SAFE-Key Device

FIPS 140-2 Non-proprietary Security Policy

Document Version 1.0
Hardware Version 1.4
Boot Module Firmware Version 1.2.0.0
BIOS Module Firmware Version 1.2.0.0
PEM Module Firmware Version 1.2.0.0
May 15, 2018

SAFE-Key Device Security Policy

Module Specification

This document describes the security policy of the SAFE-Key Device. It was submitted as part of the Federal Information Processing Standard (FIPS) 140-2 level 3 validation process.

The SAFE-Key device is a USB-based hardware security token; its primary purpose is two-factor authentication. The device supports three modalities: web-based two-factor authentication (patented), two-factor authentication for protected file decryption and two-factor authentication for digital signature generation. The two factors are *something you have* (i.e. a specific SAFE-Key device) and *something you know* (i.e. a password). It also supports one-factor, *something you have* (i.e. a specific SAFE-Key device) authentication.

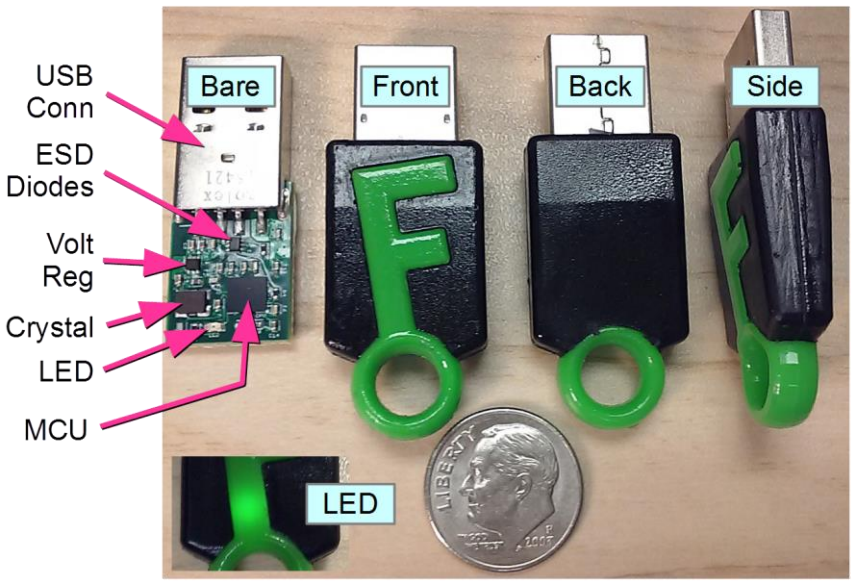

Figure 1 – SAFE-Key Device Front and Back

Figure 1 shows the SAFE-Key device. From left to right, the image shows the bare board, the front / top, the back / bottom and the side of the device. The green plastic is translucent which allows light from an LED to be seen. The inset image (lower left) shows the location of the LED and provides an idea of the brightness. Note that the green plastic is not so translucent that components on the circuit board can be seen.

The SAFE-Key device is a multi-chip, standalone cryptographic module. The major components are labeled on the image

of the bare board. All components other than the MCU and its case are excluded. Excluded components include the circuit board, the USB connector, the voltage regulator, the ESD diodes, ferrite beads, resistors, capacitors, the crystal and the LED. Removing the case to access the bare board provides exactly the same access to the MCU internals and the CSPs as is available through the USB bus.

Table 1 shows the security level of the SAFE-Key device under each evaluation criteria.

Area	Description	Level
1	Module Specification	3
2	Ports and Interfaces	3
3	Roles and Service	3
4	Finite State Model	3
5	Physical Security	3
6	Operational Environment	N / A
7	Key Management	3
8	EMI / EMC	3
9	Self-Test	3
10	Development	3
11	Mitigation of Other Attacks	N / A
	Overall	3

Table 1 - Security Levels

Approved Algorithms

Table 2 shows the approved algorithms used within the SAFE-Key device.

