The automatic response may take various forms but generally involves direct actions (e.g., shutting a system down) rather than notification actions.

O.Trial: Trial-and-Error Resistance

The TOE authentication mechanism is resistant to spoofing by trial and error.

Authentication data must be random from a sample size of at least 1 million. Furthermore, no more than eight authentication attempts are allowed.

O.Unlink: Linkage

The TOE must provide the means of allowing an entity to make multiple uses of resources or services without other entities being able to link those uses together.

The TOE should be designed and implemented so that no information is exposed in any normal operation that would contribute to a breach in security in another operation. This objective should work to prevent such disclosure.

O.Volatile_Memory: Destruction of Volatile Memory

The contents of volatile memory cannot be retrieved after power is removed from the TOE or a failure occurs.

4.2 Security Objectives for the Environment

Security objectives for the environment should trace back to aspects of identified threats that are not completely countered by the TOE and/or organizational security policies or assumptions not completely met by the TOE. Security objectives for the enviroment may be a reiteration of the assumptions portion of the TOE security environment. Thus, environmental security objectives are statements about aspects of the TOE's environment that need to be addressed by mechanisms outside of the TOE. They assist in completely countering the threats and supporting the policies and assumptions for the TOE by specifying objectives that should be addressed in the TOE's environment. Assurance requirements address the environmental objectives. These security objectives also levy additional requirements on the environment that are outside the scope of this PP.

Each environmental objective is stated in bold type font. Most objectives are followed by an explanation, in normal font, that supplies additional information and interpretation.

OE.Con_Cont: Code Configuration Control

The TOE will be labeled with a unique instance identifier that establishes its composition, and controls will be provided to ensure that the components have not been modified.

OE.Con_Des: Control of Design

Those responsible for the TOE must ensure that design information, details of hardware security mechanisms, IC specifications, IC databases, schematics/layouts,

software specifications, detailed designs, source code, or any other information are accessible only by authorized personnel.

Information that could lead to a compromise in security during TOE operation is routinely available during the design and manufacture of the TOE. This information must be protected to prevent its availability to hostile parties.

OE.Con_Prod: Control of Product

The manufacturing process shall ensure the protection of the TOE from any kind of unauthorized use such as tampering or theft.

During various stages of manufacture and preparation for use, the TOE may exist in a variety of incomplete through finished forms. These instantiations of the TOE must be protected to prevent their becoming available to hostile parties.

OE.Con_Tools: Control of Tools

The TOE development process shall ensure the protection of the development tools from any kind of unauthorized use such as tampering or theft.

A variety of tools are routinely used during the development and test of the TOE. These tools could provide significant information to a hostile party regarding the functionality of the TOE security systems and, thus, must be protected to prevent them from becoming available to hostile parties.

OE.Dlv_Aud: Delivery Audit

Procedures shall ensure that all nonconformance to mandated delivery processes are detected and that corrective actions are taken in case of improper operations.

OE.Dlv_Proc: Delivery Procedures

Procedures such as validation of code signatures shall ensure protection of TOE material/ information during delivery.

Numerous IC manufacturers, chip embedders, smart card personalizers, issuers, and others may have access to the TOE and its various support information prior to issuance. This information may be particularly vulnerable during transport between the various representatives. This objective should prevent this information from becoming available to hostile parties. Prevention includes

checking the verification of signed code that is downloaded prior to execution. A well-known example is checking digital signatures on signed Java applets.

OE.Dlv_Trn: Delivery Training

Procedures shall ensure that people dealing with the procedures for delivery (shipping department, carriers, reception department) have the required skill, training, and knowledge to meet the procedure requirements and to act fully in accordance with the above expectations.

Numerous IC manufacturers, chip embedders, smart card personalizers, issuers, and others may have access to the TOE and its various support information prior to issuance. This information may be particularly vulnerable during transport between the various representatives. This objective should prevent this information from becoming available to hostile parties.

OE.Ident: TOE Identification

Procedures must support the recording and preservation of TOE identification information on the TOE prior to being issued to the user.

The TOE consists of hardware and software elements. The software may be stored in a hard mask (through incorporation in the ROM photomask) or in nonvolatile memory. The hardware could have optional features that might or might not be enabled. It is therefore essential that an accurate identification be established for the exact instantiation of the final product compliant to this protection profile.

OE.Key_Gen: Key Generation

Key exchange keys are generated in a secure manner in accordance with X.509 Certificate Policy.

OE.Mask_Prot: Photomask Protection

The photomask fabrication management process shall ensure the protection of the mask from any kind of unauthorized use such as tampering or theft.

The photomask represents the instantiation of the hardware elements of the TOE and may contain ROM code. Information about secure functions and mask-programmed software and codes are included in the TOE photomasks. Furthermore, availability of the photomasks could significantly reduce the effort

required to clone all or part of the TOE. The photomasks must therefore be protected to prevent them from becoming available to hostile parties.

OE.Personnel: Personnel

Personnel working as administrators or in other privileged positions shall be carefully selected and trained for reliability.

OE.SW_Develop: Software Development Process

All code to be used for the TOE will be developed using a software development process that is standard and consistent across the organization.

OE.Sample_Acs: Sample Access

Samples used to run tests shall be accessible only by authorized personnel.

The preparation of samples, sometimes in large quantities, is routine during the development of a fully operational TOE. These samples represent the TOE in a variety of incomplete through finished forms. These instantiations must be protected to prevent them from becoming available to hostile parties.

OE.Sec_Com: Secure Communication

Only a trusted host[2] can establish a secure connection with the TOE.

The secure connection implies that the TOE is in a DoD authenticated state and that the host can be trusted.

OE.Train: User Training

Users will be trained on the usage policy of the TOE in accordance with proper security procedures.

2 Device to which a token authenticates to establish a secure communication path.

5 IT Security Requirements

5.1 TOE Security Functional Requirements

This section contains the security functional requirements that must be satisfied by a TOE compliant with the DoD PKI and KMI Token PP. Security Function Policies used by the security functional requirements are listed in section 5.1.1. Components, which group related security functional requirements, are listed in Table 5-2. The security functional requirements are listed in sections 5.1.3–5.1.10. All the requirements contained in this section are selected from Part 2 of the CC. Refer to the "Conventions and Terminology" section of this PP for an explanation of the conventions used in this section.

5.1.1 Strength of Function Claims

The statement of the TOE security requirements must include a minimum strength level for the TOE security functions realized by a probabilistic or permutational mechanism, except for cryptographic functions. For this PP, the minimum level will be SoF-medium.

An explicit SoF metric is defined for FIA_UAU.1.2. SoF will be demonstrated for the authentication mechanism so that for each attempt to use the authentication mechanism, the probability that a random attempt will succeed is less than one in a million.

5.1.2 Identification of Standards Compliance Methods

For this PP, cryptographic operations and key management functions must meet FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities. The methodology used to show compliance to FIPS 140-2 standards will be specified by the designated approval authority of the TOE-user organization. Authorized certificates used by a PP-compliant TOE must be DoD PKI Class 4, X.509 certificates.

5.1.3 Security Function Policies

Several of the functional requirements in section 5.1 reference Security Function Policies (SFPs). SFPs are named pieces of requirements. They are not organizational policies. Each SFP is named below in bold type font. The short name for the SFP is preceded by a label of "**SFP**". Following each named SFP is an explanation, in normal font, that supplies additional information and interpretation.

The SFPs used by the functional requirements in this PP are listed below:

SFP.DAC: Data Access Control Security Function Policy

Table 5-1 defines access privileges by role and information type. The Data Access Control Security Policy (SFP.DAC) is used in the access control, export, and import of data, and management of security attributes requirements.

The SSO role can only be assumed after successful authentication to the TOE.

Table 5-1 Access Control Table

Information Type	Role	Function[3]
User Authentication data[4]	SSO	initialize, modify
SSO Authentication data	SSO	Set[5], update
Personalization data[6]	SSO	set, read
	User	read
DoD PKI signature, e-mail signature keys	DoD User	generate, access
Key exchange keys	DoD User	load, access
Certificate[7]	SSO, DoD User	load
	All	access
Data[8]	DoD User	encrypt,[9] decrypt, sign
DoD Directories	SSO	create, rename, read, delete
	DoD User	read, write, resize
Non-DoD Directory	SSO	create, rename, read, delete
	Non-DoD User	read, write, resize
DoD File	DoD User	as specified by application
Non-DoD File	Non-DoD User	as specified by application
Security Attributes	SSO	default, modify, delete
Application	SSO, DoD User	load

SFP.Info_Flow_Control: Information Flow Control

Information only flows between subjects by way of data objects with access specified by SFP.DAC.

[3] Operations allowed for the given type of information in the given role.
[4] Information used to verify the claimed identity of the user.
[5] SSO will update the preset (default) SSO authentication data.
[6] User name, social security number, etc.
[7] DoD PKI, e-mail, key exchange, root CA, CA, etc.
[8] E.g., e-mail.
[9] Including signing.

The information is represented as user and TSF data. This should rule out information flowing between subjects (e.g., applications and OS processes) through covert channels. Labels may be used for data to control the flow of information.

5.1.4 Security Functional Components

Table 5-2 summarizes the security functional components that appear in this PP.

Table 5-2 Security Functional Components

Component	Component Name
FCS_CKM.1	Cryptographic key generation
FCS_CKM.2	Cryptographic key distribution
FCS_CKM.3	Cryptographic key access
FCS_CKM.4	Cryptographic key destruction
FCS_COP.1	Cryptographic operation
FDP_ACC.1	Subset access control
FDP_ACF.1	Security attribute based access control
FDP_DAU.1	Basic data authentication
FDP_ETC.1	Export of user data without security attributes
FDP_IFC.1	Subset information flow control
FDP_IFF.1	Simple security attributes
FDP_IFF.3	Limited illicit information flows
FDP_ITC.1	Import of user data without security attributes
FDP_ITT.1	Basic internal transfer protection
FDP_RIP.1	Subset residual information protection
FIA_AFL.1	Authentication failure handling
FIA_ATD.1	User attribute definition
FIA_SOS.1	Verification of secrets
FIA_UAU.1	Timing of authentication
FIA_UAU.6	Re-authenticating
FIA_UAU.7	Protected authentication feedback
FIA_UID.2	User identification before any action
FMT_MOF.1	Management of security functions behavior
FMT_MSA.1	Management of security attributes
FMT_MSA.2	Secure security attributes
FMT_MSA.3	Static attribute initialization
FMT_MTD.1	Management of TSF data
FMT_MTD.2	Management of limits of TSF data
FMT_MTD.3	Secure TSF data
FMT_REV.1	Revocation
FMT_SMR.2	Security roles
FMT_SMR.3	Assuming roles

Component	Component Name
FPT_AMT.1	Abstract machine testing
FPT_FLS.1	Failure with preservation of secure state
FPT_ITI.1	Inter-TSF detection of modification
FPT_ITT.1	Basic internal TSF data transfer protection
FPT_PHP.1	Passive detection of physical attack
FPT_PHP.3	Resistance to physical attack
FPT_RCV.4	Function recovery
FPT_RVM.1	Non-bypassability of the TSP
FPT_SEP.1	TSF domain separation
FPT_TST.1	TSF testing
FRU_RSA.1	Maximum quotas
FTP_ITC.1	Inter-TSF trusted channel

5.1.5 Cryptographic support (FCS) requirements

5.1.5.1 Cryptographic key generation (FCS_CKM.1)

FCS_CKM.1.1
The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm from list of required supported cryptographic algorithms in Appendix E and specified cryptographic key sizes of

- at least 160 bit private key with at least 1024 bit prime modulus for Digital Signature Standard keys;
- at least 1024 bit public key for Key Exchange Algorithm (KEA);
- at least 2048 bit public key for RSA;
- at least 384 bit for Elliptic Curve Digital Signature Algorithm key prime field (//p//);

that meet the following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities, X.509 Certificate Policy, and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 4 (see Appendix F). Each generated key must be unique and have its own unique identification number.

Application notes:
- Throughout the requirements in this PP, references are made to requirements for FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities. If the DoD Common Access Card issuing infrastructure is not capable of issuing two different levels of cards, then all CACs will be required to meet FIPS 140-2 Level 3.

5.1.5.2 Cryptographic key distribution (FCS_CKM.2)

FCS_CKM.2.1
The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method encryption with key exchange keys for symmetric keys in a DoD

Authenticated State that meets the following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 6 (see Appendix F).

Application note: Possession of the Key Exchange Key authenticates the host to the TOE.

5.1.5.3 Cryptographic key access (FCS_CKM.3)

FCS_CKM.3.1
The TSF shall perform encryption of cryptographic keys in nonvolatile memory in accordance with a specified cryptographic key access method, cryptographic key storage, that meets the following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 2 (see Appendix F).

5.1.5.4 Cryptographic key destruction (FCS_CKM.4)

FCS_CKM.4.1
The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method, zeroization, that meets the following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 5 (see Appendix F).

5.1.5.5 Cryptographic operation (FCS_COP.1)

FCS_COP.1.1
The TSF shall perform signing e-mail hash values and wrapping or unwrapping e-mail session keys in accordance with a specified cryptographic algorithm from a list of required supported cryptographic algorithms in Appendix E and cryptographic key sizes of

- at least 160 bit private key with at least 1024 bit prime modulus for Digital Signature Standard keys;
- at least 1024 bit public key for Key Exchange Algorithm (KEA);
- at least 2048 bit public key for RSA;
- at least 384 bit for Elliptic Curve Digital Signature Algorithm key prime field (//p//);

that meet the following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities and X.509 Certificate Policy and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 7 (see Appendix F).

5.1.6 User data protection (FDP) requirements

5.1.6.1 Subset access control (FDP_ACC.1)

FDP_ACC.1.1
The TSF shall enforce the SFP.DAC on:
Subjects: SSO, DoD users, non-DoD users;
Objects: Authentication data, personalization data, and initial security data,

- objects in DoD directory: root certificate, user certificate, user private key, directories, applications;
- objects in non-DoD directory: root certificate, user certificate, user private key, directories, applications; and

operations among subjects and objects covered by the SFP. Additional requirements are found in the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 8 (see Appendix F).

5.1.6.2 Security attribute based access control (FDP_ACF.1)

FDP_ACF.1.1
The TSF shall enforce the Data Access Control Security Function Policy (SFP.DAC) to objects based on role including those in Table 5-3, [*ST assignment: security attributes, named groups of security attributes*].

Table 5-3 Roles

Role	Role Description
SSO	System Security Officer (SSO) who performs a set of cryptographic intialization and management functions.
DoD User	User of the TOE who is allowed access to DoD data
Non-DoD User	User of the TOE who is not allowed access to DoD data

FDP_ACF.1.2
The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:
(a) Roles: as defined in SFP.DAC
(b) File Control: The process and commands for creating the application file structure, including file access, shall be defined in the ST in accordance with SFP.DAC.
(c) Crypt: The platform must be capable of securely storing PINs and other secret data, including cryptographic data, using access control provisions which ensure that such data cannot be read from outside.

d) First-Use: Initial authentication should be performed on first use (as specified in the ST). Indication of first use shall not be alterable.

FDP_ACF.1.3
The TSF shall explicitly authorize access of subjects to objects based on the following additional rules:
(a) The TOE must maintain a list of allowed roles or individuals per directory and file.
(b) The list must contain (at a minimum) a cross-reference of access privileges (e.g., read, write, execute, etc.) to be associated with each role, the authentication method and data used for authentication required for each role.
(c) The TOE must maintain single-bit integrity of its access information. The data integrity of the access control information and file must be verified before granting access to the directory or file.
(d) A DoD-defined state prescribes those DoD applications that may execute on the TOE when the TOE is in that state. The Application Specification levies additional requirements for the use of applications by the TOE (see Appendix F).

Application Note: Appendix D, section 4, defines the DoD states. As stated in the Application Specification, an application can only be used by the TOE after its signature by a DoD entity is validated.

FDP_ACF.1.4
The TSF shall explicitly deny access of subjects to objects based on the:
(a) The TOE must maintain a list of allowed roles or individuals per directory and file.
(b) The list must contain (at a minimum) a cross-reference of access privileges (e.g., read, write, execute, etc.) to be associated with each role, the authentication method and data used for authentication required for each role.
(c) The list must be protected by the TOE's data integrity mechanism. The data integrity of the file (and list) must be verified before granting access to the directory or file.

5.1.6.3 Basic data authentication (FDP_DAU.1)

FDP_DAU.1.1
The TSF shall provide a capability to generate evidence that can be used as a guarantee of the validity of DoD data.

FDP_DAU.1.2
The TSF shall provide the SSO with the ability to verify evidence of the validity of the indicated information.

5.1.6.4 Export of user data without security attributes (FDP_ETC.1)

FDP_ETC.1.1

The TSF shall enforce the SFP.DAC when exporting user data, controlled under the SFP(s), outside of the TSC.

FDP_ETC.1.2
The TSF shall export the user data without the user data's associated security attributes.

5.1.6.5 Subset information flow control (FDP_IFC.1)

FDP_IFC.1.1
The TSF shall enforce the Information Flow Control Security Function Policy (SFP.Info_Flow_Control) on [*ST assignment: list of subjects, information, and operations that cause controlled information to flow to and from controlled subjects covered by the SFP*].

5.1.6.6 Simple security attributes (FDP_IFF.1)

FDP_IFF.1.1
The TSF shall enforce the SFP.Info_Flow_Control based on the following types of subject and information security attributes: application attribute.

Application note: "Application attribute" is a subject attribute for the application being executed. See Common Criteria, Version 2.1, Part 2, Section 1.3, paragraphs 30-37 for a discussion of attributes.

FDP_IFF.1.2
The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: as specified by the applications.

FDP_IFF.1.3
The TSF shall enforce the: none.

FDP_IFF.1.4
The TSF shall provide the following: none.

FDP_IFF.1.5
The TSF shall explicitly authorize an information flow based on the following rules: none.

FDP_IFF.1.6
The TSF shall explicitly deny an information flow based on the following rules: none.

5.1.6.7 Limited illicit information flows (FDP_IFF.3)

FDP_IFF.3.1
The TSF shall enforce the SFP.Info_Flow_Control to limit the capacity of information leaked over power, ground, clock, reset, or I/O lines to a [*ST assignment: maximum capacity*].

5.1.6.8 Import of user data without security attributes (FDP_ITC.1)

FDP_ITC.1.1
The TSF shall enforce the SFP.DAC when importing user data, controlled under the SFP, from outside of the TSC.

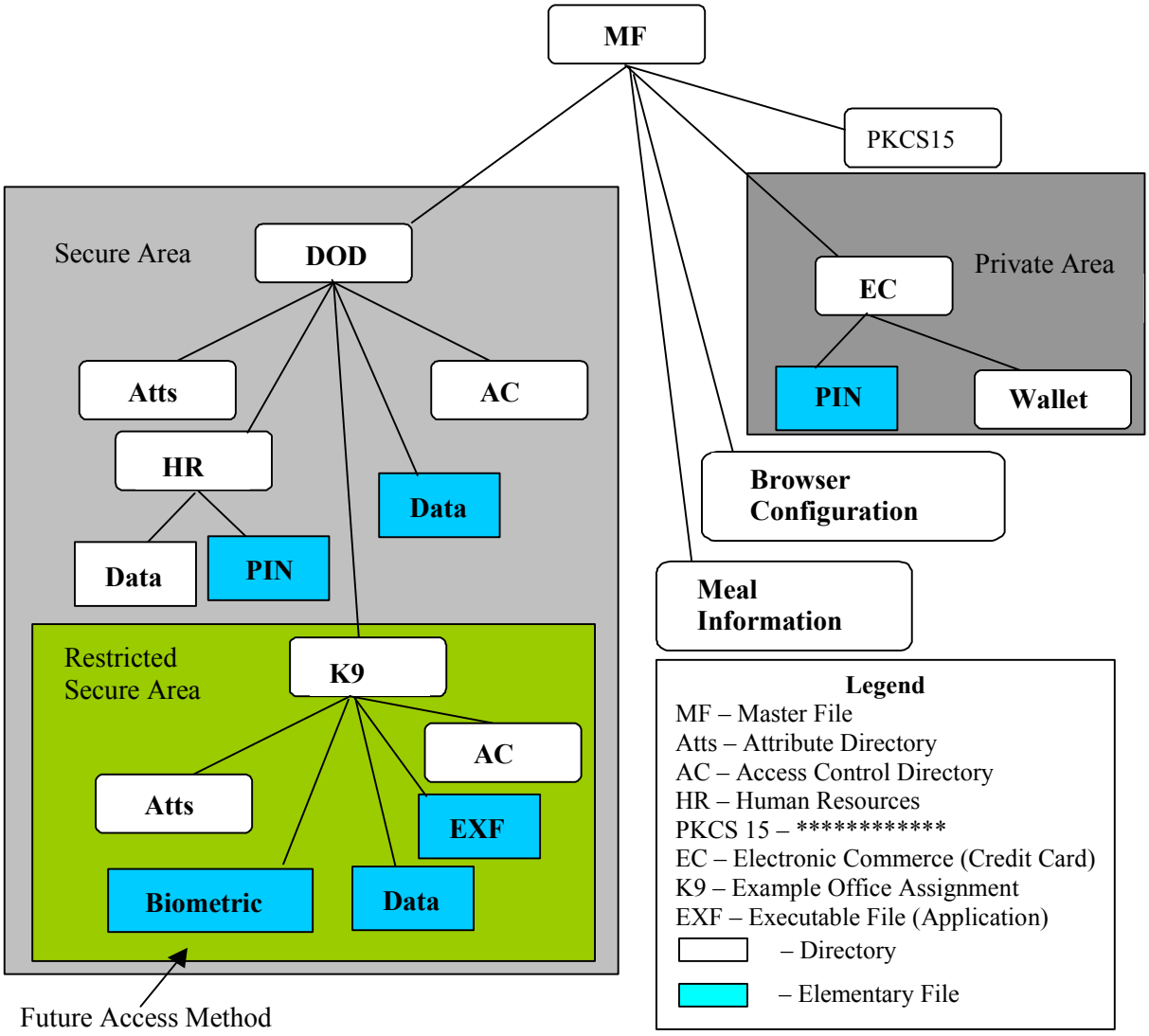
FDP_ITC.1.2
The TSF shall ignore any security attributes associated with the user data when imported from outside the TSC.

FDP_ITC.1.3
The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC:
(a) Data Load: The SSO controls the loading of data into the Master File (MF) (top-level directory), including the loading of non-DoD applications. Non-DoD applications have control within their directories. See Figure 5-1 in the application note below.
(b) Application Load: All loading of applications onto the TOE requires signature verification by the TOE as detailed in the Application Specification, Section 2.4 (see Appendix F).
Any proposed DoD PKI and KMI Token application signature algorithm and key length used must comply with Appendix E of this Protection Profile.
(c) Application Manager: As detailed in the Application Specification, Section 2.3 (see Appendix F), an application manager must be used by the TOE to:

1) Ensure that the application code signature is verified prior to card execution.
2) Associate an application with card security states or domains.
3) Provide a method for selecting an application to execute.
4) Provide a method for deleting an application.
5) Limit the number of load attempts of any given Application ID (AID) to three unsuccessful attempts.

Application note: An example of a token's directory structure is illustrated below in Figure 5-1. The directory structure shown is compliant with ISO 7816-4. Although this standard is intended for smart cards, many other types of tokens have adopted its use. The DoD directory (secure area) allocates resources for DoD objects (files). The DoD directory also controls the MF, granting resources on the token.

Figure 5-1 Directory Structure Example



5.1.6.9 Basic internal transfer protection (FDP_ITT.1)

FDP_ITT.1.1
The TSF shall enforce the SFP.DAC and SFP.Info_Flow_Control to prevent the disclosure or modification of user data when it is transmitted between physically separated parts of the TOE.

Application note: For a token, "physically separated parts of the TOE" means portions of the IC separated by bus lines or modules on the token.

5.1.6.10 Subset residual information protection (FDP_RIP.1)

FDP_RIP.1.1
The TSF shall ensure that any previous information content of a resource is made unavailable upon the deallocation of the resource from the following objects: DoD applications.

5.1.7 Identification and authentication (FIA) requirements

5.1.7.1 Authentication failure handling (FIA_AFL.1)

FIA_AFL.1.1
The TSF shall detect when eight (by the user) or four (by the SSO) unsuccessful authentication attempts occur related to login.

FIA_AFL.1.2
When the defined number of unsuccessful authentication attempts has been met or surpassed, the TSF shall cause the TOE to enter a locked state.

5.1.7.2 User attribute definition (FIA_ATD.1)

FIA_ATD.1.1
The TSF shall maintain the following list of security attributes belonging to individual users: role (e.g., SSO, user), and [*ST assignment: list of additional security attributes*].

5.1.7.3 Verification of secrets (FIA_SOS.1)

FIA_SOS.1.1 (Iteration 1)
The TSF shall provide a mechanism to verify that secrets meet a biometric or a password or PIN with minimum length of 6 bytes for the user. **The token must generate the 6-byte PIN, i.e., the user cannot determine his or her own PIN.**

FIA_SOS.1.1 (Iteration 2)
The TSF shall provide a mechanism to verify that secrets meet a biometric or a password or PIN with minimum length 16 bytes for the SSO. **While the user will have a personal password or PIN, each SSO can have a unique password/PIN. The SSO must establish a secure session to authenticate himself. The token must generate the 16-byte PIN, i.e., the SSO cannot determine his or her own PIN.**

Application notes:
- The SSO and user authentication mechanisms specified in this requirement serve as the authentication mechanisms required for entering the token's authenticated states as defined in Appendix D.

5.1.7.4 Timing of authentication (FIA_UAU.1)

FIA_UAU.1.1
The TSF shall allow:

- Information requests (e.g., request for help on the login procedure) about the TOE
- Initialization and entry of user personalization data by the SSO
- Signature key pair generation
- Obtaining directory or file information about non-DoD directories if the application that owns the directory allows it
- Accessing non-DoD data
- Use of non-DoD applications (e.g., e-commerce)
- [*ST assignment: additional TSF-mediated actions*] on behalf of the user to be performed before the user is authenticated.

FIA_UAU.1.2
The TSF shall require each user to be successfully authenticated before allowing any other TSF-mediated actions on behalf of that user.

Application note: Users and SSOs must be successfully authenticated prior to assuming any roles on the TOE.

5.1.7.5 Re-authenticating (FIA_UAU.6)

FIA_UAU.6.1
The TSF shall re-authenticate the user under the conditions during which token security may have been breached or when the user performs an action for which he or she may be held legally responsible, including

(a) periods of inactivity of at least 10 minutes
(b) use of signature keys for e-commerce applications
(c) storage of a prescription on the token by a doctor

5.1.7.6 Protected authentication feedback (FIA_UAU.7)

FIA_UAU.7.1
The TSF shall provide only nothing to the user while the authentication is in progress.

5.1.7.7 User identification before any action (FIA_UID.2)

FIA_UID.2.1

The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

5.1.8 Security management (FMT) requirements

5.1.8.1 Management of security functions behavior (FMT_MOF.1)

FMT_MOF.1.1
The TSF shall restrict the ability to determine, disable, enable, or modify the behavior of the functions:
(a) Changes to cryptographic key attributes including key type (e.g., public, private, secret), validity period, and use (e.g., digital signature, key encryption, key agreement, data encryption)
(b) Actions to be taken in the event of user authentication failure
(c) Actions that can be taken before the user is authenticated
(d) Rules for authentication
(e) Revocation rules
(f) List of actions that need to be taken in case of repetitive penetration attempts
(g) Conditions under which TSF self-testing occurs, such as during initial start-up, regular interval, or under specified conditions
to the SSO role.

5.1.8.2 Management of security attributes (FMT_MSA.1)

FMT_MSA.1.1 (Iteration 1)
The TSF shall enforce the SFP.DAC to restrict the ability to change_default, modify, or delete the security attributes roles, object owner, and the application to which an object belongs to the SSO.

FMT_MSA.1.1 (Iteration 2)
The TSF shall enforce the SFP.Info_Flow_Control to restrict the ability to access the security attributes data objects to roles specified in Table 5-1.

5.1.8.3 Secure security attributes (FMT_MSA.2)

FMT_MSA.2.1
The TSF shall ensure that only secure values are accepted for security attributes.

5.1.8.4 Static attribute initialization (FMT_MSA.3)

FMT_MSA.3.1 (Iteration 1)

The TSF shall enforce the SFP.DAC to provide restrictive default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.1 (Iteration 2)
The TSF shall enforce the SFP.Info_Flow_Control to provide restrictive default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2
The TSF shall allow the SSO to specify alternative initial values to override the default values when an object or information is created.

