VIA3 VkCrypt 4.2 and 6.0 ***Security Policy***

VIACK Corporation

Document Revision: 1.1
Date: 01/30/08

TABLE OF CONTENTS

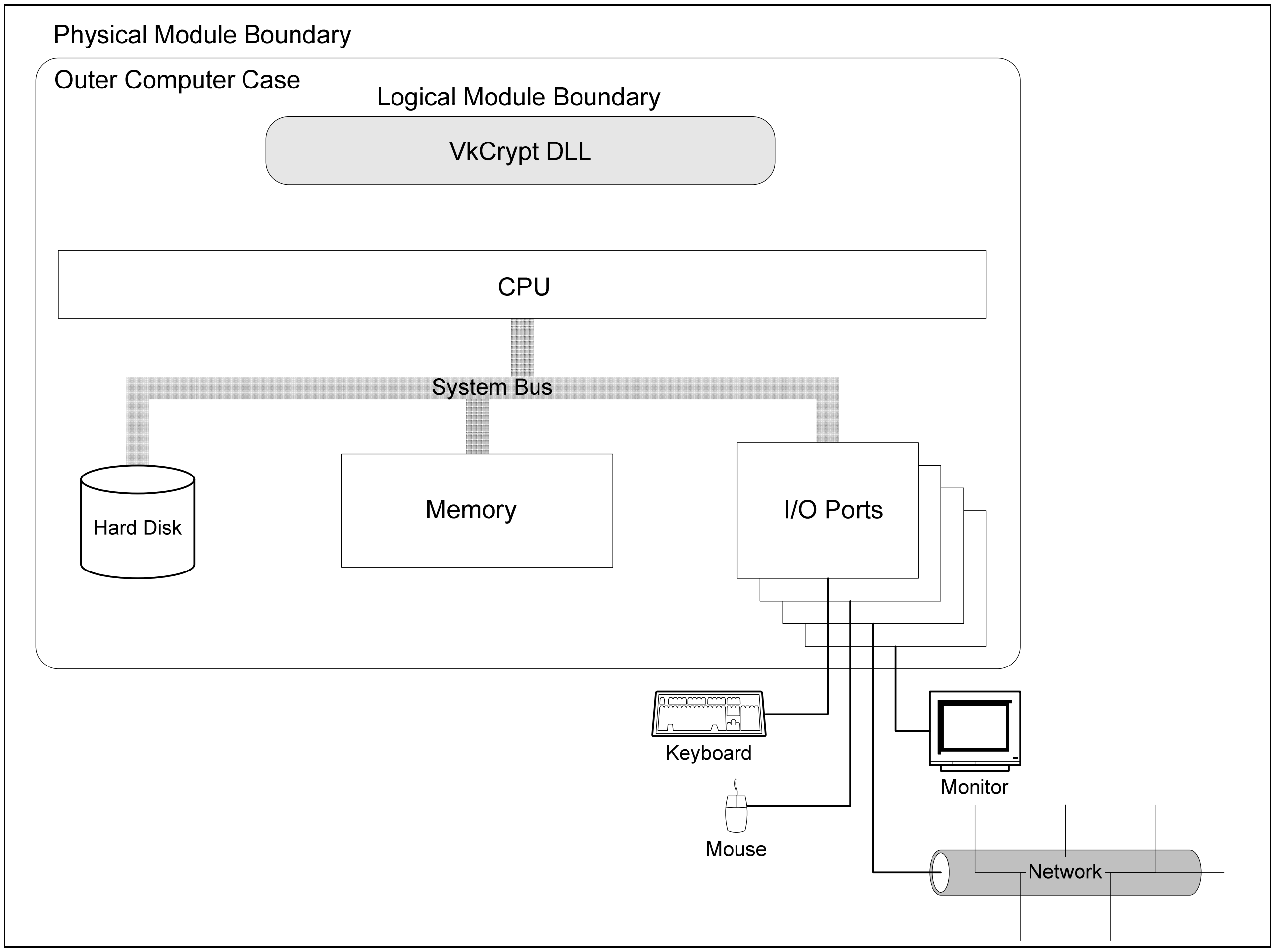
1. Module Overview

VIA3 VkCrypt is a software-based cryptographic module packaged as a single dynamically linked library (DLL) named VkCrypt.dll (Versions 4.2 and 6.0) that runs on Microsoft Windows operating systems. It provides a C++ Application Programming Interface (API) to access the cryptographic functionality. The module has been validated on Microsoft Windows XP. Although not officially tested by the testing laboratory, the *validated* module can also execute (without modification) on Windows 2000, Windows 2003, and Windows Vista.