CAVP Cert	Algorithm	Standard	Mode / Method	Key Lengths	Use
#5373	AES	FIPS 197, SP 800-38A	CBC	256	Encryption, Decryption
#5373	AES	FIPS 197, SP 800-38F	KW	256	Authenticated key export
CVL #1839	RSADP	FIPS 186-4; RSADP		2048	Decryption primitive for key wrapping
Vendor Affirmed	CKG	SP 800-133			Key generation
#2080	DRBG	SP 800-90A	Hash SHA-256		Random bit generation
#3561	HMAC	FIPS 198-1	SHA-256	256	PB KDF

CAVP Cert	Algorithm	Standard	Mode / Method	Key Lengths	Use
#2874	RSA	FIPS 186-4	SHA-256, PKCS #1	2048	Signature generation, Signature verification
Vendor Affirmed with CVL #1839	KTS	SP 800-56B	RSA with OAEP	2048	Key agreement of key transport
#4313	SHS	FIPS 180-4	SHA-256		Message Digest

Table 2- Approved Algorithms

Allowed Algorithms

Table 3 shows the allowed algorithms.

Algorithm	Use
NDRNG	Entropy Generation
PBKDF (no security claimed)	Protection of intellectual property within firmware updates

Table 3 – Allowed Algorithms

The Non-Deterministic Random Number Generator (NDRNG) is the entropy generator. It uses the MCU hardware RNG circuit. The NDRNG passes the NIST Statistical Test Suite outlined in SP 800-22Rev1a. The NDRNG was tested using the NIST Non-IID Test, which estimated that 5.92 bits of entropy exist in each byte from the entropy generator. The DRBG is seeded using 1536 bits of NDRNG data and reseeded using 1024 bits of NDRNG output. Therefore, it is estimated that the DRBG is seeded with 1136 bits and reseeded with 757 bits of entropy, both of which are far in excess of the maximum key length of 256 bits generated by the SAFE-key device. The output of the NDRNG is used directly, without modification, by the DRBG. The output of the DRBG is used directly, without modification, by the cryptographic processes and other services.

The firmware update process is protected by an RSA-2048 signature. Firmware updates are also AES-256 encrypted; the sole purpose of the encryption is the protection of the BiObex's intellectual property contained within the firmware. The PBKDF algorithm, specified in SP 800-132 Option 1, is used to generate the AES key that decrypts an update. The *Salt* and *Password* bytes each consist of 256 bits of randomly generated data. No security is claimed.

Ports and Interfaces

Table 4 enumerates the SAFE-Key device's physical and logical interfaces.

Logical Interface	Port	Function
Data Input	USB	Command input, Data input
Data Output	USB	Data output, Error output, Status output
Control Input	USB	USB protocol control inputs
Control Output	USB	USB protocol control outputs
Status Output	USB, LED	Operation progress, Operating status, Error condition
Power	USB	+5V power

Table 4- Physical and Logical Interfaces

The USB port provides the majority of the functionality of the SAFE-Key device. The data port does ***NOT*** provide access to critical security parameters (CSPs). All CSPs remain within the cryptographic boundary of the device.

The LED provides status feedback to the user. It blinks rapidly while performing the boot verification. If the integrity or self-test checks fail, the device never leaves the boot mode and the LED blinks rapidly until the SAFE-Key device is unplugged. The blink rate reduces when SAFE-Key device enters either the PEM or BIOS mode; it continues to blink while the cryptographic library is being validated. Once the device enters a FIPS approved mode of operation, the LED remains in a steady *On* state.

In PEM or BIOS modes, the LED blinks continuously (non-rapid mode) if any library self-test failed, or if a subsequent error causes the cryptographic library to enter the lockout state. It will continue to blink until a subsequent self-test (caused by either a USB command or a full reset) returns the library to the *Idle* state. If the Boot module fails its self-test or the both the PEM and BIOS integrity tests fail, the device stays in Boot mode (indicated by a rapidly blinking LED). Since the USB functionality is not available, only a power-cycle reset can trigger a retest.

The LED blinks twice and then pauses when the device is gathering a password. It blinks five times and then pauses if the device's USB connection has not completed its USB handshake. It blinks ten times and then pauses if the device has been zeroized.

Figure 2 shows the SAFE-Key device's overall architecture. Virtually the entire SAFE-Key device is contained within the physical boundaries of the STM32F415 MCU (micro-controller unit). The STM32F415 is a system-on-a-chip with a CPU, RAM, FLASH memory and peripherals all implemented on one piece of silicon.

