Attachmate Security Component for Java

FIPS 140-2 Security Policy, version 2.10

7 April 2008

table of contents

1 Module Overview

Attachmate Security Component for Java (ASCJ), V. 1.32, is a software component that provides data encryption and integrity services through its implementation of the SSL/TLS protocols for secure communication over TCP/IP networks. Its primary use is as a building block in software systems that require such services. It provides data input and output, control input, and status output logical interfaces via a Java-based Application Programming Interface. Interface separation is provided and enforced through single-purpose, type-checked, minimally-scoped API methods and parameters.

For FIPS 140-2 classification purposes the module is considered to be a "multi-chip standalone module."

The module may be used for implementation of SSL/TLS clients, SSL/TLS servers, or both. Server authentication and client authentication are supported. All data are encrypted over the SSL/TLS connection using AES, DES or Triple-DES ciphers.

A typical implementation of a system for end-to-end encryption employs one instance of the module as a client and another instance of the module as a server. Client and server negotiate a secure connection, after which plaintext data written to the data input interface of one module instance are encrypted for transport over the network, then decrypted for reading as plaintext through the data output interface of the second module instance. The module also fully supports interoperability as a client or server with other implementations of the SSL/TLS protocol.

The module supports both an Approved and a non-Approved mode of operation. In Approved mode the module performs power-up self-tests, operator authentication, enforces Level 1 or Level 2 security requirements, and prohibits the use of SSL. This Security Policy applies to Level 1 operation of the module. The mode of operation and enforcement level are selected at power-up (module instantiation) and are in effect for the lifetime of the instance. Approved mode is selected by providing a value of *true* for the *inApprovedMode* argument of the module's *TLSModule* constructor. The security level is selected by providing a value of *FIPS140_LEVEL_1* or *FIPS140_LEVEL_2* for the *inEnforcedSecurityLevel* argument of the module's *TLSModule* constructor. The module indicates whether it is in an Approved mode of operation and the enforced security level when the operator invokes the *Show Status* service.

The module supports the following cryptographic algorithms in both modes of operation:

- AES
- DES (although it is configurable, the use of DES is not permitted in Approved mode)
- Triple-DES
- RSA encryption (for use with the TLS key establishment protocol only; key establishment methodology provides between 80 and 112 bits of encryption strength)

- RSA PKCS#1 digital signature
- DSA digital signature
- SHA-1
- MD5 (for use with the TLS key establishment protocol only)
- HMAC-SHA-1
- Diffie-Hellman (for use with the TLS key establishment protocol only; key establishment methodology provides between 80 and 128 bits of encryption strength)
- FIPS 186-2 Appendix 3.1 PRNG
- HMAC-MD5 (for use with the TLS key establishment protocol only)

Operation in Approved mode at Level 1 is permitted when the module is installed on any tested hardware and software configuration, or on any hardware and software configuration on which the module executes without modification.

Attachmate Security Component for Java has been tested on the following systems:

- Red Hat Linux 4 x 64 (RHELx64) and Sun Java Runtime 1.5.0 and AMD 275 Opteron 2.2GHz, Dual Core processor (HP ProLiant DL145R2 2G Server)
- Mac OS X 10.4.3 and Apple Java Runtime 1.5.0 and G4 processor (Apple Power Macintosh)
- Windows XP and Sun Java Runtime 1.5.0 and Intel Xeon 2.80GHz/800MHz, dual processor (HP ProLiant DL140)

Figure 1-1 is a block diagram illustrating the cryptographic module, its relationships to other components, and the cryptographic boundaries. The cryptographic module includes Attachmate Cryptographic Library for Java (ACLJ) which implements the validated cryptographic algorithms. The logical cryptographic boundary is denoted by the dotted blue line. The cryptographic module client component outside the logical boundary illustrates the means by which the module may be used to secure data over a network. The physical cryptographic boundary is denoted by the dotted red line.