For the purposes of FIPS 140-2 validation, the module is considered a *multi-chip standalone* device. The *physical boundary* is defined as the outer case of the general purpose computer system on which the module is executed. The *logical boundary* is the API of the module.

The following diagram below illustrates the module boundaries.

Figure 1 – Block Diagram of the Cryptographic Module



2. Security Level

The cryptographic module meets the overall requirements applicable to Level 1 security of FIPS 140-2.

Table 1 - Module Security Level Specification

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

3. Modes of Operation

Approved mode of operation

In FIPS mode, the cryptographic module only supports FIPS Approved algorithms as follows:

- ANSI X9.31-1998 Appendix A.2.4 RNG for deterministic random number generation
- AES (CBC) (128 bit keys) for symmetric encryption and decryption of bulk data
- RSA Signature (RSASSA-PKCS1-v1_5) (1024 bit keys) for digital signature
- RSA (RSAES-OAEP) (1024 bit keys) for asymmetric key establishment
- AES Key Wrap (128 bit keys) for symmetric key establishment
- SHA-1 for hashing
- HMAC-SHA-1 for keyed hashing

The cryptographic module relies on the implemented deterministic random number generator (RNG) that is compliant with ANSI X9.31-1998 Appendix A.2.4 for generation of all cryptographic keys.

The cryptographic module may be configured for FIPS mode by first setting the Microsoft Windows registry DWORD **HKLM\Software\VIACK\Product1\FIPSMode** entry to 1. After this has been done, when the cryptographic module is loaded, the module runs in FIPS mode. The user can determine if the cryptographic module is running in FIPS vs. non-FIPS mode by calling the *VkCryptGetFIPSMode* function.

Non-Approved mode of operation

In non-FIPS mode, the cryptographic module also supports the following non-FIPS approved algorithms:

- RC2 block encryption for encryption/decryption of RSA private keys.

4. Ports and Interfaces

The *logical interfaces* to the cryptographic module consist of a C++ Application Programming Interface (API) exported by the VkCrypt DLL (Versions 4.2 and 6.0). The *VIA3 VkCrypt API Reference* document describes the functions and parameters that behave as the module's data input, data output, control input, and status output interfaces.

The *physical ports* are the standard I/O ports for connecting external devices to the computer such as keyboards/mouse and monitors. These devices are outside the *physical boundary* of the cryptographic module and are excluded from the validation.

All data entered into and output from the cryptographic module are logically separated through the API and the software design maintains separation of the module's logical paths.

The following table summarizes the cryptographic module's logical interfaces and physical ports.

Table 2 – Logical Interfaces and Physical Ports

Interface	Logical Interface	Physical Port
Data Input	Input parameters passed to the API function call.	Standard input port (ie. keyboard/mouse).
Data Output	Output parameters returned by the API function call.	Standard output port (ie. monitor).
Control Input	Exported API function calls.	N/A
Status Output	Function return code from the API function call.	Standard output port (ie. monitor).

The computer on which the cryptographic module runs obtains power from an external power source via the *Power Interface*.

5. Identification and Authentication Policy

Assumption of Roles

The cryptographic module supports a *User* role and a *Crypto Officer* role. The module does not support a *Maintenance* role. The module is intended to run on Windows operating systems in single user mode and does not support concurrent operators.

