AKCode Cryptographic Module

Anonymous Key Technology (AKT)

C++ Module (CPP)

and

Anonymous Key Technology (AKT)

Java Module (JAVA)

Non-Proprietary Security Policy

FIPS 140-2 Level 1

Document Version 1.9
Module Version 1.0.2

By: Lynn Spraggs, Ph.D., P.Eng
AKCode Inc.
http://www.akcode.com

Table of Contents

1. Background

1.1. Purpose

This is the non-proprietary FIPS 140-2 security policy for the Anonymous Key Technology ("AKT") software Cryptographic Module ("Module") developed by AKCode Inc. ("AKCode"). The AKCode Security Policy details the secure operation of the AKT C++ Module and the AKT JAVA Module as required in Federal Information Processing Standards Publication 140-2 (FIPS 140-2) as published by the National Institute of Standards and Technology (NIST) of the United States Department of Commerce.

1.2. References

For more information on the AKT C++ Module and the AKT JAVA module, please visit http://www.akcode.com. For more information on the FIPS 140-2 cryptographic module validation program and the validation process or information on NIST, please visit

- https://csrc.nist.gov/projects/cryptographic-module-validation-program

2. Overview

The AKT C++ Module is a software module packaged as a Dynamic-link Library (DLL) on Windows, or as a shared object (.so) on Linux/Solaris. The AKT JAVA Module is a software module packaged as a jar or cab container of class files that are dynamically loaded by other executables. The packaged library is for use on specified operating systems (see section 3.0). The AKT C++ Module and the AKT JAVA Module provide an intuitive, high-level approach that can be used to develop new secure applications, customize existing applications or by utilizing the supplied functionality to do much of the secure processing required in distributed systems today.

2.1. FIPS 140-2 Compliance:

The AKT C++ Module and the AKT JAVA Module were designed and implemented to meet FIPS 140-2 requirements. As such, there are no special steps required to ensure applications using the modules are FIPS 140-2 compliant. The functionality delivered with the AKT C++ Module and the AKT JAVA Module will allow most organizations to utilize the suites as validated.

2.2. Access:

The AKT C++ Module and the AKT JAVA Module authentication strength varies according to the particular method chosen for authentication. A validation of the authentication strength was not conducted at this time, thus AKCode makes no particular claim as pertains to the FIPS 140-2 identity or role based authentication requirements.

The AKT C++ Module and the AKT JAVA Module allow access to the functionality for operators using usernames, passwords and optional biometrics. Once access is granted, operators assume one of two primary roles: Crypto Officer (CO) or User. COs have access

to user management and key database management functionality. Users have access to general cryptographic functionality. The access control procedure is performed inside the AKT C++ Module and the AKT JAVA Module.

2.3. FIPS-Approved Algorithms:

The AKT C++ Module and the AKT JAVA Module provide confidentiality, integrity and message digest services. The The AKT C++ Module and the AKT JAVA Module natively supports the following algorithms: AES, SHA-1 and HMAC-SHA-1. The table following list the AKT NIST validated algorithms:

Table 1 FIPS-Approved Algorithms Table

Algorithm	CAVP Certificates	Modes/Methods	Usage
FIPS 197 AES	#3193 (C++), #3194 (Java)	256-bit AES-ECB Encrypt and Decrypt	All symmetric encryption
FIPS 180-4 SHS	#2640 (C++), #2641 (Java)	SHA-1	Hash for obfuscation
FIPS 198-1 HMAC	#C1669 (C++), #C1670 (Java)	HMAC-SHA-1	Integrity

2.4. Non-Approved Algorithms:

The following non-Approved algorithms are supported by the module. Any usage of one of the non-Approved and disallowed algorithms or services implicitly places the module in a non-Approved mode of operation:

Table 2 Non-Approved Algorithms Table

Algorithm	Modes/Methods	Usage	Allowed?
PRNG FIPS 186-2	256 bit random number generator	Deterministic keys	No
PPP Spraggs Key Establishment Algorithm[1]	32 byte encryption	Key exchange	Yes

2.5. Data Management:

Data required by the cryptographic module is maintained within the cryptographic boundary or it is stored in an encrypted state outside the module or it is entered through an input device. All keys remain inside the AKT C++ Module or the AKT JAVA Module or they are imported or exported in an encrypted state. All encryption and/or decryption of data is performed within the cryptographic boundary.

[1] This algorithm is allowed until June 30, 2022 per FIPS 140-2 IG D.8 scenario 4 and appears on the certificate as "Diffie-Hellman (key agreement; key establishment methodology provides 112 or 128 bits of encryption strength)". This algorithm is a Diffie-Hellman-like algorithm that is a special case of Shamir's Three-Pass Protocol, with the KDF being the identity transform (i.e.: the shared secret is directly used as the key).

