Rosetta CSI sToken

Security Policy

SPYRUS®

Rosetta CSI sToken

SECURITY POLICY

SPYRUS

1 October 2004

SPYRUS®

<info@spyrus.com>

<http://www.spyrus.com>

Document Number: 550-040004-07

Revision History

REV. #	DATE	DESCRIPTION
2 May 02	D0	Initial release to development team
24 May 02	D1	Update
20 Sept 02	D2	Release for Validation Bid
27 Sept 02	D3	Release for Lockheed Martin
16 Dec 02	D4	Pre-Validation Release for InfoGard
29 Jan 03	D5	Product name change to Rosetta CSI sToken. Update addressing initial comments from InfoGard
25 Mar 03	1.0	Update addressing validation comments from InfoGard
2 June 03	2.0	Final revisions addressing validation comments
25 June 03	3.0	Change of name of HMAC algorithm to in accordance with NIST implementation guidance as of 6/10/03, to "HMAC-SHA-1" on page 3.
9 Dec 03	4.0	Changes to sections 1.4, 3.1, 3.2, 5.2; tables 4.1, 6.1, 6.2 addressing comments from InfoGard
16 Dec 03	5.0	Changes to section 1.4, addressing comments from InfoGard
6 Aug 04	6.0	Requirement added to section 3.2 for feature change; Addition to RESTORE command description in section 4.2 to emphasize the support of hardware token restore operation.
24 Sept 04	7.0	Version numbers for Rosetta CSI sToken version 4.2.2.1 removed per CMVP comments.

Contents

1 Introduction

1.1 Rosetta CSI sToken Overview

Rosetta CSI sToken is intended to emulate, as closely as possible, the capabilities, processes, and procedures of the existing hardware-based FORTEZZA PCMCIA cryptographic token.

1.2 Rosetta CSI sToken Implementation

The Rosetta CSI sToken is a Software-only cryptographic module operating on a generic PC. As such, the Rosetta CSI sToken's evaluated physical embodiment is:

- Multi-chip Stand-alone as defined by FIPS 140-2.

The Rosetta CSI sToken operational environment is a Windows-based O/S including, but not limited to, Windows NT, 98, 2000, and XP.

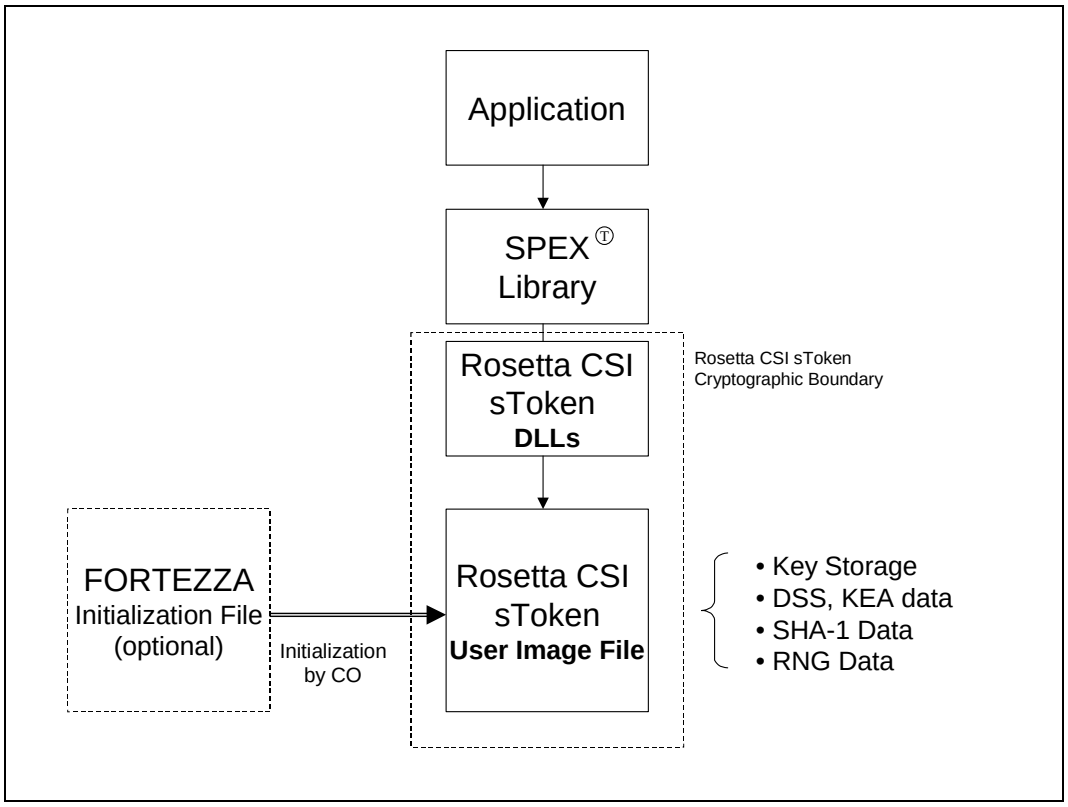

Figure 1 System Architecture and Logical Cryptographic Boundary of Rosetta CSI sToken.