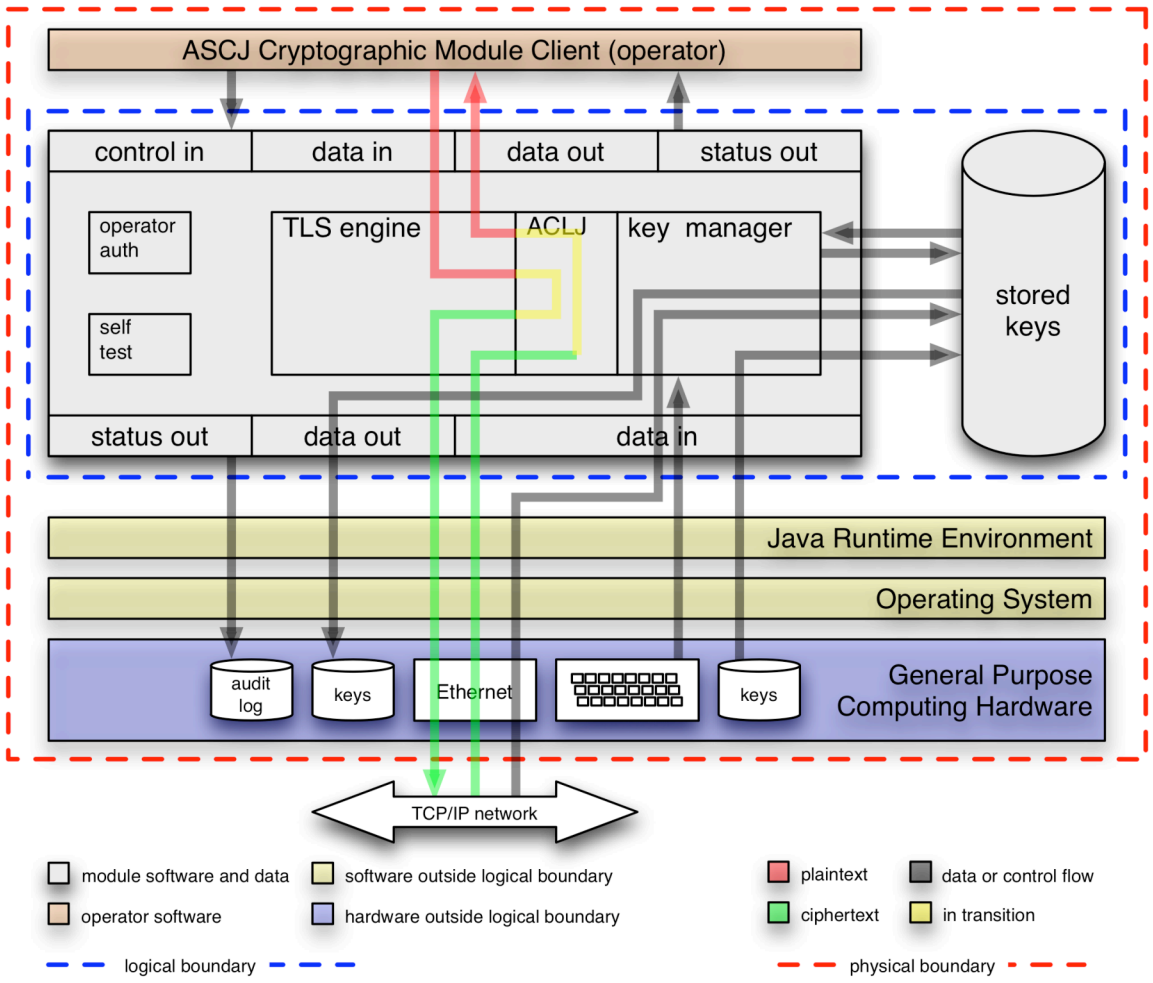

Figure 1-1: module block diagram

2 Security Level

The cryptographic module meets the overall requirements applicable to Level 1 security of FIPS 140-2. The following table identifies the level attained for each security requirements section.

Table 2-1: Module Security Level Specification

Security Requirements Section	Level
Cryptographic Module Specification	3
Module Ports and Interfaces	1
Roles, Services, and Authentication	3
Finite State Model	1
Physical Security	N/A
Operational Environment	1
Cryptographic Key Management	1
EMI/EMC	1
Self-Tests	1
Design Assurance	3
Mitigation of Other Attacks	N/A

3 Identification and Authentication Policy

The cryptographic module shall support a User operator role and a Cryptographic Officer operator role. The module shall support identity-based authentication. Only one role at a time shall be active. The module shall not allow concurrent operators.

The module shall not support a maintenance role.