5.1.8.5 Management of TSF data (FMT_MTD.1)

FMT_MTD.1.1
The TSF shall restrict the ability to change default, query, modify, delete, clear [*ST assignment: other operations*] the SSO authentication data, [*ST assignment: list of additional TSF data*] to the SSO.

5.1.8.6 Management of limits on TSF data (FMT_MTD.2)

FMT_MTD.2.1
The TSF shall restrict the specification of the limits for [*ST assignment: list of TSF data*] to the SSO.

FMT_MTD.2.2
The TSF shall take the following actions, if the TSF data are at, or exceed, the indicated limits: cause the TOE to enter the DoD Locked State.

5.1.8.7 Secure TSF data (FMT_MTD.3)

FMT_MTD.3.1
The TSF shall ensure that only secure values are accepted for TSF data.

5.1.8.8 Revocation (FMT_REV.1)

FMT_REV.1.1
The TSF shall restrict the ability to revoke security attributes associated with the users, subjects, objects, and other additional resources within the TSC to SSO.

FMT_REV.1.2
The TSF shall enforce the rules upon exit from the DoD SSO Authenticated State.

5.1.8.9 Restrictions on security roles (FMT_SMR.2)

FMT_SMR.2.1
The TSF shall maintain the roles specified in Table 5-3.

FMT_SMR.2.2
The TSF shall be able to associate users with roles.

FMT_SMR.2.3
The TSF shall ensure that the conditions of successfully authenticating users and SSOs prior to assuming any roles on the TOE are satisfied.

5.1.8.10 Assuming roles (FMT_SMR.3)

FMT_SMR.3.1
The TSF shall require an explicit request to assume the following roles: SSO.

5.1.9 Protection of the TOE Security Functions (FPT) requirements

5.1.9.1 Abstract machine testing (FPT_AMT.1)

FPT_AMT.1.1
The TSF shall run a suite of tests in accordance with FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities during initial start-up, when conditions are outside the normal range, or when requested by the user to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF.

5.1.9.2 Failure with preservation of secure state (FPT_FLS.1)

FPT_FLS.1.1
The TSF shall preserve a secure state when the following types of failures occur: power supplied outside of normal operating range, physical tampering, faulty clock signal, or temperature outside normal operating range.

Application note: As an example, in cases of out-of-range temperatures and power and faulty clock signals, the TOE should automatically enter the Power-On State, requiring user authentication before any more actions. In cases of the detection of physical probing and

manipulation, the TOE should automatically enter the Totally Locked state. In these states, the TOE's volatile memory should not contain valid information. See Appendix D for more information about token states.

5.1.9.3 Inter-TSF detection of modification (FPT_ITI.1)

FPT_ITI.1.1
The TSF shall provide the capability to detect modification of all TSF data during transmission between the TSF and a remote trusted IT product within the following metric: cryptographic checksum.

FPT_ITI.1.2
The TSF shall provide the capability to verify the integrity of all TSF data transmitted between the TSF and a remote trusted IT product and perform an ignore of the TSF data and request that the originating trusted product to send the TSF data again if modifications are detected.

5.1.9.4 Basic internal TSF data transfer protection (FPT_ITT.1)

FPT_ITT.1.1
The TSF shall protect TSF data from disclosure when it is transmitted between separate parts of the TOE.

5.1.9.5 Passive detection of physical attack (FPT_PHP.1)

FPT_PHP.1.1
The TSF shall provide unambiguous detection of physical tampering that might compromise the TSF. **Physical protection measures in accordance with FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities must be taken to meet this requirement.**

Application note: The term "unambiguous" means detectable by the user. Physical tampering must be evident by inspection by the average user of the token.

FPT_PHP.1.2
The TSF shall provide the capability to determine whether physical tampering with the TSF's devices or TSF's elements has occurred.

Application note: Detection of tampering should place the TOE in the Totally Locked State.

5.1.9.6 Resistance to physical attack (FPT_PHP.3)

FPT_PHP.3.1
The TSF shall resist physical probing and manipulation, exposure to temperatures outside the normal operating range, faulty clock signals, and exposure to supplied power outside the operating range to the authentication information, private key(s), random number generator, cryptographic circuits, memory, and [*ST assignment: other devices/elements*] by responding automatically such that the TSP is not violated.

Application note: As an example, in cases of out-of-range temperatures and power and faulty clock signals, the TOE should automatically enter the Power-On State, requiring user authentication before any more actions. In cases of the detection of physical probing and manipulation, the TOE should automatically enter the Totally Locked state. In these states, the TOE's volatile memory should not contain valid information. See Appendix D for more information about token states.

5.1.9.7 Function recovery (FPT_RCV.4)

FPT_RCV.4.1
The TSF shall ensure that power supplied outside of normal operating range, physical tampering, faulty clock signal, temperature outside normal operating range, and [*ST assignment: other failures*] have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

Application note: As an example, in cases of out-of-range temperatures and power and faulty clock signals, the TOE should automatically enter the Power-On State, requiring user authentication before any more actions. In cases of the detection of physical probing and manipulation, the TOE should automatically enter the Totally Locked state. In these states, the TOE's volatile memory should not contain valid information. See Appendix D for more information about token states.

5.1.9.8 Non-bypassability of the TSP (FPT_RVM.1)

FPT_RVM.1.1
The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

Application note: TSP enforcement functions control the TOE's transition between states. Each state has prescribed functions that are allowed in that state. TSP enforcement functions also include those measures used to control the access to objects by subjects, as defined by SFP.Info_Flow_Control and SFP.DAC.

5.1.9.9 TSF domain separation (FPT_SEP.1)

FPT_SEP.1.1
The TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.

Application note: SFP.Info_Flow_Control and SFP.DAC prescribe the creation of a security domain that protects the TOE from untrusted subjects.

FPT_SEP.1.2
The TSF shall enforce separation between the security domains of subjects in the TSC.

5.1.9.10 TSF testing (FPT_TST.1)

FPT_TST.1.1
The TSF shall run a suite of self tests during initial start-up, **and** at the conditions when power, input frequency, or temperature are outside of their normal operating conditions to demonstrate the correct operation of the TSF.

FPT_TST.1.2
The TSF shall provide authorized users with the capability to verify the integrity of TSF data.

FPT_TST.1.3
The TSF shall provide authorized users with the capability to verify the integrity of stored TSF executable code.

5.1.10 Resource utilization (FRU) requirements

5.1.10.1 Maximum quotas (FRU_RSA.1)

FRU_RSA.1.1
The TSF shall enforce maximum quotas of the following resources:
memory, program space, [*ST assignment: controlled resources*] that individual user, defined group of users **(all SSOs)**, subjects **(applications)** can use simultaneously **and** over a specified period of time.

Application note: The defined group of users is SSOs, and executables

5.1.11 Trusted path/channels (FTP) requirements

5.1.11.1 Inter-TSF trusted channel (FTP_ITC.1)

FTP_ITC.1.1
The TSF shall provide a communication channel between itself and a remote trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of channel data from modification or disclosure. **Additional requirements are found in the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 9 (see Appendix F).**

Application note: As detailed in the Key Management and SSO Authentication Specification, all DoD data transmitted between the TOE and a host will be encrypted with a session key shared between the host and the TOE. Successfully sharing a session key implies the host is trusted. A session key, when used, shall be destroyed by both parties upon removal of the token from its reader, upon shutdown or closure of the supported host application, or upon a power failure. A new session key will be generated upon commencement of a new session.

FTP_ITC.1.2
The TSF shall permit the TSF, the remote trusted IT product to initiate communication via the trusted channel.

Application note: The remote trusted IT product is the trusted host.

FTP_ITC.1.3
The TSF shall initiate communication via the trusted channel for initial user authentication, cryptographic and DoD data exchange, loading applications, and [*ST assignment: other services for which trusted channel is required*].

5.2 TOE Security Assurance Requirements

The requirements in this section support specific objectives or are included to be consistent with an Evaluation Assurance Level (EAL) of 4. The TOE has an EAL4 *augmented* assurance level, which means that the addition of assurance components or the substitution of higher assurance components has been made to the group of EAL4 assurance requirements. The TOE Security Assurance Requirements for EAL4 have been augmented with AVA_VLA.3 (substitution for AVA_VLA.2) and ALC_TAT.3 (substitution for ALC_TAT.1). All of the assurance requirements contained in this section are selected from Part 3 of the CC.

Five of the assurance requirements have been refined. These refined assurance requirements are as follows:

ACM_CAP.4
ADV_IMP.1
ADV_LCD.1
ALC_TAT.3
AVA_VLA.3

Table 5-4 Assurance Requirements: EAL(4) Augmented

Assurance Class	Assurance Components
ACM	ACM_AUT.1, ACM_CAP.4, ACM_SCP.2
ADO	ADO_DEL.2, ADO_IGS.1
ADV	ADV_FSP.2, ADV_HLD.2, ADV_IMP.1, ADV_LLD.1, ADV_RCR.1, ADV_SPM.1
AGD	AGD_ADM.1, AGD_USR.1
ALC	ALC_DVS.1, ALC_LCD.1, ALC_TAT.3
ATE	ATE_COV.2, ATE_DPT.1, ATE_FUN.1, ATE_IND.2
AVA	AVA_MSU.2, AVA_SOF.1, **AVA_VLA.3**

5.2.1 Configuration management (ACM)

5.2.1.1 Partial CM automation (ACM_AUT.1)

Developer Action Elements:

ACM_AUT.1.1D
The developer shall use a CM system.

ACM_AUT.1.2D
The developer shall provide a CM plan.

Content and Presentation of Evidence Elements:

ACM_AUT.1.1C
The CM system shall provide an automated means by which only authorized changes are made to the TOE implementation representation.

ACM_AUT.1.2C
The CM system shall provide an automated means to support the generation of the TOE.

ACM_AUT.1.3C
The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.1.4C
The CM plan shall describe how the automated tools are used in the CM system.

Evaluator Action Elements:

ACM_AUT.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.1.2 Generation support and acceptance procedures (ACM_CAP.4)

Developer Action Elements:

ACM_CAP.4.1D
The developer shall provide a reference for the TOE. **Each unique copy of the TOE shall have its own serial number. Additional requirements are found in the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 1 (see Appendix F).**

ACM_CAP.4.2D
The developer shall use a CM system.

ACM_CAP.4.3D
The developer shall provide CM documentation.

Content and Presentation of Evidence Elements:

ACM_CAP.4.1C
The reference for the TOE shall be unique to each version of the TOE. **Each unique copy of the TOE shall have its own serial number.**

ACM_CAP.4.2C
The TOE shall be labeled with its reference. **Each unique copy of the TOE shall be labeled with its own serial number.**

ACM_CAP.4.3C
The CM documentation shall include a configuration list, a CM plan, and an acceptance plan.

ACM_CAP.4.4C
The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.4.5C
The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.4.6C
The CM system shall uniquely identify all configuration items.

ACM_CAP.4.7C
The CM plan shall describe how the CM system is used.

ACM_CAP.4.8C
The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.4.9C

The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.4.10C
The CM system shall provide measures such that only authorized changes are made to the configuration items.

ACM_CAP.4.11C
The CM system shall support the generation of the TOE.

ACM_CAP.4.12C
The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

Evaluator Action Elements:

ACM_CAP.4.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.1.3 Problem tracking CM coverage (ACM_SCP.2)

Developer Action Elements:

ACM_SCP.2.1D
The developer shall provide CM documentation.

Content and Presentation of Evidence Elements:

ACM_SCP.2.1C
The CM documentation shall show that the CM system, as a minimum, tracks the following: the TOE implementation representation, design documentation, test documentation, user documentation, administrator documentation, CM documentation, and security flaws.

ACM_SCP.2.2C
The CM documentation shall describe how configuration items are tracked by the CM system.

Evaluator Action Elements:

ACM_SCP.2.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2 Delivery and operation (ADO)

5.2.2.1 Detection of modification (ADO_DEL.2)

Developer Action Elements:

ADO_DEL.2.1D
The developer shall document procedures for delivery of the TOE or parts of it to the user. **Additional requirements are found in the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 10 (see Appendix F).**

ADO_DEL.2.2D
The developer shall use the delivery procedures.

Content and Presentation of Evidence Elements:

ADO_DEL.2.1C
The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

ADO_DEL.2.2C
The delivery documentation shall describe how the various procedures and technical measures provide for the detection of modifications, or any discrepancy between the developer's master copy and the version received at the user site.

ADO_DEL.2.3C
The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

Evaluator Action Elements:

ADO_DEL.2.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2.2 Installation, generation, and start-up procedures (ADO_IGS.1)

Developer Action Elements:

ADO_IGS.1.1D
The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

Content and Presentation of Evidence Elements:

ADO_IGS.1.1C

The documentation shall describe the steps necessary for secure installation, generation, and start-up of the TOE.

Evaluator Action Elements:

ADO_IGS.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_IGS.1.2E
The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

5.2.3 Development (ADV)

5.2.3.1 Fully defined external interfaces (ADV_FSP.2)

Developer Action Elements:

ADV_FSP.2.1D
The developer shall provide a functional specification.

Content and Presentation of Evidence Elements:

ADV_FSP.2.1C
The functional specification shall describe the TSF and its external interfaces using an informal style.

ADV_FSP.2.2C
The functional specification shall be internally consistent.

ADV_FSP.2.3C
The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

ADV_FSP.2.4C
The functional specification shall completely represent the TSF.

ADV_FSP.2.5C
The functional specification shall include rationale that the TSF is completely represented.

Evaluator Action Elements:

ADV_FSP.2.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.2.2E
The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.2 Security enforcing high-level design (ADV_HLD.2)

Developer Action Elements:

ADV_HLD.2.1D
The developer shall provide the high-level design of the TSF.

Content and Presentation of Evidence Elements:

ADV_HLD.2.1C
The presentation of the high-level design shall be informal.

ADV_HLD.2.2C
The high-level design shall be internally consistent.

ADV_HLD.2.3C
The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.2.4C
The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.2.5C
The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.2.6C
The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.2.7C
The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.2.8C
The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_HLD.2.9C
The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

Evaluator Action Elements:

ADV_HLD.2.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.2.2E
The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.3 Subset of the implementation of the TSF (ADV_IMP.1)

Developer Action Elements:

ADV_IMP.1.1D
The developer shall provide the implementation representation for a selected subset of the TSF **to include at least the following subsets:**

- **(a) the subset of the physical structure of the TOE related to**
  1. **structure size, organization, and layout**
  2. **interconnects and data bus layout**
  3. **fuse locations**
  4. **physical structure including shielding layers and packaging**
  5. **EEPROM manipulation**
  6. **RAM access**
- **(b) the subset of the logical structure of the TOE related to**
  1. **command range and validity checking**
  2. **interrupts and reset function**
  3. **secure data checking and manipulation**
  4. **availability of commands outside of defined application**
  5. **transfer of information between applications or functions**
- **(c) the subset of the structure of the TOE related to unalterability of**
  1. **unique serial number and other life-cycle identifiers**
  2. **blocking or elimination of debugging functions**

Content and Presentation of Evidence Elements:

ADV_IMP.1.1C
The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.1.2C
The implementation representation shall be internally consistent.

Evaluator Action Elements:

ADV_IMP.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.1.2E
The evaluator shall determine that the least abstract TSF representation provided is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.4 Descriptive low-level design (ADV_LLD.1)

Developer Action Elements:

ADV_LLD.1.1D
The developer shall provide the low-level design of the TSF.

Content and Presentation of Evidence Elements:

ADV_LLD.1.1C
The presentation of the low-level design shall be informal.

ADV_LLD.1.2C
The low-level design shall be internally consistent.

ADV_LLD.1.3C
The low-level design shall describe the TSF in terms of modules.

ADV_LLD.1.4C
The low-level design shall describe the purpose of each module.

ADV_LLD.1.5C
The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.1.6C
The low-level design shall describe how each TSP-enforcing function is provided.

ADV_LLD.1.7C
The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.1.8C
The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.1.9C
The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_LLD.1.10C
The low-level design shall describe the separation of the TOE into TSP-enforcing and other modules.

Evaluator Action Elements:

ADV_LLD.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.1.2E
The evaluator shall determine that the low-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.5 Informal correspondence demonstration (ADV_RCR.1)

Developer Action Elements:

ADV_RCR.1.1D
The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

Content and Presentation of Evidence Elements:

ADV_RCR.1.1C
For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

Evaluator Action Elements:

ADV_RCR.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.6 Informal TOE security policy model (ADV_SPM.1)

Developer Action Elements:

ADV_SPM.1.1D
The developer shall provide a TSP model.

ADV_SPM.1.2D
The developer shall demonstrate correspondence between the functional specification and the TSP model.

Content and Presentation of Evidence Elements:

ADV_SPM.1.1C
The TSP model shall be informal.

ADV_SPM.1.2C
The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.1.3C
The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.1.4C
The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

Evaluator Action Elements:

ADV_SPM.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 Guidance documents (AGD)

5.2.4.1 Administrator guidance (AGD_ADM.1)

Developer Action Elements:

AGD_ADM.1.1D
The developer shall provide administrator guidance addressed to system administrative personnel.
Content and Presentation of Evidence Elements:

AGD_ADM.1.1C
The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

AGD_ADM.1.2C
The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.3C
The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.4C
The administrator guidance shall describe all assumptions regarding user behavior that are relevant to secure operation of the TOE.

AGD_ADM.1.5C
The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

AGD_ADM.1.6C
The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C
The administrator guidance shall be consistent with all other documentation supplied for evaluation.

AGD_ADM.1.8C
The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator.

Evaluator Action Elements:

AGD_ADM.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4.2 User guidance (AGD_USR.1)

Developer Action Elements:

AGD_USR.1.1D
The developer shall provide user guidance.

Content and Presentation of Evidence Elements:

AGD_USR.1.1C
The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

AGD_USR.1.2C
The user guidance shall describe the use of user-accessible security functions provided by the TOE.

AGD_USR.1.3C
The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C
The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behavior found in the statement of TOE security environment.

AGD_USR.1.5C
The user guidance shall be consistent with all other documentation supplied for evaluation.

AGD_USR.1.6C
The user guidance shall describe all security requirements for the IT environment that are relevant to the user.

Evaluator Action Elements:

AGD_USR.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.5 Life cycle support (ALC)

5.2.5.1 Identification of security measures (ALC_DVS.1)

Developer Action Elements:

ALC_DVS.1.1D
The developer shall produce development security documentation.

Content and Presentation of Evidence Elements:

ALC_DVS.1.1C
The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

ALC_DVS.1.2C
The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

Evaluator Action Elements:

ALC_DVS.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.1.2E
The evaluator shall confirm that the security measures are being applied.

5.2.5.2 Developer defined life-cycle model (ALC_LCD.1)

Developer Action Elements:

ALC_LCD.1.1D
The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE. **The developer's software process that produces the TOE operating system, executable files (EXF), or interface software for the TOE shall be assessed against the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) through a Software Capability Evaluation conducted by an SEI authorized lead assessor. The developer's software process must meet CMM Level 1.**

Application note: The refinement to this requirement is a "placeholder" for stronger CMM requirements in future revisions of this protection profile. Over the next few years, the DoD

intends to require that the software process for the developer of the DoD PKI and KMI Token meet CMM Level 3.

This requirement was created to address the threat of an attacker exploiting a development flaw in code used by the TOE. The Capability Maturity Model (CMM) for Software describes the key elements of an effective software process for a developer's organization. Following processes outlined by the CMM equips a developer to develop code that is less likely to contain flaws that can be exploited. This requirement provides assurance that the developer's organization has started preparing to meet the eventual DoD requirement of CMM Level 3. CMM Level 3 will provide assurance that code developed for the TOE does what it is designed to do.

ALC_LCD.1.2D
The developer shall provide life-cycle definition documentation.

Content and Presentation of Evidence Elements:

ALC_LCD.1.1C
The life-cycle definition documentation shall describe the model used to develop and maintain the TOE. **The developer's standard software process at CMM Level 1 for developing and maintaining TOE software shall be documented. The results of the developer's Software Capability Evaluation shall be provided. Results include the assessed CMM level, and the specific Key Processes Areas (KPAs) that were satisfied within each CMM level (whether or not all KPAs for the level were met). Additionally, the "Findings Report" that documents the current state of the developer's software development process shall be provided.**

ALC_LCD.1.2C
The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

Evaluator Action Elements:

ALC_LCD.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.5.3 Compliance with implementation standards – all parts (ALC_TAT.3)

Developer Action Elements:

ALC_TAT.3.1D
The developer shall identify the development tools being used for the TOE. **Additionally, the token platform developer must provide guidance to develop secure applications on the developer's operating system platform.**

ALC_TAT.3.2D
The developer shall document the selected implementation-dependent options of the development tools.

ALC_TAT.3.3D
The developer shall describe the implementation standards for all parts of the TOE.

Content and Presentation of Evidence Elements:

ALC_TAT.3.1C
All development tools used for implementation shall be well-defined. **The developer shall provide at least the following development evidence:**

(a) Design Information

1. **IC specification and technology**
2. **IC design**
3. **IC hardware security mechanisms**
4. **IC software security mechanisms**
5. **photomask**
6. **development tools**
7. **initialization procedures**
8. **access control mechanisms**
9. **authentication systems**
10. **data protection systems**
11. **memory partitioning**
12. **cryptographic programs**

(b) Data

1. **initialization data**
2. **personalization data**
3. **passwords**
4. **cryptographic keys**

(c) Test Information

1. **test tools**
2. **test procedures**
3. **test programs**
4. **test results**

(d) Physical Instantiations

1. **silicon samples**
2. **bond-out chips**
3. **pre-initialized cards**
4. **pre-personalized cards**
5. **personalized but unissued cards**

(e) Guidance documentation for developing secure applications on the

developer's operating system platform.

ALC_TAT.3.2C
The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.3.3C
The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

Evaluator Action Elements:

ALC_TAT.3.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_TAT.3.2E
The evaluator shall confirm that the implementation standards have been applied.

5.2.6 Tests (ATE)

5.2.6.1 Analysis of coverage (ATE_COV.2)

Developer Action Elements:

ATE_COV.2.1D
The developer shall provide an analysis of the test coverage.

Content and Presentation of Evidence Elements:

ATE_COV.2.1C
The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

ATE_COV.2.2C
The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

Evaluator Action Elements:

ATE_COV.2.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.2 Testing: high-level design (ATE_DPT.1)

Developer Action Elements:

ATE_DPT.1.1D
The developer shall provide the analysis of the depth of testing.

Content and Presentation of Evidence Elements:

ATE_DPT.1.1C
The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design.

Evaluator Action Elements:

ATE_DPT.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.3 Functional testing (ATE_FUN.1)

Developer Action Elements:

ATE_FUN.1.1D
The developer shall test the TSF and document the results.

ATE_FUN.1.2D
The developer shall provide test documentation.

Content and Presentation of Evidence Elements:

ATE_FUN.1.1C
The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

ATE_FUN.1.2C
The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C
The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on

the results of other tests.

ATE_FUN.1.4C
The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.1.5C
The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

Evaluator Action Elements:

ATE_FUN.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.4 Independent testing—sample (ATE_IND.2)

Developer Action Elements:

ATE_IND.2.1D
The developer shall provide the TOE for testing.

Content and Presentation of Evidence Elements:

ATE_IND.2.1C
The TOE shall be suitable for testing.

ATE_IND.2.2C
The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

Evaluator Action Elements:

ATE_IND.2.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E
The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

ATE_IND.2.3E
The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

5.2.7 Vulnerability assessment (AVA)

5.2.7.1 Validation of analysis (AVA_MSU.2)

Developer Action Elements:

AVA_MSU.2.1D
The developer shall provide guidance documentation.

AVA_MSU.2.2D
The developer shall document an analysis of the guidance documentation.

Content and Presentation of Evidence Elements:

AVA_MSU.2.1C
The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences, and implications for maintaining secure operation.

AVA_MSU.2.2C
The guidance documentation shall be complete, clear, consistent and reasonable.

AVA_MSU.2.3C
The guidance documentation shall list all assumptions about the intended environment.

AVA_MSU.2.4C
The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

AVA_MSU.2.5C
The analysis documentation shall demonstrate that the guidance documentation is complete.

Evaluator Action Elements:

AVA_MSU.2.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.2.2E
The evaluator shall repeat all configuration and installation procedures, and other procedures selectively, to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.2.3E

The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

AVA_MSU.2.4E
The evaluator shall confirm that the analysis documentation shows that guidance is provided for the secure operation in all modes of operation of the TOE.

5.2.7.2 Strength of TOE security function evaluation (AVA_SOF.1)

Developer Action Elements:

AVA_SOF.1.1D
The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.

Content and Presentation of Evidence Elements:

AVA_SOF.1.1C
For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST.

AVA_SOF.1.2C
For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.

Evaluator Action Elements:

AVA_SOF.1.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E
The evaluator shall confirm that the strength claims are correct.

5.2.7.3 Moderately resistant (AVA_VLA.3)

Developer Action Elements:

AVA_VLA.3.1D

The developer shall perform and document an analysis of the TOE deliverables searching for ways in which a user can violate the TSP. **The analysis shall take into account the following generic vulnerabilities:**

(a) The TOE may be subject to deconstruction to reveal internal circuits and structures.
(b) The TOE may be subject to tampering with the structure and content of internal memories, data transport mechanisms, security functions, and test methods.
(c) The TOE may be subject to analysis of information that is internal to the device through monitoring of connections between elements of the circuits and structures.
(d) The TOE may be subject to use of logical commands to produce responses that lead to security vulnerabilities.
(e) The TOE may be subject to manipulations outside defined operational boundaries that lead to security vulnerabilities.
(f) **The TOE may be subject to analysis of information that is available external to the device, through monitoring emanations or any of the connections to the device including power, ground, clock, I/O, and reset.**
(g) **The TOE may be subject to vulnerabilities that have been identified in preceding generations of the same, or a similar, TOE.**

AVA_VLA.3.2D
The developer shall document the disposition of identified vulnerabilities.

Content and Presentation of Evidence Elements:

AVA_VLA.3.1C
The documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.3.2C
The documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

AVA_VLA.3.3C
The evidence shall show that the search for vulnerabilities is systematic.

Evaluator Action Elements:

AVA_VLA.3.1E
The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.3.2E
The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

AVA_VLA.3.3E
The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.3.4E
The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.

AVA_VLA.3.5E
The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a moderate attack potential.

6 Rationale

6.1 TOE Description Rationale

The target of evaluation, a DoD PKI and KMI Token, has been defined. This TOE has a unique set of threats relating to its character as a small, self-contained microprocessor that is manufactured in large quantities and may ultimately be issued to untrusted token holders for their long-term retention. The description of the TOE supports the statement of threats, policies, and assumptions discussed earlier in this PP.

6.2 Security Objectives Rationale

This section demonstrates, through coverage rationale, that the stated security objectives counter all identified threats, policies, or assumptions.

Table 6-1 and Table 6-2 map the security objectives to the security environment defined by the threats, policies, and assumptions. The mappings illustrate that each security objective covers at least one threat, policy, or assumption, and that each threat, policy, and assumption is covered by at least one security objective.