Table 3 DEK Management Table

KEY	Description/Usage	Generation	Storage	Entry/Output	Destruction	Establishment
Enterprise Server DEK 256 bit AES	Key used to encrypt or decrypt all data being secured for transmission outside the boundary.	None (External)	Stored in plaintext in SRAM	Imported (plaintext) through API.[2] Not exported.	Destroyed if used in SRAM	N/A
Application Server DEK 256 bit AES	Key used to encrypt or decrypt all data being secured for transmission outside the boundary.	None (External)	Stored in plaintext in SRAM	Imported (plaintext) through API. Not exported.	Destroyed if used in SRAM	N/A
Operator DEK 256 bit AES	Key used to encrypt operator transmissions	None (External)	Stored in plaintext in SRAM while user is active	Imported (plaintext) through API Output encrypted (AES-ECB) with the PPP KEK through API	Is destroyed after closure	N/A
PPP KEK 256 bits AES	Key used to encrypt operator key during key distribution.	None (External)	Stored in plaintext in SRAM	Imported (plaintext) through API	Is destroyed after usage.	PPP Key Establishment Protocol[3]
PPP Public and Private Key Pairs 2048, 3072 bits	These are the exponents (public & private) and modulus (public) used in the PPP key establishment protocol	None (External)	Stored in plaintext in SRAM Stored in plaintext in module binary	Not Entered or Output	Destroyed if used in SRAM Procedurally for persistent copies	Pre-loaded

[2] For all items in this table noted as being imported or exported in any way through the API, it falls under FIPS 140-2 IG 7.7 scenario "CM Software to/from App Software via GPC INT Path", thus they are N/A and not considered a form of Key Entry or Establishment under FIPS 140-2 IG D.2

[3] The PPP KEK is the direct output from the PPP Key Establishment Protocol in that the KDF is the identity transform. The PPP KEK is the shared secret that is established and it is converted to bytes to be directly used as an AES key. Note that both keys with the 'PPP' prefix are associated with the Diffie-Hellman-like algorithm described in section 2.4.

3. Security Specification

This document specification describes the AKT C++ Module and the AKT JAVA Module version 1.7 Security Policy submitted for validation in accordance with the FIPS publication 140-2. It is implemented as a multi-chip stand-alone application.

AKT C++ Module and the AKT JAVA Module consist of the following generic components:

- A commercially available general-purpose hardware-computing platform that is shown schematically in Figure 2.0. This platform must conform to FCC regulations.
- A commercially available Operating System (OS) that is consistent with and runs on the above platform. For the purposes of this validation only Windows 7 has been fully and formally tested and is listed on the certification. Windows NT, Windows XP, Windows 10 and Windows 2000 are all vendor-affirmed platforms that are covered by this certification as well, but with the following caveat:
    - *"No claim can be made as to the correct operation of the module or the security strengths of the generated keys when ported to an operational environment which is not listed on the validation certificate."*

The AKT C++ Module has been operationally tested on the following hardware computing platform and Operating Systems:

Microsoft Windows 7 (x86-64) running on an Acer Aspire with an Intel Core i5
Microsoft Windows 7 (x86-64) running on a Dell Inspiron with an Intel Core i5
Microsoft Windows 7 (x86-64) running on a Gateway DX4860 with an Intel Core i5

The AKT JAVA Module has been operationally tested on the following hardware computing platform and Operating Systems:

Microsoft Windows 7 (x86-64) running on an Acer Aspire with an Intel Core i5
Microsoft Windows 7 (x86-64) running on a Dell Inspiron with an Intel Core i5
Microsoft Windows 7 (x86-64) running on a Gateway DX4860 with an Intel Core i5

Table 4 – Module Security Level Specification

Security Requirements Section	Level
Cryptographic Module	**1**
Module Interfaces	**1**
Roles and Services	**1**
Finite State Model	**1**
Physical Security	**N/A**
Operational Environment	**1**
Key Management	**1**
Cryptographic Algorithms	**1**
EMI/EMC	**3**
Self Test	**1**
Design Assurance	**1**
Mitigation of Other Attacks	**N/A**

3.1. Cryptographic Boundary and Physical Interfaces

The AKT C++ Module and the AKT JAVA Module is a software cryptographic system designed to comply with FIPS 140-2 Level 1 for a single user stand-alone cryptographic module. Single user mode is configured on the host PC at installation which enforces this. The AKT C++ Module and the AKT JAVA Module have an embodiment which is "Multi-chip Standalone".