1.3 Rosetta CSI sToken Cryptographic Boundary

The Cryptographic Boundary for the Rosetta CSI sToken is comprised of:

1. A 'physical' cryptographic boundary -- defined to be the edge of the generic PC, and
2. A 'logical' cryptographic boundary -- defined to be the Rosetta CSI sToken DLLs (the Algorithm and Crypto DLL's of the SPEX/ Library) written in "C" Language, and the Rosetta CSI sToken User Image File.

The logical cryptographic boundary is shown in Figure 1. No software components that comprise the Rosetta CSI sToken DLLs are excluded from the requirements of FIPS 140-2.

1.4 Approved Modes of Operations

The Rosetta CSI sToken 'FIPS-approved mode of operations' is comprised of the Rosetta CSI sToken command set. All commands that use FIPS 140-2 approved security functions (e.g. algorithms) are defined to be in the "FIPS approved mode of operation." By invoking the services of the sToken command set (section 4.2) using an initialized User Image File that is associated with the session, the user remains in an approved mode of operation. The Rosetta CSI sToken also has a "non-FIPS approved mode of operation".

The Rosetta CSI sToken supports the following FIPS 140-2 approved security functions in a FIPS approved mode of operation:

Encryption & Decryption
Skipjack
Digital Signatures
DSA / SHA-1
Key Transport / Key Agreement
KEA
Random Number Generation
Deterministic / X9.31, Appendix A.2.4 (TDES)

The following “non-FIPS approved mode” algorithms are present in the Rosetta CSI sToken:

Encryption & Decryption
DES, RC2, TDES
Digital Signatures
RSA
Hashing
MD5
Message Authentication Code
DES-MAC, HMAC-SHA-1

Non-FIPS approved algorithms may only be invoked using a DLL call to one of the Rosetta CSI sToken component DLLs if no User Image File is associated with the session.

2 FIPS 140-2 Security Levels

The Rosetta CSI sToken cryptographic module passes FIPS 140-2 validation to the levels defined in Table 2.1. The FIPS 140-2 overall rating of the Rosetta CSI sToken is Level 1.

Table 2-1
FIPS 140-2 Certification Levels

FIPS 140-2 Category	Level
1. Cryptographic Module Specification	1
2. Cryptographic Module Ports and Interfaces	1
3. Roles, Services, and Authentication	2
4. Finite State Model	1
5. Physical Security	N/A
6. Operational Environment	1
7. Cryptographic Key Management	1
8. EMI/EMC	3
9. Self-tests	1
10. Design Assurance	3
11. Mitigation of Other Attacks	N/A

3 Security Rules

The Rosetta CSI sToken enforces the following security rules. These rules are separated into two categories, 1) those imposed by FIPS 140-2 and, 2) those imposed by SPYRUS, Inc.

3.1 FIPS 140-2 Imposed Security Rules

1. The Rosetta CSI sToken shall indicate an approved mode of operation with a "0x0000" return code if the command was successfully executed. If the command is unsuccessful in the approved mode, the return code is "0xFFFF".
2. The Rosetta CSI sToken interfaces shall be logically distinct from each other.
3. The Rosetta CSI sToken shall support the following four (4) interfaces:
   - data input
   - data output
   - control input
   - status output
4. The Rosetta CSI sToken shall inhibit all data output via the data output interface whenever an error state exists and during self-tests.
5. The Rosetta CSI sToken shall logically disconnect the output data path from the circuitry and processes performing the following key functions:
   - key generation,
   - manual key entry, and
   - key zeroization
6. The Rosetta CSI sToken shall support a User role and a Cryptographic Officer role.
7. The Rosetta CSI sToken shall re-authenticate when it is powered-up after being powered-off.
8. The Rosetta CSI sToken shall provide the following services:
   - Reference Table 4.1.
9. The Rosetta CSI sToken shall provide a zeroization mechanism that can be performed either procedurally by the operator *or* automatically by the Rosetta CSI sToken (See also Rule #22 of this Section).
10. The Rosetta CSI sToken shall be entirely contained in a metal or hard plastic production-grade enclosure. The physical boundary of the Rosetta CSI sToken is the hard enclosure. The logical boundary of the Rosetta CSI sToken is the set of files (DLLs and Image File) shown in Figure 1.
11. The OS on which the Rosetta CSI sToken runs shall be restricted to a single operator mode of operation.
12. The Rosetta CSI sToken shall prevent access by other processes to plaintext private and secret keys, CSPs, and intermediate key generation values during the time the Rosetta CSI sToken is executing / operational.
13. Non-cryptographic processes shall not interrupt the Rosetta CSI sToken during execution.

14. The Rosetta CSI sToken shall implement a DSA digital signature on all cryptographic software within the Rosetta CSI sToken.
15. The Rosetta CSI sToken shall protect the following keys from unauthorized disclosure, modification and substitution:
    - secret keys.
    - private keys.
16. The Rosetta CSI sToken shall protect public keys against unauthorized modification and substitution.
17. The Rosetta CSI sToken shall generate keys using an approved FIPS 140-2 deterministic random number generator.
18. The Rosetta CSI sToken shall run a known answer test (KAT) on the approved deterministic random number generator at power-up.
19. The Rosetta CSI sToken shall use a 'seed input' into the deterministic random number generator of sufficient length that ensures at least the same amount of operations are required to determine the value of the generated key.
20. The Rosetta CSI sToken shall use a key establishment methodology that ensures at least the same amount of operations are required to determine the value of the transported/agreed upon key.
21. The Rosetta CSI sToken shall provide that:
    - a key entered into,
    - stored within, or
    - output from