Table 3 - Roles

User	The *User* is any entity that can access cryptographic services provided by the cryptographic module. The User role is *implicitly selected* when a process calls a User role API function exported by the module (as listed in Table 5 below).
Crypto Officer	The *Crypto Officer* is any entity that can perform tasks such as configuring the operating system, installing the cryptographic module on the computer system, and performing zeroization of the module. Other tasks performed by the *Crypto Officer* include initiating power-on self tests on demand, and checking the status of the module. The *Crypto Officer* can configure the module for FIPS mode operation, but has no special access to any keys or data. The *Crypto Officer* role is *implicitly selected* when performing the above tasks and when calling a Crypto Officer role API function (as listed in Table 5 below).

Authentication

The cryptographic module does not directly implement authentication to control access to the module (this is acceptable for FIPS 140-2 level 1 validation). The operating system provides functionality to require a user to be successfully authenticated prior to using any service provided by the module.

Table 4 – Roles and Required Identification and Authentication

Role	Type of Authentication	Authentication Data
User	Not required.	Not required.
Cryptographic-Officer	Not required.	Not required.

6. Access Control Policy

Roles and Services

The following table lists the services available within the cryptographic module.

Input and output of RSA private keys, AES symmetric keys, and certificates is done using handles. The handles are opaque references to the private key portion of a 1024 bit RSA public/private key pair, the 128 bit AES key, and the validated X.509 certificate respectively.

See the *VIA3 VkCrypt API Reference* document for more detailed descriptions of these services.

Table 5 – Services Authorized for Roles

Role	Authorized Services
User	• Generate Random Number: This service generates and fills a buffer with random bytes. • Hash Data: This service hashes data using SHA-1. • Protect Private Key: This service wraps an RSA private key using a global AES session key. • UnProtect Private Key: This service unwraps an RSA private key using a global AES session key. • Encrypt Private Key: This service wraps an RSA private key using an AES key. • Decrypt Private Key: This service unwraps an RSA private key using an AES key. • Escrow Private Key: This service wraps an RSA private key using an RSA public key. • UnEscrow Private Key: This service unwraps a RSA private key using an RSA private key. • Sign Data: This service signs data using an RSA private key. • Verify Data: This service verifies a signature using an RSA public key. • Clear Private Key: This service zeroizes and removes an RSA private key from volatile memory. • Get marshal key: This service returns a well-known AES key. • Generate AES key: This service generates a new AES key. • Encrypt Key: This service wraps an AES key with another AES key. • Decrypt Key: This service unwraps an AES key with another AES key. • Escrow Key: This service wraps an AES key using an RSA public key. • UnEscrow Key: This service unwraps an AES key using an RSA private key. • Protect Key: This service wraps an AES key using a global AES session key.

	• UnProtect Key: This service unwraps an AES key using a global AES session key. • Encrypt Data: This service encrypts plaintext data using an AES key. • Decrypt Data: This service decrypts ciphertext data using an AES key. • Clear AES Key: This service zeroizes and removes an AES key from volatile memory. • Request Certificate: This service generates an RSA public/private keypair and uses the generated keypair to request a certificate from a certificate server. • Import Certificate: This service validates and stores an X.509 encoded certificate in memory. • Export Certificate: This service returns the X.509 encoded certificate stored in memory. • Delete Certificate Requests: This service deletes certificate requests for the specified user. • Get Certificate Subject Key Identifier: This service retrieves the unique subject key identifier for the certificate. • Get Certificate Public Key: This service retrieves the RSA public key for the certificate. • Clear Certificate: This service zeroizes and removes a certificate from volatile memory.
Crypto Officer	• Install Module: The module is installed as part of the application's installation package. See the *Crypto Officer and User Guidance* document for more information. • Zeroization: Module CSPs can be explicitly zeroized. See section on Key Zeroization below. • Do Power Up Self Test: This service performs on demand power up self tests required by FIPS 140-2. • Get Power Up Self Test Status: This service provides the current status of the cryptographic module. • Get FIPS Mode: This service returns the current mode of operation of the cryptographic module (FIPS or non-FIPS).

Definition of Critical Security Parameters (CSPs)

The following are CSPs contained in the cryptographic module:

A. User Signing Private Key: This is the private part of the user's signing public/private key pair (1024 bit RSA key).

B. User Encrypting Private Key: This is the private part of the user's encrypting public/private key pair (1024 bit RSA key).