The User role permits access to basic TLS services: opening sockets, reading and writing data, etc. It also permits access to a set of services for basic module operation: power management, status reporting, etc.

The Cryptographic Officer role permits access to services for configuring and testing the module: generating and importing cryptographic keys and certificates, executing self tests, etc. It also permits access to a set of services for basic module operation: power management, status reporting, etc.

The module supports identity-based authentication through digital signatures. In order to instantiate the module for the purpose of invoking services the operator (a programmatic entity, e.g. an AttachmateWRQ application or other Attachmate-authorized application) must supply to the module a digitally signed value derived from its executable bytecode. The module computes this value independently by examining the bytecode of the caller, and ensures that it matches the digitally signed value. Module instantiation will succeed only in the event that this Attachmate-signed bytecode is authenticated.

Operator code must implement one or more tagging interfaces identifying the roles for which it has been authorized. Module services will refuse to execute if they are not permitted for the roles implemented by the instantiating operator. Operator code that requires User role services must implement the *com.wrq.fipsmodule.roles.IFipsUserRole* interface. Operator code that requires Cryptographic Officer services must implement the *com.wrq.fipsmodule.roles.IFipsCryptoOfficerRole* interface.

The public key corresponding to the private key used to sign the authentication data is stored within the module. The digital signature algorithm used by the signing certificate may be either RSA or DSA.

There is no method of manual data entry for purposes of authentication. If an attempt is made to instantiate the module with incorrect authentication data then the module throws an exception following a one second delay. This exception contains no sensitive data that could weaken the strength of the authentication method.

No authentication state survives power cycling.

Table 3-1 summarizes the roles and authentication requirements.

Table 3-1: Roles and Required Identification and Authentication

Role	Type of Authentication	Authentication Data
User	identity-based operator authentication	tagging interface, module bytecode, digital signature
Cryptographic Officer	identity-based operator authentication	tagging interface, module bytecode, digital signature

Table 3-2 summarizes the strengths of the authentication mechanisms.

Table 3-2: Strengths of Authentication Mechanisms

Authentication Mechanism	Strength of Mechanism
digital signature	The digital signature employs a 20 byte SHA-1 hash, or 2^{160} discrete values. A single authentication attempt therefore has less than a 1 in 2^{80} chance of false acceptance. The module operator authentication process delays for one second before responding to a failed authentication attempt, preventing more than one attempt per second. The Java synchronization mechanism is employed during the delay to prevent a multithreaded attack. The chance of a false acceptance in the period of one second is therefore equal to the chance of any single attempt resulting in false acceptance. The chances of a false acceptance in the period of one minute are therefore less than 60 in 2^{80}.

4 Access Control Policy

4.1 Roles and Services

The User role has no access to cryptographic parameters or keys. All services are programmatically invoked through a Java-based Application Programming Interface. The following table identifies the services that are available to each authorized role.

Table 4-1: Services Authorized for Roles

Authorized Role	Authorized Service
User	***Close Connection*** – This service closes an SSL/TLS connection. The operator supplies a socket created by the Open Connection service.
User, Cryptographic Officer	***Data Encryption*** – This service performs data encryption and decryption. The operator supplies plaintext or ciphertext data and identifies a keystore and key. This service makes use of RSA encryption security functions. This service may not be used in Approved mode.
User, Cryptographic Officer	***Data Signing*** – This service performs digital signing and signature verification. For signing, the operator supplies data to be signed and identifies a key. For verification, the operator supplies signed data, a signature, and identifies a keystore and key. This service makes use of DSA signature and RSA signature security functions.
Cryptographic Officer	***Delete Keys*** - This service deletes a key entry (public key or public/private keypair) from a module keystore. The operator identifies the keystore and the key's alias. The operator or an interactive user supplies passphases for encrypted keystores. This service makes use of Triple-DES security functions for managing encrypted PKCS #12 keystores. This encryption is not used to fulfill any FIPS requirements and for the purposes of FIPS 140-2 the keystores are considered to be plaintext.
Cryptographic Officer	***Enter Keys*** - This service imports a public key or public/private key pair from a keystore within the physical boundary and adds it to a module keystore. The operator or an interactive user supplies passphases for encrypted keystores. This service can also process a returned Certificate Signing Request. This service makes use of Triple-DES security functions for managing encrypted PKCS #12 keystores. This encryption is not used to fulfill any FIPS requirements and for the purposes of FIPS