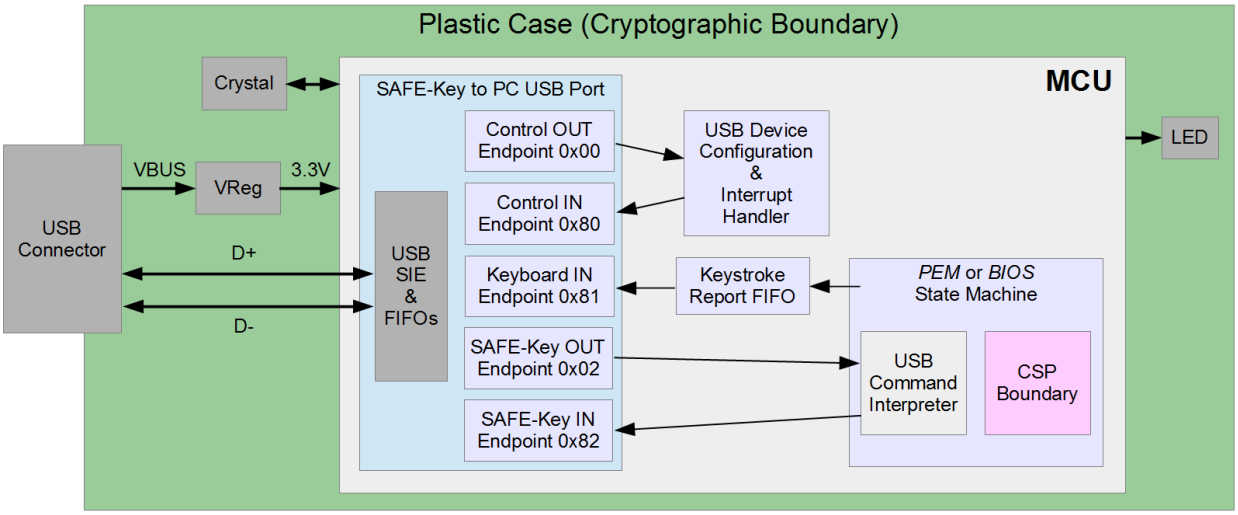

Figure 2 – Physical and Logical Device Model

By USB-IF (USB Implementer Forum) convention, endpoints are labeled IN or OUT based on the direction of the data flow to the host, which in this case is the PC (e.g. IN means data flowing into the PC). The USB transport layer protocol moves the data from the PC to the USB SIE (serial interface engine) endpoints. Each IN endpoint is serviced by a different FIFO. All OUT endpoints use the same FIFO. The first word read from the OUT FIFO contains the endpoint number and the size of the packet. The firmware steers each packet appropriately. (This is described in more detail in the *SAFE-Key State Machines* document.)

The SAFE-Key device implements five total USB endpoints that define three distinct channels. The control channel, channel 0, consists of end-points 0x00 and 0x80. The control channel is a fundamental part of every USB device; it implements the USB enumeration and control protocol. The keyboard channel, channel 1, consists of end-point 0x81. It implements a USB HID (human input device) channel that mimics a keyboard. Keystrokes are one transport mechanism used by the SAFE-Key device to pass information to the PC.

The SAFE-Key channel, channel 2, consists of end-points 0x02 and 0x82. It is the conduit between the PEM or BIOS state machine and the PC. Command messages are passed on channel 0x02 from the PC to the command interpreter and replies are transmitted back to the PC on channel 0x82. Simple messages, such as status requests are answered by the interpreter. Each command that requires the use of CSPs is passed to the PEM or BIOS state machines, which authenticate and service the command and return a reply to the interpreter.

The LED is driven by MCU I/O pins. It provides a visual indication of the device's status. Interpretation of the LED blinks is described elsewhere in this document.

Initialization and Delivery

The MCU within a SAFE-Key device remains blank until it is being prepared for shipment, at which time each device receives its firmware and RSA keys. FIPS 140-2 certified SAFE-Key devices are delivered via a trusted shipping service agreed between vendor and customer.

The customer (the device's owner) should examine a new SAFE-Key device's case for visible signs of tampering. If tampering is suspected, the device should be returned immediately. A new device contains only its RSA keys. Any "new" SAFE-Key device that contains any other CSPs should be returned to BiObex immediately. The customer should plug the SAFE-Key device into a PC; the LED should blink, indicating that the self-tests are being performed, and should reach a steady-on state after about 2 seconds. Any SAFE-Key device that fails to present this behavior should be returned immediately.

The customer should create the crypto officer role as soon as possible by creating a master device password (MDP) for the SAFE-Key device. It is ***IMPERATIVE*** that the customer remember and / or escrow the MDP, or create an MDP Reset Officer. There are no backdoor or password reset means other than those described in this document; a lost or forgotten MDP will permanently restrict the capabilities of the SAFE-Key device, ***FOREVER***.

