PostX Corporation

Security Policy

Version 3.5

February 9, 2006

Contact Information

PostX Corporation
3 Results Way
Cupertino, CA 95014 - 5924
Phone: 408-861-3500
Website: www.postx.com

Trademarks

Distribution

Table of Contents

1. Introduction

This is a non-proprietary cryptographic module security policy for PostX Corporation's PostX FIPS Cryptography Kernel version 3.5 (Cryptography Kernel). This security policy describes how the Cryptography Kernel meets the security requirements of FIPS 140-2, and how to securely operate the Cryptography Kernel. This policy was prepared as part of the level 1 FIPS 140-2 validation of the Cryptography Kernel.

FIPS 140-2 (Federal Information Processing Standards Publication 140-2 — *Security Requirements for Cryptographic Modules*) details the U.S. Government requirements for cryptographic modules. More information about the FIPS 140-2 standard and validation program is available on the NIST website at http://csrc.nist.gov/cryptval/.

1.1. References

This document deals only with operations and capabilities of the Cryptography Kernel in the technical terms of a FIPS 140-2 cryptographic module security policy. More information is available on the Cryptography Kernel from the following resources:

- The PostX website contains information on their full line of products and services at http://www.postx.com
- An overview of the RSA BSAFE Crypto-J JCE module, from which the PostX FIPS Cryptography Kernel is derived, is located at http://www.rsasecurity.com/node.asp?id=1204
- The PostX product overview is provided at http://www.postx.com/products
- For answers to technical or sales related questions please refer to the contact details on page 2 of this document.

1.2. Document Organization

In addition to this document, the complete Submission Package contains:

- Executive summary
- Vendor evidence document
- Finite state machine
- Module software listing
- Other supporting documentation as additional references.

This document explains the Cryptographic Module's FIPS 140-2 relevant features and functionality. This first section, Introduction, provides an overview and introduction to the Security Policy. The section Cryptographic Module on page 6 describes the Cryptographic Module and how it meets the FIPS 140-2 requirements. Secure Operation of the Cryptography Kernel on page 10 specifically addresses the required configuration for the FIPS-mode of operation. Services on page 11 list all of the functions provided by the Cryptographic Module. Acronyms on page 12 define the acronyms used in this document.

With the exception of this Non-Proprietary Security Policy, the FIPS 140-2 Certification Submission Documentation is PostX-proprietary and releasable only under appropriate non-disclosure agreements. For access to these documents, please contact the PostX Corporation.