    The Rosetta CSI sToken is associated with the correct entities to which the key is assigned.
22. The Rosetta CSI sToken shall provide the capability to zeroize all plaintext cryptographic keys and other unprotected critical security parameters within the Rosetta CSI sToken.
23. The Rosetta CSI sToken shall conform to the EMI/EMC requirements specified in FCC Part 15, Subpart J, Class B.
24. The Rosetta CSI sToken shall return the status of a command via the return code from each command.
25. The Rosetta CSI sToken shall perform the following self-tests:
    Power-up and on-demand tests:
    - Cryptographic algorithm KAT on all approved FIPS Algorithms and the deterministic random number generator:
      - Skipjack Algorithm KAT for modes ECB (64-bit), CBC (64-bit), OFB (64-bit), CFB (64-bit), CFB (32-bit), CFB (16-bit), CFB (8-bit);
      - SHA-1 KAT;
      - DSA KAT; and,
      - Power-up RNG test with known sample input and known output.
    - DSA Software Integrity test on loading of Rosetta CSI sToken DLLs.

    Conditional tests:
    - Pairwise consistency test (DSA).
    - Continuous random number generator test.

26. The Rosetta CSI sToken shall output a success return code (CI_OK, value = 0) via the status interface whenever self-test is successful.
27. The Rosetta CSI sToken shall enter an Error State and output an error indicator via the status interface whenever self-test is failed.
28. The Rosetta CSI sToken shall not perform any cryptographic functions while in an Error State.
29. The Rosetta CSI sToken shall inhibit all data via the data output interface when in an Error State.
30. The power-up tests shall not require operator intervention in order to run.
31. The Rosetta CSI sToken shall provide an indication via the "status output" interface if all of the power-up tests are passed successfully.
32. The Rosetta CSI sToken shall be under a configuration management system and each configuration item shall be assigned a unique identification number.
33. The Rosetta CSI sToken source code shall be annotated.
34. The Rosetta CSI sToken documentation shall provide Crypto Officer and User Guidance per FIPS 140-2, Section 4.10.4.

3.2 SPYRUS Imposed Security Rules

1. The Rosetta CSI sToken shall not provide a maintenance role/interface.
2. The Rosetta CSI sToken shall not support a bypass mode.
3. The Rosetta CSI sToken shall not be required to mitigate any specific attacks.
4. Reconfiguration of the Windows Registry shall allow storage in the Rosetta CSI sToken for a fixed number greater than the default maximum of 50 certificates, and less than a highest recommended maximum of 8000 certificates.
5. The Rosetta CSI sToken shall allow the following operations without authentication:
    - Random Number Generation;
    - Zeroization of the Token;
    - Opening a socket connection and Initialization of the Rosetta CSI sToken (Open Image File);
    - Closing a socket connection;
    - Selecting a socket;
    - Terminating a connection (closing the library and socket);
    - Locking the Rosetta CSI sToken (for the current application and socket);
    - Unlocking the Rosetta CSI sToken (for the current application and socket);
    - Getting the current configuration (Get Configuration);
    - Getting the current state (Get State);
    - Getting the current status (Get Status);
    - Getting the current time (Get Time); and,
    - Resetting the Rosetta CSI sToken (Reset).

4 Rosetta CSI sToken Roles and Services

4.1 Roles

The Rosetta CSI sToken supports the following two roles:
- Crypto-officer (also called Site Security Officer (SSO)) and
- User

The Rosetta CSI sToken enforces the separation of these roles by requiring a PIN to be input to access the set of services available to each role.

Crypto-officer Role: The Crypto-officer (CO) is responsible for initializing the Rosetta CSI sToken. Initialization is typically performed using a Certificate Authority Workstation (CAW) that is secured according to the site security policy of the deploying organization. In some of the FORTEZZA literature, the CO is also referred to as the site security officer (SSO).

Before issuing a Rosetta CSI sToken to an end user, the Crypto-officer initializes the Rosetta CSI sToken with private keying material and certificate information.

User Role: The User role is available after the Rosetta CSI sToken has been initialized by the Crypto-officer.

4.2 Services

The following table describes the services provided by the Rosetta CSI sToken

Table 4.1
Rosetta CSI sToken Services

Service	Description
Change PIN Phrase	The *Change PIN Phrase* command enables the CO to change either the User PIN or CO PIN in a Rosetta CSI sToken. The CO must provide the original and new PIN phrases. Only the CO is authorized to execute the *Change PIN Phrase* command. When the User role PIN is successfully changed or the command fails, the CO is automatically logged out. When the CO role PIN is changed, the CO remains logged on.
Check PIN Phrase	The *Check PIN Phrase* command inputs a PIN Phrase to authenticate the CO or the User.
Close	The *Close* command closes the specified Socket. The *Close* command will also release a