If requested by the customer, the *Decryption Private Key* can be encrypted and saved in a file during the manufacturing process. The file is typically *self-encrypted*, which means that only the SAFE-Key device itself can decrypt the private key material. BiObex restricts access to the encrypted file to just the SAFE-Key device's owner.

The device's crypto officer role is required in order to decrypt that device's self-encrypted file. This allows the crypto officer to opt-in to an escrow arrangement whereby SAFE Protected Files can be decrypted using the escrowed key file in the event that the SAFE-Key device is lost or broken. The escrowed key file MUST be generated BEFORE the SAFE-Key device is lost or broken. (BiObex allows the customer to request escrow file generation during manufacturing.) The escrowed key file does not need to be distributed until it is needed, which gives the SAFE-Key device's owner additional control over the escrow arrangement.

Either scenario, generating a file or not generating a file, involves significant risk; it is up to the customer to decide which risk if preferable. BiObex is ***NOT*** responsible for these risks. The risk chosen must be accepted by the customer and should be mitigated to the extent possible.

When a device is manufactured ***with*** an encrypted key file, BiObex warns that decrypting the *Decryption Private Key* allows decryption of SPF files and decryption-based device identification without authentication. However, domain login, MDP Reset Officer authentication, signature generation and signature-based device identification all remain secure. Even with the *Decryption Private Key*, three other secrets continue to secure domain login: the user's password, the domain key and the SAFE-Key device's signing key. MDP Reset Officer authentication, signature generation and signature-based device identification all require the device's signing key and are therefore not compromised by exposure of the *Decryption Private Key*. Risk can be mitigated by limiting access to the decrypted private key files. Access can also be limited by controlling who is a device's crypto officer, by physically securing any electronic media that contains self-encrypted and/or escrowed files, and by using multi-person and multi-device authentication to access escrowed files.

When a device is manufactured ***without*** an encrypted key file, BiObex warns that loss of or damage to the SAFE-Key device will permanently render all SPF encrypted material

inaccessible. This risk can be mitigated by either backing-up all plaintext documents (perhaps within other secure storage) or by using a second SAFE-Key device to provide an alternative decryption option.

Roles, Services and Authentication

Table 5 shows the roles used by the SAFE-Key device. The *Crypto Officer Role* is authenticated using the *Master Device Password* (MDP). The *Crypto Officer Role* is required in order to perform all administrative functions, hence the MDP must be created before a SAFE-Key device will create any other type of CSP. Only a blank SAFE-Key device will accept an MDP without an authentication step. The MDP account is permanent (unless the device is zeroized). Each SAFE-key device has exactly one crypto officer. The MDP should never be shared. If an organization requires back-up support for the crypto officer, it may require the crypto officer to escrow the MDP or create appropriate MDP Reset officers.

A second type of crypto officer exists in the *MDP Reset Officer* role. An *MDP Reset Officer* can only be created by the SAFE-Key device's crypto officer. Authentication of the MDP Reset Officer role is performed by a second SAFE-Key device; the second device verifies the MDP Reset Officer's password. This is an *opt-in* capability; it is NOT necessary to create an MDP Reset Officer. If no one is assigned to the role, the device's MDP can only be set or reset by first knowing the current MDP.

Another type of account is the ordinary user account. These accounts are primarily used for decrypting SAFE Protected Files (SPF) and producing digital signatures. Password authentication is performed within the SAFE-Key device. Since SPF files and SAFE-Key digital signatures are device-specific, this constitutes two-factor authenticated key delivery and signature generation, since both a specific device and a password are required.

Some operations are intended for use within an authentication operation. These operations are shown in the *No Auth* columns in Table 7 and Table 8. An example is the web-based authentication protocol. The authentication packet is generated by the device without the user authenticating to the device; the actual authentication is performed by the external server that receives the packet. Another example is device identification. The SAFE-Key device cryptographically verifies its identity but provides no evidence as to the operator's identity, which is device-based, one-factor authentication.

Other operations in that can be performed without assuming an authenticated role are non-cryptographic operations, such as checking the status of the device or triggering the device's internal self-test.

Role	Type of Authentication	Authentication Mechanism
Crypto Officer	Identity-based authentication	Password verification
MDP Reset Officer	Identity-based authentication	2-factor (password plus a 2nd SAFE-Key device) authentication
User	Identity-based authentication	Password verification
No Auth[1]	None	None

Table 5- Roles and Authentication Mechanisms

Authentication Strength

Password-based authentication requires strong passwords. Table 6 shows the information space represented by purely random characters. The internal buffering of the SAFE-Key device limits the maximum password to 240 characters.