The physical cryptographic boundary is the host PC and the address space of the process within which the AKT C++ Module or the AKT JAVA Module is loaded, the logical boundary, is the software (uiscm.dll and uiscm.bin for the AKT C++ Module and uiscm.jar for the AKT Java Module).

Figure 1.0 provides a schematic of the cryptographic boundary and Figure 2.0 illustrates the AKT C++ Module and the AKT JAVA Module resident in the hardware. The hardware interfaces are defined to be:

3.1.1 Interfaces into the Cryptographic Boundary

- Keyboard Input Device
- PAD storage devices
- Wireless Storage Device

3.1.2 Biometrics Input Device(s)

- Fingerprint Reader
- Keyboard Cadence

3.1.3 Interfaces Out of the Cryptographic Boundary

- Standard PC Ports
- TCP Port

3.1.4 Interfaces that are both Input and Output

- Smart Card Reader/Writer
- USB storage device
- Database Storage Device
- Wireless Reader/Writer

Being a software application, the AKT C++ and the AKT JAVA Module defines their logical interfaces in terms of the API that it provides. The Data Input Interface is defined to be all the API calls that accept, as their arguments, data to be used or processed by the AKT C++ Module and the AKT JAVA Module. Data Output Interfaces are defined to be the API calls that return, by means of return value or arguments of appropriate types, data generated or otherwise processed by the AKT C++ Module and the AKT JAVA Module to the caller. Finally, Control Input Interfaces are comprised of the calls used to initiate the AKT C++ Module and the AKT JAVA Module and the API calls used to control their operation. The Status Output Interface is defined as special API calls that provide information about return

values and the status to the requesting user. A functional block diagram is shown in Figure 3.0.