Service	Description
	Lock.
Decrypt	The *Decrypt* command supports many decryption modes. The command requires that an 8-byte header precede data to be encrypted or decrypted. The Rosetta CSI sToken software adds the header to both plaintext and ciphertext data areas. The host software must reserve 4 bytes.
Delete Certificate	The *Delete Certificate* command overwrites with nulls the data in an indexed storage location. The *Delete Certificate* command passes the certificate index to the Rosetta CSI sToken; after the data is deleted, the corresponding certificate index flag is cleared.
Delete Key	The *Delete Key* command clears the key in the indexed key register. The *Delete Key* command passes the register index to the Rosetta CSI sToken. Key Register 0 contains K_s; therefore, Key Register 0 is not valid for the *Delete Key* command. Deleting an unused Key Register is permitted and in this case, the Delete Key command will indicate success.
Encrypt	The *Encrypt* command supports many encryption modes. Each mode requires a specific byte length and the host software pads data with zeros to reach the required length. The command requires that an 8-byte header precede data to be encrypted or decrypted. The Rosetta CSI sToken software adds the header to both plaintext and ciphertext data areas.
Extract X	The *Extract X* commands archives a private X value, supports remote re-key operations and duplicates personalities. A password protects the key material. Only X values loaded or generated by the CO can be extracted.
Generate IV	The *Generate IV* command generates, checks and writes the 192-bit IV value into the cryptographic data structures. The output from the Generate IV command is also written into the Data-Out area. A Set Key command must be executed prior to this command.
Generate MEK	The *Generate MEK* (Message Encryption Key) command uses the internal random number generator to produce a key for encrypting messages. The command results are stored in

Service	Description
	the key register index designated for future use. The MEK is not available in plaintext outside the Rosetta CSI sToken. The MEK is available for use immediately after it is generated.
Generate Ra	The *Generate Ra* command is used with *Generate TEK* to support the public/private key exchange. The *Generate Ra* command generates a 1024-bit random value that is written to the Data-Out Block.
Generate Random Number	The *Generate Random Number* command generates a 32-bit random number and returns it in the Data-Out Block. This command can be executed before logging on to a Rosetta CSI sToken.
Generate TEK	The *Generate TEK* (Token Encryption Key) command supports public/private key exchanges. Both initiator and recipient use the command after the key exchange. The *Generate TEK* command generates an 80-bit key based on the parameters passed with the Data-In Block. The key is stored in the key register index indicated.
Generate X	The *Generate X* commands allows the CO or user to generate unique X & Y components before loading a certificate. The certificate index and the component type to be generated (KEA, DSA or both) are parameters passed in the Data-In Block.
Get Certificate	The *CI_GetCertificate* command returns the 2048 bytes of data associated with the certificate index specified by CertificateIndex. Any 2048-byte block of data can be stored. The certificate label returned by the *CI_GetPersonalityList* command indicates the contents of the data. Certificate data may be in any format because the Rosetta CSI sToken does not read the certificate.
Get Configuration	The *Get Configuration* command returns a structure which contains: • Crypto Interface Library Version. • The Rosetta CSI sToken Manufacturer's name. • The Rosetta CSI sToken 's product name. • The number of bytes of User RAM. • The size, in bytes of the largest block of data

Service	Description
	that may be passed to a function. • Key Register Count, The number of Key Registers on the Rosetta CSI sToken. • Certificate Count, The maximum number of Certificates that the Rosetta CSI sToken can store (including Certificate 0). • A flag that if non-zero indicates that there is a Rosetta CSI sToken associated with the socket. If this value is zero then there is **not** a Rosetta CSI sToken associated with the socket. • The ICD Compliance level. • The Manufacturer's Software Version Device Driver Version.
Get Hash	The *Get Hash* command returns the current hash value for a block(s) of hashed data. The command expects the last or only block of data to be hashed as input. If the block hash is on a 512-bit boundary, a last block of zero length is valid. The Rosetta CSI sToken maintains the entire length of the hashed message.
Get Personality List	The *Get Personality List* command provides the host system with a list of stored personalities. The Rosetta CSI sToken returns an entry for each certificate location, including Index 0. All the certificate names stored in User Image File memory are sent to the host for the user to select and are 32 bytes in length. Locations not loaded with a valid certificate return a NULL (0000 0000h) value for all 32 bytes.
Get State	The *Get State* command returns the execution state of the Rosetta CSI sToken. This function may be called at any time, regardless of if the Rosetta CSI sToken has been initialized or logged on to.
Get Status	The *Get Status* command allows the CO or user to obtain the current status of the Rosetta CSI sToken. Status information returned includes current state, serial number of the Rosetta CSI sToken instantiation, mode, personality; key registers in use and certificate slots in use.
Get Time	The *Get Time* command enables a user or CO to retrieve the time from the operating system. In the command format, "XX" indicates unused digits filled with zeros (XX=00H).