	140-2 the keystores are considered to be plaintext.
User, Cryptographic Officer	***Export Keys*** - This service exports one or more public key certificates from a PKCS #12 keystore. The operator identifies the keystore, the key's alias, and a destination file specification. Certificates can be written to a file or returned directly to the caller of the API. The operator or an interactive user supplies passphases for encrypted keystores. This service makes use of Triple-DES security functions for managing encrypted PKCS #12 keystores. This encryption is not used to fulfill any FIPS requirements and for the purposes of FIPS 140-2 the keystores are considered to be plaintext.
Cryptographic Officer	***Generate Keys*** - This service generates public/private key pairs, saving the private key and public key certificate in a module keystore. The operator supplies all pertinent certificate data and identifies the keystore and key alias. The operator or an interactive user supplies passphases for encrypted keystores. This service can also generate a Certificate Signing Request for submission to a Certificate Authority. This service makes use of RSA and DSA signature, PRNG, and Triple-DES security functions. The PRNG implements a FIPS 186 approved algorithm. Triple-DES encryption is not used to fulfill any FIPS requirements and for the purposes of FIPS 140-2 the keystores are considered to be plaintext.
Cryptographic Officer	***Manage Keystores*** - This service returns keystore passphrases to the operator or modifies passphrases of existing keystores. The operator identifies the keystore. This service makes use of Triple-DES security functions for managing encrypted PKCS #12 keystores. This encryption is not used to fulfill any FIPS requirements and for the purposes of FIPS 140-2 the keystores are considered to be plaintext.
User	***Open Connection*** – This service opens an SSL/TLS connection. The operator supplies an open socket over which SSL/TLS negotiation will occur, SSL/TLS configuration parameters, and identifies keystores containing client and server authentication keys and certificates. The service returns an instance of a socket which has been secured with SSL/TLS. This service implements a bypass capability by permitting a null cipher to be specified and negotiated for encryption of session data. This service makes use of DSA signature, RSA signature, RSA encryption, Diffie-Hellman key establishment, SHA-1, MD5, HMAC, and PRNG security functions. The PRNG implements a FIPS 186 approved algorithm. Key establishment has 80 to 128 bits

	of strength, depending on cipher suite selection.
Cryptographic Officer	***Perform Self-Tests*** - This service initiates the power-up self-tests. This service makes use of all security functions supported by the module.
User, Cryptographic Officer	***Power-Down*** - This service shuts down the module, performing any necessary zeroization. This service does not make use of any security functions.
User, Cryptographic Officer	***Preflight*** – This service is used to instruct an instance of the module to acquire keystore passphrases for subsequent use. The operator identifies the keystore. This service does not make use of any security functions.
User	***Read Data*** – This service receives encrypted data from an SSL/TLS socket and decrypts it. The operator provides a buffer into which the service stores the plaintext. This service makes use of AES and DES or Triple-DES decryption security functions. DES decryption may not be used in Approved mode.
User, Cryptographic Officer	***Show Status*** – This service presents the status of the module through a logging facility, by setting properties which may be accessed programmatically by the operator, and by direct return of results to API method calls. This service does not make use of any security functions.
User	***Write Data*** – This service encrypts data and transmits the encrypted data over an SSL/TLS socket. The operator provides a buffer of plaintext data for encryption and transmission. This service makes use of AES and DES or Triple-DES encryption security functions. DES encryption may not be used in Approved mode.
Cryptographic Officer	***Zeroize*** – This service zeroizes all cryptographic keys and critical security parameters in both volatile memory and and any nonvolatile keystores identified by the operator. This service does not make use of any security functions.

4.2 Definition of Cryptographic Keys and Critical Security Parameters

The following table lists the critical security parameters, including secret and private cryptographic keys, contained in the module.