Figure 1 AKT Module

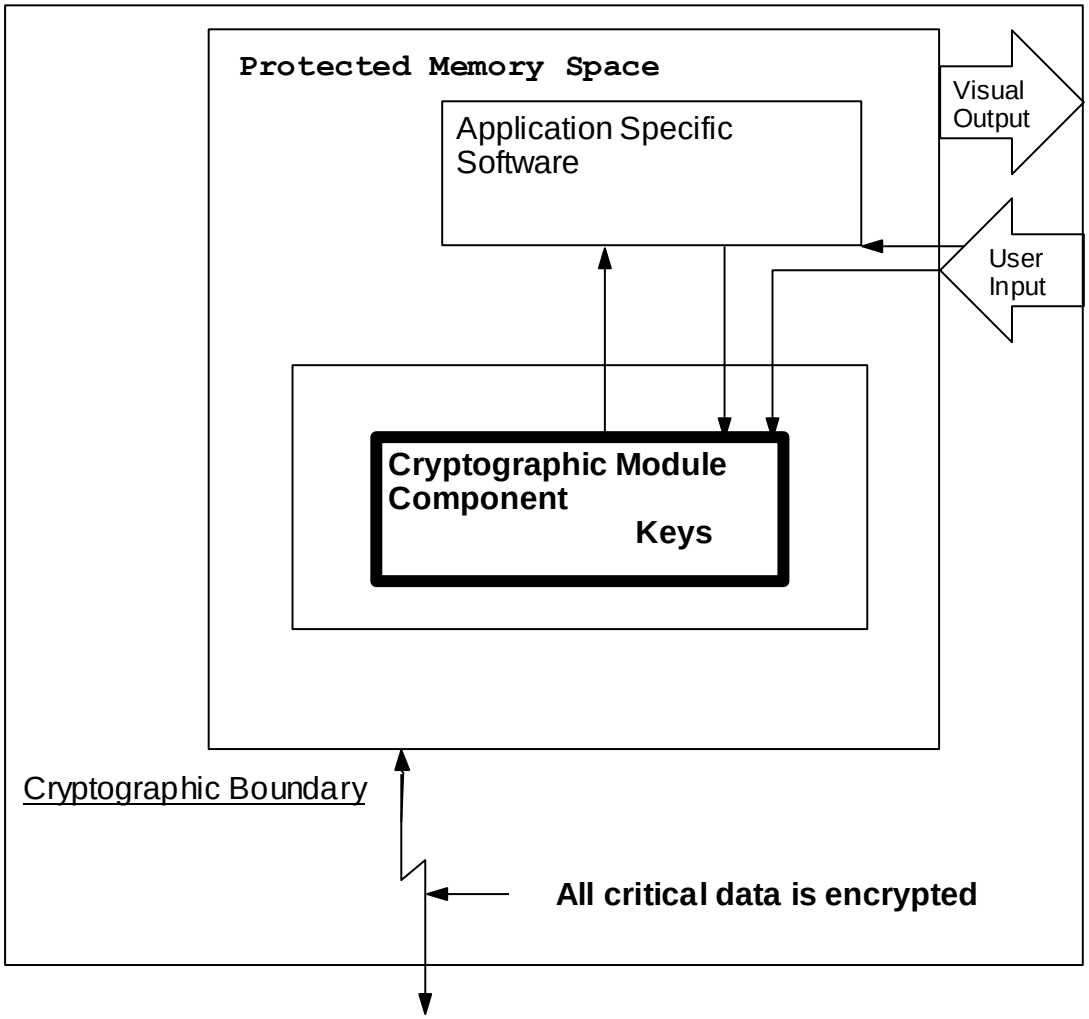


Figure 2 Module Hardware and Software