Set Name	Number of Elements	Bits Per Element	Min Len	Max Len	34 bits	50 bits
Numeric	10	3.32	8	240	11	16
Basic Alphabet	26	4.70	6	240	8	11
Numeric and Alphabetic	36	5.17	5	240	7	10
Mixed Case (MC)	52	5.70	5	240	6	9
Mixed Case with Numbers (MCwN)	62	5.95	5	240	6	9
Mixed Case with Numbers and Special characters (MCwN&SC)	94	6.55	4	240	6	8
Complete Words[2]	8,000	12.96	2	240	3	4

Table 6- Character Space and Minimum Password Lengths

The FIPS 140-2 authentication strength requirement is that a good password should represent a 1 in 1,000,000 chance of being guessed. That means the password must have 20 bits of random data.

The SAFE-Key logic delays the USB reply a minimum of a quarter second after an incorrect password guess. Worst case, this equates to about 240 guesses per minute.[3] The FIPS 140-2

[1] *No Auth* is a pseudo-role; a limited number of services can be accessed without authenticating.

[2] While there are roughly 400,000 words in the English language, the average vocabulary is only about 8,000 words. Using this as a benchmark, choosing a passphrase password made up of complete words provides roughly 13 bits of guess space per word. Short words (words less than 4 characters) may not provide sufficient protection and should be avoided or not counted when estimating the strength of a passphrase. Given the non-random nature of whole words and the *Basic Alphabet* requirement of 6 characters, BiObex recommends that passphrases be a minimum of 12 characters long.

[3] The actual rate is WELL below 240 guesses per minute this since 10 incorrect guesses results in an MCU reset with a simulated boot error, which requires that the device be unplugged and reinserted to clear the error.

requires that the chances of hitting the password within one minute be less than 1 in 100,000. Multiply this requirement times the worst case guess rate equates to a password with a 1 in 24,000,000 chance of being guessed. This equates to roughly 25 bits of random data. This value was used to populate the minimum length column in Table 6. SAFE-Key operators ***SHALL*** use passwords that meet or exceed this minimum length requirement. Operators ***SHOULD*** use passwords that meet or exceed the 34-bit lengths.

The above list is optimistic. It assumes that the characters are chosen at random. Unfortunately, random data is hard to remember so people use mnemonics, such as the first letter in a phrase, "2" instead of "to" and "4" instead of "for", etc. to generate and memorize their passwords. These techniques increase the chance that letters such as A, S, M, etc. (think of the fat volumes of an Encyclopedia or the thick sections of a dictionary) will appear in such a password. Thus, the information space used in practice is less than ideal; based on text compression results, BiObex estimates that the information space might be as little as half the number of bits per character of the above values. BiObex therefore recommends that a purely random password generator should be used and then the user should create a mnemonic to remember it.

The length of a password that will achieve 34 and 50 bits are shown. These password sizes equate to $1.7*10^{10}$ and $1.1*10^{15}$ possible passwords. Ignoring the reset safeguard, 240 guess per minute equates to $1.3*10^{8}$ guesses per year. On average a correct guess will be made half way through the complete search. When the number of possible passwords is divided by the guess rate, and then the result is halved, the average search periods for 35- and 50-bit passwords are computed to be 65 years and 4.2 million years, respectively.

Mitigation of Remote Brute Force Attacks

The SAFE-Key device counts the number of password authentication or decrypt operations that have failed. If there have been more than 10 failures, the device calls for one of the Boot Known Answer Test tests, i.e. it signals one of the Boot KATs to fail. The device then resets itself. Since the Boot KAT fails, the device remains in the Boot Error mode. This effectively locks the device until it is power cycled. Since the Boot Error mode doesn't provide USB functionality, there's no way to reset the device without physically removing it from the USB slot and reinserting it. Therefore an attacker must have physical access to a device to make more than a few password guesses.

This check is intended to greatly limit any attempt to brute force attack the password authentication, particularly from a remote location. If an attacker places malware on a user's computer with the intent to sending password guesses to the device, the malware will be limited to only a few guesses, after which, the blinking LED and locked state of the device will alert the user to the attack.