Table 4-2: Critical Security Parameters

Critical Security Parameter	Description
SSL/TLS client private key	An instance of the module acting as an authenticated SSL/TLS client must have access to its private key to encrypt authentication data during the SSL/TLS handshake. RSA keys between 512 and 2048 bits in length or DSA keys between 512 and 1024 bits in length may be used. In Approved mode RSA keys of less than 1024 bits may not be used.
SSL/TLS server private key	An instance of the module acting as an SSL/TLS server must have access to its private key to decrypt data encrypted by the client with the server's public key during the SSL/TLS handshake. RSA keys between 512 and 2048 bits in length or DSA keys between 512 and 1024 bits in length may be used. In Approved mode RSA keys of less than 1024 bits may not be used.
Diffie-Hellman private keys	DH public/private keys between 512 and 3072 bits in length are used for key exchange and session key generation during the SSL/TLS handshake. In Approved mode DH keys of less than 1024 bits may not be used.
SSL/TLS session key	Session keys are used for encryption and decryption of transmitted data using AES (128-bit or 256-bit key), DES (56-bit key), and Triple-DES (168-bit key) ciphers. DES keys may not be used in Approved mode.
SSL/TLS HMAC key	HMAC keys are used for authentication of transmitted session data. The key size is the same as the SSL/TLS session key size.
master secret	The master secret is used for session key and message initialization vector generation.
message initialization vector (IV)	The cipher used by each SSL/TLS socket is initialized with a unique IV.
PRNG state	The internal state of the pseudo random number generator.
data signing and encryption	The module may contain any number of private keys for use with the Data Signing and Data Encryption

	services.

The following table lists the public cryptographic keys contained in the module.

Table 4-3: Public Cryptographic Keys

Cryptographic Key	Description
CA root certificate	Certificate Authority root certificates are used in establishing the validity of other public key certificates.
trusted certificate	Trusted certificates are used in establishing the validity of other public key certificates. RSA keys between 512 and 2048 bits in length or DSA keys between 512 and 1024 bits in length may be used. In Approved mode RSA keys of less than 1024 bits may not be used.
SSL/TLS server	An instance of the module acting as an SSL/TLS server must provide a certificate containing its public key for server authentication during the SSL/TLS handshake. An instance of the module acting as an SSL/TLS client receives a server's public key and uses it for authentication and encryption during the SSL/TLS handshake. RSA keys between 512 and 2048 bits in length or DSA keys between 512 and 1024 bits in length may be used. In Approved mode RSA keys of less than 1024 bits may not be used.
SSL/TLS client	An instance of the module acting as an SSL/TLS client must provide a certificate containing its public key for client authentication during the SSL/TLS handshake if the server is configured for client authentication. An instance of the module acting as an SSL/TLS server receives a client's public key and uses it for authentication during the SSL/TLS handshake. RSA keys between 512 and 2048 bits in length or DSA keys between 512 and 1024 bits in length may be used. In Approved mode RSA keys of less than 1024 bits may not be used.
Diffie-Hellman public keys	DH keys between 512 and 3072 bits in length are used for key exchange and session key generation during the SSL/TLS handshake. In Approved mode DH keys of

	less than 1024 bits may not be used.
Attachmate Corporation signing certificate	The module must have access to the public key of the Attachmate Corporation signing certificate for authentication of digitally signed operator code and module software integrity message digests. This is a 1024-bit RSA key.
data signing and encryption	The module may contain any number of public keys for use with the Data Signing and Data Encryption services.

4.3 Definition of CSPs Modes of Access

4.3.1 Access Operations

Cryptographic keys and critical security parameters are accessed during the performance of a number of module services. The access operations are defined below, including the type of access (in bold type) for each key and CSP. Refer to Section 4.2 for definitions of the keys and CSPs, and Table 4-4 for a mapping of access operations to services.

Authenticate Client – This operation, performed by the SSL/TLS client, **reads** the *SSL/TLS client public key* and *SSL/TLS client private key* and uses them to authenticate a client to a server. This operation is optionally performed when an SSL/TLS socket is created.

Authenticate Server – This operation, performed by the SSL/TLS server, **reads** the *SSL/TLS server public key* and *SSL/TLS server private key* and uses them to authenticate a server to a client. This operation is performed when an SSL/TLS socket is created.

Create IV – This operation creates and **writes** the *message initialization vector* that will be used to initialize a cipher. This operation is performed when an SSL/TLS socket is created.

Create Master Secret – This operation creates and **writes** the *master secret* used for generating session keys and initialization vectors. This operation is performed when an SSL/TLS socket is created.

Decrypt Session Data – This operation **reads** the *SSL/TLS session key* and uses it to decrypt data received from an SSL/TLS connection.