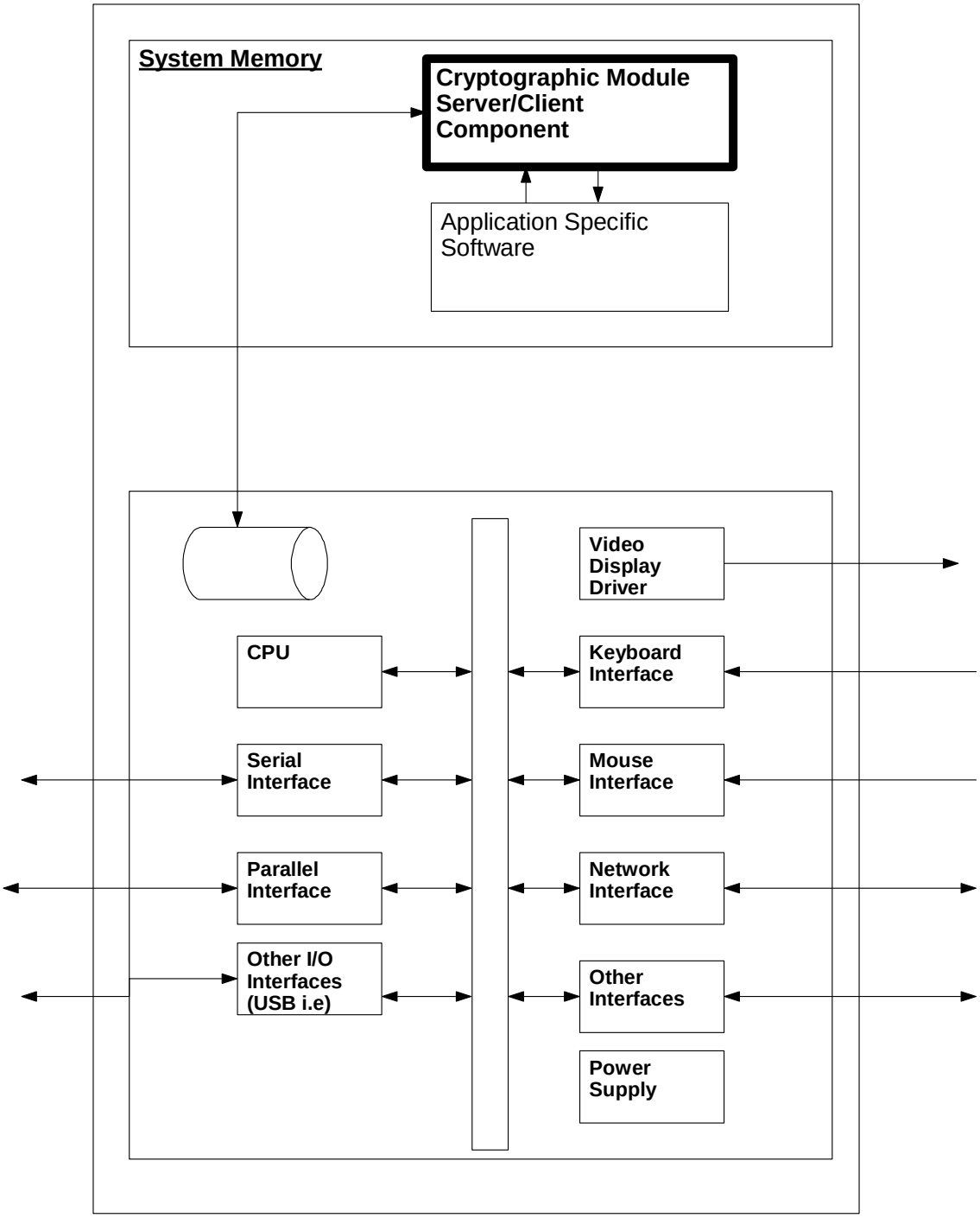
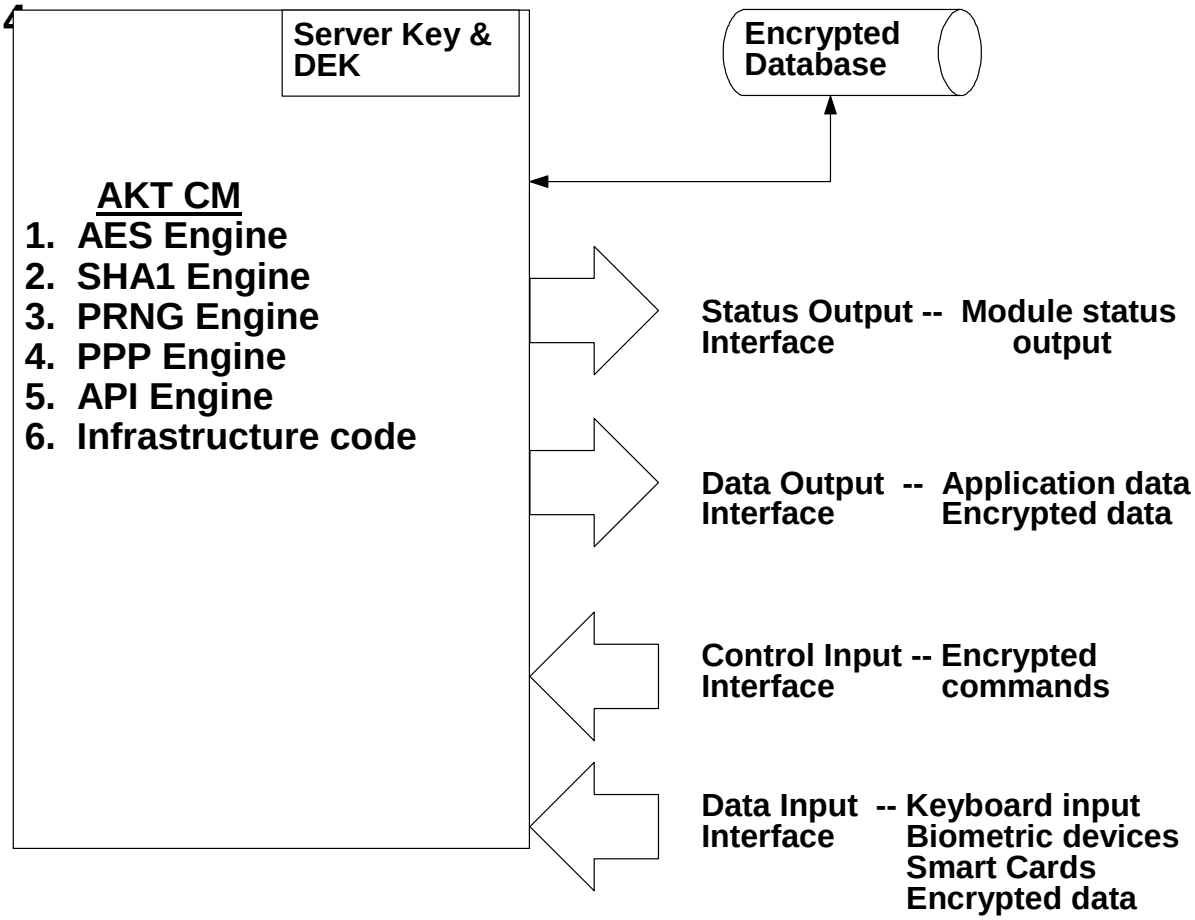