Table 6-1 Mapping the TOE Security Environment to Security Objectives

Policy/Threat/Assumptions	Security Objectives for the TOE
P.Control_of_Applications	O.Control_of_Applications
P.Protection_Mechanisms	O.DAC, O.Authenticate, O.Key_Encrypt, O.Auth_Protect, O.User_Data_Control, O.Role_Man
P.Key_Length	O.Crypt
T.App_Ftn	O.Mult_App
T.Bad_Load	O.Self_Test, O.Authenticate, O.User_Data_Control
T.Clon	O.Phys_Prot, O.Tamper_Response
T.Component_Fail	O.Fail_Secure
T.Crypt_Attk	O.Crypt
T.E_Manip	O.Phys_Prot, O.Tamper_Response
T.Env_Strs	O.Env_Strs
T.Fail_Secure	O.Log_Prot, O.Env_Strs, O.Init
T.First_Use	O.Set_Up
T.Flt_Ins	O.Input_Probe
T.Forced_State_Change	O.Init
T.Hacker_Comm_Eavesdrop	O.I_Leak, O.Secure_Host_Comms, O.Tamper_Response, O.Data_Exchange_Conf

Policy/Threat/Assumptions	Security Objectives for the TOE
T.I_Leak	O.I_Leak, O.Env_Strs
T.Impers	O.Trial, O.Auth_Protect, O.Authenticate
T.Inv_Inp	O.Init, O.Log_Prot
T.LC_Ftn	O.Life_Cycle
T.Link	O.Unlink
T.Lnk_Att	O.Log_Prot
T.P_Modify	O.Tamper_Response, O.Phys_Prot
T.P_Probe	O.Tamper_Response, O.Key_Encrypt, O.Volatile_Memory, O.Auth_Protect, O.D_Read, O.Phys_Prot
T.Power_Clock	O.Env_Strs, O.Fail_Secure
T.Rep_Atk	O.Env_Strs, O.Log_Prot, O.Trial, O.DAC
T.Res_Con	O.Res_Access
T.Spoof	O.Secure_Host_Comms
T.UA_Use	O.Init, O.Log_Prot, O.Self_Test
Policy/Threat/Assumptions	**Security Objectives for the Environment**
A.Dev_Protect	OE.Con_Prod, OE.Con_Tools, OE.Mask_Prot
A.Key_Gen	OE.Key_Gen
A.Secure_Host_Comms	OE.Sec_Com, OE.Dlv_Proc
T.Bad_Load	OE.Personnel, OE.Sec_Com
T.Clon	OE.Con_Des, OE.Dlv_Aud, OE.Dlv_Proc, OE.Ident, OE.Sample_Acs
T.Developer_Flawed_Code	OE.Con_Cont, OE.Con_Des, OE.Con_Prod, OE.Dlv_Aud, OE.Dlv_Proc, OE.Dlv_Trn, OE.SW_Develop
T.Hacker_Social_Engineer	OE.Personnel, OE.Train
T.Inv_Inp	OE.Train
T.Privilege	OE.Personnel

Table 6-2 Tracing of Security Objectives to the TOE Security Environment

Security Objectives for the TOE	Policy/Threat/Assumptions
O.Auth_Protect	P.Protection_Mechanisms, T.Impers, T.P_Probe
O.Authenticate	P.Protection_Mechanisms, T.Bad_Load, T.Impers
O.Control_of_Applications	P.Control_of_Applications
O.Crypt	T.Crypt_Attk, P.Key_Length
O.DAC	P.Protection_Mechanisms, T.Rep_Atk
O.D_Read	T.P_Probe
O.Data_Exchange_Conf	T.Hacker_Comm_Eavesdrop
O.Env_Strs	T.Env_Strs, T.Fail_Secure, T.I_Leak, T.Power_Clock, T.Rep_Atk
O.Fail_Secure	T.Component_Fail, T.Power_Clock
O.I_Leak	T.Hacker_Comm_Eavesdrop, T.I_Leak
O.Init	T.Fail_Secure, T.Forced_State_Change, T.Inv_Inp, T.UA_Use
O.Input_Probe	T.Flt_Ins
O.Key_Encrypt	P.Protection_Mechanisms, T.P_Probe
O.Life_Cycle	T.LC_Ftn
O.Log_Prot	T.Fail_Secure, T.Inv_Inp, T.Lnk_Att, T.Rep_Atk, T.UA_Use
O.Mult_App	T.App_Ftn
O.Phys_Prot	T.Clon, T.E_Manip, T.P_Probe, T.P_Modify
O.Res_Access	T.Res_Con
O.Role_Man	P.Protection_Mechanisms
O.User_Data_Control	P.Protection_Mechanisms, T.Bad_Load
O.Secure_Host_Comms	T.Hacker_Comm_Eavesdrop, T.Spoof
O.Self_Test	T.Bad_Load, T.UA_Use
O.Set_Up	T.First_Use
O.Tamper_Response	T.Clon, T.E_Manip, T.Hacker_Comm_Eavesdrop, T.P_Modify, T.P_Probe
O.Trial	T.Impers, T.Rep_Atk
O.Unlink	T.Link
O.Volatile_Memory	T.P_Probe

Security Objectives for the Environment	Policy/Threat/Assumptions
OE.Con_Cont	T. Developer_Flawed_Code
OE.Con_Des	T.Clon, T.Developer_Flawed_Code
OE.Con_Prod	A.Dev_Protect, T.Developer_Flawed_Code
OE.Con_Tools	A.Dev_Protect
OE.Dlv_Aud	T.Clon, T.Developer_Flawed_Code
OE.Dlv_Proc	A.Secure_Host_Comms, T.Clon, T.Developer_Flawed_Code
OE.Dlv_Trn	T.Developer_Flawed_Code
OE.Ident	T.Clon
OE.Key_Gen	A.Key_Gen
OE.Mask_Prot	A.Dev_Protect

OE.Personnel	T.Bad_Load, T.Hacker_Social_Engineer, T.Privilege
OE.SW_Develop	T.Developer_Flawed_Code
OE.Sample_Acs	T.Clon
OE.Sec_Com	A.Secure_Host_Comms, T.Bad_Load
OE.Train	T.Hacker_Social_Engineer, T.Inv_Inp

6.2.1 Assumptions

A.Dev_Protect: Protection of TOE by Developer
During the development and manufacturing process, the TOE and associated development tools are assumed to be protected by the developer from any kind of unauthorized use, e.g., tampering or theft.

***Coverage Rationale*:** A.Dev_Protect (Protection of TOE by Developer) establishes that the TOE and its development tools are protected by the developer from unauthorized use during the TOE's development and manufacturing phases. OE.Con_Prod (Control of Product), OE.Con_Tools (Control of Tools) , and OE.Mask_Prot (Photomask Protection) ensure this protection.

A.Key_Gen: Key Exchange Key Generation
Key exchange keys are assumed to be generated off-TOE in a secure manner in accordance with X.509 Certificate Policy.

***Coverage Rationale*:** A.Key_Gen (Key Generation) establishes that key exchange keys were generated in a secure manner before being loaded onto the TOE. This assumption is supported by OE.Key_Gen (Key Exchange Key Generation) that ensures Key Exchange Keys are securely generated in accordance with X.509 Certificate Policy.

A.Secure_Host_Comms: Secure Host Communications
If the host establishes a secure connection between itself and the TOE that conforms to the requirements imposed by the TOE, the host, including code and security data it contains, is assumed to be trusted.

***Coverage Rationale*:** A.Secure_Host_Comms (Secure Host Communications) establishes that the host, its code, and its security data are trusted by the TOE, provided that a secure connection is established between the host and the TOE. OE.Sec_Com (Secure Communication) ensures that only a trusted host is able to establish a secure connection with the TOE. OE.Dlv_Proc (Delivery Procedures) ensures that the code and security data on the host are protected during their delivery to the host.

6.2.2 Policies

P.Control_of_Applications: Control of Applications
1) A DoD PKI and Token must contain an application manager as detailed in Appendix F in Section 2.3 of the *Application Specification for the DoD PKI and KMI Token.*
2) Application signatures must be used to sign approved applications to control the loading of applications onto the DoD PKI and KMI token. Valid applications must be signed by an approved DoD entity as detailed in Appendix F in Section 2.4 of the *Application Specification for the DoD PKI and KMI Token.*
3) Any proposed DoD PKI and KMI Token application signature algorithm and key length must comply with Appendix E of this Protection Profile.
4) The token platform developer must provide guidance to develop secure applications on the developer's operating system platform.

***Coverage Rationale*:** P.Control_of_Applications (Control of Applications) addresses the need to control the development, loading, and use of applications for the TOE. O.Control_of_Applications ensures that the loading and use of applications is controlled by an application manager and application signatures. O.Control_of_Applications also ensures that proper measures will be taken to control the development of applications for the TOE.

P.Key_Length: Cryptographic Key Length
X.509 Certificate Policy for the U. S. Department of Defense. Digital Signature Standard keys shall use at least 160 bit private key and at least 1024 bit prime modulus. Minimum public key size shall be 1024 bits for Key Exchange Algorithm (KEA). Minimum public key size shall be 2048 bits for RSA. For Class 4, Elliptic Curve Digital Signature Algorithm key prime field (//p//) shall be not less than 384 bits.

***Coverage Rationale*:** P.Key_Length (Cryptographic Key Length) addresses the required key length for digital signatures and public key cryptography. O.Crypt ensures that cryptographic operations are performed in accordance with established policies for SBU data.

P.Protection_Mechanisms: Application of Protection Mechanisms
Information Assurance Guidance and Policy Memorandum 6-8510. Protection mechanisms shall be applied such that the TOE maintains the appropriate level of confidentiality, integrity, authentication, and nonrepudiation based on mission criticality, sensitivity of information handled by the system, and need-to-know.

***Coverage Rationale*:** P.Protection_Mechanisms (Application of Protection Mechanisms) addresses unauthorized access to DoD information or resources by legitimate users and applications. O.Authenticate and O.Auth_Protect guarantee that the subject making an access to cryptographic or other DoD data is who he or she claims. O.Role_Man requires the TOE to manage identified roles in a secure manner. O.User_Data_Control (User Data Control) restricts the setting or modification of sensitive information. By O.DAC, each user has the means of

limiting access for its objects and resources to authorized users. O.Key_Encrypt provides further protection by preventing the reading of keys in nonvolatile memory.

6.2.3 Threats

T.App_Ftn: Use of Unallowed Application Functions
An attacker may exploit interactions between applications to expose sensitive TOE or user data.

***Coverage Rationale*:** T.App_Ftn (Use of Unallowed Application Functions) deals with the exploitation of inappropriate interaction of functions between applications. O.Mult_App (Multiple Applications) ensures that such interactions do not compromise security through unauthorized availability of information between applications.

T.Bad_Load: Load Bad Software or Security Data
An attacker, an SSO, or the user may load improper software (operating system, executable files) or security data (authentication information, keys, access control information) onto the TOE that could modify or expose software (e.g., security functions) or data on the TOE.

***Coverage Rationale*:** T.Bad_Load (Load Bad Software or Security Data) addresses the risk that improper software or security data could cause software or data on the TOE to be improperly modified or exposed. O.Authenticate (Authentication of Users and SSOs) and O.User_Data_Control (User Data Control) ensure that security data with which the TOE is initialized is supplied by an authenticated SSO, and OE.Personnel ensures that the SSOs are carefully selected and trained for reliability. OE.Sec_Com (Secure Communication) guarantees that the host from which the software or security data is downloaded is trusted and, therefore, the software or data are appropriate for the TOE. Additionally, O.Self_Test (Self-Test) checks that the execution of bad code does not modify other code.

T.Clon: Cloning
An attacker may clone part or all of a functional TOE to develop further attacks.

***Coverage Rationale*:** T.Clon (Cloning) represents the threat that an attacker may manufacture all or a usable portion of the IC that is then used for fraudulent purposes. This threat is countered by O.Phys_Prot (Physical Protection) through a construction that makes it difficult to understand any information derived from physical attacks on the TOE. By O.Tamper_Response (Respond to Tamper), the TOE automatically responds to physical attempts to extract information, thereby preventing an attacker from obtaining the design of the IC. This is also supported by OE.Con_Des (Control of Design), which ensures the protection of design and fabrication information supporting the construction of the IC. The objectives OE.Dlv_Proc (Delivery Procedures) and OE.Dlv_Aud (Delivery Audit) also provide support through ensuring that information, designs, and product (in various states of completion) are not available to attackers.

OE.Sampl_Acs (Sample Access) limits access to the samples used to run tests to authorized personnel, preventing others from gaining access to this privileged information. OE.Ident (TOE Identification) counters cloning attacks to ensure TOE identification information is recorded and preserved on the TOE prior to being issued to the user.

T.Component_Fail: Failure of a Critical System Component
An attacker exploits a failure of one or more system components resulting in the loss of system-critical functionality.

***Coverage Rationale*:** T.Component_Fail (Failure of a Critical System Component) addresses the failure of one or more system components, resulting in the failure of security-critical functionality. O.Fail_Secure (Preservation of secure state) ensures that if a security component does fail, the system will remain secure.

T.Crypt_Attk: Cryptographic Attack
An attacker may defeat security functions through a cryptographic attack against the algorithm, through cryptanalysis on encrypted data, or through a brute-force attack.

***Coverage Rationale*:** T.Crypt_Attk (Cryptographic Attack) addresses direct attacks on the cryptographic mechanisms employed in the TOE. This threat is countered by O.Crypt (Cryptography), which ensures that the available cryptographic functions are of appropriate strength for the sensitivity of the data processed by the TOE.

T.Developer_Flawed_Code: Software containing security related flaws
An attacker exploits code delivered by a system or application developer that does not perform according to specifications, contains security flaws, or is not appropriate for operational use.

***Coverage Rationale*:** T.Developer_Flawed_Code (Software containing security-related flaws) addresses flaws in the system or developer's application code. OE.SW_Develop (Software Developer Process) ensures that the developer's software development process for the code is reliable. The design information, by OE.Con_Des (Control of Design), and the code, by OE.Con_Prod (Control of Product) and OE.Con_Cont (Code Configuration Control), are protected from unauthorized modification. OE.Dlv_Proc (Delivery Procedures), OE.Dlv_Aud (Delivery Audit), and OE.Dlv_Trn (Delivery Training) protect the software while it is transferred from one facility or supplier to another.

T.E_Manip: Electrical Manipulation of the IC
An attacker may use electrical probing and manipulation of the TOE to modify security-critical data so that the TOE can be used fraudulently.

***Coverage Rationale*:** T.E_Manip (Electrical Manipulation of the IC) addresses attempts in which the TOE is modified so that it can be directly fraudulently used. This differs from T.P_Modify in that the goal of the former threat is to derive information and not to reuse the TOE. This threat is countered directly by O.Phys_Prot (Physical Protection), which ensures that the TOE is resistant to physical attack. The threat is additionally countered by O.Tamper_Response, which provides capabilities to automatically respond to physical attacks against specified parts of the TOE. The automatic response may take varying forms but generally involves direct actions (e.g., shutting a system down) rather than notification actions. The combination of O.Phys_Prot and O.Tamper_Response provides a two-phased approach to countering E.Manip by protecting the TOE from manipulation and invoking a TOE response to manipulation when detected.

T.Env_Strs: Environmental Stress
An attacker may exploit failures in the TOE induced by environmental stress.

***Coverage Rationale*:** T.Env_Strs (Environmental Stress) deals with the imposition of environmental extremes on the TOE with the intent to cause a direct or indirect failure in the

security mechanisms. This threat is countered by O.Env_Strs (Environmental Stress), which ensures that the TOE performs in an acceptable fashion (i.e., does not reveal secure information) when exposed to out-of-design-specification conditions.

T.Fail_Secure: Failing in a nonsecure state
An attacker may cause failure of the TOE security functions by exposing the TOE to conditions outside of its normal operating range, causing the TOE to enter a nonsecure state.

***Coverage Rationale*:** T.Fail_Secure addresses forcing the TOE into a nonsecure state by causing the failure of TOE security functions by exposing the TOE to conditions outside of its normal operating range. This threat is countered by O.Log_Prot, which ensures that the TOE is constructed to be resistant to logical manipulation. This threat is also countered by O.Env_Strs, which ensures that the TOE performs in a secure fashion (i.e., does not reveal security information) when exposed to out-of-design-specification conditions. Additionally, this threat is addressed by O.Init, which requires that the TOE enter a defined initial state upon experiencing a reset condition.

T.First_Use: Fraud on First Use
An attacker may gain access to TOE information by unauthorized use of a new, previously unissued TOE.

***Coverage Rationale*:** T.First_Use (Fraud on First Use) deals with fraud perpetrated through the use of TOEs that have not been officially issued. This threat is countered directly by O.Set_Up (Set-up Sequence), which ensures that a defined and controlled sequence of events is completed before the TOE is enabled for use.

T.Flt_Ins: Insertion of Faults
An attacker may determine security-critical information through observation of the results of repetitive insertion of selected data.

***Coverage Rationale*:** T.Flt_Ins addresses the situation when the TOE is actively being probed through the deliberate insertion of selected inputs with the intent of observing the outputs. This is normally performed over multiple repetitions with small changes in the selected inputs. This is countered through O.Input_Probe (Probing by Selected Input), which ensures that such attacks are resisted.

T.Forced_State_Change: Forced State Change
An attacker may force the TOE into a nonsecure state through inappropriate termination of selected operations.

***Coverage Rationale*:** T.Forced_State_Change (Forced State Change) addresses the situations in which the TOE is reset during operation. This may occur at any time including during a reset

operation itself. This threat is countered directly by O.Init (Initialization), which ensures that the TOE always enters its defined initial state upon reset.

T.Hacker_Comm_Eavesdrop: Hacker Eavesdrops on User Data Communications
Hacker obtains user data by eavesdropping on communications lines.

***Coverage Rationale*:** T.Hacker_Comm_Eavesdrop addresses the tapping of cables between the host and the TOE (reader) to extract sensitive data. This threat is addressed by O.Tamper_Response, which responds to physical tampering against system devices and components. O.I_Leak addresses this threat by providing the means to control and limit the leakage of information in the TOE so that no useful information is revealed over the power, ground, clock, reset, or I/O lines. O.Data_Exchange_Conf helps in countering the tapping of communication lines by protecting user data confidentiality when exchanging data. O.Secure_Host_Comms counters this threat by establishing secure communications with the host before cryptographic or other DoD data are passed between the TOE and the host.

T.Hacker_Social_Engineer: Social Engineering
A hacker uses social engineering techniques to gain information about system entry, system use, system design, or system operation.

***Coverage Rationale*:** T.Hacker_Social_Engineer addresses the use of social engineering techniques to gain information about the system. This threat is countered by OE.Train and OE.Personnel, which requires TOE users and administrators to be trained on the proper usage of the TOE and its security procedures.

T.I_Leak: Information Leak
An attacker may exploit information that is leaked from the TOE during normal usage.

***Coverage Rationale*:** T.I_Leak (Information Leakage) deals with the exploitation of information inadvertently available from emanations or variations in power consumption or other operating parameters as a function of the operation being performed, i.e., Simple Power Analysis and Differential Power Analysis. This threat is countered by O.I_Leak (Information Leakage), which provides the means to control and limit leakage of information in the TOE and ensures that such information is not exposed. O.Env_Strs (Environmental Stress) ensures that the TOE performs in an acceptable fashion (i.e., does not reveal secure information) when exposed to out-of-design-specification conditions.

T.Impers: Impersonation
An attacker may gain access to TOE information by impersonating an authorized user of the TOE.

***Coverage Rationale*:** T.Impers (Unauthorized use of TOE) addresses the use of the TOE by an attacker impersonating an authorized user or SSO. This threat is countered directly by O.Trial (Trial and Error Resistance) and O.Authenticate (Authentication of User and SSOs), which require that the user be authenticated with a mechanism resistant to spoofing by trial and error.

T.Inv_Inp: Invalid Input
An attacker or authorized user of the TOE may compromise the security features of the TOE through the introduction of invalid inputs.

***Coverage Rationale*:** T.Inv_Inp (Invalid Input) addresses the introduction of input that does not conform to the required style, content, or format. This input may have the look of accidental or erroneous entries (and that may be, in fact, the source of the data), but the result may be the misperformance of the TOE such that security is compromised. Attackers may use nonconforming data, existing but inappropriate commands, or well-formatted commands with data requests that refer to locations that are outside of range or not to be used in that operation. This threat is countered directly by O.Log_Prot (Logical Protection), which ensures that the TOE is constructed such that it responds in a secure manner to all probing represented by data, commands, or other input. This threat is also countered by O.Init (Initialization), which provides additional protection against this threat by ensuring the TOE starts in a defined and controlled state after a restart condition. This threat is also countered by OE.Train (User Train), which requires TOE users to be trained on the proper usage of the TOE and TOE-related security procedures.

T.LC_Ftn: Use of Unallowed Life-Cycle Functions
An attacker may exploit interactions between life-cycle functions to expose sensitive TOE or user data.

***Coverage Rationale*:** T.LC_Ftn (Use of Unallowed Life-Cycle Functions) deals with the exploitation of inappropriate interactions of functions between various life-cycle operations. O.Life_Cycle (Life-Cycle Functions) ensures that such interactions do not compromise security through unauthorized availability of information between elements used in different parts of the life cycle.

T.Link: Linkage of Multiple Observations
An attacker may observe multiple uses of resources or services and, by linking these observations, deduce information that would reveal critical security information.

***Coverage Rationale*:** T.Link (Linkage of Multiple Observations) addresses the observation and linking of a variety of operations, leading to the attacker being able to deduce useful information. This threat is differentiated from T.Alt_Ftn and T.Gen_Attk, since it purely entails observation of normally visible operations and not the manipulation entailed in using operations across defined boundaries. This threat is countered by O.Unlink (Linkage), which ensures that

information exposed in a combination of operations is of no use to an attacker in understanding and attacking the TOE.

T.Lnk_Att: Linked Attacks
An attacker may perform successive attacks with the result that the TOE becomes unstable or some aspect of the security functionality is degraded. A following attack may then be successfully executed.

***Coverage Rationale*:** T.Lnk_Att (Linked Attacks) deals with multiple attacks synergistically causing a degradation and failure of TOE security. O.Log_Prot ensures that the TOE remains secure in the event of logical probing attacks.

T.P_Modify: Physical Modification of the IC
An attacker may physically modify the TOE in order to reveal design- or security-related information.

***Coverage Rationale*:** T.P_Modify (Physical Modification) deals with attempts to physically modify the TOE such that information relating to the secure operation of the TOE is revealed. This is an extension of T.P_Probe, since it may involve physical changes to the IC such as rerouting connections or repairing fuses. This threat is countered directly by O.Phys_Prot (Physical Protection), which ensures that the TOE is resistant to physical attack or is able to create difficulties in understanding the information derived from such an attack. Additionally, O.Tamper_Response (Tamper Response) provides capabilities to automatically respond to physical attacks against specified parts of the TOE, thereby resisting such attacks.

T.P_Probe: Physical Probing of the IC
An attacker may perform physical probing of the TOE to reveal design information and operational contents.

***Coverage Rationale*:** T.P_Probe (Physical Probing) deals with direct probing of the TOE using IC failure analysis and IC reverse engineering efforts to reveal critical hardware and software design information. This threat is countered directly by O.Phys_Prot (Physical Protection), which ensures that the TOE is resistant to physical attack or is able to create difficulties in understanding the information derived from such an attack. Additionally, O.Tamper_Response (Tamper Response) provides automatic response to physical attacks against specified parts of the TOE deemed critical, thereby resisting such attacks. This threat is also partially countered by O.D_Read (Data Read Format), which ensures that data available on data buses inside the TOE provide no information beyond that which would be available through statically reading the memory, that is, information is transferred in the same format in which it is stored. If a key is stored in nonvolatile memory, it is encrypted by O.Key_Encrypt. Thus, plain text keys are only stored in volatile memory. By O.Volatile_Mem, these keys are destroyed when the TOE is removed from a CAD. Likewise, O.Auth_Protect ensures that authentication data cannot be modified by physical probing.

T.Privilege: Abuse by Privileged Users
A careless, willfully negligent, or hostile administrator or other privileged user may compromise the TOE assets through execution of actions that expose, change, or destroy the security functions or the protected/security-critical data.

***Coverage Rationale*:** The threat of abuse by a privileged user is completely addressed by the OE.Personnel objective. OE.Personnel requires personnel working as administrators or in other privileged positions to be selected and trained for reliability.

T.Power_Clock: Power and Clock
An attacker may interrupt, reset, or alter TOE power or clock to disrupt security-critical functions.

***Coverage Rationale*:** T.Power_Clock (Power and Clock) deals with the interruption, reset, or alteration of TOE power or clock. This threat is countered by O.Env_Strs (Environmental Stress), which ensures that the TOE performs in an acceptable fashion (i.e., does not reveal secure information) when exposed to out-of-design-specification conditions. This threat is also addressed by O.Fail_Secure, which preserves the secure state of the system in the event of reset or interruption of power or clock.

T.Rep_Atk: Repetitive Attack
An attacker may utilize repetitive undetected attempts at penetration to expose memory contents or to change security-critical elements in the TOE.

***Coverage Rationale*:** T.Rep_Atk addresses repeated attempts at penetration aimed at exposing memory contents or to change security-critical elements in the TOE. This threat is addressed by several objectives. O.Trial protects the TOE against spoofing by trial and error. O.DAC addresses the access control rules for accessing the TOE. This threat is also countered by O.Log_Prot, which ensures that the TOE is constructed such that it responds in a secure manner to all probing represented by data, commands, or other input not fully conforming to the anticipated style and content. Finally, O.Env_Strs works to prevent disclosure of security-related information in the presence of environmental stress.

T.Res_Con: Resource Contention
A user or attacker may willfully, or through negligence, monopolize resources of the TOE, denying service to another user.

***Coverage Rationale*:** T.Res_Con (Resource Contention) addresses the utilization of an excessive amount of memory, program space, or other resource by a negligent user or an attacker, precluding further normal use of the TOE. This threat is countered by O.Res_Access

(Resource Access), which ensures that limits on resource allocations are established to preclude this denial of service.

T.Spoof: Spoofing Legitimate System Services
An attacker tricks users into interacting with spurious system services, e.g., an unauthorized (bogus) terminal, that request sensitive information from the TOE.

***Coverage Rationale*:** This threat is countered by ensuring that only trusted hosts may communicate with the TOE. O.Secure_Host_Communications requires that DoD and cryptographic data be transmitted only over a secure channel. By A.Secure_Host_Comms, only trusted hosts can establish such a path.

T.UA_Use: Unauthorized Program Use
An attacker may utilize unauthorized programs to penetrate or modify the security functions of the TOE.

***Coverage Rationale*:** T.UA_Use (Unauthorized Program Use) addresses the situations in which legitimate programs may exist in the TOE that are not to be used in the application then being performed. This threat is countered directly by O.Log_Prot (Logical Protection), which ensures the TOE is constructed such that it responds in a secure manner to all probing represented by data, commands, or other input that is not fully conforming to the anticipated style and content. O.Init (Initialization) provides additional protection against this threat by ensuring the TOE starts in a defined and controlled state after a restart condition. This threat is also addressed by O.Self_Test (Self-Test), which track and detect the use of legitimate operations used at times that are not allowed.

6.3 Security Requirements Rationale

This section demonstrates, through coverage rationale, that the stated security requirements support all the objectives in the PP.

Table 6-3 and 6-4 map this PP's security objectives to the security requirements that support them. The mappings illustrate that each security objective is supported by at least one requirement, and that each requirement supports at least one objective.

Table 6-3 Requirements to Security Objectives Mapping

Objectives	Requirements
O.Auth_Protect	FPT_PHP.3
O.Authenticate	FIA_AFL.1, FIA_SOS.1, FIA_UAU.1, FIA_UAU.6, FIA_UAU.7
O.Control_of_Applications	FDP_ITC.1, ALC_TAT.3
O.Crypt	FCS_CKM.1, FCS_CKM.2, FCS_CKM.4, FCS_COP.1

Objectives	Requirements
O.DAC	FDP_ACC.1, FDP_ACF.1, FIA_ATD.1, FIA_UAU.1, FIA_UAU.6, FMT_MOF.1, FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, FMT_MTD.1, FMT_MTD.2, FMT_MTD.3,FMT_REV.1, FDP_ITC.1, FIA_UID.2, FDP_ETC.1
O.D_Read	FDP_ITT.1, FPT_ITT.1
O.Data_Exchange_Conf	FDP_ITC.1, FDP_ETC.1
O.Env_Strs	FPT_FLS.1, FPT_PHP.3, AVA_VLA.3
O.Fail_Secure	FPT_PHP.3, FPT_FLS.1, ADV_SPM.1, AVA_VLA.3
O.I_Leak	FPT_ITT.1, FDP_ITT.1, FDP_IFF.3
O.Init	FIA_UID.2, FPT_RCV.4
O.Input_Probe	FDP_DAU.1, FPT_PHP.3
O.Key_Encrypt	FCS_CKM.3
O.Life_Cycle	FDP_ACC.1, FDP_ACF.1
O.Log_Prot	FPT_PHP.3, ADV_IMP.1, FPT_FLS.1, FPT_PHP.1, FPT_RVM.1, FPT_SEP.1, AVA_VLA.3
O.Mult_App	FDP_IFF.1, FDP_ACC.1, FDP_IFC.1
O.Phys_Prot	FDP_DAU.1, FPT_PHP.1, FPT_PHP.3
O.Res_Access	FRU_RSA.1
O.Role_Man	FMT_SMR.2, FMT_SMR.3
O.User_Data_Control	FDP_ACC.1, FDP_ACF.1
O.Secure_Host_Comms	FTP_ITC.1, FCS_CKM.1, FPT_ITI.1
O.Self_Test	FPT_TST.1, FPT_AMT.1
O.Set_Up	FDP_ACC.1, FDP_ACF.1
O.Tamper_Response	FDP_RIP.1, FPT_PHP.3
O.Trial	FIA_AFL.1, FIA_SOS.1
O.Unlink	FDP_ETC.1, FDP_IFC.1, FDP_IFF.1, FDP_ITT.1
O.Volatile_Memory	FPT_FLS.1
OE.Con_Cont	ACM_AUT.1, ACM_CAP.4, ACM_SCP.2
OE.Con_Des	ACM_SCP.2, ADO_DEL.2, ADV_LLD.1, ADV_HLD.2, ALC_DVS.1
OE.Con_Prod	ADO_DEL.2, ALC_DVS.1
OE.Con_Tools	ALC_DVS.1, ALC_TAT.3
OE.Dlv_Aud	ATE_DPT.1
OE.Dlv_Proc	ADO_DEL.2
OE.Dlv.Trn	ADO_DEL.2
OE.Ident	ACM_CAP.4
OE.Key_Gen	ALC_DVS.1
OE.Mask_Prot	ALC_DVS.1
OE.Personnel	AGD_ADM.1, AGD_USR.1
OE.SW_Develop	ALC_LCD.1
OE.Sample_Acs	ADV_RCR.1, ATE_FUN.1
OE.Sec_Com	ADO_IGS.1
OE.Train	AGD_ADM.1, AGD_USR.1, AVA_MSU.2

Table 6-4 shows the support relationship between the security requirements and objectives.