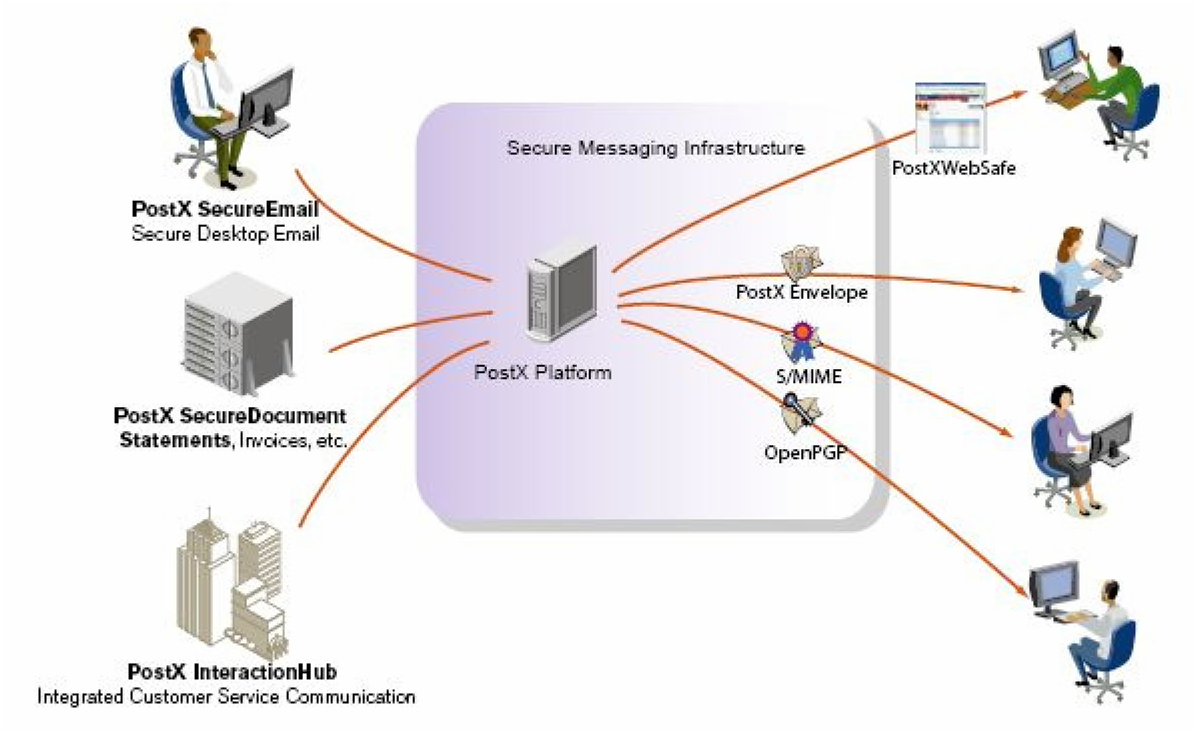
2. PostX FIPS Cryptography Kernel

This section provides an overview of the Cryptography Kernel. The following topics are discussed:

- Introduction
- Cryptographic Module
- Module Interfaces
- Roles and Services
- Cryptographic Key Management
- Cryptographic Algorithms
- Self-Test.

2.1. Introduction

Since PostX Corporation's inception as a pioneer in secure messaging, PostX has pursued a singular vision: to provide enterprises with secure email delivery of information essential to their business and customer relationships. Our production experience spans the financial services, telecommunications, insurance, and health care industries. PostX has enabled global organizations to meet the challenge of reconciling reliable and secure messaging with ease of use.



PostX security is based on the strongest, most widely accepted encryption algorithms. Each message is encrypted using a unique secure cryptographic key. Multiple encryption algorithms are supported, including ARC4 and AES, so you can select the encryption method that best meets your needs. For customers who require validated FIPS compliance, PostX has partnered with RSA Security Inc. to provide this cryptographic kernel, which is based on RSA BSAFE technology.

2.2. Cryptographic Module

This Cryptographic Module is classified as a multi-chip standalone module for FIPS 140-2 purposes. As such, the module must be tested upon a particular operating system and computer platform. The cryptographic boundary includes the Cryptographic Module running on selected platforms running selected operating systems while configured in "Single user" mode. The Cryptographic Module was validated as meeting all FIPS 140-2 level 1 security requirements, including cryptographic key management and operating system requirements. The Cryptography Kernel is packaged in a Java Archive (JAR) file, which contains all the code for the module. Additionally, the Cryptography Kernel relies on the physical security provided by the host PC in which it runs.

The JCE application programmer interface to the Cryptography Kernel is provided in the jsafeJCEFIPS.jar file.

The Cryptography Kernel was tested on the following platforms:

- Microsoft Windows
  - Microsoft Windows XP (SP2) – Sun JDK 1.4.2 (32 bit).

Compliance is maintained on platforms for which the binary executable remains unchanged including (but not limited to):

- Microsoft Windows
  - Microsoft Windows NT 4 Sun JDK
  - Microsoft Windows 2000 Service Pack 4 Sun JDK
  - Microsoft Windows XP (SP1) Sun JDK
  - Microsoft Windows 2003 Server. Sun JDK.