Figure 3 Functional Diagram (NOTE: PRNG is not compliant, PPP will become non-compliant on June 30, 2022)



Roles and Services

The AKT C++ and the AKT JAVA Modules support the following two distinct operator roles:

- User Role
- Cryptographic Operator Role

The AKT C++ and the AKT JAVA Modules will enforce the separation of roles using access control. An operator must enter their individual characteristics for access. Upon correct entry of their user characteristics, the role is selected based on the identity of the operator. At the end of a session, the operator will be logged-out.

The AKT C++ and the AKT JAVA Modules are being validated at Level 1, therefore, no FIPS approved authentication requirements apply. The authentication strength varies according to the particular method chosen for authentication. A validation of the authentication strength was not conducted at this time, thus AKCode makes no particular claim as pertains to the FIPS 140-2 authentication requirements.

4.1. User Role

The User role shall provide all of the services necessary for the secure transport of data over an insecure network. These services are the following:

4.1.1. Enter User Name:

This service presents the operator with an input dialogue box wherein the operator can input their assigned user name that is used as a key tag in the server database, which is located outside of the logical boundary of the AKT C++ Module and the AKT JAVA Module.

4.1.2. Enter Password:

This service presents the operator with an input dialogue box wherein the operator can input their assigned password. The entry is plain text and is obscured by asterisks from the operator, as it is input.

4.1.3. Input Additional Biometrics Data:

The biometrics data is input to the The AKT C++ Module or the AKT JAVA Module through devices that are configured to read an individual biometrics characteristic and subsequently to convert it into a unique digital format that can be transmitted.

4.1.4. Encrypt Data:

This service utilizes the FIPS-197 AES function to perform encryption of the data.

4.1.5. Decrypt Data:

This service utilizes the FIPS-197 AES function to perform decryption of the data.

4.1.6. Transmit data outside the boundary:

This service allows data to be securely transmitted outside of the AKT C++ Module or the AKT JAVA Module.

4.1.7. Create User:

The user has the ability to create themselves as a new user. A user is created when a profile, a username and a password have been successfully stored in the secure user database, which is located outside of the logical boundary of the AKT C++ Module and the AKT JAVA Module, and the user activity has been initiated.

4.1.8. Show Status:

It is possible for the user to show the current status by invoking the show status command. If an error state is processed an error code is returned, represented as a 32-bit unsigned long, to the calling application via the status output interface when an error occurs. The application using the The AKT C++ Module or the AKT JAVA Module determines how to communicate the error to the operator. If no error is determined, then a state of 0 is returned which is the ready state.

4.1.9. Perform Self Tests:

Self-testing the AKT C++ Module or the AKT JAVA Module can be performed to verify that the components are functioning as per the specification. If an error state is processed, the same methodology is used that is used in 4.1.8 above. If no error is determined, then a state of 0 is returned which is the ready state.

4.2. Cryptographic Officer:

The CO role shall provide all of the services necessary for the secure transport of data over an insecure network. These services are as follows (some services are defined earlier and are the same):

4.2.1. Enter User Name

4.2.2. Enter Password

4.2.3. Input Additional Biometrics Data

4.2.4. Encrypt Data

4.2.5. Decrypt Data

4.2.6. Transmit data outside the boundary

4.2.7. Create User:

The CO has the ability to create new users. A user is created when a profile, a username and a password have been successfully stored in the secure user database, which is located outside of the logical boundary, and the user activity has been initiated.

4.2.8. Initiate User:

In some instances, it is possible for users to enroll themselves in the system. However, the user must obtain permission from the CO to utilize the system. The user remains in an inert state until the user is initiated.

4.2.9. Deactivate User:

If a user is removed from the system, the CO has the power to deactivate them. In this state, it is possible for the user profile to remain in the database, which is

located outside of the boundary, for archival purposes, but the data would be in an inert state.

4.2.10. Spawn New CO:

The CO has the power to spawn other CO's to share the duties.

4.2.11. Show Status:

See section 4.1.8 for a discussion of how the error state is processed.

4.2.12. Perform Self Test:

The role of the CO is created when the software is installed the first time in a trusted environment. Thereafter, the CO can elect to spawn other CO's, and/or create users or, have the system established so users are allowed to create their own profiles but the CO activates the user profiles.

5. Security Rules

This section documents the security rules enforced by the AKT C++ and the AKT JAVA Modules to implement the security requirements of this FIPS 140-2 Level 1 module when used in conjunction with the specified hardware and software.

5.1. Distinct Operator Roles:

The AKT C++ and the AKT JAVA Modules shall provide two distinct operator roles. These are the User role and the CO role.

5.2. Module Access Control:

The AKT C++ and the AKT JAVA Modules shall provide access using a username and password, or additional unique user information. The cryptographic module is being validated at Level 1, therefore, no FIPS approved authentication requirements apply. The AKT C++ Module and the AKT JAVA Module authentication strength varies according to the particular method chosen for authentication. A validation of the authentication strength was not conducted at this time, thus AKCODE makes no particular claim as pertains to the FIPS 140-2 identity or role based authentication requirements.

5.3. Keys:

All keys are output in encrypted form. All keys imported through the API interface are encrypted.

5.4. Data Encryption Key (DEK):

The AKT C++ and the AKT JAVA Modules uses the AES algorithm running in 256-bit mode. All data transmitted outside the AKT C++ or the AKT JAVA Modules is encrypted using one of the following Data Encryption Keys.