C. Random Number Generator Seed Key: This is the seed key for the random number generator (168 bit three key triple DES key).

D. Random Number Generator Seed Values: These are seed values for the random number generator.

E. Global Marshal Key: This is a global cryptographic module symmetric key (128 bit AES key).

F. Global Session Key: This is a global cryptographic module symmetric key (128 bit AES key).

G. Generated Symmetric Key: This is a module generated symmetric key (128 bit AES key).

Definition of Cryptographic Public Keys:

The following are the public keys contained in the module:

H. User Signing Public Key: This is the public part of the user's signing public/private key pair (1024 bit RSA key).

I. User Encrypting Public Key: This is the public part of the user's encrypting public/private key pair (1024 bit RSA key).

Definition of CSPs Modes of Access

Table 6 defines the relationship between access to cryptographic keys/CSPs and the different module services. The referenced cryptographic keys/CSPs in each module service are denoted using the letters above. The modes of access shown in the table are defined as follows:

- A cryptographic key or CSP that is provided as an input parameter to the module API. This indicates read/write access (**RW**) to that cryptographic key or CSP.
- A cryptographic key or CSP that is returned from the module API as an output parameter. This indicates read access (**R**) to that cryptographic key or CSP.
- A cryptographic key or CSP that is not accessible from the module API. This indicates no access (**NONE**) to that cryptographic key or CSP.

Table 6 – CSP Access Rights within Roles & Services

Role		Service	Cryptographic Keys and CSPs	Types of Access
User	C.O.			
X		Generate Random Number	seed key (C), seed values (D)	NONE
X		Hash Data	NONE	N/A
X		Protect Private Key	private key(A or B) symmetric global session key (F) encrypted private key	RW NONE R
X		UnProtect Private Key	encrypted private key symmetric global session key (F) private key (A or B)	RW NONE R
X		Encrypt Private Key	private key (A or B) symmetric key (G) encrypted private key	RW RW R
X		Decrypt Private Key	encrypted private key symmetric key (G) private key (A or B)	RW RW R
X		Escrow Private Key	private key (A or B) public key (I) symmetric key (G) encrypted private key	RW RW NONE R
X		UnEscrow Private Key	encrypted private key private key (B) symmetric key (G) private key (A or B)	RW RW NONE R
X		Sign Data	private key (A)	RW
X		Verify Data	public key (H)	RW
X		Clear Private Key	private key (A or B)	RW
X		Get Marshal Key	symmetric global marshal key (E)	R
X		Generate AES Key	symmetric key (G)	R
X		Encrypt Key	symmetric key (G) symmetric key (G) encrypted symmetric key	RW RW R
X		Decrypt Key	encrypted symmetric key symmetric key (G) symmetric key (G)	RW RW R
X		Escrow Key	symmetric key (G)	RW

			public key (I) encrypted symmetric key	RW R
X		UnEscrow Key	encrypted symmetric key private key (B) symmetric key (G)	RW RW R
X		Protect Key	symmetric key (G) symmetric global session key (F) encrypted symmetric key	RW NONE R
X		UnProtect Key	encrypted symmetric key symmetric global session key (F) symmetric key (G)	RW NONE R
X		Encrypt Data	symmetric key (G)	RW
X		Decrypt Data	symmetric key (G)	RW
X		Clear AES Key	symmetric key (G)	RW
X		Request Certificate	public/private key pair (H/A or I/B) private key (A or B)	NONE R
X		Import Certificate	public key (H or I) public key (H or I)	RW R
X		Export Certificate	public key (H or I)	R
X		Delete Certificate Requests	NONE	N/A
X		Get Certificate Subject Key Identifier	public key (H or I)	RW
X		Get Certificate Public Key	public key (H or I) public key (H or I)	RW R
X		Clear Certificate	public key (H or I)	RW
	X	Install Module	NONE	N/A
	X	Zeroization	NONE	N/A
	X	Do Power Up Self Test	NONE	N/A
	X	Get Power Up Self Test Status	NONE	N/A
	X	Get FIPS Mode	NONE	N/A