- Sun Microsystems
  - Sun Microsystems Solaris 8, Sparc v9 (32-bit) Sun JDK 1.4.2(32 bit)
  - Sun Microsystems Solaris 9, Sparc v9 (32-bit) Sun JDK 1.4.2(32 bit)
  - Sun Microsystems Solaris 10, Sparc v9 (32-bit) Sun JDK 1.4.2(32 bit)
  - Sun Microsystems Solaris 10, Sparc v9 (64-bit) Sun JDK 1.5 (64 bit)
  - Sun Microsystems Solaris 10, Intel x86 (32-bit) Sun JDK 1.4.2(32 bit).

- Red Hat Linux 7.2 x86 (32-bit)
  - Red Hat Linux 7.2 x86 (32-bit) Sun JDK 1.4.2
  - Red Hat Advanced Server 3.0 x86 (32-bit). Sun JDK 1.4.2.

- IBM AIX 5L v5.3 (32bit)
  - IBM AIX 5L v5.3, PowerPC (32-bit) IBM JDK 1.4.2.

Refer to the NIST document, *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program*, for resolution on the issue of "Multi user" modes. This document is located at: http://csrc.nist.gov/cryptval/140-1/FIPS1402IG.pdf.

2.3. Module Interfaces

As a multi-chip standalone module, the Cryptography Kernel's physical interfaces consist of the keyboard, mouse, monitor, serial ports, network adapters, and so on. However, the underlying logical interface to the cryptographic module is the Application Program Interface (API). The module provides for Control Input through the API calls. Data Input and Output are provided in the variables passed with API calls, and Status Output is provided in the returns and error codes that are documented for each call.

2.4. Roles and Services

The Cryptography Kernel meets all FIPS140-2 level 1 requirements for Roles and Services, implementing both a User role and Crypto-Officer role. As allowed by FIPS 140-2, the Cryptography Kernel does not require user identification or authentication for these roles. Only one role may be active at a time and the Cryptography Kernel does not allow concurrent operators.

The application programmer interfaces for control of the module are via the "CryptoJ" class.

JARFILE	PREFIX
jsafeJCEFIPS.jar	com.rsa.jsafe.crypto

2.4.1. Crypto-Officer Role

An operator may assume the Crypto-Officer (CO) role by invoking `<PREFIX>.CryptoJ.setRole()` method with the argument CRYPTO_OFFICER_ROLE.

Once in the CO role, the operator has the ability to start the power-up self-tests on demand by calling the `<PREFIX>.CryptoJ.runSelfTests()` method.

The CO can perform this operation manually at the command prompt by navigating to the directory containing the appropriate jar file, and typing:

```
java –cp <JARFILE> <PREFIX>.CryptoJ –testAll
```

Alternatively, the operation can be called programmatically:

```
<PREFIX>.CryptoJ.runSelfTests();
```

Note When the Cryptography Kernel is loaded, the power-up self-tests start to run. Once the integrity check is passed, they will not be run again unless the Cryptography Kernel is unloaded and reloaded. So, after the initial loading of the module, calling the self-tests on demand only results in the power-up known-answer tests and pairwise consistency checks being performed.

2.4.2. User Role

By default, an operator is in the User role. However, an operator may explicitly assume the User role by invoking the `<PREFIX>.CryptoJ.setRole()` method with the argument USER_ROLE.

The Cryptography Kernel API and its functions and capabilities are documented in the Services section on page 11.

2.5. Cryptographic Key Management

2.5.1. Key Generation

The Cryptography Kernel supports generation of the DSA, RSA, and Diffie-Hellman (DH) public and private keys. Furthermore, the module employs a FIPS 186-2 compliant random number generator for generating asymmetric and symmetric keys used in algorithms such as AES, DES, TDES, RSA, DSA or Diffie-Hellman.

2.5.2. Key Storage

The Cryptography Kernel does not provide long-term cryptographic key storage. If a User chooses to store keys, the User is responsible for storing keys returned to the calling application.