Table 6-4 Security Objectives to Requirements Mapping

Requirements	Objectives
ACM_AUT.1	Supporting Selection of EAL4, OE.Con_Cont
ACM_CAP.4	Supporting Selection of EAL4,O.Ident, OE.Con_Cont, OE.Ident
ACM_SCP.2	Supporting Selection of EAL4, OE.Con_Cont, OE.Con_Des
ADO_DEL.2	Supporting Selection of EAL4, OE.Con_Des, OE.Con_Prod, OE.Dlv_Proc, OE.Dlv_Trn
ADO_IGS.1	Supporting Selection of EAL4,OE.Sec_Com
ADV_FSP.2	Supporting Selection of EAL4
ADV_HLD.2	Supporting Selection of EAL4, OE.Con_Des
ADV_IMP.1	Supporting Selection of EAL4, O.Ident, O.Log_Prot
ADV_LLD.1	Supporting Selection of EAL4, OE.Con_Des
ADV_RCR.1	Supporting Selection of EAL4, OE.Sample_Acs
ADV_SPM.1	Supporting Selection of EAL4, O.Fail_Secure
AGD_ADM.1	Supporting Selection of EAL4, OE.Personnel, OE.Train
AGD_USR.1	Supporting Selection of EAL4, OE.Personnel, OE.Train
ALC_DVS.1	Supporting Selection of EAL4, OE.Con_Des, OE.Con_Prod, OE.Con_Tools, OE.Key_Gen, OE.Mask_Prot
ALC_LCD.1	Supporting Selection of EAL4, OE.SW_Develop
ALC_TAT.3	O.Control_of_Applications, OE.Con_Tools
ATE_COV.2	Supporting Selection of EAL4
ATE_DPT.1	Supporting Selection of EAL4, OE.Dlv_Aud
ATE_FUN.1	Supporting Selection of EAL4, OE.Sample_Acs
ATE_IND.2	Supporting Selection of EAL4
AVA_MSU.2	Supporting Selection of EAL4, OE.Train
AVA_SOF.1	Supporting Selection of EAL4
AVA_VLA.3	O.Log_Prot, O.Env_Strs, O.Fail_Secure
FCS_CKM.1	O.Crypt, O.Secure_Host_Comms
FCS_CKM.2	O.Crypt
FCS_CKM.3	O.Key_Encrypt
FCS_CKM.4	O.Crypt
FCS_COP.1	O.Crypt
FDP_ACC.1	O.DAC, O.Life_Cycle, O.Set_Up, O.Mult_App, O.User_Data_Control
FDP_ACF.1	O.DAC, O.Life_Cycle, O.Set_Up, O.User_Data_Control
FDP_DAU.1	O.Input_Probe, O.Phys_Prot
FDP_ETC.1	O.DAC, O.Data_Exchange_Conf, O.Unlink
FDP_IFC.1	O.Mult_App, O.Unlink
FDP_IFF.1	O.Mult_App, O.Unlink
FDP_IFF.3	O.I_Leak
FDP_ITC.1	O.DAC, O.Data_Exchange_Conf, P.Control_of_Applications
FDP_ITT.1	O.D_Read, O.I_Leak, O.Unlink
FDP_RIP.1	O. Tamper_Response
FIA_AFL.1	O.Authenticate, O.Trial
FIA_ATD.1	O.DAC
FIA_SOS.1	O.Authenticate, O.Trial
FIA_UAU.1	O.Authenticate, O.DAC

Requirements	Objectives
FIA_UAU.6	O.Authenticate, O.DAC
FIA_UAU.7	O.Authenticate
FIA_UID.2	O.DAC, O.Init
FMT_MOF.1	O.DAC
FMT_MSA.1	O.DAC
FMT_MSA.2	O.DAC
FMT_MSA.3	O.DAC
FMT_MTD.1	O.DAC
FMT_MTD.2	O.DAC
FMT_MTD.3	O.DAC
FMT_REV.1	O.DAC
FMT_SMR.2	O.Role_Man
FMT_SMR.3	O.Role_Man
FPT_AMT.1	O.Self_Test
FPT_FLS.1	O.Env_Strs, O.Fail_Secure, O.Log_Prot, O.Volatile_Memory
FPT_ITI.1	O.Secure_Host_Comms
FPT_ITT.1	O.D_Read, O.I_Leak
FPT_PHP.1	O.Phys_Prot, O.Log_Prot
FPT_PHP.3	O.Env_Strs, O.Fail_Secure, O.Log_Prot, O.Tamper_Response, O.Phys_Prot, O.Input_Probe
FPT_RCV.4	O.Init
FPT_RVM.1	O.Log_Prot
FPT_SEP.1	O.Log_Prot
FPT_TST.1	O.Self_Test
FRU_RSA.1	O.Res_Access
FTP_ITC.1	O.Secure_Host_Comms

6.3.1 Functional Security Requirements Rationale

O.Auth_Protect: Protection of Authentication Data
Authentication data maintained by the TOE will be protected from disclosure and modification.

***Coverage Rationale*:** O.Auth_Protect is provided by FPT_PHP.3 (Resistance to physical attack). This requirement provides features that prevent or resist physical tampering with authentication data.

O.Authenticate: Authentication of Users and SSOs
Before cryptographic or other DoD data are accessed, either the user's identity or an administrative role will be authenticated by the TOE.

***Coverage Rationale*:** O.Authenticate is provided by FIA_AFL.1 (Authentication failure handling), FIA_SOS.1 (Verification of secrets), FIA_UAU.1 (Timing of authentication),

FIA_UAU.6 (Re-authenticating), and FIA_UAU.7 (Protected authentication feedback). FIA_SOS.1 specifies the minimum length of secrets (passwords). FIA_UAU.1 covers when authentication is necessary. The minimum strength level for the TOE will be SoF-medium. FIA_UAU.1 has an explicit SoF metric defined. SoF will be demonstrated for the authentication mechanism so that for each attempt to use the authentication mechanism, the probability that a random attempt will succeed is less than one in a million. FIA_UAU.6 requires re-authentication of the user before actions for which the user can be held legally responsible or during periods of inactivity. FIA_UAU.7 requires that no feedback information is provided to the user during authentication. FIA_AFL.1 requires the termination of the session establishment process after a specified number of unsuccessful user authentication attempts. FIA_AFL.1 further requires that the TSF places the TOE into the Locked state when the number of allowed unsuccessful user authentication attempts is exceeded.

O.Control_of_Applications: Control of Applications
A DoD PKI and Token must contain an application manager that will control the loading and use of applications on the TOE. The loading of applications onto the TOE will be controlled by using application signatures from an approved DoD entity. Any proposed DoD PKI and KMI Token application signature algorithm and key length must comply with Appendix E of this Protection Profile. Lastly, the token platform developer must provide guidance to develop secure applications on the developer's operating system platform.

***Coverage Rationale*:** O.Control_of_Applications is implemented by FDP.ITC.1 (Import of user data without security attributes) that details three related requirements that control the loading and use of applications on the TOE when importing these applications from outside the TOE's scope of control. By FDP.ITC.1, all loading of applications onto the TOE requires signature verification by the TOE, any proposed DoD PKI and KMI Token application signature algorithm and key length used must comply with Appendix E of this Protection Profile, and an application manager must be used by the TOE. The application manager ensures that the application code signature is verified prior to card execution. Additionally, it associates an application with card security states or domains, provides a method for selecting an application to execute, provides a method for deleting an application, and limits the number of load attempts of any given Application ID (AID) to three unsuccessful attempts. The combination of these related requirements ensures that applications will be secure loaded and used by the TOE.

O.Crypt: Cryptography
The TOE must perform cryptographic functions with sufficient strength for Sensitive But Unclassified (SBU) data.

***Coverage Rationale*:** O.Crypt is implemented by FCS_COP.1 (Cryptographic operation) that specifies how the TOE will perform specific cryptographic operations. FCS_CKM.1 (Cryptographic key generation), FCS_CKM.2 (Cryptographic key distribution), and FCS_CKM.4 (Cryptographic key destruction) require that cryptographic keys be generated, distributed, and destroyed in accordance with specified methods.

O.DAC: Data Access Control
The TOE must provide each authorized user with the means of controlling and limiting access to the objects and resources it owns or for which it is responsible, on the basis of user identity or role and in accordance with the P.Protection_Mechanisms Security Policy.

***Coverage Rationale*:** O.DAC (Data Access Control) is provided by a combination of requirements. FDP_ACF.1 (Security attribute based access control) set the basic access rule through the Data Access Control Security Function Policy (SFP.DAC). FDP_ACC.1 (Subset access control) provides the definition of to whom these apply, while FIA_ATD.1 (User attribute definition) provides the list of user security attributes. Import and export of user data are controlled through FDP_ITC.1 (Import of user data with security attributes) and FDP_ETC.1 (Export of user data with security attributes). The requirement FIA_UID.2 (User identification before any action) ensures that users identify themselves before any action will be allowed by the TSF, while FIA_UAU.1 (Timing of authentication) covers when authentication is necessary. The minimum strength level for the TOE will be SoF-medium. FIA_UAU.1 has an explicit SoF metric defined. SoF will be demonstrated for the authentication mechanism so that for each attempt to use the authentication mechanism, the probability that a random attempt will succeed is less than one in a million. FIA_UAU.6 (Re-authenticating) requires re-authentication of the user before actions for which the user can be held legally responsible or during periods of inactivity. FMT_MOF.1 (Management of security functions behavior), FMT_MSA.1 (Management of security attributes), and FMT_MTD.1 (Management of TSF data) allow the management of these functions. FMT_MSA.2, FMT_MSA.3, FMT_MTD.2, and FMT_MTD.3 guarantee that only secure values of attributes and TSF data are used. Finally, FMT_REV.1 (Revocation) identifies the roles that are allowed to revoke the security attributes necessary to have access.

O.D_Read: Data Read Format
The TOE shall format data passing between modules on the IC such that information is not exposed.

***Coverage Rationale*:** O.D_Read (Data Read Format) is provided by FDP_ITT.1 (Basic internal transfer protection). This requirement provides the means of preventing the disclosure or modification of user data when they are transmitted between parts of the TOE according to policies expressed in the SFP.DAC and the P.IFC. FPT_ITT.1 (Basic internal TSF data transfer protection) further protects TSF data from modification.

O.Data_Exchange_Conf: Enforce Data Exchange Confidentiality
Protect user data confidentiality when exchanging data with a remote system.

***Coverage Rationale*:** O.Data_Exchange_Conf (Enforce data exchange confidentiality) is provided by FDP_ITC.1 (Import of user data with security attributes) and FDP_ETC.1 (Export of user data with security attributes), which controls the import and export of user data.

O.Env_Strs: Environmental Stress
The TOE must protect itself against compromise by having a structure that neither reveals security information nor operates in an insecure fashion when exposed to out-of-standard conditions (high or low) in the environment, including such factors as temperature, voltage, clock frequency, and external energy fields.

***Coverage Rationale*:** O.Env_Strs (Environmental stress) is provided by FPT_PHP.3 (Resistance to physical attack) and FPT_FLS.1 (Failure with preservation of secure state). These requirements protect against identified vulnerabilities, including those that deal with manipulations outside defined operational boundaries and preserves a secure state of operation in the event that a failure does occur. This objective is ensured by AVA_VLA.3 (Moderately resistant).

O.Fail_Secure: Preservation of Secure State for Failures in Critical Components
Preserve the secure state of the system in the event of a secure component failure.

***Coverage Rationale*:** O.Fail_Secure (Preservation of secure state for failures in critical components) is provided by FPT_PHP.3 (Resistance to physical attack) and FPT_FLS.1 (Failure with preservation of secure state). The definition of a "secure state" should be provided by the security model documentation (ADV_SPM.1). These requirements protect against identified vulnerabilities, including those that deal with manipulations outside defined operational boundaries, and preserve a secure state of operation in the event that a failure does occur. This objective is ensured by AVA_VLA.3 (Moderately resistant).

O.I_Leak: Information Leak
The TOE must provide the means of controlling and limiting the leakage of information in the TOE so that no useful information is revealed over the power, ground, clock, reset, or I/O lines.

***Coverage Rationale*:** O.I_Leak (Information Leak) is provided by FDP_ITT.1 (Basic internal transfer protection). This requirement provides the means of preventing the disclosure or modification of user data when they are transmitted between parts of the TOE according to policies expressed in the SFP.DAC and the P.IFC. FPT-ITT.1 further protects TSF data from disclosure. FDP-IFF.3 (Limited illicit information flows) limits covert information flows as defined in FDP-IFC.1 (Subset information flow control).

O.Init: Initialization
An initialized TOE not in the Totally Locked state must assume the Nonauthenticated state immediately upon power-up, reset, or after other restart conditions.

***Coverage Rationale*:** By FIA_UID.2, the user will be identified before any actions are performed, which occurs in the Nonauthenticated state. By FPT_RCV.4 (Function recovery),

upon reset or other restart condition, the TOE enters the secure Power-on state and from there, it enters the Nonauthenticated state.

O.Input_Probe: Probing by Selected Inputs
The TOE must be resistant to repeated probing through insertion of erroneous data.

***Coverage Rationale*:** O.Input_Probe (Probing by selected inputs) is provided by FPT_PHP.3 (Resistance to physical attack). This requirement protects against identified vulnerabilities, including those that deal with manipulations outside defined operational boundaries. FDP_DAU.1 (Basic data authentication) provides protection against using inserted erroneous data.

O.Key_Encrypt: Encryption of Stored Keys (TSRD)
Keys stored in nonvolatile memory on the TOE must be encrypted.

***Coverage Rationale*:** O.Key_Encrypt (Encryption of stored keys) is provided by FCS_CKM.3 (Cryptographic key access). This requirement ensures that access to cryptographic keys is in accordance with a specified access method and based on an assigned standard.

O.Life_Cycle: Life-Cycle Functions
The TOE must provide a means of controlling and limiting the use of life-cycle-specific commands to the life-cycle stages in which they are intended.

***Coverage Rationale*:** O.Life_Cycle (Life-cycle functions) is provided by FDP_ACF.1 (Security-attribute based access control), which sets the basic access rules through the SFP.DAC and FDP_ACC.1 (Subset access control), which provide the definition of to whom these apply.

O.Log_Prot: Logical Protection
The TOE must protect itself against logical compromise by having a structure that is resistant to logical manipulation or modification.

Implementation Application: Updated versions of the TOE should counter vulnerabilities discovered in previous TOE versions.

***Coverage Rationale*:** O.Log_Prot (Logical Protection) is provided by the requirements and assurances discussed below. FPT_PHP.1 (Passive detection of physical attack) and FPT_PHP.3 (Resistance to physical attack) detect and protect against identified vulnerabilities, including those that deal with manipulations outside defined operational boundaries. FPT_FLS.1 (Failure with preservation of secure state) preserves a secure state of operation in the event that a failure does occur. FPT_SEP.1 (TSF domain separation) and FPT_RVM.1 (Non-bypassability of the TSP) ensure that the TSP is always invoked and that the TSF is protected from modification or damage by providing domain separation required for TSF execution. This objective is further

supported by ADV_IMP.1 (Subset of the implementation of the TSF), specifically in the implementation of unique serial number and other life-cycle identifiers. This objective is further ensured by AVA_VLA.3 (Moderately resistant), which reviews the identified vulnerabilities, including those involving the deconstruction and manipulation of the IC.

O.Mult_App: Multiple Applications
The TOE must support an application (or applications) while providing and maintaining security between and among the various resident elements.

Coverage Rationale: By FDP_IFF.1, all application-specific data are identified with that application. By FDP_ACC.1, these data are only available to that application and by FDP_IFC.1, an application defines which other applications can have access to its application-specific information.

O.Phys_Prot: Physical Protection
The TOE must be resistant to physical attack or be able to create difficulties in understanding the information derived from such an attack.

Coverage Rationale: By FPT_PHP.1, the TOE will detect physical tampering. By FPT_PHP.3, the TOE will automatically enter the Totally Locked state to prevent violation of the TOE Security Policy due to physical tampering. By FDP_DAU.1, DoD data are able to be verified to ensure the information content has not been forged or fraudulently modified.

O.Res_Access: Resource Access
The TOE shall protect its resources against monopolization by a user or attacker to the detriment of other users of the TOE.

Coverage Rationale: By FRU_RSA.1, the TOE enforces maximum quotas on memory, program space, and other resources on defined groups of users.

O.Role_Man: Role Management
Management of roles for the TOE is performed in a secure manner.

Coverage Rationale: By FMT_SMR.2, the TSF shall maintain identified security roles, be able to associate users with roles, and ensure that conditions for different roles are satisfied. By FMT_SMR.3, the TSF shall require an explicit request to assume identified roles.

O.User_Data_Control: User Data Control
The setting or modification of authentication, initial security, and personalization data must be controlled.

Coverage Rationale: The policy for SSO access is established by SFP.DAC. FDP_ACC.1 specifies that the TOE will enforce SFP.DAC on the specified data. FDP_ACF.1 specifies that access is controlled by role, and that all access information is properly protected.

O.Secure_Host_Comms: Secure Host Communications
The TOE and the host shall establish a secure channel, using a session key composed of components created by the TOE and the host, before exchanging cryptographic or other DoD data.

Coverage Rationale: By FTP_ITC.1, all cryptographic and DoD data are exchanged by way of a trusted channel that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from modification or disclosure. By FPT_ITI.1, any modification during transmission is detected. By FCS_CKM.1, the session key used for this exchange must be created by the TOE and the host. By FTP_ITC.1, DoD data can only be exchanged between a trusted host and the TOE by way of the channel, and the channel protects it from modification or disclosure.

O.Self_Test: Self Test
Self-tests shall ensure the TOE is functioning properly. Integrity of all code on the TOE shall be checked. Cryptographic and other security-critical functions shall be tested. These tests shall be performed during power-up and under certain conditions.

Coverage Rationale: By FPT_TST.1, the TSF shall run a suite of self-tests during initial start-up, when power, input voltage, input frequency, or temperature are outside their normal range, or when requested by the user. By FPT_AMT.1, these tests demonstrate the correct operation of the TSF security assumptions.

O.Set_Up: Set-Up Sequence
The TOE shall require that the SSO updates the preset (default) SSO authentication data prior to entering the Nonauthenticated state.

Coverage Rationale: SFP.DAC only allows the SSO to update the preset SSO authentication data. FDP_ACC.1 specifies that the TOE will enforce SFP.DAC on the specified data. FDP_ACF.1 specifies that access is controlled by role, and that all access information is properly protected.

O.Tamper_Response: Respond to Tamper
The TOE shall respond to physical tampering against specified system devices and components.

Coverage Rationale: By FPT_PHP.3, the TOE automatically responds to physical probing and

manipulation such that the TSP is not violated. By FDP_RIP.1, the contents of all resources are made unavailable when the resources are deallocated by a reset or other restart condition.

O.Trial: Trial and Error Resistance
The TOE authentication mechanism is resistant to spoofing by trial and error.

Coverage Rationale: By FIA_SOS.1, authentication data for both users and SSOs are sufficiently long to prevent it from easily being guessed. By FIA_AFL.1, repeated wrong guesses cause the TOE to enter a locked state.

O.Unlink: Linkage
The TOE must provide the means of allowing an entity to make multiple uses of resources or services without other entities being able to link those uses together.

Coverage Rationale: SFP.DAC does not permit non-DoD users to access DoD directories that contain resource usage information. By FDP_ETC.1, this is enforced when data are exported from the TOE, and by FDP_ITT.1 it is enforced when data are transferred between different parts of the TOE. FDP_IFC.1 and FDP_IFF.1 guarantee that the SFP.Info_Flow_Control holds and that data, including resource usage, only flow between applications as allowed by those applications.

O.Volatile_Memory: Destruction of Volatile Memory
The contents of volatile memory cannot be retrieved after power is removed from the TOE or a failure occurs.

Coverage Rationale: By FPT_FLS.1, a lack of power or a failure requires that a secure state be preserved. Since there are no protections on volatile memory, the only way that it can be secure is to be zeroized.

6.3.2 Assurance Security Requirements Rationale

The assurance level for this protection profile is EAL4 augmented.

EAL4 allows a developer to attain a reasonably high assurance level without the need for highly specialized processes and practices. It is considered to be the highest level that could be applied to an existing token line without undue expense and complexity. As such, EAL4 is appropriate for the DoD PKI and KMI Token.

Additionally, the assurance security requirements that implement EAL4 assist in supporting all of TOE's environmental security objectives as illustrated in Tables 6-3 and 6-4. See Section 4.2 for a discussion of environmental security objectives.

Evaluation Assurance Level (EAL) 4 is implemented in the TOE by:

1. ACM_AUT.1: Partial CM automation
2. ACM_CAP.4: Generation support and acceptance procedures
3. ACM_SCP.2: Problem tracking CM coverage
4. ADO_DEL.2: Detection of modification
5. ADO_IGS.1: Installation, generation, and start-up procedures
6. ADV_FSP.2: Fully defined external interfaces
7. ADV_HLD.2: Security enforcing high-level design
8. ADV_IMP.1: Subset of the implementation of the TSF
9. ADV_LLD.1: Descriptive low-level design
10. ADV_RCR.1: Informal correspondence demonstration
11. ADV_SPM.1: Informal TOE security policy model
12. AGD_ADM.1: Administrator guidance
13. AGD_USR.1: User guidance
14. ALC_DVS.1: Identification of security measures
15. ALC_LCD.1: Developer defined life-cycle model
16. ATE_COV.2: Analysis of coverage
17. ATE_DPT.1: Testing: high-level design
18. ATE_FUN.1: Functional testing
19. ATE_IND.2: Independent testing—sample
20. AVA_MSU.2: Validation of analysis
21. AVA_SOF.1: Strength of TOE security function evaluation

Evaluation Assurance Level (EAL) 4 is augmented with:

1. ALC_TAT.3: Compliance with implementation standards – all parts
2. AVA_VLA.3: Moderately Resistant

Augmentation results from the selection of:

AVA_VLA.3 Vulnerability Assessment—Vulnerability Analysis—Moderately resistant

The TOE is intended to function in a variety of applications, which may include secure messaging and identification systems. As such, it could contain, represent, or provide access to sensitive DoD data. In addition, the TOE will not always be directly under the control of trained and dedicated administrators. It may be subjected to a hostile environment for long periods of time. As a result, it is imperative that the TOE is shown to be moderately resistant to penetration attacks.

EAL4 requires vulnerability assessment through imposition of AVA_VLA.2. This dictates a review of only the identified vulnerabilities. Component AVA_VLA.3 requires, in addition, that a systematic search for vulnerabilities be documented and presented. This provides a significant increase in the consideration of vulnerabilities over that provided by AVA_VLA.2.

The rationale for this augmentation is based on the Common Evaluation Methodology (CEM) definitions of basic/medium/high attack potentials. These definitions apply most directly to

information processing systems that exist in small numbers and that are offered some form of external protection. The TOE, as discussed above, may be issued in large quantities, is exposed for prolonged periods of time, and is subject to short duration secondary attacks based on longer term development of sophisticated capabilities. As a result, the attack potentials, as stated, are not appropriate. They need to be redefined in this context for the TOE described in this protection profile. With that understanding, a moderate attack potential would address the most reasonably expected competent attacks. Addressing all attacks at all levels (e.g., AVA_VLA.4) introduces cost and complexity higher than justified for all but the most secure applications. It is also questionable if, given the current CEM definitions, this level can be achieved.

AVA_VLA.3 has the following dependencies:

- ADV_FSP.1 Informal functional specification
- ADV_HLD.2 Security enforcing high-level design
- ADV_IMP.1 Subset of the implementation of the TSF
- ADV_LLD.1 Descriptive low-level design
- AGD_ADM.1 Administrator guidance
- AGD_USR.1 User guidance

All of these are met or exceeded in the EAL4 assurance package.

ALC_TAT.3 Life Cycle Support—Tools and techniques—Compliance with implementation standards – all parts

The TOE is intended to host a variety of applications, which may include secure messaging, financial services, and access control. These applications will be developed outside of the TOE. Corrupt applications could compromise the secure operation of the TOE. Consequently, ALC_TAT.3 requires that the developer provide guidance for the development of secure applications on the developer's operating platform. The functional security requirement, FDP_ITC.1, compliments this requirement by requiring that such applications, developed following the TOE developer's guidance, is approved and signed by a DoD entity. The TOE will not allow the loading of applications without the DoD entity's signature.

EAL4 requires life cycle support through imposition of ALC_TAT.1. Tools and techniques distinguishes between the implementation standards applied by the developer (ALC_TAT.2) and the implementation standards for "all parts of the TOE" (ALC_TAT.3) that additionally inclues third party software. Thus, ALC_TAT.3 is needed to require implementation guidance be given by the TOE developer to third parties developing applications for the TOE.

AVA_TAT.3 has the following dependencies:

- ADV_IMP.1 Subset of the implementation of the TSF

This dependency is met in the EAL4 assurance package.

6.4 Dependency Rationale

This section demonstrates that the security requirements set forth in this PP form a mutually supportive and internally consistent whole. Internal consistency is shown through an analysis of dependencies. Mutual support is shown through consideration of the interactions between and among the security requirements.

The requirements in Table 6-5 are listed with requirements on which they are dependent. All of the dependencies identified below are met in this PP.

Table 6-5 Functional and Assurance Requirements Dependencies

Component	Depends On
Functional Requirements	
FCS_CKM.1	FCS_CKM.2 or FCS_COP.1, FCS_CKM.4, FMT_MSA.2
FCS_CKM.2	FCS_CKM.1, FCS_CKM.4, FMT_MSA.2
FCS_CKM.3	FCS_CKM.1, FCS_CKM.4, FMT_MSA.2
FCS_CKM.4	FCS_CKM.1, FMT_MSA.2
FCS_COP.1	FCS_CKM.1, FCS_CKM.4, FMT_MSA.2
FDP_ACC.1	FDP_ACF.1
FDP_ACF.1	FDP_ACC.1, FMT_MSA.3
FDP_DAU.1	-
FDP_ETC.1	FDP_IFC.1
FDP_IFC.1	FDP_IFF.1
FDP_IFF.1	FDP_IFC.1, FMT_MSA.3
FDP_IFF.3	FDP_IFC.1, AVA_CCA.1
FDP_ITC.1	FDP_IFC.1, FMT_MSA.3
FDP_ITT.1	FDP_IFC.1
FDP_RIP.1	-
FIA_AFL.1	FIA_UAU.1
FIA_ATD.1	-
FIA_SOS.1	-
FIA_UAU.1	FIA_UID.1
FIA_UAU.6	-
FIA_UAU.7	FIA_UAU.1
FIA_UID.2	-
FMT_MOF.1	FMT_SMR.2
FMT_MSA.1	FDP_ACC.1, FMT_SMR.2
FMT_MSA.2	FDP_ACC.1 or FDP_IFC.1, FMT_MSA.1, FMT_SMR.2, ADV_SPM.1
FMT_MSA.3	FMT_MSA.1, FMT_SMR.2
FMT_MTD.1	FMT_SMR.2
FMT_MTD.2	FMT_MTD.1, FMT_SMR.2
FMT_MTD.3	FMT_MTD.1, ADV_SPM.1
FMT_REV.1	FMT_SMR.2
FMT_SMR.2	FIA_UID.1
FMT_SMR.3	FMT_SMR.2
FPT_AMT.1	-
FPT_FLS.1	ADV_SPM.1
FPT_ITI.1	-
FPT_ITT.1	-
FPT_PHP.1	FMT_MOF.1
FPT_PHP.3	-
FPT_RCV.4	ADV_SPM.1
FPT_RVM.1	-
FPT_SEP.1	-

Component	Depends On
Functional Requirements	
FPT_TST.1	FPT_AMT.1
FRU_RSA.1	-
FTP_ITC.1	-
Assurance Requirements	
ACM_AUT.1	ACM_CAP.3
ACM_CAP.4	ACM_SCP.1, ALC_DVS.1
ACM_SCP.2	ACM_CAP.3
ADO_DEL.2	ACM_CAP.3
ADO_IGS.1	AGD_ADM.1
ADV_FSP.2	ADV_RCR.1
ADV_HLD.2	ADV_FSP.1, ADV_RCR.1
ADV_IMP.1	ADV_LLD.1, ADV_RCR.1, ALC_TAT.1
ADV_LLD.1	ADV_HLD.2, ADV_RCR.1
ADV_SPM.1	ADV_FSP.1
AGD_ADM.1	ADV_FSP.1
AGD_USR.1	ADV_FSP.1
ALC_TAT.3	ADV_IMP.1
ATE_COV.2	ADV_FSP.1, ATE_FUN.1
ATE_DPT.1	ADV_HLD.1, ATE_FUN.1
ATE_IND.2	ADV_FSP.1, AGD_ADM.1, AGD_USR.1, ATE_FUN.1
AVA_MSU.2	ADO_IGS.1, ADV_FSP.1, AGD_ADM.1, AGD_USR.1
AVA_SOF.1	ADV_FSP.1, ADV_HLD.1
AVA_VLA.3	ADV_FSP.1, ADV_HLD.2, ADV_IMP.1, ADV_LLD.1, AGD_ADM.1, AGD_USR.1

6.5 Rationale for Strength of Function Medium

The strength of function rating of SOF-medium is based on the potentially high value of information protected by the TOE, as well as the level of threat to the TOE as described in section 3.2 of this PP. Medium is specified to counter the assumption that attackers have a medium level of expertise, resources, and motivation. This strength of function rating is in turn consistent with the security objectives described in section 4.