5.4.1 Enterprise/Application Server DEK

A DEK generated outside the cryptographic boundary and imported to the server through an API interface. These DEKs are utilized on the server to encrypt data transmitted to and from the secure user database, which is located outside of the logical boundary.

5.4.2 Operator DEK

A DEK generated outside the cryptographic boundary and imported to the server through an API interface. This DEK is used to transmit data outside of the logical boundary or to transmit data between an AKT C++ Module or the AKT JAVA Module and another AKT C++ Module or the AKT JAVA Module.

5.5. Key Generation:

In order to operate the AKT C++ Module or the AKT JAVA Module in an Approved mode, an operator shall import approved keys using the SetCipherKey API. Any usage of the non-Approved PRNG to generate keys implicitly places the module in a non-Approved mode of operation.

Keys established in an Approved mode of operation shall not be accessed or shared while in a non-Approved mode of operation.

5.6. Pre - Validation State:

When the AKT C++ and the AKT JAVA Modules have not been placed in a valid role, the operator shall not have access to any cryptographic services.

5.7. Power Up:

Upon the application of power, or when commanded on demand by the operator (via the PerformSelfTest function), the AKT C++ or the AKT JAVA Modules shall perform the following tests:

- Integrity test:

  A HMAC-SHA-1 of its on-disk image is compared against the HMAC-SHA-1 generated on the AKT C++ Module or the AKT JAVA Module when it was built by AKCode. The specific procedure is:

  Performed by AKCode:

  1) Build (compile) the AKT Module library.
  2) A post-build step runs GenEDC.exe on the AKT C++ Module or the AKT JAVA Module built in step 1.
  3) GenEDC calculates a HMAC-SHA-1 from the AKT C++ Module or the AKT JAVA Module and writes the result to a plaintext output file in the same directory as the code. This file is shipped with the package.

  When the the AKT C++ Module or the AKT JAVA Module is loaded by an application:

  1) A HMAC-SHA-1is calculated from its image on disk.
  2) The HMAC-SHA-1 is loaded from the plaintext file and compares it to the calculated HMAC-SHA-1.

3) An error is returned to the application via the status output interface and the AKT C++ Module or the AKT JAVA Module locks its interface if the HMAC-SHA-1s are not equal.

- SHA-1 Algorithm Known Answer Test (KAT)
- SHA-1 HMAC Known Answer Test
- AES Encryption Algorithm KAT
- AES Decryption Algorithm KAT
- PPP Algorithm KAT (will become non compliant on June 30, 2022)

Upon successful completion of the self-tests, an indication is given that a "Ready" state has been achieved. If any of the above tests fail then the AKT C++ Module and the AKT JAVA Module will not continue to function, any keys that have been generated will be zeroized and the error indicator, "Power-up Self Test Failed." will be displayed on the Video Display Unit (VDU). To exit the error state, a power cycle must be performed.

At any time the AKT C++ Module and the AKT JAVA Module is in an idle state, the operator, if logged in, will be capable of performing a power-up self-test.

5.8. Zeroized Transitions:

The AKT C++ and the AKT JAVA Modules will have all ephemeral cryptographic keys zeroized under the following conditions:

5.8.1. Upon Destruction:

Destruction refers to the unloading of the library from memory.

5.8.2. Upon CO command:

The command to zeroized is "ZeroizeCipherKey".

5.8.3. Upon any Error Condition:

Any error that causes a shut down state will result in unloading and zeroizing.

The AKT C++ and the AKT JAVA Modules will have all persistent cryptographic keys zeroized under the following conditions:

5.8.4. Upon Procedural Zeroization:

Procedural Zeroization refers to the uninstallation, reformatting and overwriting (at least once) of the persistent storage medium that is used to store the module's binary.

5.9. Module Status:

The AKT C++ and the AKT JAVA Modules status can be by querying the module for the status. In the majority of cases, the status will be normal, a state which will be indicated by a message returned by the module. If the status is abnormal, the message will be changed to indicate that a problem has arisen and an error code will be returned. If no error is found, the module returns a value of 0.

5.10. Concurrent Users:

The AKT C++ and the AKT JAVA Modules is intended to be single user per instance and it does not support concurrent users of each instance.

5.11. Definition of Critical Security Parameters (CSP):

The following are critical security parameters contained in the AKT C++ and the AKT JAVA Modules:

- Operator DEK
- Application Server DEK
- Enterprise Server DEK
- PPP Key Encryption Key (KEK)
- PPP Public and Private Key Pairs

5.12. Error State:

When an error state is entered (due to a self test failure, for example) the following steps occur:

5.12.1. CSPs cleared:

The CSPs of the faulting component are cleared.