Volatile (short term) memory storage of cryptographic keys and CSPs employed by the cryptographic module is handled in the following manner:

Note The User and CO roles have equal and complete access to all keys and CSPs listed in the following table.

Table 1. Key and CSP storage

Item	Storage
AES keys	In volatile memory only (plaintext)
DES[1] keys	In volatile memory only (plaintext)
Triple DES keys	In volatile memory only (plaintext)
HMAC with SHA1 and SHA2 keys	In volatile memory only (plaintext)
Diffie-Hellman public key	In volatile memory only (plaintext)
Diffie-Hellman private key	In volatile memory only (plaintext)
RSA public key	In volatile memory only (plaintext)
RSA private key	In volatile memory only (plaintext)
DSA public key	In volatile memory only (plaintext)
DSA private key	In volatile memory only (plaintext)
PRNG seeds(FIPS 186-2)	In volatile memory only (plaintext)

2.5.3. Key Protection

All key data resides in internally allocated data structures and can only be output using the module's defined API. The operating system and Java Runtime Environment protects memory and process space from unauthorized access.

2.5.4. Key Zeroization

All key data resides in internally allocated data structures that are "cleaned up" by the Java Virtual Machine's (JVM) garbage collector. Java often handles memory in ways that are unpredictable and transparent to the User, a User can take additional steps to ensure sensitive data is properly zeroized by making use of the clearSensitiveData method for clearing sensitive data."

2.6. Cryptographic Algorithms

The Cryptography Kernel supports a wide variety of cryptographic algorithms. FIPS 140-2 requires that FIPS-defined algorithms be used whenever there is an applicable FIPS standard for the module operating in FIPS mode. Therefore, as the following table summarizes, only a subset of the algorithms provided by the Cryptography Kernel may be used in compliance with FIPS 140-2 requirements.

For more information on using the Cryptography Kernel in a FIPS compliant manner, refer to Secure Operation of the Crypto on page 10.

Table 3 lists the non-FIPS-approved algorithms.

[1] For legacy system use only; transitional phase only – valid until May 19th, 2007.

Table 2. Cryptography Kernel FIPS-approved algorithms

Algorithm	**Certificate Number**
AES – ECB, CBC, CFB (128), OFB (128) – [128, 192, 256 bit key sizes]	271
DES - ECB, CBC, CFB (64bit) , and OFB (64 bit) (for legacy use only)	326
Diffie-Hellman Key Agreement	Non-Approved (Allowed in FIPS mode)
Digital Signature Algorithm (DSA)	140
FIPS 186-2 General Purpose (x-Change Notice(SHA-1))	106
HMAC-SHA*x* (where x is 1, 224, 256, 384, or 512)	86
RSA PKCS#1 v1.5 (sign/verify) (SHA-1,SHA-224,SHA-256,SHA-384,SHA-512)	71
RSASSA-PSS (sign, verify) (SHA-1)	71
Secure Hash Standard (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512)	356
Triple DES - ECB, CBC, CFB (64 bit), and OFB (64 bit)	354
RSA X9.31 (keygen, sign, verify)	71

Table 3. Cryptography Kernel non-FIPS-approved algorithms

Algorithm
DESX
MD2
MD5
Random Number Generators (ANSI X9.31, MD5Random, SHA1Random)
The RC2® block cipher
The RC4® stream cipher
The RC5® block cipher
PBE with SHA1And3DES
RSA OAEP for key transport
Raw RSA encryption and decryption
RSA Keypair Generation MultiPrime (2 or 3 primes)
RIPEMD160
HMAC-MD5

2.7. Self-Test

The Cryptography Kernel performs a number of power-up and conditional self-tests to ensure proper operation. If any of these tests fails, the module throws a SecurityException, providing status output, and aborts the operation that caused the conditional self-tests to fail.