Service	Description
Hash	The *Hash* command hashes the data provided with the command. It continues to hash using the current hash state as a starting point. The hash value may be reset by the *Initialize Hash* command or restored by the *Restore* command. The *Hash* command does not output the hash value. To obtain the hash value, use the *Get Hash* command.
Initialize	The *Initialize* command initializes the Rosetta CSI sToken Library. All other function calls will return an error code if they are called before this function.
Initialize Hash	The *Initialize Hash* command initializes the on-board hash function according to the NIST Secure Hash Standard.
Install X	The *Install X* command is used by either the CO or a user to: ▪ Restore an archived, private DSA or KEA *X* value ▪ Support remote re-key operations ▪ Duplicate personalities between Rosetta CSI sToken s A password is required to verify positive control of the instantiation of key material.
Load Certificate	The *Load Certificate* command enables the CO or a user to load user certificates for storage in a Rosetta CSI sToken 's non-volatile memory. The certificates loaded define the personalities available to the Rosetta CSI sToken user. The Get Configuration command indicates the maximum number of certificates that can be stored in a Rosetta CSI sToken. After a *Generate X* or *Load X* command has generated the certificate's unique X and Y values, the user's certificate is loaded in the index location containing the X value and associated with a user personality name. The certificate label is included as a 32-byte character string.
Load DSA Parameters	The *Load DSA Parameters* command enables a user to load externally supplied p, q and g parameters for signature verification outside the domain of the currently selected personality.
Load Initialization Values	The *Load Initialization Values* command enables

Service	Description
	the CO to load the Rosetta CSI sToken 's initialization parameters—a 64-bit random seed value and a 80-bit user storage key, K_s.
Load IV	The Load IV command writes the 192-bit value of an IV into the decryption data structures. A Set Key command must be executed prior to this command. The command is used before an *Encrypt* or *Decrypt* command.
Load SWF InitValues	The *Load SWF Init Values* command allows the CO to load the parameter for the random number generator and to load the Initialization PINPhrase for use with the Rosetta CSI sToken Initialization File.
Load X	The *Load X* commands enables the CO or user to load the secret X value in the Rosetta CSI sToken during initialization. For the X value loaded, the command must specify the associated certificate and the mode (DSA or KEA).
Lock	The *Lock* command grants an application with exclusive access to the currently selected socket and its Rosetta CSI sToken.
Open	The *Open* command opens a Socket. Sockets are numbered from one (1) to SocketCount. (SocketCount is returned by the *Initialize* command, and indicates the number of available sockets at initialization time). All subsequent commands will be issued to the socket opened. A Rosetta CSI sToken does not need to be in the socket before executing this command.
Open Image File	The *Open Image File* command opens a Socket and associates a Rosetta CSI sToken User Image File with the Socket. Pre-configured sockets are numbered from one (1) to SocketCount. However, Rosetta CSI sToken also allows the creation of new software sockets by simply specifying a socket number that is not in the pre-configured range. All subsequent commands will be issued to the socket opened.
Relay	The *Relay* command is used to restore an archived private X value, support remote re-key operations and duplicate personalities. A password is required to verify positive control of the instantiation of key material.
Reset	The *Reset* command will reset the Rosetta CSI

Service	Description
	sToken. All registers and common memory are zeroized. The User or SSO Enabled User is logged off of the Rosetta CSI sToken. The *Reset* command does not terminate the Rosetta CSI sToken Library.
Restore	The *Restore* command renews the state of the cryptologic operation as specified by the Crypto Type input parameter. If the Data-In Block contains data, the Rosetta CSI sToken uses the value of the input parameter as the input value. If the length of the restore data is zero, the Rosetta CSI sToken uses the stored data (the result of a *Save* command). The saved cryptologic of a Fortezza hardware token can be loaded with this command, in compliance to the Software Fortezza requirements.
Save	The *Save* command keeps the state of the cryptologic operation specified by the Crypto Type input parameter.
Select	The *Select* command allows the user to select from any of the currently open sockets.
Set Key	The *Set Key* command selects a Key Register for following commands that will use the contents of that register for encryption/decryption and related operations.
Set Mode	The *Set Mode* command sets the encrypt and decrypt command modes.
Set Personality	The *Set Personality* command enables a user to select a personality and can be executed any time during a session.
Sign	The *Sign* command computes the digital signature values r and s over the host provided data. The data is signed with the private key associated with the certificate selected by the *Set Personality* command.
Terminate	The *Terminate* command closes the Rosetta CSI sToken Library. The *Terminate* command first closes any open Sockets, and then closes the communication link with the Rosetta CSI sToken Socket Services. Note that *Terminate* calls the Close function and resets the Rosetta CSI sToken, for each open socket.
Timestamp	The *Timestamp* command computes the digital signature values r and s over the host provided

Service	Description
	hash value and the date and time obtained from the on-board clock.
Unlock	The *Unlock* command releases an application's exclusive access, established by the *Lock* command, to the currently selected socket and its Rosetta CSI sToken.
Unwrap Key	The *Unwrap Key* command reveals the key data using the key indicated by the register index. After the key is unwrapped, the Rosetta CSI sToken performs a checkword test and compares the generated and unwrapped values.
Verify Signature	The *Verify Signature* command validates a digital signature that has been received.
Verify Timestamp	The *Verify Timestamp* command validates a digitally signed date/time value that has been received.
Wrap Key	The *Wrap Key* command uses the key indicated by the first register index to wrap the key indicated by the second register index. The result of the *Wrap Key* command is returned in the Rosetta CSI sToken 's Data-Out Block as a 96-bit value. A 16-bit checkword is generated over the key that will be wrapped. Both the checkword and the key are wrapped.
Zeroize	The *Zeroize* command clears the Rosetta CSI sToken 's data and internal buffers, key management information, and puts the Rosetta CSI sToken in the zeroized state.