7. Operational Environment

Operating System Requirements

1. The module is loaded by a commercially available general purpose operating system. Therefore from a FIPS 140-2 perspective, the operational environment is a *modifiable operational environment*. The module was tested on Microsoft Windows XP.
2. The module is restricted to a *Single Operator Mode of Operation*, per FIPS 140-2 level 1

requirements. Note: It is the responsibility of the Crypto Officer to configure the operating system for a single operator mode of operation. See the *Crypto Officer and User Guidance* document for information on how to do this.

3. The module is loaded in its own independent process by the operating system. The module does not communicate with other processes. Other processes cannot interrupt the module while the module is executing. Other processes cannot access module CSPs.
4. The integrity of the module is verified through the use of self-tests as described in Section 10, “Self-Tests”.

8. Security Rules

The cryptographic module’s design corresponds to the cryptographic module’s security rules. This section documents the security rules enforced by the cryptographic module to implement the security requirements of this FIPS 140-2 Level 1 module.

1. The cryptographic module shall provide two distinct operator roles. These are the *User* role, and the *Crypto Officer* role.
2. When the module has not been placed in a valid role, the operator shall not have access to any cryptographic services.
3. The cryptographic module shall perform power-up self-tests as described in Section 10, “Self Tests”.
4. At any time the cryptographic module is in an idle state, the operator shall be capable of commanding the module to perform the power-up self-test.
5. After generating an RSA public/private key pair, the public/private keys are tested using the conditional test specified in FIPS 140-2 §4.9.2.
6. Prior to each use, the internal RNG shall be tested using the conditional test specified in FIPS 140-2 §4.9.2.
7. Data output shall be inhibited during key generation, self-tests, zeroization, and error states.
8. Status information shall not contain CSPs or sensitive data that if misused could lead to a compromise of the module.
9. The module shall not support concurrent operators.

9. Cryptographic Key Management

Random Number Generators (RNGs)

The module generates random numbers using the FIPS approved *Deterministic Random Number Generator* specified by ANSI X9.31-1998 – Appendix A.2.4.

Key Generation

The seed key and seeds for the random number generator are generated using the *CryptGenRandom* API call provided by the Windows operating system.

The module generates keys using the random number generator above, in the following manner:

- RSA keys are generated according to the procedures described in ANSI X9.31. RSA keys must be a minimum of 1024 bits.
- Symmetric AES keys are generated by generating a random sequence of suitable size according to the procedures specified in ANSI X9.31-1998 – Appendix A.2.4.

Intermediate key generation values are not output from the module during key generation.

Key Establishment

There is no manual key establishment. All key establishment is automated and the input and output of all private and symmetric keys through VkCrypt API is in wrapped form only.

Symmetric key establishment uses AES Key Wrap, a FIPS Approved symmetric key establishment technique.

Asymmetric key establishment uses RSA Key Wrapping, which is allowed for use, in the absence of a FIPS Approved asymmetric key establishment technique.

Key Entry and Output

All cryptographic keys entered into and output from the module through the API are encrypted as follows:

- RSA private keys (1024 bit) are wrapped using AES Key Wrap. The resulting wrapped key retains a key strength of 80 bits of security.
- Symmetric AES keys (128 bit) are wrapped using AES Key Wrap. The resulting wrapped key retains a key strength of 128 bits of security.
- Symmetric AES keys (128 bit) are wrapped using RSA Key Wrap (RSAES-OAEP). The resulting wrapped key has a key strength of 80 bits of security.

The minimum key strength supported by the module is 80 bits of security.

Key Storage

The module provides key storage on persistent storage media and RAM.

Key Zeroization

The global marshal key and global session key can be explicitly zeroized by deleting the module (VkCrypt.dll) from the disk using a secure delete utility. The utility overwrites the deleted file's on-disk image using techniques that are shown to make disk data unrecoverable, even using recovery technology that can read patterns in magnetic media that reveal weakly deleted files. See the *Crypto Officer and User Guidance* document for information on how to do this.