2.7.1. Power-Up Self-Tests

The power-up self-tests implemented in the Cryptography Kernel are:

- PRNG KATs
- AES KATs
- DES KATs
- TDES KATs
- SHA-1 KATs
- SHA-224 KATs
- SHA-256 KATs
- SHA-384 KATs
- SHA-512 KATs

- HMAC SHA-1 KATs
- HMAC SHA-224 KATs
- HMAC SHA-256 KATs
- HMAC SHA-384 KATs
- HMAC SHA-512 KATs
- pairwise consistency checks for DSA and RSA
- software/firmware integrity check.

Power-up self-tests are executed automatically when the module is loaded by the Java Runtime Environment (JRE).

2.7.2. Conditional Self-Tests

The Cryptography Kernel performs two conditional self-tests: a pair-wise consistency tests each time the module generates a DSA or RSA public/private key pair, and a continuous random number generator test each time the module produces random data per the FIPS 186-2 standard.

2.7.3. Mitigation of Other Attacks

RSA key operations implement blinding by default, providing a defense against timing attacks. Blinding is implemented through blinding modes, and the following options are available:

- Blinding mode off
- Blinding mode with no update, where the blinding value is constant for each operation
- Blinding mode with full update, where a new blinding value is used for each operation.

3. Secure Operation of the Cryptography Kernel

The Cryptography Kernel does not require any special configuration to operate in conformance with FIPS 140-2 requirements. However, the following guidance must be followed to achieve a FIPS mode of operation.

3.1. Crypto-Officer Guidance

The CO is responsible for installing the module.

Note The module's state is set to FIPS_MODE by default, and COs must not take the module out of FIPS_MODE.

3.2. User Guidance

The User must only use algorithms approved for use in a FIPS-mode of operation. Table 2 on page 9 lists the approved algorithms. The FIPS-approved bit length for a DSA key pair must be between 512-1024 bits in multiples of 64, and the FIPS-approved RNGs must be seeded with values of at least 160 bits in length.

The FIPS-approved bit lengths for an RSA[2] key pair must be between 1024 – 4096 bits in multiples of 512.

The FIPS-approved bit lengths for the Diffie-Hellman[3] key agreement must be between 1024-2048 bits.

The FIPS-approved bit lengths for an HMAC key must be between 80-4096 bits.

If RSA Key generation is requested in FIPS mode, the module always uses the FIPS-approved RSA X9.31 key generation procedure.

Note The module's state is set to FIPS_MODE by default, and Users must not take the module out of FIPS_MODE. Users should take care to zeroize CSPs when they are no longer needed.

4. Services

The Cryptography Kernel meets all FIPS 140-2 level 1 requirements for Roles and Services, implementing both a User role and Crypto-Officer role. As allowed by FIPS 140-2, the Crypto-J Module does not require user identification or authentication for these roles. Only one role may be active at a time and the Cryptography Kernel does not allow concurrent operators.

The following table details the services provided by the Cryptography Kernel in terms of the interface into the cryptographic module.

Table 4. Services for Crypto-J (jsafeJCEFIPS.jar)

Service	Service	Service
CryptoJ.runSelfTests*	AlgorithmParameters	Mac
CryptoJ.setRole	AlgorithmParameterGenerator	MessageDigest
CryptoJ.getRole	Cipher	SecretKeyFactory
CryptoJ.setMode	KeyAgreement	SecureRandom
CryptoJ.getMode	KeyFactory	Signature
CryptoJ.getState	KeyGenerator	
CryptoJ.selfTestPassed	KeyPairGenerator	

* Only available to the Crypto Officer role.

[2] When used for transporting keys and using the minimum allowed modulus size; the minimum strength of encryption provided is 80 bits.

[3] Using the minimum allowed modulus size; the minimum strength of encryption provided is 80 bits.