5 Identification and Authentication

5.1 Initialization Overview

The FORTEZZA Crypto Card is discussed in this section to make a comparison between the initialization of the Rosetta CSI Token and those used in the FORTEZZA card technology. The FORTZZA Crypto Card is a PCMCIA card that contains a crypto processor and a set of data maintained in non-volatile memory. The processor is programmed to manage all the interactions between the FORTEZZA Card and an external/host processor, as well as to perform all of the required cryptographic operations. The data on the card is the information necessary to identify the card and its user in all the crypto operations being performed. This data includes unique card identifier information, public/private key pairs, certificates, certificate labels, a random seed, a storage key, PIN data, and state management information. To get this data onto a card the CO programs the card with initialization information, causing the card to transition through a series of *state*s. This information and these states and their transitions are fully described in the *FORTEZZA Application Implementor's Guid*e (AIG). The source of the data and the procedures enforced during the initialization of cards are a matter of security policy and may vary from one operational implementation to the next. In a typical operational scenario the information for a card is generated using a Certificate Authority Workstation (CAW). This information is directly programmed onto the card. The card is then issued to the user who is provided with the PIN used to access the card.

In Rosetta CSI sToken, this initialization sequence is varied slightly because of the difference between a "hardware" token and a "software" token. The Rosetta CSI sToken token is an implementation specific file on the user's workstation that contains the appropriate data necessary to define the user for Rosetta CSI sToken. This file is called the *User Image Fil*e. It contains the equivalent of the information stored in a FORTEZZA Crypto Card's non-volatile memory. This Image File could be created directly on the user's workstation using commands found in the Rosetta CSI sToken Library. For security, convenience, portability, and flexibility in key management, an intermediate-platform independent data interchange file has been defined. This file is called the Rosetta CSI sToken Initialization File. Making use of the Rosetta CSI sToken Initialization File allows the Rosetta CSI sToken initialization process to be divided into two steps:
(1) generation of a Rosetta CSI sToken Initialization File, and
(2) creation of the User Image File from a Rosetta CSI sToken Initialization File.

To support the Rosetta CSI sToken initialization process as described in this document, two new commands have been added to the SPEX/ Cryptographic Library:

- CI_LoadSWFInitValues
- CI_OpenImageFile

See Table 4.1 for descriptions of the services provided by the above functions.

5.1.1 Generating Rosetta CSI sToken Initialization Files

Generation of a Rosetta CSI sToken Initialization File may be accomplished in a variety of ways; but for most operational systems it will involve the use of a Certificate Authority Workstation (CAW). A common series of steps is generally performed during the generation and issuance of a Rosetta CSI sToken Initialization File:

1. The user begins this procedure by requesting a new Rosetta CSI sToken Certificate.
2. The Certificate Request is processed.
3. Upon approval of the Certificate Request, a platform independent file that will be used to initialize Rosetta CSI sToken is created. This platform independent file is the Rosetta CSI sToken Initialization File. It is a data interchange file and contains the data required to create an instance of Rosetta CSI sToken regardless of the implementation or the platform on which it is running. This data includes public/private key pairs, certificates, certificate labels, a random seed, and a storage key. Sensitive information in this file will be encrypted using a storage key (Ks), which in turn is encrypted using a PIN. Access to this storage key will then be protected using an initialization key that is protected using an Initialization PIN.
4. The Rosetta CSI sToken Initialization File will be distributed to the user via an appropriate channel subject to security policies that are in place. This may be on a floppy disk or some other portable medium.
5. Two PINs will be generated for the Rosetta CSI sToken Initialization File. Depending on security policies in effect these may be different or the same.
6. Once received by the CO with initializer access to the Initialization File (i.e. the Initialization PIN), the Rosetta CSI sToken Utility program will use the contents of an Rosetta CSI sToken Initialization File to create a User Image File as described below. The User Image File defines the user for Rosetta CSI sToken. The contents of the Rosetta CSI sToken Initialization File, which are established by the CAW are not updated nor used directly by Rosetta CSI sToken.

5.1.2 Creating the User Image File

Within Rosetta CSI sToken, the token which contains a user's credentials is a User Image File. This is simply a file on the user's workstation that contains the appropriate data necessary to define the user for Rosetta CSI sToken. It may be thought of as the user's Rosetta CSI sToken token because it is equivalent to the non-volatile memory portion of a hardware FORTEZZA Crypto Card. All of the

commands necessary to create a User Image File are available within Rosetta CSI sToken. This allows the initialization sequence used to program a FORTEZZA Crypto Card to also be used to create and initialize a Rosetta CSI sToken User Image File directly on the user's workstation. The actual data that is loaded will depend on some options selected by the CO. The standard utility that creates and initializes a new User Image File with the data provided within an Initialization File is not provided.

5.2 Role Authentication

Role authentication is accomplished by PIN entry by the user. On invocation by the user, the Rosetta CSI sToken waits for authentication of the user or CO role by entry of a PIN phrase. Once a valid PIN phrase has been accepted the Rosetta CSI sToken waits for the Personality to be set. Setting a Personality renders the Rosetta CSI sToken ready for user commands. If the user fails to logon to the Rosetta CSI sToken in 10 consecutive attempts, the Rosetta CSI sToken will zeroize the User PIN and then transitions to the CAW Initialized State. It is possible for the user to fail to logon in this way, since the CO sets the initial User PIN value. To restore operation to the Rosetta CSI sToken, the CO Enabled User or CO will have to reload the initialization parameters and User PIN phrase. If the SSO Enabled User fails to logon to the Rosetta CSI sToken in 10 consecutive attempts, the Rosetta CSI sToken will zeroize all of the certificates, Private Components, Key Registers and disallow User access. When the Rosetta CSI sToken is invoked after a zeroize command, it will power up and transition to the Zeroize State, where it will only accept the Zeroize Default PIN phrase. After the Zeroize Default PIN phrase has been accepted, the Rosetta CSI sToken transitions to the Un-initialized State and must be reinitialized, as described in section 5.1.