Delete Key – This operation **erases** an *SSL/TLS client private key* or *SSL/TLS server private key* from a keystore.

Destroy Diffie-Hellman Keys – This operation **erases** the *Diffie-Hellman keys* for an SSL/TLS socket. This operation is performed when the master secret is generated.

Destroy HMAC Key – This operation **erases** the *SSL/TLS HMAC key* for an SSL/TLS socket. This operation is performed when the socket is closed.

Destroy IV – This operation **erases** the *message initialization vector* for an initialized encryption algorithm. This operation is performed when cipher initialization is completed.

Destroy Master Secret – This operation **erases** the *master secret* for an SSL/TLS socket. This operation is performed when an SSL/TLS handshake is completed if

the session is not resumable. If session resumption is enabled for the socket then its master secret is erased by the Destroy Volatile Keys operation.

Destroy Nonvolatile Keys – This operation **erases** *all cryptographic keys and critical security parameters* in nonvolatile storage within the module. This operation is performed on command of the Crypto Officer.

Destroy Session Key – This operation **erases** the *SSL/TLS session key* for an SSL/TLS socket. This operation is performed when the socket is closed.

Destroy Volatile Keys – This operation **erases** *all cryptographic keys and critical security parameters* in volatile storage within the module. This operation is performed during power-down or on command of the Crypto Officer.

Encrypt Session Data – This operation **reads** the *SSL/TLS session key* and uses it to encrypt data for transmission over an SSL/TLS connection.

Generate Diffie-Hellman Keys – This operation generates and **writes** the *Diffie-Hellman public/private keys* used for session key establishment. This operation is performed when an SSL/TLS socket is created. The PRNG implements a FIPS 186 approved algorithm.

Generate HMAC Key – This operation establishes through SSL/TLS and **writes** the *SSL/TLS HMAC key* for authenticating SSL/TLS connection data. This operation is performed when an SSL/TLS socket is created. The PRNG used in this operation implements a FIPS 186 approved algorithm.

Generate Random Number – This operation **reads** and **writes** the *PRNG state*. The PRNG implements a FIPS 186 approved algorithm.

Generate Session Key – This operation establishes through SSL/TLS and **writes** the *SSL/TLS session key* for encrypting and decrypting SSL/TLS connection data. This operation is performed when an SSL/TLS socket is created. The PRNG used in this operation implements a FIPS 186 approved algorithm.

Generate Server Keys – This operation generates and **writes** the *SSL/TLS server private key* and corresponding self-signed public key certificate for an SSL/TLS server. The PRNG implements a FIPS 186 approved algorithm.

Initialize Cipher – This operation **reads** the *SSL/TLS session key* and *message initialization vector* and uses them to initialize the cipher for SSL/TLS data encryption and decryption. This operation is performed when an SSL/TLS socket is created.

Load Key – This operation **reads** an *SSL/TLS client private key, SSL/TLS server private key*, or *data signing and encryption key* from nonvolatile storage within the module.

Store Key – This operation **writes** a *SSL/TLS client private key, SSL/TLS server private key*, or *data signing and encryption key* and corresponding public keys to nonvolatile storage within the module.

4.3.2 Services and Access Operations

The following table defines the relationships between module services defined in Section 4.1, the keys and the access operations defined in Section 4.3.1.

Table 4-4: Access Operations for Services

Service	Access Operation(s)	Type(s) of Access
Close Connection	Destroy HMAC Key Destroy Session Key	erase erase
Data Encryption	Load Key	read
Data Signing	Load Key	read
Delete Keys	Delete Key	erase
Enter Keys	Store Key	write
Export Keys	N/A	
Generate Keys	Generate Random Number Generate Server Keys Store Key	read, write write write
Manage Keystores	N/A	
Open Connection	Authenticate Client Authenticate Server Create IV Destroy Diffie-Hellman keys Destroy IV Destroy Master Secret Generate HMAC Key Generate Master Secret Generate Random Number Generate Session Key Initialize Cipher Load Key	read read write erase erase erase write write write write read read
Perform Self-Tests	N/A	
Power-Down	Destroy Volatile Keys	write
Preflight	N/A	
Read Data	Decrypt Session Data	read
Show Status	N/A	
Write Data	Encrypt Session Data	read
Zeroize	Destroy Nonvolatile Keys Destroy Volatile Keys	write write

5 Security Rules

The following rules are enforced by the cryptographic module in order to implement FIPS 140-2 Level 1 security requirements.