5. Acronyms

Acronym	Definition
AES	Advanced Encryption Standard. A fast block cipher with a 128-bit block, and keys of lengths 128, 192 and 256 bits. This will replace DES as the US symmetric encryption standard.
API	Application Programming Interface.
Attack	Either a successful or unsuccessful attempt at breaking part or all of a cryptosystem. Various attack types include an algebraic attack, birthday attack, brute force attack, chosen ciphertext attack, chosen plaintext attack, differential cryptanalysis, known plaintext attack, linear cryptanalysis, and middleperson attack.
CBC	Cipher Block Chaining. A mode of encryption in which each ciphertext depends upon all previous ciphertexts. Changing an IV alters the ciphertext produced by successive encryptions of an identical plaintext.
CFB	Cipher Feedback. A mode of encryption that produces a stream of ciphertext bits rather than a succession of blocks. In other respects, it has similar properties to the CBC mode of operation.
CSP	Cryptographic Service Provider.
DES	Data Encryption Standard. A symmetric encryption algorithm with a 56-bit key. See also Triple DES.
Diffie-Hellman	The Diffie-Hellman asymmetric key exchange algorithm. There are many variants, but typically two entities exchange some public information (for example, public keys or random values) and combines them with their own private keys to generate a shared session key. As private keys are not transmitted, eavesdroppers are not privy to all of the information that composes the session key.
DSA	Digital Signature Algorithm. An asymmetric algorithm for creating digital signatures.
ECB	Electronic Code Book. A mode of encryption in which identical plaintexts are encrypted to identical ciphertexts, given the same key.
Encryption	The transformation of plaintext into an apparently less readable form (called ciphertext) through a mathematical process. The ciphertext may be read by anyone who has the key that decrypts (undoes the encryption) the ciphertext.
FIPS	Federal Information Processing Standards.
HMAC	Keyed-Hashing for Message Authentication Code.
KAT	Known Answer Test.
Key	A string of bits used in cryptography, allowing people to encrypt and decrypt data. Can be used to perform other mathematical operations as well. Given a cipher, a key determines the mapping of the plaintext to the ciphertext. Various types of keys include: distributed key, private key, public key, secret key, session key, shared key, subkey, symmetric key, and weak key.
MD5	A secure hash algorithm created by Ron Rivest. MD5 hashes an arbitrary-length input into a 16-byte digest.
NIST	National Institute of Standards and Technology. A division of the US Department of Commerce (formerly known as the NBS) which produces security and cryptography-related standards.
OFB	Output Feedback. A mode of encryption in which the cipher is decoupled from its ciphertext.
OS	Operating System.

Acronym	**Definition**
PC	Personal Computer.
private key	The secret key in public key cryptography. Primarily used for decryption but also used for encryption with digital signatures.
PRNG	Pseudo Random Number Generator.
RC2	Block cipher developed by Ron Rivest as an alternative to the DES. It has a block size of 64 bits and a variable key size. It is a legacy cipher and RC5 should be used in preference.
RC4	Symmetric algorithm designed by Ron Rivest using variable length keys (usually 40 bit or 128 bit).
RC5	Block cipher designed by Ron Rivest. It is parameterizable in its word size, key length and number of rounds. Typical use involves a block size of 64 bits, a key size of 128 bits and either 16 or 20 iterations of its round function.
RNG	Random Number Generator.
RSA	Public key (asymmetric) algorithm providing the ability to encrypt data and create and verify digital signatures. RSA stands for Rivest, Shamir, and Adleman, the developers of the RSA public key cryptosystem.
SHA	Secure Hash Algorithm. An algorithm which creates a unique hash value for each possible input. SHA takes an arbitrary input which is hashed into a 160-bit digest.
SHA-1	A revision to SHA to correct a weakness. It produces 160-bit digests. SHA-1 takes an arbitrary input which is hashed into a 20-byte digest.
SHA-2	The NIST-mandated successor to SHA-1, to complement the Advanced Encryption Standard. It is a family of hash algorithms (SHA-256, SHA-384 and SHA-512) which produce digests of 256, 384 and 512 bits respectively.
TDES	Triple-DES