5.3 Strength of Authentication

The strength of the authentication mechanism conforms to the following specifications.

5.3.1 Single Random Attempt

For each attempt to use the authentication mechanism the probability is less than one in 1,000,000 that a random attempt will succeed or a false acceptance will occur (e.g., guessing a password or PIN).

The mandatory minimal PIN size is 4 characters. If a random attempt to perform the CheckPIN command for either a CO or user (i.e., login), the probability of success on that single attempt is equal to the twice the reciprocal of the size of the PIN-space. This is due to the fact that there are normally two operant PINs

that allow access at any time (unless the Rosetta CSI sToken has not been initialized for the user, in which case only the CO PIN is operable). Thus if there are N possible PINs available, the probability of success in entering the CO or User PIN is $2 / N^4$. The character set available for PINs is at least all Alphanumeric characters (upper and lower cases) and 31 special keyboard characters comprising the set {~ ! @ # $ % ^ & * ( ) _ + - = { } [ ] | \ : ; ” ’ < , > . ? /}. This results in a PIN-space of $(26 + 26 + 10 + 31)^4 = 93^4 = 74,805,201$. The probability of a single successful random attempt is therefore: $1.336805445 \times 10^{-8}$ to 10 significant figures, which is less than one in 1,000,000, or 1.0×10^{-6}.

5.3.2 Multiple Attempts

For multiple attempts to use the authentication mechanism during a one-minute period, the probability is less than one in 100,000 that a random attempt will succeed or a false acceptance will occur.

The probability of success of a multiple entry attack in one minute is limited by the policy that no more than 10 attempts at login are permitted. Any more attempts would result in transition to the user un-initialized state, or in the case of the CO, transition to the zeroized state. In either case, no further PIN entries could be successful.

To determine the probability of success of a multiple attack, the following observation is relevant: the attack is decomposed into ten mutually exclusive events that entail the success on the i^{th} try after i - 1 unsuccessful attempts. Thus the probability of ultimate success over a set of ten mutually exclusive and exhaustive events is the sum of the probabilities of the 10 attack events described.

For i ranging from 1 to 10, the i^{th} attack event has a probability equal to the probability P_i of i - 1 unsuccessful previous attacks multiplied by the probability of a successful attack on the i^{th} try. Thus $P_i = [(N – i)/N] \times [1 / (N – i)]$, or by cancellation, 1 / N, where N is the size of the PIN-space calculated in 5.3.1. The total probability of a multiple attack, given that either CO or User PIN may be guessed, is then 2 x 10 / N, or 20/N. Thus, the probability of a multiple attack resulting in success (i.e., entering either the user or CO PIN), is $2.673610890 \times 10^{-7}$ to 10 significant figures. This is less than one in 100,000, or 1.0×10^{-5}, as required.

5.3.3 Obscuration of Feedback

Feedback of authentication data to an operator is obscured during authentication (e.g., no visible display of characters result when entering a password). The PIN value is input to the CheckPIN command as a parameter by the calling application. No return code or pointer to a return value that contains the PIN is

provided. The PIN value is furthermore not stored in plaintext or encrypted form within the Rosetta CSI sToken, and cannot be returned at a later time.

5.3.4 Non-weakening Effect of Feedback

Feedback provided to an operator during an attempted authentication shall not weaken the strength of the authentication mechanism. The only feedback provided by the CheckPIN command is a return code denoting success or failure of the operation. As is evident in the calculations of 5.3.1 and 5.3.2, this information in no way affects the probability of success or failure in either single or multiple attacks.

6 Access Control

6.1 Critical Security Parameters (CSPs)

Table 6.1
Rosetta CSI sToken CSPs

CSP Designation	Algorithm(s) / Standards	Symbolic Form	Description
Private Key	DSA/KEA	X	The Private Key of the User employed in digital signing operations; generated by RNG.
Secret Keys	SKIPJACK	(see also K_s, MEK, TEK)	The symmetric encryption key used in Skipjack encryption and decryption operations
Storage Key	SKIPJACK	K_s	Internal Storage Key.
Key Encryption Key	SKIPJACK	KFEK	Internal key for encrypting the internal storage key K_s
Message Encryption Key (MEK)	SKIPJACK	MEK	Generated by the Rosetta CSI sToken RNG for data encryption or wrapping of keys.
Token Encryption Key (TEK)	KEA, SKIPJACK	TEK	Used with KEA for public/private key exchange; requires Ra, Rb and either initiator's or recipient's Y value depending on whether the user is the initiator or the recipient in the key transfer.
KEA 1024-bit Random Numbers	KEA, SKIPJACK	Ra, Rb	Generated by the Rosetta CSI sToken RNG for key exchange operation (See TEK).
Initialization Random Seed	X9.31		Internally, a 64-bit RNG seed is generated at Initialization. This is transferred to the User Image File in wrapped form.
Random Number Generator key	TDES, X9.31	*K	A 112-bit constant-value TDES key that is uniquely generated at Rosetta CSI sToken installation. .
User Role PIN Phrase	N/A	PIN	A secret 12-byte value used for user authentication.
Zeroize Default PIN	N/A	PIN	A secret 12-byte value used by CO recovery from a zeroized Rosetta CSI sToken
CO Role PIN Phrase	N/A	PIN	A secret 12-byte value used for CO authentication.