Appendix A: References

[1] Common Criteria, Version 2.1, International Standard 15408, http://csrc.nist.gov/cc, August 1999.

[2] Department of Defense Chief Information Officer Guidance and Policy Memorandum 6-8510. *Department of Defense Information Assurance.* Draft, January 2000.

[3] International Standards Organization. ISO 7816 - I*dentification Cards - Integrated Circuit Cards with Contacts*.

[4] Kocher, Paul. "Differential Power Analysis and Problems in Applied Cryptography." In *CardTech/SecurTech '99 Gateway to Practical Innovation*. Chicago, 1999.

[5] National Institute of Standards and Technology. FIPS 140-2. *Security Requirements for Cryptographic Modules*. (Draft) 1999.

[6] National Security Agency. The Information Assurance Technical Framework, Release 2.0.1, September 1999. http://www.iatf.net/.

[7] *Smart Card Security User Group Smart Card Protection Profile*, Draft Version 2.0 May 1, 2000.

[8] U. S. Department of Defense. *Consideration of Smart Cards as the DoD PKI Authentication Device Carrier*, 10 January 2000.

[9] U. S. Department of Defense. *DoD Target Token Requirements Document*, (Draft), 8 March 2000.

[10] U. S. Department of Defense. *Public Key Infrastructure Target Class 4 Token Security Requirements*, Draft version 1.01, April 10, 2000.

[11] U. S. Department of Defense. *X.509 Certificate Policy for the United States Department of Defense*, version 5.2, 13 November 2000.

Appendix B: Acronyms

Common Criteria-Related Acronyms

CC	Common Criteria
CEM	Common Evaluation Methodology
DoD	United States Department of Defense
EAL	Evaluation Assurance Level
ISO	International Standards Organization
IT	Information Technology
PP	Protection Profile
SCSUG	Smart Card Security User Group
SF	Security Function
SFP	Security Function Policy
SML	Strength of Mechanism Level
SOF	Strength of Function
ST	Security Target
TOE	Target of Evaluation
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSFI	TSF Interface
TSP	TOE Security Policy

Token-Related Acronyms

AID	Application ID
CAD	Card Acceptor Device
CM	Configuration Management
CSP	Cryptographic Security Parameter
DAP	Data Authentication Pattern
DES	Data Encryption Standard
DPA	Differential Power Analysis
EC	Electronic Commerce
EEPROM	Electrically Erasable Programmable Read Only Memory
EXF	Executable file
IC	Integrated Circuit
ICC	Integrated Circuit Card
ISO	Information Security Officer
KMI	Key Management Infrastructure
OP	Open Platform
PCMCIA	Personal Computer Memory Card International Association
PIN	Personal Identification Number
PKI	Public Key Infrastructure
POI	Point of Issuance
RAM	Random Access Memory
ROM	Read Only Memory
RSA	Rivest, Shamir, and Adleman (encryption algorithm)

SBU	Sensitive But Unclassified
SFP	Security Function Policy
SHA	Secure Hash Algorithm
SIM	Subscriber Identity Module
SPA	Simple Power Analysis
SSO	System Security Officer
UA	Unauthorized Agent
USB	Universal Serial Bus

Appendix C: Glossary of Terms

The Glossary of Terms is subdivided into two sections: Common Criteria Terminology and Token Terminology.

Common Criteria Terminology

This section contains terms that are used in a specialized way in the CC. The majority of terms in the CC are used either according to their accepted dictionary definitions or commonly accepted definitions found in ISO security glossaries or other well-known collections of security.

Administration	Administrative responsibilities will be split between a system administrator and a security administrator who together will be able to administer the entire system. This is done to prevent any one person having too much control and to provide for two person integrity (checks and balances).
Assets	Information or resources to be protected by the countermeasures of a TOE (e.g., user data and cryptographic keys).
Assignment	The specification of an identified parameter in a component.
Assurance	Ground for confidence that an entity meets its security objective.
Attack potential	The perceived potential for success of an attack, should an attack be launched, expressed in terms of an attacker's expertise, resources, and motivation.
Augmentation	The addition of one or more assurance component(s) from Common Criteria Part 3 to an EAL or assurance package.
Authorized user	A user who may, in accordance with the TSP, perform an operation.
Component	The smallest selectable set of elements that may be included in a PP, an ST, or a package.
Dependency	A relationship between requirements such that the requirement that is depended upon must normally be satisfied for the other requirements to be able to meet their objectives.

Element	Most detailed refinement of a CC functional requirement. Similar elements when grouped together form a CC component requirement. When a CC component functional requirement is included in a PP, all associated elements must be included.
Evaluation Assurance Level (EAL)	A collection of assurance components from CC, Part 3, which when selected, represents a point on the CC predefined assurance scale.
Extension	The addition to an ST or PP of functional requirements not contained in CC, Part 2 and/or assurance requirements not contained in Part 3 of the CC.
Identity	A representation (e.g., a string) uniquely identifying an authorized user that can be either the full or abbreviated name of that user or a pseudonym.
Information	Defined as user data, regardless of its format.
Information Security Officer (ISO)	A person responsible for creating, maintaining, interpreting, and overseeing consistent implementation of site security policy and procedures.
Internal Communication Channel	A communication channel between separated parts of the TOE.
Internal TOE transfer	Communicating data between separated parts of the TOE.
Object	An entity within the TSC that contains or receives information and on which subjects perform operations.
Organizational Security Policies	One or more security rules, procedures, practices, or guidelines imposed by an organization upon its operations.
Protection Profile (PP)	An implementation-independent set of security functional and assurance requirements for a category of TOEs that meet specific consumer needs.
Refinement	The addition of details to a component.
Resources	Any system asset required for the correct operation of the TOE.
Role	A predefined set of rules establishing the allowed interactions between a user and the TOE.

Secret	Information that must be known only to authorized users and/or the TSF in order to enforce a specific SFP.
Security attribute	Information associated with subjects, users, and/or objects that is used for the enforcement of the TSP.
Security Function (SF)	A part or parts of the TOE that have to be relied on for enforcing a closely related subset of the rules from the TSP.
Security Function Policy	The security policy enforced by part or parts of the TOE that have to be relied upon for enforcing a closely related subset of rules from the TSP.
Security objective	A statement of intent to counter identified threats and/or satisfy identified organizational security policies and assumptions.
Security Target (ST)	A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
Selection	The specification of one or more items from a list in a component.
Strength of Function (SOF)	A qualification of a TSF expressing the minimum efforts assumed necessary to defeat its expected security behavior by directly attacking its underlying security mechanisms.
SOF-basic	A level of the TOE strength of function in which analysis shows that the function provides adequate protection against casual breach of TOE security by attackers possessing a low attack potential.
SOF-medium	A level of the TOE strength of function where analysis shows that the function provides adequate protection against straightforward or intentional breach of TOE security by attackers possessing a moderate attack potential.
SOF-high	A level of the TOE strength of function where analysis shows that the function provides adequate protection against deliberately planned or organized breach of TOE security by attackers possessing a high attack potential.
Strength of Mechanism Level (SML)	A scale for measuring the relative strength of a security mechanism hierarchically ordered from SML 1 through SML 3.
Subject	An entity within the TSC that causes operations to be performed.
Target of Evaluation	An IT product or system and its associated administrator and user

(TOE)	guidance documentation that is the subject of evaluation.
TOE Security Functions (TSF)	A set consisting of all hardware, software, and firmware of the TOE that must be relied on for the correct enforcement of the TSP.
TOE Security Functions Interface (TSFI)	A set of interfaces, whether interactive (man-machine interface) or programmatic (application programming interface), through which TOE resources are accessed and mediated by the TSF, or information is obtained from the TSF.
TOE Security Policy (TSP)	A set of rules that regulate how assets are managed, protected, and distributed within a TOE.
TOE Security Policy Model	A structured representation of the security policy to be enforced by the TOE.
TSF data	Data created by and for the TOE that might affect the operations of the TOE.
TSF Scope of Control (TSC)	The set of interactions that can occur with or within a TOE and are subject to the rules of the TOE site security policy.
Trusted Channel	A means by which a TSF and a remote trusted IT product can communicate with necessary confidence to support the TSP.
Unauthorized Agent (UA)	Any person (or process acting on behalf of a person) that is not authorized, under the TOE site security policy, to access the TOE resources or information processed by the TOE. This person includes anyone from a "hacker" to a determined foreign adversary, and security administrators, system administrators or authorized users who are untrustworthy.
User	Any entity (human user or external IT entity) outside the TOE that interacts with the TOE.
User data	Data created by and for the user that does not affect the operation of the TSF.

Token Terminology

This section contains terms that are used in a specialized way in the token authentication device industry. The majority of terms are used either according to their accepted dictionary definitions or commonly accepted definitions found in ISO security glossaries or other well-known collections of security.

Access control — Process of granting access to information system resources only to authorized users, programs, processes, or other systems.

Application — (1) An application may also be called an Executable file (EXF), Applet, or Cardlet (for Java Cards). An application is to be run on the token that may be downloaded onto the token during enrollment, or just prior to execution invoked by the host.
(2) Intended final use for the token. This may include (but is not limited to) such activities as payment, telephony, identification, secure information storage, or access.

Attack — An attempt to gain unauthorized access to an information system's services, resources, or information or the attempt to compromise an information system's integrity, availability, or confidentiality. There are several forms of attacks including:
Malicious attacks – virus, worm, Trojan horse, masquerading
Unintentional attacks – malfunction, human error
Physical attacks – fire, water, battle damage, power loss

Biometrics — Automated methods of authenticating or verifying an individual based on a physical or behavioral characteristic.

Cell Family — The group of building blocks used in the fabrication of any IC. A custom-made cell family will hinder the attacker attempting the reverse engineering of a token.

Class 4 Token — A hardware device that contains an operating system, uses an access control mechanism, performs key pair generation, and has protected storage capability in accordance with the requirements of the DoD PKI and KMI Token Protection Profile.

Class 4 Applications — Intended final uses for the DoD PKI and KMI token. This may include (but is not limited to) such activities as payment, telephony, identification, secure information storage, or access. Class 4 refers to the assurance level intended for applications handling high value, UNCLASSIFIED information (i.e., Mission Critical, National Security System Information) in a minimally protected environment.

Contact card	A type of smart card that communicates with the outside world via a reader connected to a standard (e.g., serial, USB, or PCMCIA) interface.
Contact-less card	A type of smart card that communicates with the outside world via radio frequency (RF) electromagnetic waves.
Cryptographic module	The set of hardware, software, firmware, or some combination thereof that implements cryptographic logic processes, including cryptographic algorithms, and is contained within the cryptographic boundary of the module.
Cryptographic Security Parameter (CSP)	Security-related information (e.g., secret and private cryptographic keys and authentication data such as biometrics, passwords, PINs) appearing in plain text or otherwise unprotected form and whose disclosure or modification can compromise the security of a cryptographic module or the security of the information protected by the module.
Data Encryption Standard	A widely-used method of data encryption using a private (secret) key that was judged so difficult to break by the U.S. government that it was restricted for exportation to other countries. There are 72,000,000,000,000,000 (72 quadrillion) or more possible encryption keys that can be used. For each given message, the key is chosen at random from among this enormous number of keys. Like other private key cryptographic methods, both the sender and the receiver must know and use he same private key.
Differential Power Analysis (DPA)	A technique combining physical measurement of such things as power consumption with statistical signal processing techniques to identify IC operating details. DPA can, in some instances, provide information leading to recovery of internal operational parameters, keys, etc.
DoD data	All data on the TOE located below the DoD directory. These data are owned by DoD. It includes DoD executables, DoD PINs, DoD cryptographic keys, and DoD user personal information.
Electrically Erasable Programmable Read Only Memory (EEPROM)	A non-volatile memory technology where data can be electrically erased and rewritten.

Failure analysis	The compilation of techniques used by semiconductor development and testing labs to identify the operating problems in newly designed or modified ICs. Such techniques include not only observation (to determine what is not functioning properly) but also modification of IC internal structure (to determine fixes).
FORTEZZA®	A type of PCMCIA cryptographic card produced by NSA. See Personal Computer Memory Card International Association Card.
Host	Device to which a token authenticates to establish a secure communication path.
Ibutton®	Type of token made by Dallas Semiconductor. A computer chip encased in a 16-mm stainless steel case that can be attached to articles of clothing, wallets, etc.
Integrated Circuit (IC)	Electronic component(s) designed to perform processing and/or memory functions contained on a single chip.
Integrated Circuit Card (ICC)	A card into which has been inserted one or more ICs.
Initialization	The process of writing specific information into nonvolatile memory during IC manufacturing and testing as well as executing security protection procedures by the IC manufacturer.
Java Ring®	Type of token. A Java Ring® is a ring with an iButton® attached to it.
Key Management Infrastructure	DoD program that will unify existing and planned key management systems with the DoD PKI to create a single, integrated whole.
Key Exchange Algorithm (KEA)	Algorithm used by cryptoprocessors (e.g., FORTEZZA®) to produce key exchange keys. See the following Web site for more details: http://csrc.nist.gov/encryption/skipjack/skipjack-kea.htm .
Nonvolatile memory	A semiconductor memory that retains its content when power is removed (i.e., ROM, EEPROM, FLASH).
Operational keys	The cryptographic keys loaded onto the assembled token product for use by the token holder during normal operations.
Password	A string of characters (letters, numbers, and other symbols) used to authenticate an identity or verify access authorization.

Personal Computer Memory Card International Association Card (PCMCIA)	A hardware device that supports specific dedicated functions. Examples of PCMCIA card functions include memory devices, input/output devices (e.g., modems and fax modems), and portable disk drives. PCMCIA cards are most commonly used to provide additional computing features for portable computers such as laptops. NSA's FORTEZZA® Crypto Card is an example of a PCMCIA card. PCMCIA cards provide the strongest security and largest memory storage capacity of available tokens.
Personal Identification Number (PIN)	A 4- to 16- character alphanumeric code or password used to authenticate an identity (commonly used in banking applications).
Personalization	The process of writing specific information into the nonvolatile memory preparing the IC for issuance to users.
Photomask	A mask used during chip manufacturing to protect selected parts of a silicon wafer from a light source while allowing other parts of the surface of the wafer to be exposed. The purpose is to expose the photoresist on the surface so that subsequent etching processes can generate the desired substrate structure. The photomask is the means by which the chip's circuits and, therefore, its functionality are placed on the chip.
Point of Issuance (POI)	A facility or system at which personalized User Keys for a token are defined and requested from the KMI, and the token is provided to a specific end user.
Post-issuance	The time period during which the token is in the hands of the user. On some tokens, additional functionality can be loaded onto the token post-issuance.
Private key	A cryptographic key used with a public key cryptographic algorithm, uniquely associated with an entity and not made public.
Public key	A cryptographic key used with a public key cryptographic algorithm, uniquely associated with an entity, and that may be made public.
Public Key Infrastructure (PKI)	A PKI enables users of a basically unsecure public network such as the Internet to securely and privately exchange data and money through the use of a public and a private cryptographic key pair that is obtained and shared through a trusted authority. The public key infrastructure provides for a digital certificate that can identify an individual or an organization and directory services that can store and, when necessary, revoke the certificates.

Public key (asymmetric) cryptographic algorithm	A cryptographic algorithm that uses two related keys for encryption and decryption—a public key and a private key. The two keys have the property that, given the public key, it is computationally infeasible to derive the private key.
Production Keys	The cryptographic keys loaded onto the IC for security during production.
Random Access Memory (RAM)	A volatile, randomly accessible memory (used in the IC) that requires power to maintain data.
Read Only Memory (ROM)	A nonvolatile memory (used in the IC) that requires no power to maintain. ROM data are often contained in one of the numerous masks used during manufacture.
Reverse engineering	The compilation of techniques used by semiconductor development and testing labs to generate design documentation and specifications for an unknown IC. Reverse engineering, in its most complete sense, would allow the identification of a complete fabrication package given only an (unidentified) IC as a starting point.
Rivest, Shamir, and Adleman (RSA)	RSA is an Internet encryption and authentication system that uses an algorithm developed in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. The RSA algorithm is the most commonly used encryption and authentication algorithm.
Semiconductor IC	An integrated circuit (IC), sometimes called a chip or microchip, is a semiconductor wafer on which thousands or millions of tiny resistors, capacitors, and transistors are fabricated. An IC can function as an amplifier, oscillator, timer, counter, computer memory, or microprocessor.
Simple Power Analysis (SPA)	A technique in which physical measurements of power consumption over time are used to identify IC operating details. SPA can, in some instances, provide information leading to recovery of internal operational parameters, keys, etc.
Subscriber Identification Module (SIM)	A token having a shape in accordance with ISO 7812, designed to be inserted into a special cavity in a mobile phone.
System Security Officer (SSO)	The role assumed to perform a set of cryptographic initialization or management functions (e.g., cryptographic key and parameter entry, and alarm resetting).

Tamper detection	The automatic determination by a cryptographic module that an attempt has been made to compromise its physical security.
Tamper response	The automatic action taken by a cryptographic module when it detects that a physical tampering has occurred (minimum response action is zeroization of plain text keys and other CSPs).
Terminal	The device capable of reading or writing to a token.
Token	An authentication device carrier that is used to store and carry cryptographic keys and certificates supporting user identity authentication. This technology can consist of (but is not limited to) smart cards, USB tokens, PCMCIA Card, and iButtons® /JavaRing® technology.
Token holder	A person to whom a token has been legitimately issued (a user).
Token issuer	An institution who issues tokens.
Token Operating System	Operating system developer-specific code, written in the microprocessor's native or machine code.
Token reader	A machine capable of reading and/or writing to a token.
Transport keys	The cryptographic keys loaded onto the IC for security during transport of ICs, modules, and assembled products prior to issuance.
Universal Serial Bus	An interface incorporating the high-speed external bus for PCs. A Universal Serial Bus (USB) token is a device containing an embedded microprocessor IC that interfaces directly with a PC's USB port without any additional hardware, e.g., a card reader. The microprocessors used in USB tokens are just as powerful as those in smart cards.
Zeroization	A method of erasing electronically stored data by altering or deleting the contents of the data storage so as to prevent the recovery of the data.

Appendix D: Description of Token States

Some states of the DoD PKI and KMI Token need to be defined to effectively describe the conditions under which some of the token security requirements apply. The following diagram, Figure D-1, illustrates the states of the token and the relationships between the states.

Security critical functions will only be executable in the appropriate authentication state. Each description of a state will contain a list of allowable host-commanded functions (functions that the host commands the token to perform). If a given function is listed under a given state, then it cannot be run under any other state unless it is explicitly stated.

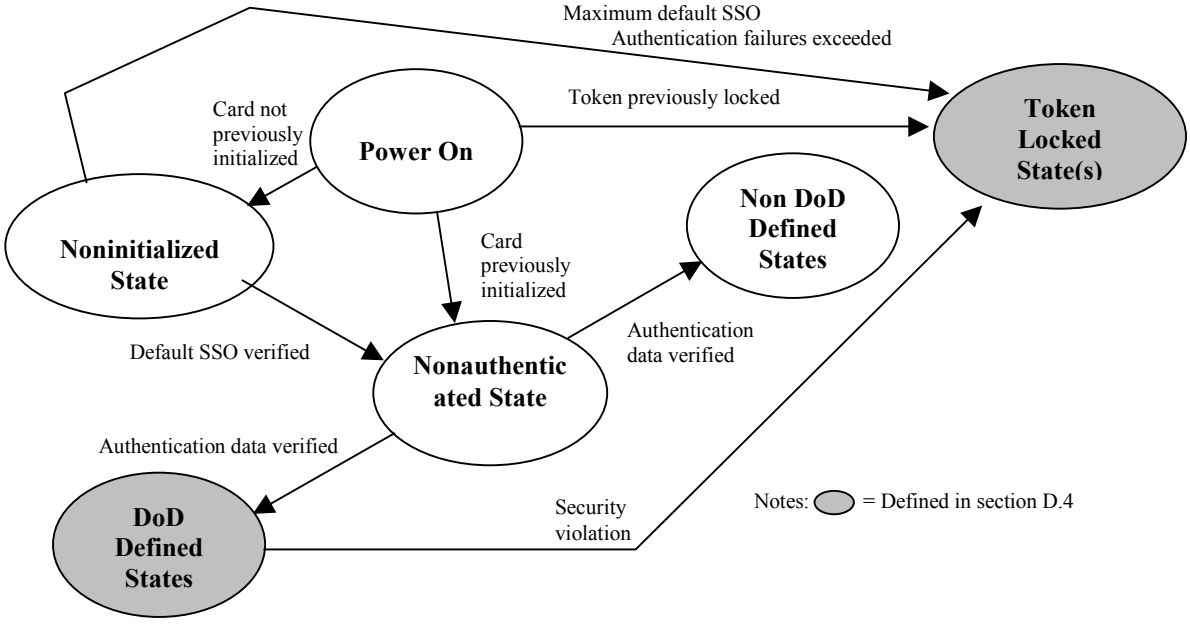


Figure D-1 Token Top Level State Diagram

D.1 Power-On State

The Power-On state will perform required power on tests and determine the next state. Out-of-range temperatures and power and faulty clock signals will place the TOE in the Power-On state. The following requirements apply to the Power-On state:

1. The token must determine if the token has been previously initialized or locked.
2. If the token has been locked, then the specific locked state must be entered (refer to section D.4.3 for a description of locked states).
3. If the token has not been locked, and if the token has not been initialized, then the Noninitialized state must be entered.
4. If the token has not been locked, and if the token has been initialized, then the Nonauthenticated state must be entered.
5. Power-on self-tests (in compliance with FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities) must be run prior to exiting the power-on state.
6. Self-tests must include data integrity on all code and tests on cryptographic functions and all security-critical functions.

Host commanded functions allowed in this state:

- None.

D.2 Noninitialized State

The Noninitialized state is the state of the token after manufacture. The following requirements apply to the Noninitialized State:

1. The Noninitialized state will implement default SSO authentication data.
2. Updating the SSO authentication data is the only token function allowed in this state.
3. Upon receiving the token, the token issuer must update the SSO authentication data.
4. Updating the SSO data must include verification of the default data. Four successive authentication failures will place the token in the Totally Locked state (refer to section D.4.3.3 for details).
5. Successfully updating the SSO data will result in placing the token in the Nonauthenticated state.
6. The default SSO data must be destroyed (i.e., the Noninitialized state must never be used again).

Host commanded functions allowed in this state:

- Update of the SSO's authentication data.

D.3 Nonauthenticated State

This is the state of the token after power-on and a successful SSO update. The following requirements apply to the token in the Nonauthenticated state:

1. The Nonauthenticated state will enter a Token Locked state if any security violation (e.g., a PIN or biometric authentication attempt exceeding the maximum number of attempts allowed) occurs.
2. The Nonauthenticated state shall only be exited when an authentication mechanism has been successful or a security violation has occurred.

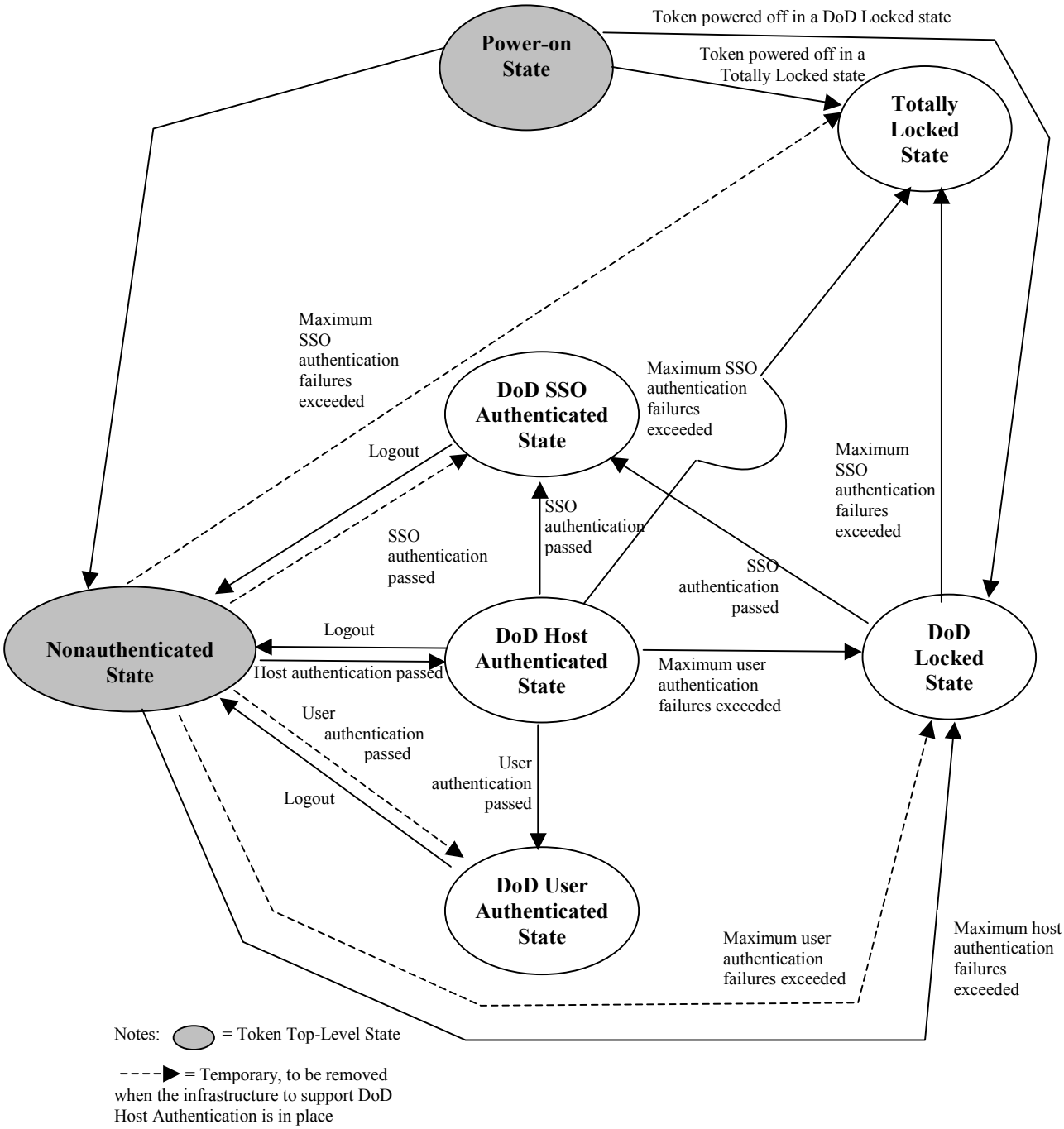
Host commanded functions that are allowed in this state:

- General status information about the token.
- Directory or file information about non-DoD directories, if the application that owns the directory allows it (e.g., the electronic purse).
- Functions that do not have a requirement to operate in an Authenticated state.