1. The design of the cryptographic module shall correspond to these security rules.

2. The cryptographic module shall provide two distinct operator roles. These are the User role and the Cryptographic Officer role.

3. The cryptographic module shall provide identity-based authentication.

4. An operator may be authorized for one or both roles.

5. At power-up (cryptographic module instantiation) the cryptographic module instance shall be instructed to assume either non-Approved or Approved mode of operation.

6. The cryptographic module shall not permit instantiation in Approved mode unless the operator has been authenticated.

7. The cryptographic module instance shall not perform any service for which the operator is not authorized.

8. The cryptographic module instance shall remain in the non-Approved or Approved mode of operation determined at power-up until power-down.

9. The cryptographic module instance shall indicate whether it is in non-Approved or Approved mode via the status output interface.

10. At power-up the cryptographic module instance shall be instructed to enforce Level 1 security requirements.

11. The cryptographic module instance shall enforce the Level 1 security requirements determined at power-up until power-down.

12. If the cryptographic module instance has been instructed to enforce Level 1 security requirements then the operational environment shall be configured for single-user mode.

13. If the cryptographic module instance has been instructed to operate in its Approved mode of operation then it shall establish only TLS connections and refuse to establish SSL connections.

14. When the cryptographic module instance is in Approved mode the DES cipher shall not be used.

15. Upon cryptographic module instance power-down the cryptographic module instance shall zeroize all memory-resident cryptographic keys and critical security parameters.

16. Upon cryptographic module instance power-down the cryptographic module instance shall inhibit all cryptographic operations.

17. Upon cryptographic module instance power-down the cryptographic module instance shall inhibit the data output interface.

18. The cryptographic module shall perform cryptographic functions when operating in Approved mode using the following FIPS Approved algorithms:
    a. AES
    b. Triple-DES
    c. DSA digital signature
    d. HMAC-SHA-1
    e. RSA PKCS#1 digital signature
    f. SHA-1
    g. FIPS 186-2 Appendix 3.1 PRNG

19. The cryptographic module shall perform cryptographic functions when operating in Approved mode using the following non-FIPS Approved algorithms:
    a. Diffie-Hellman (for use with the TLS key establishment protocol only)
    b. MD5 (for use with the TLS key establishment protocol only)
    c. RSA encryption (for use with the TLS key establishment protocol only)
    d. HMAC-MD5 (for use with the TLS key establishment protocol only)

20. The cryptographic module shall perform random number generation using a FIPS 186-2 Approved algorithm.

21. The cryptographic module shall perform the following conditional self-tests:
    a. continuous random number generator self-test
    b. Diffie-Hellman pair-wise consistency test for key generation during SSL/TLS negotiation
    c. DSA signature pair-wise consistency test during SSL/TLS negotiation
    d. RSA signature pair-wise consistency test during SSL/TLS negotiation
    e. DSA signature pair-wise consistency test during generation of a self-signed certificate
    f. RSA signature pair-wise consistency test during generation of a self-signed certificate
    g. bypass test when a socket is opened or closed (Approved mode only)

22. All keys shall be generated using the FIPS 186-2 PRNG.

23. All DSA keys shall be generated in accordance with the FIPS 186-2 DSA key generation requirements.

24. The RSA key generation process shall be compliant with PKCS #1.

25. Upon power-up in Approved mode or when commanded by the operator the cryptographic module shall perform the following self-tests:
    a. software integrity test through a SHA-1 message digest and RSA signature verification
    b. DES encryption algorithm known answer test (encryption and decryption)
    c. Triple-DES encryption algorithm known answer test (encryption and decryption)
    d. AES encryption algorithm known answer test (encryption and decryption)
    e. RSA encryption algorithm pair-wise consistency test
    f. SHA-1 hash known answer test (verify)
    g. MD5 known answer test (verify)
    h. HMAC-SHA-1 known answer test (verify)
    i. HMAC-MD5 known answer test (verify)
    j. DSA signature pair-wise consistency test
    k. RSA signature pair-wise consistency test
    l. Diffie-Hellman pair-wise consistency test
    m. PRNG known answer test

26. Upon completion or failure of power-up self-tests the cryptographic module shall output the results via the status output interface.