6.2 Other Key Management Parameters

Table 6.2
Other Rosetta CSI sToken Key Management Parameters

CSP Designation	Algorithm(s) / Standards	Symbolic Form	Description
Public Key	DSA/KEA	Y	The Public Key of the user employed in digital signature verification operations; generated from X, p, and g -values.
DSA Private Prime Factor	DSA/KEA	g	A public selected element of the group Z_p* used in generation of the DSA public key
DSA 64 -128 byte Prime Modulus	DSA/KEA	p	A public selected prime number used as the modulus in the generation of the DSA public key
DSA 160-bit Prime Divisor	DSA/KEA	q	A public selected prime number used in the generation of the DSA private key
DSA 320-bit Digital Signature		(r,s)	A public pair of 160-bit values, denoted r and s, that authenticate a data block or message to have been signed using the originator's private key.
Hash Inputs	SHA-1		The input to SHA-1; may be public or secret, according to the use to which the hashed value is put.
Hash outputs	SHA-1		The output to SHA-1; may be public or secret, according to the use to which the hashed value is put.
PIN Phrase Initialization Vector (IV)	N/A	IV	Internally, an IV is generated at Initialization to wrap the Ks key. This is transferred to the User Image File in wrapped form.
FORTEZZA Certificate	FORTEZZA-specific design		The Public Key certificate storage structure in the sToken User Image File
X.509 Certificate	X.509		The X.509 Certificate field in the FORTEZZA Certificate
Real-Time Clock Value	N/A		The current system time value used in time stamping operations. This value is a CSP once it is imported within the sToken cryptographic boundary

6.3 CSP Access Type

Table 6.3
Rosetta CSI sToken Access Types

Access Type	Description
Generate (G)	“Generate” is defined as the creation of a CSP
Delete (D)	“Delete” is defined as the zeroization of a CSP
Use (U)	“Use” is defined as the process in which a CSP is employed. This can be in the form of loading, encryption, decryption, signature verification, or key wrapping.

6.4 Access Matrix

The following table shows the services (see section 4.2) of the Rosetta CSI sToken, the roles (see section 4.1) capable of performing the service, the CSPs (see section 6.1) that are accessed by the service and the mode of access (see section 6.3) required for each CSP.

Table 6.4
Rosetta CSI sToken Access Matrix

Service Name	Roles			
	CO	User	CSPs	Access Mode
Change PIN Phrase	X		PIN (old) PIN (new) Ks KFEK	U,D G U G,U,D
Check PIN Phrase	X	X	PIN Ks KFEK	U U G, U, D
Close	X	X		
Decrypt		X	Secret Key	U
Delete Certificate	X	X		
Delete Key		X	MEK/TEK	D
Encrypt		X	Secret Key	U
Extract X	X		X Ks KFEK	U U G, U, D
Generate IV		X		
Generate MEK		X	MEK	G
Generate Ra		X	Ra	G
Generate Random Number	X	X	Random Number *K	G U
Generate TEK		X	Ra or Rb TEK X KFEK	U G U G, U, D
Generate X	X	X	X *K KFEK	G U G, U, D
Get Certificate	X	X		
Get Configuration	X	X		
Get Hash		X		
Get Personality List	X	X		
Get State	X	X		

Service Name	Roles			
	CO	User	CSPs	Access Mode
Get Status	X	X		
Get Time	X	X		
Hash		X		
Initialize	X	X		
Initialize Hash		X		
Install X	X	X	X Ks KFEK	U U G, U, D
Load Certificate	X	X		
Load DSA Parameters		X		
Load Initialization Values	X		Initialization Random Seed *K Ks KFEK	U G U G, U, D
Load IV		X		
Load SWF InitValues	X			
Load X	X	X	X Ks KFEK	U U G, U, D
Lock	X	X		
Open	X	X		
Open Image File	X	X		
Relay	X	X	X Ra TEK Ks KFEK	U U G U G, U, D
Reset	X	X		
Restore		X		
Save		X		
Select	X	X		
Set Key		X	MEK/TEK	U
Set Mode		X		
Set Personality	X	X		
Sign		X	X Ks KFEK	U U G, U, D
Terminate	X	X		

Service Name	Roles			
	CO	User	CSPs	Access Mode
Timestamp		X	X Ks KFEK	U U G, U, D
Unlock	X	X		
Unwrap Key		X	TEK/MEK	U
Verify Signature		X	X Ks KFEK	U U G, U, D
Verify Timestamp		X	X Ks KFEK	U U G, U, D
Wrap Key		X	TEK MEK	U U
Zeroize	X	X	All User Image File CSP values (see Table 6.1)	D