D.4 DoD Defined States

This section defines token states specific to DoD security requirements. Figure D-2 below illustrates the relationships between the Token Top Level states (Figure D-1) and DoD specific states.

Figure D-2 DoD Level State Diagram



D.4.1 DoD Authentication States

The token will require several authentication states.

The following requirements pertain to achieving these states:

1. The directory owner may determine how the user-authenticated state is achieved.
2. All non-DoD applications (e.g., electronic commerce) run in a nonauthenticated state (the states of those applications are not trusted).

Access control for the DoD directories on the DoD PKI and KMI token is based on establishing roles that require an authentication method to place the token in an authenticated state. The following diagram illustrates the relationship between the user, the authentication method, and the state(s). The states for which authentication is necessary prior to entry are the DoD Host Authenticated state and the DoD Human Authenticated states consisting of the DoD SSO Authenticated state, and the DoD User Human Authenticated state.

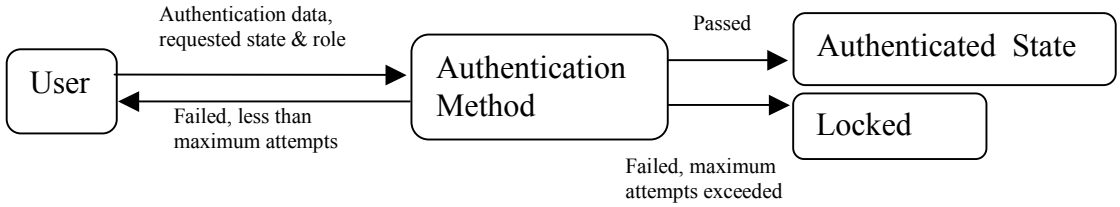


Figure D-3 Role Between User, Authentication Method, and Authenticated State

D.4.1.1 DoD Host Authenticated State

The DoD Host Authenticated state exists when the DoD Host Access Method has been successfully completed. The DoD Host Authenticated state is intended to provide a state in which trust can be placed in the host communicating with the token. Once the host has been authenticated with the token and a secure session has been established, then a state will exist in the token in which the token is protected from many host-based attacks.

The User Authenticated state can be accessed from within the Host Authenticated state. If infrastructure exists to support the DoD Host Access Method, then the User state should only be accessed from the Host Authenticated state.

Host commanded functions that are allowed in this state:
- Directory or file information about DoD directories
- Access to certificates
- Access to the DoD Human Authenticated states

The User Authenticated state can be accessed from within the Host Authenticated state. If infrastructure exists to support the DoD Host Access Method, then the User state should only be accessed from the Host Authenticated state.

Host commanded functions that are allowed in this state:
- Directory or file information about DoD directories
- Access to certificates
- Access to the DoD Human Authenticated states

D.4.2 DoD Human Authenticated States

The DoD Human Authenticated states are the DoD SSO Authenticated state or the DoD User Authenticated state.

Host commanded functions that are allowed in these states:
- Generating private/public key pairs
- Loading private keys
- Loading certificates
- Loading EXFs
- Creating directories or files within the Master File (the top level directory)
- Creating directories or files under the DoD directory

D.4.2.1 DoD SSO Authenticated State

The SSO Authenticated state exists after an SSO has been authenticated using the SSO authentication mechanism.

Host commanded functions that are allowed in this state:
- Token intialization (updating the SSO authentication data from the default after manufacture)
- Unlocking the token after the Token Locked state has been entered
- Creating, modifying, and updating DoD user authentication data

D.4.2.2 DoD User Authenticated State

The User Authenticated state exists after a user has been authenticated using the established user authentication mechanism.

Host commanded functions that are allowed in this state:
- Signing data
- E-mail operations

D.4.2.3 Exiting a Human Authenticated State

Any Human Authenticated state will revert to the Nonauthenticated state if:

1. A logout command is received from the host.
2. A directory or file is accessed (selected by the host) that is not within the same directory or below the directory that was selected when the authentication mechansism was invoked.
3. The token is removed from the token reader.
4. Power down condition occurs.
5. The token is reset.

D.4.3 Locked States

The DoD PKI and KMI token must employ several Locked states to allow for enabling and disabling access control to the token.

D.4.3.1 DoD Locked State

The DoD Locked state will disable the functions that require DoD user authentication after security violation has occurred (e.g., a PIN or biometric authentication attempt exceeded the maximum number of attempts allowed).

1. The token must have the ability to be forced into a Token Locked state via an authentication mechanism.
2. The token must have the ability to be forced into a Token Locked state from an EXF.

Host commanded functions that are allowed in this state:
- An attempt to get to the DoD SSO Authenticated state
- Functions that do not require DoD user privileges

D.4.3.2 Totally Locked State

The Totally Locked state will exist to disable the token's functionality after failed attempts are made to enter the DoD SSO authentication state (e.g., a PIN or biometric authentication attempt exceeding the maximum number of attempts allowed). The TOE will enter the Totally Locked state when tampering is detected. The Totally Locked state inherits all the requirements of the Token Locked state. In addition to those requirements, the following requirements apply:

1. The token must have the ability to be forced into a Totally Locked state via an SSO authentication mechanism.
2. The token must have the ability to be forced into the Totally Locked state from an EXF.
3. It shall not be possible to leave the Totally Locked state.
4. The only host commanded functionality allowed in the Totally Locked state is the authentication of an SSO.

Host commanded functions that are allowed in this state:
- SSO authentication

D.5 Non-DoD States

Non-DoD applications will be able to define separate authentication states (via EXFs), or use existing states in the token's operating system.

The security requirements relating the non-DoD states are:
1. Non-DoD authentication states will have no effect on DoD authentication states.
2. Non-DoD applications cannot use DoD authentication states.

Host commanded functions that are allowed in this state:
- Functions that are not required to operate in DoD authenticated states

D.6 Additional States

Applications (DoD and Non-DoD) will be able to define separate authentication states (via EXFs) that run on the token for access to the files within the application's directory.

Appendix E: Required Supported Cryptographic Algorithms

The following cryptographic algorithms must be supported in the DoD PKI and KMI Token:

Signature Algorithms:
1024 bit RSA
2048 bit RSA
DSA 1024
Elliptic Curve Digital Signature Algorithm 384

Key Exchange Algorithms:
1024 bit RSA
2048 bit RSA
Diffie-Hellman 1024 (FIPS approved version)
KEA 1024
Elliptic Curve Key Exchange Algorithm 384

Symmetric Algorithms:
AES (128, 192, and 256 bit keys)
DES 64
Triple DES 128
Skipjack

Hash Algorithms:
SHA-1
MD-5
SHA 256
SHA 384
SHA 512

Appendix F: DoD PKI and KMI Specifications

The DoD PKI and KMI Token PP refers to two specifications:

- Application Specification for the DoD PKI and KMI Token
- Key Management Specification for the DoD PKI and KMI Token

The Application Specification for the DoD PKI and KMI Token details the requirements for developing non-DoD applications and the requirements for loading DoD applications. This specification requires the token platform developer to provide guidance to develop secure applications on the platform.

The Key Management Specification details the DoD PKI and KMI Token's key management requirements, procedures, and policies. This specification will discuss key loading based on guidance in the TSRD, section 5.3.4.3 – Key Loading. The private key generated and stored on the token must never leave the token. It should be unchangeable and stored in nonvolatile EEPROM. This specification also details the handling of the private key on the token. The initial application verification key should be stored in ROM, and subsequent application verification keys should be stored in EEPROM.

Application Specification for the DoD PKI and KMI Token

References

[1] Global Platform. Open Platform Card Specification, Version 2.0.1, 7 April 2000.
[2] Common Criteria, Version 2.1, International Standard 15408, http://csrc.nist.gov/cc, August 1999.

1 Introduction

Compliance with this specification is required for certification as a Department of Defense (DoD) Public Key Infrastructure (PKI) and Key Management Infrastructure (KMI) Token or as a DoD PKI and KMI Token application. This specification is for use with the *Department of Defense Public Key Infrastructure and Key Management Infrastructure Token Protection Profile* only and will be revised as the protection profile (PP) is updated.

The DoD PKI and KMI token allows for multiple applications to support the DoD mass population (i.e., all DoD military, civilian, and contractor personnel operating in the SBU environment). Security functions present will be of an appropriate level and protection. An application can only be used by the DoD PKI and KMI token after its signature by a DoD entity has been validated. Typical applications for tokens within the DoD include: financial, secure messaging, identification, secure information storage, and access control. Each of these may have somewhat different security requirements, security features, roles, and environmental considerations (e.g., whether always on-line, always used off-line, usually off-line with the capability of going on-line, etc.). The security requirements for operating software, applications, and procedures for adding or deleting those applications must therefore be clearly identified, and the security functions that are present must be appropriate to the type and intended use of the token. This document specifies security-related mechanisms required on the token in order to develop, load, and execute applications on the DoD PKI and KMI Token.

2 Token Requirements

2.1 Token Overview

A token is used to store and carry cryptographic keys and certificates supporting user identity authentication. There are various types of tokens including smart cards, universal serial bus (USB) tokens, Personal Computer Memory Card International Association (PCMCIA) cards, and iButtons®/Java Rings®. The DoD PKI and KMI Token will contain an integrated circuit (IC) and

an operating system.

A semiconductor (silicon) IC is fabricated in a complex microelectronic process, which involves repeatedly masking and doping the surface of a silicon substrate to form transistors, followed by patterning metal connections, and applying a protective overcoat. This process eventually yields a design typically comprising several hundred thousand transistors, arranged in an area less than 25 square millimeters. The design consists of a central processing unit, input and output lines, and volatile and non-volatile memory.

The IC itself is packaged in a token. The current predominant packaging method is die bonding in a module. A module consists of a carrier board on which the IC is seated. Wire bonds are connected from the IC's input/output (I/O) pads to the carrier, which has contacts on its reverse side.

The token also contains an operating system that may be stored in Read Only Memory (ROM). The DoD PKI and KMI Token's operating system will allow authorized applications to be added to the token. Examples of operating systems are MultOS®, Java Virtual Machine®, and Smart Cards for Windows® with MEL®, Java®, and Visual Basic® as their programming languages respectively.

2.2 DoD vs. non-DoD

DoD applications are those applications sponsored by a DoD service or agency and written to support an inherently governmental function. These applications frequently require access to data protected within a DoD security state or domain or perform a service with no commercial counterpart.

Non-DoD applications would include applications permitted to reside on the DoD PKI and KMI Token as a convenience to the government, the cardholder, and trusted commercial partners such as entities in the financial and travel industries.

2.3 Application Manager

A DoD PKI and KMI Token must contain an application manager. An application manager is a module of code embedded in the OS of a token which controls the load, verification, selection, and execution of a token application. The application manager provides the following services:

- It ensures that the application code signature is verified prior to card execution.
- It provides a means to associate an application with card security states or domains.
- It provides a method for selecting an application to execute.
- It provides a method for deleting an application.
- It limits the number of load attempts of any given Application ID (AID) to three unsuccessful attempts.

The *Open Platform Card Specification* [1] has been selected as the Application Manager for the DoD PKI and KMI Token. To achieve certification as a DoD PKI and KMI Token, the *Open Platform Card Specification* [1] must be implemented.

Note: Other DoD PKI and KMI Token requirements documents may call out additional Global Platform requirements.

2.4 Application Signatures

Application signatures must be used to sign approved applications to control the loading of applications onto the DoD PKI and KMI token. Valid applications must be signed by an approved DoD entity. Any proposed DoD PKI and KMI Token application signature algorithm and key length must comply with Appendix E of this protection profile. To provide maximum interoperability with COTS products and to simplify Card Management, support for 1024 bit RSA with SHA-1 is required of all proposed DoD PKI and KMI Tokens for signatures on DoD applications. Support for DES signatures is permitted for non-DoD applications. This is referred to as the Data Authentication Pattern (DAP) in the *Open Platform Card Specification* [1]. Support for additional algorithms and key lengths is optional.

2.5 Future Guidance on Signature Verification

While noting the current state of the commercial practice as displayed by the *Open Platform Card Specification* [1], it is intended that future versions of this specification will require full certificate-chain validation of signatures for application code.

2.6 Guidance for Secure Application Development

The DoD will provide required applets to the token platform developer. However, provisions must be made to load applets onto the token after the token is manufactured. The token platform developer must provide guidance to develop secure applications on the developer's operating system platform.

2.7 Summary of Application Specification Requirements

1. A DoD PKI and KMI Token must contain an application manager (see FDP_ITC.1.3 in the Requirements Section of this PP).
2. Application signatures must be used to sign approved applications to control the loading of applications onto the DoD PKI and KMI token. Valid applications must be signed by an approved DoD entity (see FDP_ITC.1.3 in the Requirements Section of this PP).
3. Any proposed DoD PKI and KMI Token application signature algorithm and key length must comply with Appendix E of this Protection Profile (see FDP_ITC.1.3 in the Requirements Section of this PP).
4. The token platform developer must provide guidance to develop secure applications on the developer's operating system platform (see ALC_TAT.3 in the Requirements Section of this PP).

Key Management Specification for the DoD PKI and KMI Token

References

A. Public Key Infrastructure Target Class 4 Token Security Requirements, Draft Version 1.00, February 14, 2000.

B. KMI 2011: Program Glossary (Draft), Version 5, 30 November 2000.

C. X.509 Certificate Policy for the United States Department of Defense, Version 5.0, 13 December 1999.

Part I. Discussion: Concept of the Cryptographic Token

1. Introduction

The purpose of this specification is to establish key management requirements for tokens to be used for Class 4 applications within DoD. The requirements are intended to support the DoD PKI and KMI Token Protection Profile. The requirements section of this document, Part II, is correlated to appropriate sections of the protection profile. This discussion section has references after each requirement that refers the reader to the related requirement in the PP or states that a particular requirment is a KMI requirement, which is outside the scope of the token requirements. The requirements in this specification, both in Part I and Part II, are additional requirements to those in the PP or give more specificity to the requirements in the PP.

Reference B gives a KMI view of a token and more specificially, a Class 4 Token. A *token* is a portable, physical device in which keys and certificates are stored. In KMI, the most common token is the DoD Common Access Card (CAC), a specific commercial smart card for applications at Class 4 assurance level. A *Class 4 Token* is a hardware device that contains an operating system, uses an access control mechanism, performs key pair generation, and has protected storage capability in accordance with the requirements of the DoD PKI and KMI Token Protection Profile.

The role of a token is to perform a set of cryptographic functions to support its user's needs for security services such as identification, authentication and data confidentiality. The token holds and protects the keys needed to perform such cryptographic operations for its user.

The key management requirements applicable to a cryptographic token must achieve a comfortable balance between the sometimes competing needs of security and operational effectiveness. Security is important because the DoD plans to use the Class 4 token to support security services for mission critical data and systems, ranging from access control to confidentiality. The adequacy of the key management approach is critical to the security of all systems that rely on the token. Weak, ineffective key management approaches could undermine the applications that rely on the token for cryptographic security services. But from the perspective of operational effectiveness, burdensome key management solutions could render the token unusable. Key management approaches for the token must be able to support diverse operational environments and the size of the DoD community to which tokens will be issued. Given a community of millions of users, automated yet secure key distribution methods must be used that will allow keys to be added or changed without necessitating a return to some central issuing authority.

2. Requirements Categories Applicable to a Cryptographic Token

As implied in the Reference B definitions, the DoD PKI and NSA's Key Management Infrastructure (KMI) initially will support the DoD Common Access Card as the standard Class 4 token. The KMI target architecture acknowledges that there may be requirements for other cryptographic tokens. Planners anticipate technical advancements, evolution of tokens, eventual introduction of new or improved token types, and use of tokens that have multiple purposes, some of which may be non-cryptographic. This specification attempts to identify the requirements for compatibility with the supporting key management infrastructure that would apply to all such tokens.

This document does not constitute a complete requirements specification for a token, since there are a number of other categories besides key management for which requirements would be levied. The requirements categories for a token may include, for example:

- Physical requirements - size, shape, weight, form factor, ruggedness, environmental constraints.
- Functional requirements - special requirements dictated by the intended use of the token, which may include some non-cryptographic functions.
- External Interface requirements - input/output data and format, connector type, pin-outs, electrical specifications, ICDs.
- Host Device requirements - any other special requirements related to the specific host device or application with which the token is intended to be used.
- Human-Readable Labeling requirements - for example, identification of manufacturer, part number, equipment applicability or intended use of the token, a "Return if Found" address, and perhaps a photograph of the token's human user.
- Security requirements - as detailed in the DoD PKI and KMI Token Protection Profile and Reference A.
- Key Management requirements.

This specification addresses only the last subject on the list. The intent is to articulate the minimum set of requirements for a token to be supportable by the KMI, without introducing any conflict with any of the other requirements categories listed above. It is also the intent to support and adopt existing industry standards whenever possible, and to do nothing to preclude the use of commercially available off-the-shelf products as tokens. If the key management requirements in this specification do reflect any inconsistencies or incompatibilities with other requirements, or unnecessarily limit the applicability of COTS products, the reader is encouraged to bring these issues to the attention of the NSA Key Management Program Office, V5.

3. Cryptographic Material Needed by a Token

The cryptographic keys contained on a token can be categorized as either *User Keys* or *Token Keys*, as described in the following two sections.

A. User Keys

As indicated in Part I.1, the purpose of a token is to perform cryptographic functions in support of its user's needs for security services. The token must hold and protect the keys needed to perform the requisite cryptographic operations for its user (FCS_CKM.3). These keys are referred to as the *User Keys*. *User Keys* are loaded onto a token when it is first personalized and issued to a user. The token must also support addition, deletion, and replacement of *User Keys* at anytime during its lifetime following initial issuance (FCS_CKM.1, FCS_CKM.4). A "user" in this context may mean a human, a device, a component, a system, or a software application running on a host computer. The token must be able to store various sets of private/public keys and corresponding certificates, as well as symmetric keys, to support its user for the following purposes (FCS_CKM.3):

- General purpose signature/identification.
- E-mail signature.
- E-mail encryption.
- Log-on for special applications to a host device, system, or network.
- KMI Manager role(s).
- Organization/group roles.
- Other token applications (e.g., access control to a medical data applet on the token).

B. Token Keys

The token also must have a set of key management functions that provide the capability to securely load, store, and maintain *User Keys* on the token. This includes certificate processing and path validation to verify the source of the *User Keys* during key load operations, protecting keys in storage, and providing confidentiality to *User Keys* during transport from the KMI to the token (KMI requirement, FCS_CKM.3). The keys used to support these functions are categorized as *Token Keys*. A token possesses the following *Token Keys*:

- DoD Root CA Certificate - The token uses the Root CA certificate for source authentication during key load operations (i.e., to verify that the signature on certificates associated with *Token Keys* and *User Keys* is traceable to the root), and for supporting certificate path validation operations when providing security services for its user. At least one Root CA certificate must be loaded on a token at the factory or other secure depot facility prior to distributing the token for issuance in the field (KMI requirement). The initial Root CA certificate load must be done in some special, protected factory/depot mode only (KMI requirement). The token then allows only *Token Keys* and *User Keys* to be loaded that are traceable to the Root CA certificate held by the token. The token must be capable of loading an additional Root CA certificate or replacing the original Root CA certificate subsequent to the factory/depot initialization in some authenticated manner (e.g., signed by a Root CA already recognized by the token), and allowance must be made for storage of two Root CA certificates to allow for their natural rollover and expiration (KMI requirement).[10]
- Token Unique Key - Most tokens would have two sets of public key material that are used for interaction with the KMI or the host device. These are referred to as Token Unique Keys (TUKs). One public key set is a signature key that is used to authenticate the token to the KMI or the host. The other is a key establishment key, used to support the generation of a symmetric key encryption key which provides confidentiality to key material delivered from the KMI to the token, or the generation of a session key with the host.[11] TUKs must be loaded from a trusted external source, or created on the token itself at the factory or a secure facility (FCS_CKM.1, KMI requirement). TUKs are good for the lifetime of the token (anticipated to be 3-5 years), although the token does support replacement of these keys. The KMI retains the association between a token serial

[10] Although this specification states a requirement for a token to store two DoD Root CA certificates and support rollover, this capability is unlikely to be needed in the near term, and the requirement possibly could be waived for first-generation DoD Class 4 tokens. The concept for CA signatures articulated in the Reference D policy is that all signatures in a given CA domain must be traceable back to that CA, and relying parties may use the CA certificate for the life of the signed subordinate certificate beyond that signing. CAs may not issue subordinate certificates with validity periods that extend beyond the expiration dates of their own certificates and public keys, and a CA certificate validity period must extend well past the last use of the CA private signing key. Thus, validity periods for certificates and signing keys are quite long at high levels in the CA hierarchy, in order to support reasonable rollover intervals at lower levels of the hierarchy. Reference D prescribes a maximum validity period for a root CA signature certificate of up to 36 years, with the lifetime of the associated signing key being 25 years. That is, the DoD root CA shall cease to use a signing key 25 years after its creation, and shall begin to use a new signing key and certificate at that point, but any certificates signed with the original key, including those issued to subordinate CAs, remains valid for up to another 11 years. Rollover of a DoD Root CA certificate is therefore expected to be an infrequent event. No rollover process has yet been defined within the KMI program, and no rollover is likely to occur during the lifetime of PKI tokens issued in the next twenty or so years. It should be noted, however, that the subject of recovery from compromise of a DoD Root CA must be considered. If a token is permitted that can accommodate only one root CA certificate, it would have to be physically replaced in event of a compromise.

[11] The KMI will support transactions with tokens that possesses only a single TUK that is used for both signature and key encryption, but unless there are token types with limited storage capacity, it is expected that most tokens would use separate signature and key establishment keys. For the case of a token that cannot accommodate a TUK key pair at all, the KMI will also support the use of a unique, symmetric KEK that can be used in lieu of the key establishment public key material to securely deliver keys from the KMI to the token. The symmetric KEK could be generated by the token and provided to the KMI, along with the token serial number, during the factory/depot initialization process, or it could be provided by the KMI and loaded onto the token in plaintext form by the factory/depot initialization facility.

number and the public portion of a TUK (perhaps by creating a certificate that is used only internal to KMI). Any certificate associated with a TUK must contain the serial number of the token (KMI requirement).

- Token Storage Key - The token uses a token-unique symmetric key to encrypt all *User Keys* and other *Token Keys* for long term, non-volatile storage on the token. This key is referred to as the Token Storage Encryption Key (TSEK). The TSEK is stored on the token in non-volatile memory in a split, or in otherwise protected form that is a function of the user's token Personal Identification Number (PIN) or pass phrase, perhaps in combination with some biometric data. The TSEK is generated by the token itself during initialization at the factory or secure facility, and it is used for the lifetime of the token (anticipated to be 3-5 years). Access to keys encrypted in the TSEK must meet the requirements of FIPS 140-2 Level 2 for user tokens or FIPS 140-2 Level 3 for Registration Authority and Certification Authority tokens (FCS_CKM.3).

C. Key Material Summary

A summary of the *User Keys* and *Token Keys* held by a token is shown in Table 1. Note that multiple keys and certificates may be required for certain cases (e.g., to support multiple algorithms for interoperability if a DoD standard is not chosen).

Key Material	Est Storage (bytes) for 2K RSA	Key Type	Notes
DoD Root CA Certificate (Current)	1,500	Token	Root certificate currently in use.
DoD Root CA Certificate (Next)	1,500	Token	Supports rollover of Root CA and compromise recovery. See footnote 1.
Token Unique Keys			KMI will support use of a single key pair for both signature and key establishment, but most tokens are expected to have both. Symmetric KEK is optional, but unique to token if used.
Signature Private/Public Key Pair	768	Token	
Key Establishment Private/Public Key Pair	768	Token	
Symmetric Key Encryption Key	32	Token	
Token Storage Encryption Key	32	Token	Used to encrypt *User Keys* and *Token Keys* in non-volatile storage.
General Purpose Identity Key and Certificate	2,000	User	Used for general identity purposes and for e-mail signature.
E-Mail Encryption Key and Certificate	2,000	User	Might also have general purpose uses, to include file encryption.
E-Mail Decryption Private Key (Recovered)	200	User	Provided as part of key recovery operation. Required for recovering old encrypted e-mail and other files.
Local Network Log-on Key and Certificate	2,000	User	Optional. Tokens may require multiple User Keys depending on applications supported.
KMI Manager Roles Key and Certificate	2,000	User	
Group/Organizational Key and Certificate	2,000	User	

Table 1: Summary of Keys and Certificates Stored in Class 4 Token

4. Token State Diagram

Figure 1 is a state diagram that illustrates the steps involved in initializing a token from a key management viewpoint. The token key management functions operate under the following state definitions:

Unkeyed State[12] - The token is completely zeroized from a keying perspective. It does not contain any *User Keys* or *Token Keys*. Active tamper detection mechanisms are not armed. It entered this state either because it is a newly manufactured device that has never been keyed, or because it executed a command for zeroization of all cryptographic content. In either case, the

[12] Reference B, Section 5.2.5, states a requirement that any executable files (applets) that can be run on a token must be signed prior to loading on the token, and that the token must hold the public value of a key to verify the signature in its non-volatile memory. The token development scenario described in Appendix A of Reference B states that this public key would be placed into the executable code of the token operating system by the OS developer. So, for a smart card or other device that is intended to become a DoD cryptographic token, it is assumed that the manufacturing process includes the embedment of a public signature key into the executable code of the operating system, followed by the loading of a key management applet that is signed by the corresponding private signature key. The device verifies the signature of the software applet, which enables it to accept keys and certificates when it is in the Unkeyed State. The public key in the OS is the same key for all devices of the same type or in a given manufacturer production run, and it is a permanent part of the device that is never changed or zeroized. The signing of the key management software applet in the private key would be done by NSA, or by the device manufacturer under NSA supervision and control. The private/public key pairs used for this purpose would most likely be generated by the NSA KMI and provided to token manufacturers, but this is not a requirement applicable to the token.

device is unusable as a cryptographic token. The token will exit this state and transfer to the Unique Key Load State after successful load of one or more DoD Root certificates.

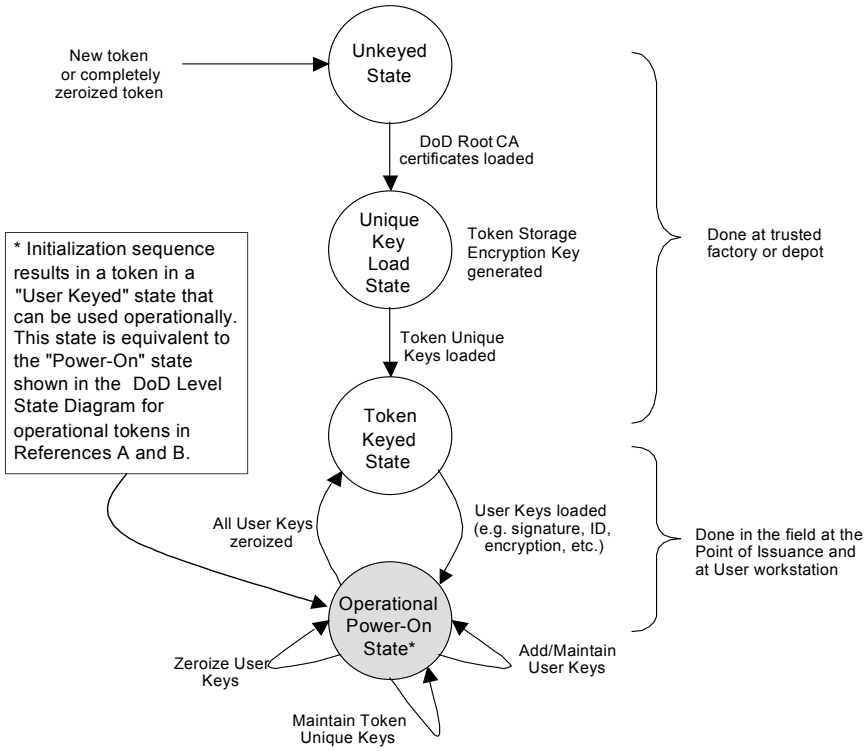


Figure 1: Initialization States of a Token

Unique Key Load State - In this state the token is ready to be initialized with its Token Unique Key(s). The token contains only the DoD Root certificate(s), which it uses to authenticate the source of the Token Unique Keys. Active tamper detection mechanisms are armed. The token can only enter this state from the Unkeyed State upon the successful load of one or more root certificates that will serve as the trust anchor(s) for certificate path validation and key source authentication activities. The token also generates its unique Token Storage Encryption Key while in this state, upon the command of the factory or depot initialization facility. It exits this state and enters the Token Keyed State upon successful TSEK generation and loading of its Token Unique Keys.