For all other keys, the module stores the keys while they are in use *in (volatile) memory* only. Key zeroization for these keys is performed by unloading the module from memory. Windows zero fills memory pages previously utilized by the module, before handing out the memory page to another process.

10. Self-Tests

The module implements both power-up tests and conditional tests to ensure proper operation of the cryptographic algorithms and authorized security services.

Power-Up Self Tests

When the VkCrypt DLL (Versions 4.2 and 6.0) is loaded into a process, power-up self-tests are automatically executed to ensure the integrity and correct operation of the cryptographic services. If any self-test fails, the module enters an error state and prevents any cryptographic service from being performed. It is then necessary for the application to unload the module from memory in order to repair the module.

The following lists the power-up self tests performed:

- Cryptographic Algorithm Test: The module performs known answer tests for AES, SHA-1, HMAC/SHA-1, RSA Sign/Verify, and RSA Key Wrap/UnWrap. A known answer test for the RNG sets all input parameters to specified values and checks for a specific output value.
- Software Integrity Test: The read-only data section of the VkCrypt DLL (Versions 4.2 and 6.0) contains an HMAC/SHA-1 secret key and value. The value is computed by computing an HMAC over the DLL image, *excluding* the stored HMAC value in the DLL. During the software integrity test, this value is recomputed and verified to match the stored value in the DLL.

The power-up self tests can be initiated on demand by calling the *VkCryptDoPowerUpSelfTest* function.

Conditional Tests

In addition to the power-up self-tests described above, the module performs the following on-going tests while executing:

- Pair-wise Consistency Test: The module runs a pair-wise consistency test each time an

RSA public/private key pair is generated. For RSA signature keys, the module signs a test message using the private key and verifies the signature using the corresponding public key. For RSA encryption keys, the module encrypts a test message with the public key, verifies that the ciphertext differs from the plaintext, decrypts the ciphertext with the corresponding private key, and verifies that the decrypted value is the same as the original test message.

- Continuous Random Number Generator Test: The module implements a continuous RNG test that executes each time the RNG is called. During the test, the previously generated random number is stored as a variable in memory. The test runs as follows:
  1. It stores the first 128 bits for comparison against the next 128 generated bits.
  2. It compares each subsequently generated 128 bits against the previously generated 128 bits.
  3. It fails if two compared 128 bit sequences are equal

If any of the conditional tests fail, the corresponding API function returns an error.

11. Physical Security Policy

For the purposes of FIPS 140-2 validation, the module is considered a *multi-chip standalone* device.

But as also stated in FIPS 140-2, for cryptographic modules implemented completely in software, physical security is provided solely by the host platform, and is not subject to the physical security requirements of the standard.

12. Mitigation of Other Attacks Policy

The module is not designed to mitigate any specific attacks.

13. References

[1] National Institute of Standards and Technology, *Security Requirements for Cryptographic Modules*, FIPS PUB 140-2, May 25, 2001.

[2] National Institute of Standards and Technology, *Derived Test Requirements for FIPS PUB 140-2*, Draft, March 02, 2004

[3] National Institute of Standards and Technology, *Advanced Encryption Standard (AES),* FIPS 197, November 26, 2001.

[4] National Institute of Standards and Technology, *Recommendation for Block Cipher Modes of Operation, Methods and Techniques,* Special Publication 800-38A, December 2001.

[5] National Institute of Standards and Technology, *Secure Hash Standard*, FIPS 180-1, April 17, 1995.

[6] National Institute of Standards and Technology, *The Keyed-Hash Message Authentication Code (HMAC)*, FIPS 198, March 06, 2002.

[7] American Bankers Association, *Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*, ANSI X9.31-1998.

[8] RSA Laboratories, *PKCS #1 v2.0: RSA Cryptography Standard*, October 1, 1998.

[9] VIACK Corporation, VIA3 4.2 VkCrypt API Reference, December 2006.

[10] VIACK Corporation, VIA3 4.2 VkCrypt Crypto Officer and User Guidance, December 2006.