27. During self-test the cryptographic module shall inhibit the data output interface for all instances of the cryptographic module.

28. Upon failure of one or more self-tests the cryptographic module shall enter an error state.

29. When in an error state the cryptographic module shall inhibit the data output interface of all instances of the cryptographic module.

30. When in an error state the cryptographic module shall not permit additional instantiations of the module.

31. Upon failing to authenticate an operator requesting instantiation of the module, the cryptographic module shall delay the operation of the module for a period of one second before accepting a new request.

32. Private keys and public key certificates shall be stored within the cryptographic module in PKCS #12 format.

33. The cryptographic module shall permit encryption of all PKCS #12 keystores using password privacy mode and password integrity mode. (Note that this does not provide FIPS-approved confidentiality.)

34. Access to the cryptographic module shall be controlled by the authorization and discretionary access control mechanisms of the operating environment.

35. Private keys shall have been manually established prior to electronic entry into the cryptographic module.

6 Non-Approved Mode of Operation

The following security functions are supported by the module but are permissible only when the module is not being operated in Approved mode.

6.1 Client Private Key Distribution

The SSL/TLS client may be configured to request that a keypair for client authentication be delivered to it on-demand from an HTTP(S) server prior to performing the SSL/TLS handshake. The keys are not encrypted using an Approved method, and the HTTPS implementation is provided by the JRE or web browser, not the module itself. Entering private keys into the module in this manner is a non-Approved security function unless the underlying HTTPS implementation has been independently Approved.

6.2 DES

The module supports DES and performs DES self-test functions, but if the module is configured such that the DES algorithm is used for TLS then it is no longer considered to be in an Approved mode of operation.

6.3 SSL

In non-Approved mode SSL may be configured and used in addition to TLS.

6.4 Short Keys

The SSL/TLS client supports the use of RSA and DH keys with lengths less than 1024 bits. When the module is operated in Approved mode it must be configured to use RSA and DH keys of 1024 bits or more.

6.5 RSA encryption

The module provides a service for data encryption and decryption using the RSA encryption algorithm. This service may not be used in Approved mode.

7 Physical Security Policy

Attachmate Security Component for Java is a cryptographic module that is implemented completely in software such that the physical security is provided solely by the host platform. It is not subject to the physical security requirements of FIPS 140-2.

8 Mitigation of Other Attacks

The Attachmate Security Component for Java module is not designed to mitigate attacks outside the scope of the FIPS 140-2 requirements.

9 Acronym List

AES – Advanced Encryption Standard
API – Application Programming Interface
CA – Certificate Authority
CSP – Critical Security Parameter
DES – Data Encryption Standard
DH – Diffie-Hellman
DSA – Digital Signature Algorithm
EMI/EMC – Electromagnetic Interference/Electromagnetic Compatibility
FIPS – Federal Information Processing Standard
FSM – Finite State Model/Machine
HMAC – Keyed-Hash Message Authentication Code
HTTP – HyperText Transmission Protocol
HTTPS – HyperText Transmission Protocol, Secure
IV – Initialization Vector
JRE – Java Runtime Environment
MD5 – Message Digest algorithm
PKCS – Public Key Cryptography Standard
PRNG – Pseudo Random Number Generator
RSA – Rivest-Shamir-Adelman
SHA-1 – Secure Hash Algorithm
SSL – Secure Socket Layer
TLS – Transport Layer Security

10 References

FIPS PUB 46-3, Data Encryption Standard (DES).

FIPS PUB 197, Advanced Encryption Standard (AES).

FIPS PUB 186-2, Digital Signature Standard (DSS).

FIPS PUB 198, The Keyed-Hash Message Authentication Code (HMAC).

FIPS PUB 140-2, Security Requirements for Cryptographic Modules.

Annex A: Approved Security Functions for FIPS PUB 140-2, Security Requirements for Cryptographic Modules.

Derived Test Requirements for FIPS PUB 140-2, Security Requirements for Cryptographic Modules.

PKCS #1 v2.1: RSA Cryptography Standard. RSA Laboratories, June 14, 2002.

ANSI X9.42, Agreement of Symmetric Keys Using Discrete Logarithm Cryptography.

RFC 1321, The MD5 Message-Digest Algorithm.