Token Keyed State - The token contains its Token Unique Keys and Token Storage Encryption Key, which subsequently can be used to load and store *User Keys*, which will personalize the token for a specific user. Those activities related to loading an initial set of *User Keys* are all that the token can perform while in this state. Active tamper detection mechanisms are armed. The token can enter this state from the Unique Key Load State upon successful load of Token Unique Keys. The token can also enter this state from the Operational Power-On State by executing a user command to zeroize all *User Keys* on the token. It exits this state and enters the Operational Power-On State upon successful load of its basic *User Keys*.

Definition: *Point of Issuance (POI) - a facility or system at which personalized User Keys for a token are defined and requested from the KMI, and the token is provided to a specific end user.* A POI is normally located in the field, near the end user, and its function is to enable the transition of a token in the Token Keyed State to the Operational Power-On State. The User Keys may be loaded at the POI prior to issuance of the token, or they may be loaded later by the user at a workstation. Both processes are described in Part I.7.

Operational Power-On State - In the Operational Power-On State the token contains its *Token Keys* and *User Keys*. It is a usable, operational device while in this state, and it can perform the subsequent operational state transitions that are depicted in the DoD-Level State Diagram shown in Reference A and the DoD PKI and KMI Token Protection Profile. It can also perform cryptographic functions in support of its user by employing *User Keys*. It can load additional *User Keys* or replacement *User Keys* through use of *Token Keys*. It can zeroize *User Keys* through direction from the user. It can generate or accept replacement *Token Keys*. Its active tamper detection mechanisms are armed. The token can enter this state only from the Token Keyed State upon successful load of its basic *User Keys*. The token can exit this state and enter the Token Keyed State if all *User Keys* are zeroized from the token.

Note that it is a basic requirement for a token to accept and execute commands from a user (human user, device, or SSO) to zeroize an individual *User Key*, zeroize all *User Keys*, or zeroize all cryptographic content, including both *User Keys* and *Token Keys*. The latter command would most likely be used only in a situation where the token is being removed from service, the token is about to be compromised, or the host device with which the token is used has been lost or compromised. That command would return the token to its Unkeyed State. The commands to zeroize some or all *User Keys* could be used for various operational reasons to change the personalized characteristics of the token. A desired change in the functionality of the token, or in the attributes or privileges of its user, for example, could dictate deletion of a key. Or a desire to reuse the token for a different purpose or a different user could prompt the deletion of all *User Keys*. That would reset the device to the Token Keyed State.

5. Key Tagging

All *Token Keys* and *User Keys* loaded onto a token must be labeled with a unique key identifier or a certificate serial number (FCS_CKM.3). The labels identify the type and purpose of each key, enabling the cryptographic functionality of the token to be determined, and they may be used for configuration control or inventory purposes, or in compromise recovery actions. The *User Key* labels also allow the host device or the supported applications to identify and select the appropriate keys for use in providing user security services.

The KMI maintains a data base which identifies, by token serial number, all items of key material that have been issued for that token. In addition, given a specific user identifier, the KMI can identify all certificates that were created for that user and associate all token serial numbers to which they were delivered. Thus, in event of a lost or compromised user or token, given either the token serial number or the user ID, KMI has the capability to identify all KMI-issued keys and certificates that need to be revoked.

Upon the command of any KMI Manager workstation or other authorized entity, a token shall electronically output its key and certificate identifiers, and the certificate content associated with any private/public key pair (KMI requirement). This data indicates the cryptographic content of the token and the intended purpose of each key set, but does not expose any private material. A Point of Issuance system typically would be expected to examine the content of a token prior to requesting any subsequent issuance of key for that token from KMI, both to assure that the token has sufficient storage capacity, and to determine that there is no duplication or incompatibility with already existing key material.

The ability of a manager workstation to view the content of a token also allows the user and the purpose of a misplaced "found" token to be determined, and may thereby avoid a potential compromise.

6. Loading Token Keys at Factory or Depot

This section describes the steps by which a token becomes initialized cryptographically and made compatible with the KMI. The initialization process is completed before the token is shipped to the field to be personalized and issued to a user. It accomplishes the loading of *Token Keys*, and transitions the token from the Unkeyed State to the Token Keyed State, as shown in Figure 1.

A. Initialization Facility

Initialization is envisioned as a process that would occur at the token production facility (factory) or at an authorized facility. The specific functional and security requirements that would apply to the facility and to the initialization process have not been fully developed. These requirements need to be documented as a follow-on activity to this specification. They must be consistent with already-stated token security requirements, so that the security posture of the end product or the supported applications is not jeopardized by inadequate process or production controls.

It is likely that a dedicated token initialization system would be required to initialize tokens, and that it would be subject to certification by NSA. The system may need to be housed in a cleared facility and operated by cleared personnel (level TBD). The system would contain a component (such as a KMI Manager workstation) that is a KMI-registered user and can order and receive private/public key pairs from KMI to be used as token keys. Communication with KMI would most likely be via web browser, but could also be accomplished using portable physical media. The token initialization system must possess a copy of the DoD Root CA certificate to load onto the tokens.

B. TSEK Generation and Root CA Certificate Loading

As implied by the definition of *Class 4 Token* stated in Part I.1 and the footnote in Part I.4, it is assumed that a token in the Unkeyed State is a working component with an operating system

and an access control mechanism. Any key management software applet that is required to support key loading and key management functions of the token has been loaded, with its signature verified, during the production process, prior to the start of key initialization.

The token is "unlocked" by an authorized human operator, who possesses an appropriate PIN or password, or other credentials required by the token's access control mechanism. The operator commands the token to generate its own unique, symmetric Token Storage Encryption Key. The TSEK is stored in non-volatile memory in a split, or otherwise protected, form where the split is a function of a PIN or pass phrase. This could be a default PIN, or the operator could command the generation of a new PIN or pass phrase unique to the token. It may also be possible, for operational reasons, to store the TSEK in unprotected form on the token until the token is personalized for a user, in order to eliminate the need to track initial PINs from the factory/depot to the Point of Issuance. The latter approach would require a protected distribution process for the token. The former would require no special distribution controls for the token, but would necessitate delivery of token and PIN in separate channels. Both approaches should be supported.

The DoD Root CA certificate is next loaded onto the token by the token initialization system. The token is then placed in a mode such that the Root CA certificate cannot be changed without completely resetting the token and zeroizing all key material from it. (Root CA certificate(s) subsequently may be changeable through an authenticated process, but a root certificate cannot be changed directly via this factory/depot initialization process unless the token is completely zeroized and reset.) The Root CA certificate is encrypted in the TSEK and stored in non-volatile memory. While the root CA certificates are 'public' known values, the encryption helps to deter a determined adversary from changing the loaded values or adding new ones. At this point the token has transitioned from the Unkeyed State to the Unique Key Load State (Figure 1).

C. Loading Token Unique Keys

Token Unique Keys may be provided by the KMI or may be generated by the token itself. The KMI, the token, and the initialization facility must support both options (FCS_CKM.1, KMI requirement).

Token Unique Keys Provided by KMI - For this case, the token initialization system has a file of Token Unique Key sets provided by the KMI. Normally each key set consists of a private/public key pair for signature and a private/public key pair for key establishment, but as discussed in Section I.3.B, a key set could consist of a single private/public key pair for both purposes, or a plaintext symmetric KEK. Each private/public key set must be unique, have a unique identification number, and be signed by a CA signature that is traceable to the DoD Root CA (KMI requirement, FCS_CKM.1). These key sets may be provided by the KMI to the token initialization system via portable physical media or electronically. The token initialization system selects and loads a key set onto the token. The token receives the key set, verifies the CA signature, and stores the keys appropriately encrypted in the TSEK. The token then generates a receipt that contains the unique identification of the key set and the token's serial number, signs the receipt with its newly acquired token signature key, and sends the signed receipt back to the

token initialization system. After successfully completing the Token Unique Key load, the token transitions into a mode such that the Token Unique keys cannot be changed without completely resetting the token and zeroizing all key material from it. The token initialization system accepts the receipt from the token, verifies the signature and the information in the receipt, and stores it for subsequent return to the KMI. The token initialization system zeroizes each key set after successfully loading it onto one and only one token. If a token rejects the load of a key set, the token initialization system tries to load the same key set again. If the load fails again, the key set is zeroized, and the results of the operation are recorded for subsequent return to the KMI. When the KMI receives the receipt from the token initialization system, it processes the receipt and registers the token by serial number in its database along with its corresponding public keys. It may use a certificate to bind the token identification to the public key, but other methods may be possible. The forwarding of receipts by the initialization system, and processing of receipts by KMI, would probably be done in a batch mode rather than on an individual basis. Any subsequent key material delivery from KMI to a token, whether *User Keys* or *Token Keys*, would require the cooperative generation of a symmetric KEK for the key download session between the token and the KMI, using the token's key exchange public key. The KMI enters the identifiers of any key sets that failed to load on the CRL.

Token Unique Keys Generated by the Token - In this case the token generates its own private/public key pairs for signature and key exchange, or may generate a unique symmetric key to be used as a KEK. The token initialization system commands the token to create the private/public key pairs or the unique symmetric key. As part of this command, the initialization system provides the token with a copy of a KMI certificate and public key component that the token will use to encrypt the public key or the symmetric key that will be sent to the KMI. The token creates a receipt containing its unique serial number, and the public keys it generated or a copy of the symmetric key it generated (encrypted in the provided KMI key). The token signs the receipt with its token signature key, and provides the receipt to the token initialization system. After successfully completing the Token Unique Key load, the token transitions into a mode such that the Token Unique Keys cannot be changed without completely resetting the token and zeroizing all key material from it. All generated keys are stored in non-volatile storage encrypted in the TSEK. The token initialization system receives the receipt from the token, verifies the signature and the information in the receipt, and stores it for subsequent return to the KMI. When the KMI receives the receipt from the token initialization system, it processes the receipt and registers the token by serial number in its database along with its corresponding public keys or symmetric key. It may use a certificate to bind the token identification to the public key, but other methods may be possible. The forwarding of receipts by the initialization system, and processing of receipts by KMI, would probably be done in a batch mode rather than on an individual basis. Any subsequent key material delivery from KMI to a token, whether *User Keys* or *Token Keys*, would require either encryption in the unique token-generated symmetric KEK or the cooperative generation of a symmetric KEK for the key download session between the token and the KMI, using the token's key exchange public key.

Token Unique Key is Symmetric Key provided by KMI - A similar process should also be supported where the Token Unique Key is a symmetric key instead of a private/public key pair. In this case the token initialization system has a file of plaintext symmetric keys provided by the KMI. Each token would be loaded with a unique symmetric key, would store the key encrypted

in the TSEK, and would provide a receipt back to the token initialization system. The receipt would contain the unique identification of the key loaded and the serial number of the token, all cryptographically bound as a function of the loaded key. The receipt would be returned to the KMI which would process it similarly to the public key case. The KMI would encrypt all key products delivered to the token using the symmetric Token Unique Key.

For any case in which the token initialization system is unable to verify the token signature on a token's receipt for a key, the token must be zeroized and discarded (KMI requirement). In any verification testing done at the factory/depot initialization facility, if any *Token Keys* that have been placed on a token become corrupted or otherwise unusable, the token must be zeroized and discarded (KMI requirement).

D. Delivery of Initialized Token

After the token is completely initialized with appropriate *Token Keys* at the factory or depot, it is ready to be shipped to a field site that serves as a Point of Issuance (POI). The shipping method is TBD, but some level of protection must be afforded to tokens during shipment (ADO_DEL.2). DCS probably would be used for OCONUS shipments. A token shipment should be tracked to the Point of Issuance, with the POI providing a receipt back to the factory/depot initialization system upon receipt, but this is not a KMI function. The level of tracking/receipting afforded by a shipping carrier may be sufficient.

7. Loading User Keys at Point of Issuance

A. POI Facility

User Keys are loaded on a token at a Point of Issuance (POI) at the time the token is first issued to a user. For the near-term, the POI systems that have a capability to issue tokens to users include RAPIDS terminals and KMI Manager workstations that will be located in various DoD operational environments. A token would typically be loaded at a POI with a general purpose signature/identification key, an e-mail signature key, and an e-mail encryption key. Other keys such as organizational or group keys, or keys for use in a KMI Manager Role may be added later.

As with the *Token Keys*, the KMI may generate private/public key pairs for *User Keys*, or the token may generate its own private/public key pairs. For most operational applications, it is expected that a token would generate its own key pairs for signature keys. With this approach, there is no need for the private key to ever leave the token or exist outside the token, so non-repudiation of the associated signature is assured. For key types other than signature key, the KMI is expected to retain the private key in a key recovery data base in order to support a subsequent key recovery service. Therefore either party may generate these key pairs. In any case, there is no token requirement for how any particular key pair must be generated, and the POI system, the key loading process, and the token must support both approaches.

All key material delivered from the KMI to a token must be encrypted in such a manner that only the token can decrypt it, even if the material consists only of a public key and/or a certificate (KMI requirement, FCS_CKM.2). This provides a consistent approach for delivery and assures confidentiality for non-public material being delivered to the token, as well as for certificate contents.

The token and the key loading process also must allow for subsequent loading of *User Keys* onto the token at a user workstation or another KMI Manager workstation after initial issuance by the POI (FCS_CKM.3, KMI requirement). This capability is necessary for subsequent addition or replacement of keys on a token. It is also needed to support the scenario in which the KMI is not available or on-line at the time of token issuance, and the loading of *User Key* has to be completed at a later time, after the token leaves the POI facility.

B. KEK Establishment

Since all key material delivered from the KMI to a token must be encrypted specifically for the token, the token must support cooperative generation of a symmetric Key Encryption Key with the KMI (FCS_CKM.1).[13] The KEK would be established by an indirect, store-and-forward key establishment method. It would be a function of the token's private key for key establishment (part of the Token Unique Key set loaded at the factory/depot), a token-generated random value, and some KMI-held key material. There must be only one such KEK active at any given time for a particular token, and this KEK may be used multiple times during the key load session to encrypt each key/certificate sent to the token (FCS_CKM.3). The KEK would be destroyed by both parties at the conclusion of the key load session.

Prior to sending a *User Key* order to KMI on behalf of a token, the POI must obtain information from the token that can be forwarded to the KMI to establish the KEK (KMI requirement). The POI system sends a request to the token requesting a KEK Establishment Response. The token provides a response to the POI system that includes the token's public key for key establishment, a token-generated random value, and the token's serial number. (The token's permanent public key for key establishment might not need to be included in the response, since the KMI already has it in its database, bound to the serial number of the token.) The token signs the response with its signature key (part of the Token Unique Key set loaded at the factory) and sends it to the POI system. The POI system receives the response from the token and then includes it as part of the request it sends to the KMI for *User Keys* to be loaded on the token.

When the KMI receives a key request from a POI, it uses the data in the token's KEK Establishment Response, along with some KMI-held material, to generate a unique KEK for use with that token. The KMI then generates the key requested in the order, and formulates a response to the POI which is encrypted in the KEK. The response also includes some public KMI data that can be used by the token, along with its own KEK Establishment data, to generate the same KEK. Thus the encrypted key package can be decrypted only by the intended token.

[13] As noted in Part I.3.B., KMI will also support the use of a unique, pre-placed, symmetric KEK for a token that doesn't support cooperative KEK generation.

C. Key Loading Process at the Point of Issuance

The sequence of steps in the key loading process, and the handling of key management data, may vary depending on the availability of the KMI and on where the private/public key pairs for *User Keys* are to be generated (by the KMI or by the token itself). There are four scenarios that must be supported: 1) KMI available and KMI generates private/public key pairs for *User Keys*; 2) KMI available and token generates private/public key pairs for *User Keys*; 3) KMI not on-line, but KMI is to generate private/public key pairs for *User Keys*; 4) KMI not on-line, and token is to generate private/public key pairs for *User Keys* (FCS_CKM.1). The following paragraphs describe the steps for each scenario. It should be noted that the KMI will support batched key orders from a POI, and an order may include requests for a mixture of key types, for keys for multiple tokens, or for keys with both token-generated key pairs and KMI-generated key pairs, even if they are intended for the same token.

KMI available and KMI generates private/public key pairs for *User Keys* - For this case, the POI system does not need any information from the token other than the KEK Establishment Response described above. The POI forms a request to the KMI for *User Keys* that contains the token's KEK Establishment Response and other information needed, such as the user's identification information to go in the certificates. The KMI is available and on-line to accept the request from the POI system. The KMI validates the POI privileges and other security relevant items, and then processes the request. It generates the requested private/public key pairs for *User Keys*, creates certificates, and encrypts the resulting keys and certificates for the token using the KEK established between the KMI and the token. The encrypted package is sent back to the POI, which in turn downloads it to the token. The token then generates the same KEK, decrypts the package, and validates that the keys came from an authorized source. The token stores the keys in non-volatile memory encrypted in the TSEK, and destroys the KEK that it generated cooperatively with the KMI for this key loading session.

KMI available and token generates private/public key pairs for *User Keys* - For this case, the POI system needs additional data from the token. The POI requests the token to generate private/public key pairs for the *User Keys*. The token generates its own key pairs and sends the public keys, along with the token serial number, back to the POI system in the form of a Token *User Key* Generation Response. The token then stores the key pairs in non-volatile memory encrypted in its TSEK. (As noted in the Part I.6.B. discussion of root CA certificates, even the public components of key pairs are normally stored in encrypted form to preclude any possibility of inadvertent additions or modifications, or tampering.) The POI system formulates a request to the KMI for the *User Keys* that includes the token's KEK Establishment Response, the public keys generated by the token for *User Keys*, and other information such as the user's identification information to go in the certificates. The KMI receives the request from the POI system, validates the POI privileges and other security relevant items, and then processes the request. The KMI creates the requested certificates, and encrypts the keys and certificates for the token using the KEK established between the KMI and the token. The encrypted package is sent to the POI and then downloaded to the token. The token generates the same KEK, decrypts the package, and validates that the keys came from an authorized source. It then stores the keys and certificates in non-volatile memory encrypted in the TSEK, and destroys the KEK that it generated cooperatively with the KMI for this key loading session.

KMI not on-line, but KMI is to generate private/public key pairs for *User Keys* - This scenario implies a staged sequence of events in which key request and key delivery occur in two separate sessions with the KMI. For the case where KMI is to generate the key, the POI system does not need any additional information from the token other than the KEK Establishment response. The POI forms a request to the KMI for *User Keys* that contains the token's KEK Establishment Response and other information needed such as the user's identification information to go in the certificates. Since the KMI is not on-line, the POI system saves the request until the KMI is available. The POI system commands the token to retain the token-generated random value portion of its KEK Establishment response in non-volatile memory encrypted in the TSEK. The token will need this data later in order to generate the KEK in which the KMI response will be encrypted. At this point the token can be given to the user, along with its PIN and instructions for completing the loading of *User Keys* at a later time. The user may also need an order number or POI-generated reference number in order to identify the expected product to the KMI at the later key delivery session. Until KMI comes on-line and the subsequent delivery session is completed, the token remains in the Token Keyed State and cannot be used operationally.

When the KMI becomes available, the POI system forwards the request created previously to the KMI. As part of this request, it tells the KMI whether the requested key is to be delivered back to the POI or directly to the user. The KMI validates the POI privileges and other security relevant items, and then processes the request. It generates the requested private/public key pairs for *User Keys*, creates certificates, and encrypts the keys and certificates for the token using the KEK established between the KMI and the token. The KMI then delivers the package to the POI system, or notifies the user via e-mail that the package is available. For the case of delivering it to the POI system, the remaining processes for initializing the token would be the same as for the scenario where the KMI is available.

For the case of delivering the package to the user, the user must initiate another session with the KMI in order to receive his *User Keys*. The user holds the PIN for the token and is the only one who can unlock the token. Once the token is unlocked, the user accesses the KMI via his user workstation or other host device. The token's TUK is used to authenticate the token to the KMI. KMI delivers the encrypted key package to the user workstation, which in turn provides it to the token. Since the *User Keys* previously ordered are all encrypted in the cooperatively generated KEK, they can only be decrypted by the token. The token generates the same KEK, decrypts the package, and validates that the key came from an authorized source. It then stores the keys in non-volatile memory encrypted in its TSEK, and destroys the token-generated copy of the KEK.

KMI not on-line, and token is to generate private/public key pairs for *User Keys* - This scenario is also a staged sequence involving two sessions with the KMI. For this case, the POI system needs some additional data from the token beyond the KEK Establishment response. The POI requests the token to generate private/public key pairs for the *User Keys*. The public keys are provided back to the POI system. The token stores the key pairs in non-volatile memory encrypted in its TSEK. The POI system formulates a request to the KMI for the *User Keys* that includes the token's KEK Establishment response, the public keys generated by the token for

User Keys, and other information such as the user's identification information to go in the certificate. Since the KMI is not on-line, the POI system saves the request until the KMI is available. The POI system commands the token to retain the token-generated random value portion of its KEK Establishment response in non-volatile memory encrypted in the TSEK. The token will need this data later in order to generate the KEK in which the KMI response will be encrypted. At this point the token can be given to the user, along with its PIN and instructions for completing the loading of *User Keys* at a later time. The user may also need an order number or POI-generated reference number in order to identify the expected product to the KMI at the later key delivery session. Until KMI comes on-line and the subsequent delivery session is completed, the token remains in the Token Keyed State and cannot be used operationally.

When the KMI becomes available, the POI system forwards the request created previously to the KMI. As part of this request, it tells the KMI whether the requested product is to be delivered back to the POI or directly to the user. The KMI validates the POI privileges and other security relevant items, and then processes the request. It creates the requested certificates, and encrypts the keys and certificates for the token using the KEK established between the KMI and the token. The KMI then delivers the package to the POI system or notifies the user via e-mail that the package is available. For the case of delivering it to the POI system, the remaining processes would be the same as for the scenario where the KMI is available.

For the case of delivering the package to the user, the user must initiate another session with the KMI in order to receive his *User Keys* (KMI requirement). The user holds the PIN for the token and is the only one who can unlock the token. Once the token is unlocked, the user accesses the KMI via his user workstation or other host device. The token's TUK is used to authenticate the token to the KMI. KMI delivers the encrypted key package to the user workstation, which in turn provides it to the token. Since the *User Keys* previously ordered are all encrypted in the cooperatively generated KEK, they can only be decrypted by the token. The token generates the same KEK, decrypts the package, and validates that the keys came from an authorized source. It then stores the keys in non-volatile memory encrypted in its TSEK, and destroys the token-generated copy of the KEK.

8. Support to User Applications

Upon completion of the loading of *User Keys* at the Point of Issuance, or at the user's workstation, the token is a fully initialized cryptographic token. It may assume the Operational Power-On State, as shown in Figure 1 and described further in the DoD PKI and KMI Token Protection Profile and Reference A, and it may be used operationally to provide security services in support of user applications.

In an operational scenario, the token would be inserted into a compatible reader (such as a smart card reader) that is either integral to the host workstation or an installed peripheral device. Depending on the design of the supported application, either the token or the host would initiate communication with the other party. A communication channel is established between the token and the host that is logically distinct from other host communication channels. Upon successful establishment of the communication channel, the token shall have the capability to

mutually generate a session key with the supported host using its private/public key pair for key establishment (FCS_CKM.1, FTP_ITC.1). If the host similarly supports session key generation with the token, then both the token and the host shall authenticate each other, generate the session key, and encrypt all subsequent data exchanged between them using that session key as a traffic encryption key (FCS_CKM.1, FCS_CKM.2, FTP_ITC.1). The certificate for the token-unique key establishment key pair would constitute the token's identification and authentication data. The ability to successfully establish such a session key implies mutual trust between the token and the host.

For applications where the host does not support mutual authentication and session key generation, the token shall operate either in a non-authenticated, non-secure channel, or in a mode in which the token provides identification or authentication data to the host upon demand, but does not receive any identification data or attempt to authenticate the host (KMI requirement). The token shall be capable of providing all requisite security services needed by the application without requiring host authentication or a secure channel (KMI requirement).[14]

A session key, when used, shall be destroyed by both parties upon removal of the token from its reader, upon shutdown or closure of the supported host application, or upon a power failure (FCS_CKM.4, FTP_ITC.1). A new session key will be generated upon commencement of a new session.

Once secure or non-secure communication with a host is established, the token must then respond to legitimate commands from the host or an application to provide the specific keys needed for supported security services (KMI requirement). Some typical services that might be supported include:

- Signing e-mail or other transactions.
- Encrypting e-mail or other transactions.
- Verifying the signature on a received e-mail or other transactions (validating the certificate path).

[14] Ultimately, the intent is to counter potential security threats such as eavesdropping, or substitution of a bogus token or a bogus host, by providing a mutually authenticated, secure channel for all communications between a token and a host. Implementation of mutual authentication and a secure channel isn't realistic in the near term, however, because it would require every DoD host computer to possess its own key and certificate to be used for that purpose. The DoD PKI would have to provide these keys and certificates, as well as the corresponding certificate management services. It is likely that the keys and certificates would be software based (Class 3), since they are to be used by an untrusted host that would not meet DoD Class 4 certificate policy for protecting user private keys. The current requirements for the DoD Class 4 PKI do not include provision of Class 3 certificates, so either the requirement would have to be added to the Class 4 PKI or the current Class 3 PKI would have to continue to be operated to provide them. The potential number of Class 3 certificates needed would be huge, given the number of computers used by DoD that would be expected to accept a token. Since the impact of a requirement for mutual authentication and secure channels could be so significant, the requirement is stated such that it can be implemented in a phased approach. The token must support mutual authentication and establishment of a secure channel, and use it with any host that supports it, but the token does not require its host to have the capability.

- Decrypting received e-mail or other transactions.
- Enabling user log-on to an external network or system.
- Controlling access to data or applications on either the host or the token.
- Verifying authorizations or privileges of the user by certificate content.

The keys used by the token to support a security service are called by their tags or identifiers from non-volatile storage into RAM and are decrypted by the TSEK. Any key component or data sent to the host device is re-encrypted in the session key, if one is used.

A token must zeroize any unencrypted key in RAM or other volatile storage upon a loss of power or detection of a tamper attempt (FCS_CKM.4). Also, while in support of an application, the token must accept a command from the host to zeroize specific *User Keys* or to zeroize all keys (FCS_CKM.4). In the event of a loss or compromise of the host device while the token is removed from it, the token must accept zeroize commands from another KMI Manager workstation or POI device (FCS_CKM.4).

9. Key Loading after Initial Issuance

Addition, deletion, or maintenance of *Token* Keys or *User Keys* on a token after initial issuance must be supported (FCS_CKM.1, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4). References B and C, as well as other KMI documents, discuss the following types of maintenance actions for public keys and certificates:

Rekey - An action that generates new private/public key pairs, and provides a new certificate serial number and validity period, but does not otherwise change the content of the certificate. Reference C requires a rekey of Class 4 signature and key establishment certificates every three years, and requires the human user to verify his identity in person.

Renew - An action that creates a new certificate with the same identification, authorizations, and key pairs as the old one, but provides a new serial number and a new validity period. The KMI will not support renewal of certificates in the implementation of its first capability increment, so this service would not be available until at least 2005.

Update - An action that changes the data for identification or authorization in a certificate, assigns a new serial number, and may or may not replace the keys.

Any action to provide a user with additional keys and certificates after initial issuance of a token, or to change the identification or authorization data in a certificate, must be authorized by a KMI Manager who is enrolled in the KMI with appropriate privileges (FDP_ACC.1, KMI requirement). In most cases the manager would initiate the request to the KMI. A request for such products also could be initiated by an individual user from his own KMI-registered workstation, but the KMI would not generate and deliver the requested product until the requisite manager approval is obtained. The manager, or the user himself, sends a request to the KMI for the key/certificate, requesting that it be made available for a specific token. The same four scenarios that are described for the initial issuance of the token *User Keys* could still apply. The

private/public key pairs for the additional keys could be generated by either the KMI or the token, and delivery of the keys could occur in the same session or at a later time. For the scenario in which a user requests keys and certificates from his own workstation that require KMI Manager approval, a second session for delivery is required. Delivery during the same KMI session in which the request is submitted could only be accomplished by returning the token to a KMI Manager workstation (which need not necessarily be the POI for initial issuance). To support secure delivery of the product from the KMI to the token in any scenario, the token must be capable of creating a new Token KEK Establishment Response as described in Section I.7.B (KMI requirement).