5.12.2. Set Fault Flag:

A ‘fault’ flag is set.

5.12.3. Error Code:

An error code relating to the encountered fault is returned to the user via the status output interface.

5.12.4. Fault Flag:

All data input interface methods check the fault flag before processing the data request. The error code is returned to the user via the status output interface if the fault flag is set.

5.12.5. Show Status:

The only method available when the fault flag is set is the ‘show status’ method.

5.12.6. Performing Self Tests:

The fault flag is set to ‘performing self test’ when the self-tests are in progress. This effectively inhibits the data input and output interface.

5.12.7. Fault Flag Reset:

The only way to reset the fault flag is to reload (i.e. unload the AKT C++ Module or the AKT JAVA Module from memory and reload it).

6. Roles, Access Control and Services

6.1. Roles and Access Control:

Access control will take the form of one of three possible mechanisms:

6.1.1. User Based Access:

User based access, uses the password and/or the username.

6.1.2. Biometrics Based Access:

Biometrics access uses the biometric template.

6.1.3. Device Based Access:

Device access uses a Personal Authentication Device, such as a smart card, an RF device, or a USB device, in conjunction with the other access methods discussed earlier.

The following table summarizes the roles and access methodologies:

Table 6.1 Roles and Required Identification and Access

Role	Type of Access	Access Data
All Operators	Password	User password
All Operators	Biometrics	Biometrics sample
All Operators	Device	Ownership of device

The AKT C++ Module and the AKT JAVA Module are being validated at Level 1, therefore, no FIPS approved authentication requirements apply. Authentication strength varies according to the particular method chosen for authentication. A validation of the authentication strength was not conducted at this time, thus AKCode makes no particular claim as pertains to the FIPS 140-2 identity or role based authentication requirements.

6.2. Roles and Services with Access Rights:

Referring to section 4.0 and 5.0 the Roles and Services can be summarized as follows (see section 7.1 for definitions):

Table 6.2(a) Roles, Services and Access Rights

Role		Service	CSP Modes of Access
C.O.	**User**		
X	X	Enter User Name	
X	X	Enter Password	
X	X	Input Additional Biometrics Data	
X	X	Encrypt Data	Use - Operator DEK Use - Application Server DEK Use - Enterprise Server DEK
X	X	Decrypt Data	Use - Operator DEK Use - Application Server DEK Use - Enterprise Server DEK
X	X	Transmit Data Outside the boundary	Use - Operator DEK Use - Application Server DEK Use - Enterprise Server DEK
X	X	Create User	Set - Operator DEK Set - Application Server DEK Set - Enterprise Server DEK Establish – PPP KEK Use - PPP Public and Private Key Pairs
X	X	Show Status	
X	X	Perform Self-Tests	
X		Initiate User	Set - Operator DEK Set - Application Server DEK Set - Enterprise Server DEK
X		Deactivate User	Zeroize - Operator DEK Zeroize - Application Server DEK Zeroize - Enterprise Server DEK
X		Spawn New CO	Set - Operator DEK Set - Application Server DEK Set - Enterprise Server DEK

6.3. Physical Security Mechanisms:

Physical security of the AKT C++ Module and the AKT JAVA Module is currently specified as production grade hardware. The AKT C++ Module and the AKT JAVA Module are software components and as such are not dependent on the physical security of the device that houses the software. Never the less, the user is expected to take reasonable measures to protect the physical security. The AKT C++ Module and the AKT JAVA Module need to be run in a secure computational environment and as much as possible, it should be run in a secure physical environment although, this latter requirement is not as important as the

computational environment. It is intended that the AKT C++ Module and the AKT JAVA Module be run under a single user operating system environment. The physical device should be in a secure location so that it is protected under normal operating conditions.

6.4. Mitigation of Other Attacks:

The AKT C++ Module and the AKT JAVA Module have not been designed to mitigate any other attacks.

7. Appendix A

7.1. Definitions

AES	Advanced Encryption Standard
CO	Crypto Officer
CPP	C++ Programming Language
CSP	Critical Security Parameter
d	Session Data
DEK	Data Encryption Key
DLL	Dynamic-link Libraries
FIPS	Federal Information Processing Standard
KAT	Known Answer Test
KEK	Key Encryption Key (PPP Establishment)
NIST	National Institute of Standards and Technology
OS	Operating System
p	Operator Password
PAD	Personal Authentication Device
PC	Personal Computer
PPP	AKCode Ping Pong Ping algorithm
SHA	Secure Hash Algorithm
AKCode	AKCode Inc.
USB	Universal Serial Bus
VDU	Video Display Unit