The KMI will support user discretionary rekeys of *Token Keys* or *User Keys*, but because of the policy requiring a user to appear physically and re-verify his identity every three years, the KMI would not normally extend the validity period of the certificate. A rekey which extends the validity period beyond the expiration of the three-year window would have to be initiated by a KMI Manager.

Part II. Key Management Requirements for the DoD PKI and KMI Token

1. Token Serialization

In the DoD PKI and KMI Token PP, Section 5.2.1.2, the Generation support and acceptance procedures (ACM_CAP.4) requirement states:

Developer Action Elements:

ACM_CAP.4.1D
The developer shall provide a reference for the TOE. **Each unique copy of the TOE shall have its own serial number. Additional requirements are found in the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 1 (see Appendix F).**

The additional requirements are:

A. Each KMI-compatible token shall have a unique serial number. The serial number must be permanent and electronic. Any externally printed serial number must be the same as the token's electronic serial number.

B. There shall be a management process that prevents duplication of serial numbers across different types of tokens, or for compatible tokens provided by different vendors.[15]

2. Key Storage

In the DoD PKI and KMI Token PP, Section 5.1.5.3, the Cryptographic key access (FCS_CKM.3) requirement states:

FCS_CKM.3.1
The TSF shall perform encryption of cryptographic keys in nonvolatile memory in accordance with a specified cryptographic key access method, cryptographic key storage, that meets the

[15] KMI maintains a data base that associates token serial numbers with the public component of a Token Unique Key, and with the identifiers of keys provided to the token. As KMI and token requirements are currently stated, there appears to be no technical barrier to duplication of serial numbers, and it appears possible that a token with a duplicate serial number could inadvertently become initialized. KMI would be unaware of the existence of two identically numbered tokens, would associate the token serial number with the most recently generated public component of a TUK, and then would accept transactions with the most recently initialized token, while rejecting the first token.

following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 2 (see Appendix F).

The additional requirements are:

A. A token shall store at least two root CA certificates, at least one of which shall be loaded at the token production or initialization facility (factory or depot).

B. A token shall support the addition of another root CA certificate, and replacement of the factory- or depot-loaded root CA certificate(s), after completion of factory or depot initialization. (See footnote 1 in Section I.3.B.)

C. A token shall store at least one Token Unique Key. The unique key shall be an asymmetric private/public key pair, with associated certificate, that is used for interaction with the KMI and the supported host workstation or application. It is used to authenticate the token to the KMI or the host, and to establish symmetric key encryption keys for subsequent downloads of additional keys from KMI or for session keys with the host.[16]

D. A token shall store a symmetric Token Storage Encryption Key in a split form protected by a PIN, password, or other unique user data.

E. A token shall store additional User Keys as required by the application it supports. The minimum requirement is one general purpose signature/identification key/certificate and one e-mail encryption key/certificate.

F. A token shall store unique key identifiers, such as key tags or certificate serial numbers, with all stored keys, by which keys can be identified and accessed.

G. A token shall provide non-volatile storage for all root CA certificates, Token Unique Keys, and User Keys. Cryptographic keys in non-volatile storage shall be encrypted in the Token Storage Encryption Key, and the key access method shall meet the requirements of FIPS 140-2 Level 2 for user tokens and FIPS 140-2 Level 3 for Registration Authority and Certification Authority tokens. See the requirements section of this protection profile, "Cryptographic Key Access (FCS_CKM.3)."

H. The token must hold and protect the keys needed to perform the requisite cryptographic operations for its user, i.e., *User Keys*.

[16] One token unique key is the minimum essential requirement. Unless there are severe memory constraints, it is expected that a token would have two different unique asymmetric key pairs, one used for signature and the other for key establishment. In addition, in the near term there may be token types that do not support asymmetric key pairs at all, and they would need to operate with a symmetric key as the token unique key.

I. The token must be able to store various sets of private/public keys and corresponding certificates, as well as symmetric keys, to support its user for the following purposes:

- General purpose signature/identification.
- E-mail signature.
- E-mail encryption.
- Log-on for special applications to a host device, system, or network.
- KMI Manager role(s).
- Organization/group roles.
- Other token applications (e.g., access control to a medical data applet on the token).

J. The token also must have a set of key management functions that provide the capability to securely load, store, and maintain *User Keys* on the token. This includes certificate processing and path validation to verify the source of the *User Keys* during key load operations, protecting keys in storage, and providing confidentiality to *User Keys* during transport from the KMI to the token.

K. Access to keys encrypted in the TSEK must meet the requirements of FIPS 140-2 Level 2 for user tokens or FIPS 140-2 Level 3 for Registration Authority and Certification Authority tokens.

L. The token and the key loading process also must allow for subsequent loading of *User Keys* onto the token at a user workstation or another KMI Manager workstation after initial issuance by the POI.

M. There must be only one such KEK active at any given time for a particular token, and this KEK may be used multiple times during the key load session to encrypt each key/certificate sent to the token. The KEK would be destroyed by both parties at the conclusion of the key load session.

N. Addition, deletion, or maintenance of *Token* Keys or *User Keys* on a token after initial issuance must be supported.

O. All *Token Keys* and *User Keys* loaded onto a token must be labeled with a unique key identifier or a certificate serial number.

3. Source Authentication

A. A new, unkeyed token that is in possession of a properly loaded and verified key management software applet shall accept a root DoD CA certificate from a factory or depot initialization system.

B. Subsequent to the successful load of a root DoD CA certificate, the token shall accept only key loads in which the source of the key is verified to be a Certification Authority whose

certificate is traceable to the DoD root. The token shall have the capability to perform certificate processing and path validation for any key/certificate subsequently loaded.

4. Key Generation

In the DoD PKI and KMI Token PP, Section 5.1.5.1, the Cryptographic key generation (FCS_CKM.1) requirement states:

FCS_CKM.1.1
The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm from list of required supported cryptographic algorithms in Appendix E and specified cryptographic key sizes of

- at least 160 bit private key with at least 1024 bit prime modulus for Digital Signature Standard keys;
- at least 1024 bit public key for Key Exchange Algorithm (KEA);
- at least 2048 bit public key for RSA;
- at least 384 bit for Elliptic Curve Digital Signature Algorithm key prime field (//p//);

that meet the following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities, X.509 Certificate Policy, and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 4 (see Appendix F).

Application notes:
- Throughout the requirements in this PP, references are made to requirements for FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities. If the DoD Common Access Card issuing infrastructure is not capable of issuing two different levels of cards, then all CACs will be required to meet FIPS 140-2 Level 3.

The additional requirements are:

A. A token shall have the capability to generate its own symmetric Token Storage Encryption Key and use it to encrypt other Token Unique Keys and User Keys to be stored in non-volatile storage.

B. A token shall have the capability to generate private/public key pairs for Token Keys or User Keys upon command of an initialization facility, a token Point of Issuance system, a KMI Manager workstation, or a host workstation.

(1) Algorithms used for key generation shall be those listed as required supported cryptographic algorithms in Appendix E.
(2) Key lengths shall be as specified in the requirements section of this protection profile, "Cryptographic Key Generation (FCS_CKM.1)."
(3) The private key of any key pair generated by a token for signature shall be unchangeable and shall never leave the token.
(4) Keys generated for channel sessions with the host must be composed of components created by both the token and the host.

(5) The asymmetric key generation process must create its own prime numbers -- i.e., it must not rely on replaced values.

(6) A hardware randomization source is required for all key generation.

C. If a token has the capability to generate a symmetric Key Encryption Key for use in downloading keys from the KMI, the KEK generation process shall also meet all applicable requirements of Paragraph 4.B.

D. A token shall support a data exchange with the KMI for the purpose of cooperative generation of a symmetric Key Encryption Key (KEK) to be used for download of Token Keys or User Keys from KMI to the token. The data exchange may occur through an intermediary key issuing system using an indirect, store-and-forward process. The token shall have the capability to generate the KEK, using an algorithm specified in Appendix E, upon receiving the appropriate data from the KMI. Only one such cooperatively generated KEK shall be active at any time, and it shall be destroyed by the token at the completion of the key loading session.

E. The token must also support addition, deletion, and replacement of *User Keys* at anytime during its lifetime following initial issuance.

F. Token Unique Keys must be loaded from a trusted external source, or created on the token itself at the factory or a secure facility.

G. Token Unique Keys may be provided by the KMI or may be generated by the token itself. The KMI, the token, and the initialization facility must support both options.

H. Each private/public key set must be unique, have a unique identification number, and be signed by a CA signature that is traceable to the DoD Root CA.

I. Since all key material delivered from the KMI to a token must be encrypted specifically for the token, the token must support cooperative generation of a symmetric Key Encryption Key with the KMI.

J. The sequence of steps in the key loading process, and the handling of key management data, may vary depending on the availability of the KMI and on where the private/public key pairs for *User Keys* are to be generated (by the KMI or by the token itself). There are four scenarios that must be supported: 1) KMI available and KMI generates private/public key pairs for *User Keys*; 2) KMI available and token generates private/public key pairs for *User Keys*; 3) KMI not on-line, but KMI is to generate private/public key pairs for *User Keys*; 4) KMI not on-line, and token is to generate private/public key pairs for *User Keys*.

K. Upon successful establishment of the communication channel, the token shall have the capability to mutually generate a session key with the supported host using its private/public key pair for key establishment.

L. If the host similarly supports session key generation with the token, then both the token and the host shall authenticate each other, generate the session key, and encrypt all subsequent data exchanged between them using that session key as a traffic encryption key.

M. Addition, deletion, or maintenance of *Token* Keys or *User Keys* on a token after initial issuance must be supported.

5. Key Destruction

In the DoD PKI and KMI Token PP, Section 5.1.5.4, the Cryptographic key destruction (FCS_CKM.4) requirement states:

FCS_CKM.4.1
The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method, zeroization, that meets the following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 5 (see Appendix F).

The additional requirements are:

A. Loss of power to the token, or detection of a tamper attempt, shall result in the zeroization of any unencrypted keys in RAM or other volatile storage. Upon zeroization, the token shall enter the Totally Locked State as described in the DoD PKI and KMI Token Protection Profile, and it can then be unlocked and restored to service only by an authorized SSO. Zeroization of encrypted keys in non-volatile storage is not required in event of a power loss or a tamper attempt.

B. A token shall accept and execute a command from a host workstation, a KMI Manager workstation, a token Point of Issuance system, or other authorized source to zeroize any specific stored User Key or all stored keys (i.e., those in non-volatile storage encrypted in the TSEK as well as any unencrypted key in RAM). A zeroize command for all stored key shall result in the complete destruction of all Token Keys and User Keys and certificates held on the token and a return to the Unkeyed State. The token is then unusable until it is again initialized in the factory or depot initialization process.

C. Key destruction shall be accomplished in accordance with FIPS 140-2 Level 2 for user tokens and FIPS 140-2 Level 3 for Registration Authority and Certification Authority tokens. See the requirements section in this protection profile, "Cryptographic Key Destruction (FCS_CKM.4)."

D. The token must also support addition, deletion, and replacement of *User Keys* at anytime during its lifetime following initial issuance.

E. A session key, when used, shall be destroyed by both parties upon removal of the token from its reader, upon shutdown or closure of the supported host application, or upon a power failure. A new session key will be generated upon commencement of a new session.

F. A token must zeroize any unencrypted key in RAM or other volatile storage upon a loss of power or detection of a tamper attempt.

G. While in support of an application, the token must accept a command from the host to zeroize specific *User Keys* or to zeroize all keys.

H. In the event of a loss or compromise of the host device while the token is removed from it, the token must accept zeroize commands from another KMI Manager workstation or POI device.

I. Addition, deletion, or maintenance of *Token* Keys or *User Keys* on a token after initial issuance must be supported.

6. Key Distribution

In the DoD PKI and KMI Token PP, Section 5.1.5.2, the Cryptographic key distribution (FCS_CKM.2) requirement states:

FCS_CKM.2.1
The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method encryption with key exchange keys for symmetric keys in a DoD Authenticated State that meets the following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 6 (see Appendix F).

Application note: Possession of the Key Exchange Key authenticates the host to the TOE.

The additional requirements are:

A. A token shall automatically validate the signature on any downloaded key package. Upon a failure to validate a signature traceable to a DoD Root CA, the token shall reject the key package and provide a notification to the user, via its host system or workstation.

B. Upon successful completion of a Token Key loading operation, the token shall provide to the factory initialization system a receipt which contains the token serial number and the unique identifier of the loaded key, signed in the token's signature key.

C. During a User Key loading operation, upon the command of the issuing system, a token shall provide its serial number, its public key establishment key, and a token-generated random value, all signed in the token's signature key.

D. A token shall accept a delivered User Key package from the KMI encrypted in the mutually generated KEK. The token shall decrypt the package, validate that the key came from an authorized source, and store the downloaded keys in non-volatile storage encrypted in the TSEK. The key distribution shall satisfy the requirements of FIPS 140-2 Level 2 for user tokens and FIPS 140-2 Level 3 for Registration Authority and Certification Authority tokens. See the requirements section in this protection profile, "Cryptographic Key Distribution (FCS_CKM.2)."

E. A token shall support a "staged" key loading scenario in which it provides requested information to an intermediary (which in turn requests key from the KMI); temporarily stores the required key establishment data in non-volatile storage, encrypted in the TSEK; and then later accepts delivery of the KMI product via either the same or a different intermediary.

F. A token shall accept addition, deletion, or replacement of any Token Key or User Key at any time subsequent to initial issuance of User Keys by a token Point of Issuance.

G. All key material delivered from the KMI to a token must be encrypted in such a manner that only the token can decrypt it, even if the material consists only of a public key and/or a certificate.

H. If the host similarly supports session key generation with the token, then both the token and the host shall authenticate each other, generate the session key, and encrypt all subsequent data exchanged between them using that session key as a traffic encryption key.

I. Addition, deletion, or maintenance of *Token* Keys or *User Keys* on a token after initial issuance must be supported.

7. Cryptographic Operation

In the DoD PKI and KMI Token PP, Section 5.1.5.2, the Cryptographic operation (FCS_COP.1) requirement states:

FCS_COP.1.1
The TSF shall perform signing e-mail hash values and wrapping or unwrapping e-mail session keys in accordance with a specified cryptographic algorithm from a list of required supported cryptographic algorithms in Appendix E and cryptographic key sizes of

- at least 160 bit private key with at least 1024 bit prime modulus for Digital Signature Standard keys;
- at least 1024 bit public key for Key Exchange Algorithm (KEA);
- at least 2048 bit public key for RSA;
- at least 384 bit for Elliptic Curve Digital Signature Algorithm key prime field (//p//);

that meet the following: FIPS 140-2 Level 2 for Subscribers/Level 3 for Registration Authorities and Certificate Authorities and X.509 Certificate Policy and the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 7 (see Appendix F).

The additional requirements are:

A. All data exchanged between a token and its host shall be encrypted in a session key shared between the token and the host. Successful sharing of a session key implies the host is trusted by the token. See the requirements section in this protection profile, "Trusted Path/Channels (FTP) Requirements."

B. A token shall accept a command from its host device, a supported application, a KMI Manager workstation, or other authorized source, to identify the keys it holds and the content of the associated certificates.

C. A token shall accept a command from its host device or a supported application to provide a specific key for use in a supported cryptographic function.

D. A token shall perform signing e-mail hash values and wrapping or unwrapping e-mail session keys in accordance with a specified cryptographic algorithm from a list of required supported cryptographic algorithms specified in Appendix E and and key lengths specified in the requirements section in this protection profile, "Cryptographic Operation (FCS_COP.1)."

8. User Data Changes by KMI Manager

In the DoD PKI and KMI Token PP, Section 5.1.6.1, the Subset access control (FDP_ACC.1) requirement states:

FDP_ACC.1.1
The TSF shall enforce the SFP.DAC on:
Subjects: SSO, DoD users, non-DoD users;
Objects: Authentication data, personalization data, and initial security data,

- objects in DoD directory: root certificate, user certificate, user private key, directories, applications;
- objects in non-DoD directory: root certificate, user certificate, user private key, directories, applications; and

operations among subjects and objects covered by the SFP. Additional requirements are found in the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 8 (see Appendix F).

The additional requirement is:

A. Any action to provide a user with additional keys and certificates after initial issuance of a token, or to change the identification or authorization data in a certificate, must be authorized by a KMI Manager who is enrolled in the KMI with appropriate privileges.

9. Host Communications

In the DoD PKI and KMI Token PP, Section 5.1.11.1, the Inter-TSF trusted channel (FTP_ITC.1) requirement states:

FTP_ITC.1.1

The TSF shall provide a communication channel between itself and a remote trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of channel data from modification or disclosure. **Additional requirements are found in the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 9 (see Appendix F).**

Application note: As detailed in the Key Management and SSO Authentication Specification, all DoD data transmitted between the TOE and a host will be encrypted with a session key shared between the host and the TOE. Successfully sharing a session key implies the host is trusted. A session key, when used, shall be destroyed by both parties upon removal of the token from its reader, upon shutdown or closure of the supported host application, or upon a power failure. A new session key will be generated upon commencement of a new session.

The additional requirements are:

A. Upon successful establishment of the communication channel, the token shall have the capability to mutually generate a session key with the supported host using its private/public key pair for key establishment.

B. If the host similarly supports session key generation with the token, then both the token and the host shall authenticate each other, generate the session key, and encrypt all subsequent data exchanged between them using that session key as a traffic encryption key.

C. A session key, when used, shall be destroyed by both parties upon removal of the token from its reader, upon shutdown or closure of the supported host application, or upon a power failure. A new session key will be generated upon commencement of a new session.

10. TOE Protection During Delivery

In the DoD PKI and KMI Token PP, Section 5.2.2.1, the Detection of modification (ADO_DEL.2) requirement states:

Developer Action Elements:

ADO_DEL.2.1D
The developer shall document procedures for delivery of the TOE or parts of it to the user. **Additional requirements are found in the Key Management Specification for the DoD PKI and KMI Token, Part II, Section 10 (see Appendix F).**

The additional requirement is:

Some level of protection must be afforded to tokens during shipment.

Appendix G: Comparison to the SCSUG's PP

The table below illustrates how SCSUG PP threats were addressed in the DoD PKI and KMI Token PP. Many of the SCSUG PP's threats were included in the Token PP with some or no modification. Additional threats added to the Token PP to cover threats identified by the DoD are listed after the table.

Table G-1 Treatment of SCSUG Threats

SCSUG Threat	How Treated
T.P_Probe	Unchanged
T.P_Modify	Unchanged
T.E_Manip	Unchanged
T.Flt_Ins	Unchanged
T.Forcd_Rst	Changed to T.Forced_State_Change; covers state changes rather than reset
T.Inv_Inp	Unchanged
T.Load_Mal	Not included, covered by T.Bad_Load
T.Reuse	Not included, covered by T. Crypt_Attk and secondary threat to T.Hacker_Comm_Eavesdrop
T.Search	Not included, covered by T.Rep_Atk
T.UA_Load	Renamed T.UA_Use
T.Access	Covered by SFP.DAC
T.First_Use	Unchanged
T.Impers	Unchanged
T.App_Ftn	Unchanged
T.LC_Ftn	Unchanged
T.Res_Con	Unchanged
T.Crypt_Atk	Added cryptanalysis
T.I_Leak	Unchanged
T.Link	Unchanged
T.Env_Strs	Unchanged
T.Lnk_Att	Unchanged
T.Rep_Atk	Unchanged
T.Clon	Unchanged
T.Carrier_Tamper	Not included, covered by T.P_Modify
T.Priv	Added verbiage at the end, changed name to T.Privilege (due to mod)

New Token PP Threats

New threats added to cover threats identified in the *Token Security Requirements* document:

T.Bad_Load
T.Component_Fail
T.Developer_Flawed_Code
T.Disable Security
T.Fail_Secure
T.Hacker_Comm_Eavesdrop
T.Hacker_Social_Engineer
T.Spoof

New threat that covers a SCSUG PP assumption:
T.Power_Clock

Comparison of Requirements

Requirements listed in the DoD PKI and KMI Token PP that are not in the SCSUG PP:

FCS_CKM.2 — Cryptographic key distribution

Signature and session keys generated by the token need to be distributed.

FCS_CKM.4 — Cryptographic key destruction

Old keys must be destroyed. FCS_CKM.1 and FCS_CKM.2 are dependent on this.

FDP_DAU.1 — Basic data authentication

The integrity of stored data must be verified.

FDP_IFF.3 — Limited illicit information flows

Prevents leakage over input or output connections.

FIA_SOS.1 — Verification of secrets

Specifies the strength of authentication.

FIA_UID.2 — User identification before any action

Instead of FIA_UID.1. We require identification before performing any actions.

FMT_SMR.2 — Security roles

Defines security roles.

FMT_SMR.3 — Assuming roles

Assuming the SSO role on the TOE requires a request from the SSO.

FPT_AMT.1 — Abstract machine testing

Used for self-test.

FPT_PHP.1 — Passive detection of physical attack

Use of coatings, etc., that provide evidence of tampering.

Requirements listed in the SCSUG PP that are not in the DoD PKI and KMI Token PP:

FAU_ARP.1 — Security alarms

The token will not respond to detection of potential security violations with an alarm. An audit list will not be generated based on activity on the TOE.

FAU_SAA.1 — Potential violation analysis

This is a dependency of FAU_ARP.1, which is also not included. The token will not apply a set of rules in monitoring audited events and indicate a potential security violation based on these rules.

FAU_SEL.1 — Selective audit

This requirement is not necessary. The TOE does not have audit requirements.

FDP_UIT.1 — Data exchange integrity

The spirit of this requirement is captured by FTP_ITC.1.

FPT_RCV.3 — Automated recovery without undue loss

This requirement is not necessary.

FPT_RPL.1 — Replay detection

This requirement is not necessary.

ADV_INT.1 — Modularity

SCSUG augmented EAL4 with this additional requirement. It is not necessary.

Appendix H: Threat Comparison

The table below compares the threats identified in the DoD paper entitled *Public Key Infrastructure Target Class 4 Token Security Requirements* (Draft version 1.01, April 10, 2000) to the threats in the DoD PKI and KMI Token PP. This table illustrates that the threats identified in the *Token Security Requirements* document have been reflected in the Token PP.

Table H-1 Threat Comparison Between TSRD and Token PP

TSRD Threat	Token PP Threat
Adversary finds token or steals token. Note: Most of the other threats assume this threat has been accomplished.	
Development/Implementation flaw allows circumvention of security mechanisms.	T.Developer_Flawed_Code T.Component_Fail T.App_Ftn
Unauthorized terminal requests sensitive information from token (adversary creates a bogus terminal).	T.Spoof
Unauthorized executable file on token violates security mechanisms.	T.Bad_Load T.Privilege
Authorized executable file on token violates security mechanisms.	T.Developer_Flawed_Code T.UA_Use
Adversary manipulates data on token.	T.UA_Use T.Forced_State_Change T.Crypt_Attk T.E_Manip T.Env_Stress T.Flt_Ins
Adversary attempts to physically extract data from the token or change data/code/hardware.	T.E_Manip T.P_Modify T.P_Probe T.Clon
Adversary uses electrical analysis to get information from the token.	T.P_Probe
Adversary places a tap on the cable between the host and the token (reader) that extracts sensitive data, or adversary replaces reader device with storage capability.	T.Hack_Comm_Eavesdrop T.I_Leak

Adversary attempts to use a stolen token.	T.Impers T.Hacker_Social_Engineer T.Rep_Atk
Adversary steals token user authentication data.	T.Hacker_Social_Engineer T.Hacker_Comm_Eavesdrop
Adversary steals card prior to initialization and attempts to create a token.	T.First_Use
Adversary attempts to steal private keys during a key initialization or update of the keys on the token.	T.P_Probe T.Hacker_Comm_Eavesdrop
Adversary exploits a failure of a security function on the token hardware or software.	T.Component_Fail T.Developer_Flawed_Code T.Env_Strs T.Fail_Secure
Adversary finds a method of successfully circumventing a security mechanism of the token.	T.Link T.Lnk_Att T.P_Modify T.Bad_Load T.Privilege
Adversary attempts to change a configuration file.	T.P_Modify T.Env_Strs T.E_Manip T.Flt_Ins

Appendix I: Smart Card Vulnerabilities

Vulnerabilities in smart cards exist at the physical level ("the silicon"), logical level ("card operating system"), and organizational level ("transport, initialization, and implementation"). Vulnerabilities associated with microprocessor-based smart cards are listed below.

Physical Level:

- The smart cards derive their power from external sources. As a result, security functions are not always active, which results in a lack of active fraud detection measures.
- Data could be read or inserted on the data bus on a chip using microscopic probes.
- The chip structure could be reverse engineered using scanning electron microscope (SEM).
- SEM could also be used to visualize voltages on the chip surface, which would then be used to thoroughly understand the functionality of the chip.
- Data on the smart cards could be read using superconducting quantum interference devices (SQDIS), electrical testing, and electron beam testing.
- Other attacks, such as UV or X-rays or high temperatures, could cause erasure of memory. However, erasure of selected bits is not allowed without disabling the card.
- Physical parameters available outside the chip could be used to spoof or tamper data on the EEPROM, RAM, or even ROM. Some of the attacks used in the industry include Differential Power Analysis (DPA), Simple Power Analysis (SPA), and radiation.
- Physical parameters could also be used to change program flow or change data on EEPROM, ROM, and even RAM. Some of the techniques used in the industry include glitching—inserting spikes on power, clock, reset, and I/O lines, and voltage manipulation.

Logical Level:

- Due to advances in the semiconductor industry, operating systems (OS) are evolving significantly. Hence, the secure OS today may be primitive in nearly a year.
- Availability of hidden or unspecified commands could cause the OS to expose unauthorized data.
- Incorrect implementation of commands could produce unexpected and unintended results.
- Inappropriate use of cryptography would result in insecure data on the card.
- Due to the lack of memory partitioning, the operating system should ensure that each application is separately protected.
- Use of static authentication, as opposed to dynamic authentication, does not provide security against card counterfeiting.
- Many times security is obtained by obscurity. For example, vendors may try to hide security holes by not revealing the test results or the architecture of their operating system.

Organizational Level:

- Systems and applications do not provide or use the functionality necessary to implement good security. For example, the OS may support multiple access level; however, the card issuer does not protect files using this feature.

- From the time the chip is created to the time it is deployed to the end-users, the chip changes many hands. The weakest link in the transport of this chip could cause a breach in smart card security.
- Many applications are implemented so poorly that compromise of one card could compromise the entire system.

Reference List:

1. Ayer, Ken. "Risk Management & Smart Cards." In *CardTech/SecurTech '99 Gateway to Practical Innovation*. Chicago: 1999.
2. Bovelander, Ernst. "Evaluations of Smart Card Based Security Systems - Is Your Smart Card Really Secure?" In *CardTech/SecurTech '95 Strategies for the Millennium*. Washington, D.C.: 1995.
3. Bovelander, Ernst, and Jan Pieters. "A Structured Approach to Smart Card Security." In *CardTech/SecurTech '96 Applications in Action*. Atlanta: 1996.
4. Garon, Gilles. "Overview of Smart Card Security and Standards." In *CardTech/SecurTech '95 Strategies for the Millennium*. Washington, D.C.: 1995.
5. Johnson, Lance. "Smart Card Chip Security: Classification and Evaluation." In *CardTech/SecurTech '97 The Art of Implementation*. Orlando: 1997.
6. Kocher, Paul. "Differential Power Analysis and Problems in Applied Cryptography." In *CardTech/SecurTech '99 Gateway to Practical Innovation*. Chicago: 1999.
7. Krueger, Julie. "The Evolution of Personal Tokens in a More Secure Future." In *CardTech/SecurTech '96 Applications in Action*. Atlanta: 1996.
8. Pieters, Jan. "External Attacks: An Overview." In *CardTech/SecurTech '00 Sizzling Solutions for a Digital World*. Miami: 2000.
9. Russell, James. "Smart Card Security – Threats and Safeguards." In *CardTech/SecurTech '99 Gateway to Practical Innovation*. Chicago: 1999.
10. Schneier, Bruce. "Security Threats in Smart Card Systems." In *CardTech/SecurTech '99 Gateway to Practical Innovation*. Chicago: 1999.
11. Thomasson, Jean-Paul. "Developments and New Architectures for High Security Smartcard Applications." In *CardTech/SecurTech '96 Applications in Action*. Atlanta: 1996.
12. Thomasson, Jean-Paul. "Smartcards and Fraud." In *CardTech/SecurTech '95 Strategies for the Millennium*. Washington, D.C.: 1995.
13. Van Gimst, Pascal. "Functionality Stress Testing of Smart Cards." In *CardTech/SecurTech '00 Sizzling Solutions for a Digital World*. Miami: 2000.