Modes of Operation

The SAFE-Key device only runs in one of two distinct approved modes, the password encoding mode, a.k.a. the PEM mode, or the basic I/O mode, a.k.a. the BIOS mode. In the PEM mode, the device can perform the full range of functions. The BIOS mode is limited to those functions necessary to authenticate the crypto officer and then load a firmware update. It is a back-up mode to prevent some form of failure during a PEM firmware update from permanently disabling the device.

The SAFE-Key device powers up into the Boot module, which performs integrity and self-tests on itself. It then determines which approved mode to switch to. It runs the integrity test on the PEM firmware before operating in the PEM mode. It runs the integrity test on the BIOS firmware before operating in the BIOS mode. If neither the PEM nor BIOS module pass the integrity test, or if the Boot module fails its integrity test, the device remains in Boot mode forever. In that case, a power cycle is needed to rerun the integrity tests. The PEM and BIOS modes perform their own self-tests before performing operations that access CSPs.

The PEM mode is defined by the PEM module firmware. The BIOS mode is defined by the BIOS module firmware. The two modules are completely independent of one another. The mode that the device is operating in is reported in the USB status reply.

Services and Authentication Requirements

Table 7 shows the operations that a SAFE-Key device can perform while in the PEM mode, the authenticated roles that can access each operation and the CSPs that the operation accesses. See Table 9 for descriptions of the CSPs. The term *Use* in the table below indicates that the CSP is the input to a cryptographic operation. For instance, in the case of password entries, the CSP is used to authenticate the user. In the case of a key, the CSP is the input to a cryptographic operation. The term *Read* indicates that an entry is read but is not used as the input to a cryptographic process. For instance the *Read Public Key* operation reads the key and transmits it over a USB channel without otherwise using it in a cryptographic process.

Service Name / Description	Crypto Officer	MDP Reset Officer	User	No Auth	CSP Access
Create Master Device Password (MDP)[4]	Yes	No	No	No	Write MDP
Change MDP	Yes	No	No	No	Use MDP; Write MDP
Add MDP Reset (MDPR) Officer	Yes	No	No	No	Use MDP; Write MDP Reset (MDPR) Entry
Reset MDP [5]	No	Yes	No	No	Use MDPR Entry; Write MDP
Remove MDPR Entry	Yes	No	No	No	Use MDP; Write MDPR Entry
Create user account	Yes	No	No	No	Use MDP; Write Password Entry
Change user password	No	No	Yes[6]	No	Use Password Entry; Write Password Entry
Reset user password	Yes	No	No	No	Use MDP; Write Password Entry
Remove user account	Yes	No	No	No	Use MDP; Write Password Entry

[4] This operation is only available if the entire password / domain database is blank, which only occurs with a brand-new device or a device that has been zeroized. The cryptographic office is NOT authenticated before performing this operation.

[5] The MPD Reset protocol relies on a second SAFE-Key device to authenticate the MDP Reset Officer.

[6] Users can only change their own password

Service Name / Description	Crypto Officer	MDP Reset Officer	User	No Auth	CSP Access
Zero all cryptographic secrets	Yes	No	No	No	Use MDP; Write Zeros[7] over all CSPs
Read SAFE-Key device status	Yes[8]	Yes[8]	Yes[8]	Yes	None
Read Public Key	Yes[8]	Yes[8]	Yes[8]	Yes	Read public key
Local signature-based device identification	Yes[8]	Yes[8]	Yes[8]	Yes	Use Signing Private Key
Local decryption-based device identification	Yes[8]	Yes[8]	Yes[8]	Yes	Use Decryption Private Key
Local KEK creation	Yes	No	Yes	No	Use MDP; Generate Random Bits; Use Decryption Private key; Write KEK entry
Local KEK removal	Yes	No	No	No	Use MDP; Write KEK entry
Retrieve SAFE Protected File (SPF) Keying Material	Yes	No	Yes	No	Use Decryption Private Key; Use Password Entry; Read SPF Keying Material (decrypt and export), Use KEK entry
Produce SPF signature	Yes	No	Yes	No	Use Password Entry; Use Signing Private Key
Web-based password authentication[9]	Yes[8]	Yes[8]	Yes[8]	Yes	Use Signing Private Key; Use Domain Key
Web-based secure text[9]	Yes[8]	Yes[8]	Yes[8]	Yes	Use Signing Private Key; Use Domain Key
Web-based key exchange	Yes	No	No	No	Use MDP; Use Decryption Private Keys; Read Signing Public Key; Write Domain Key

[7] See Table 9 and Table 10 for the list of CSPs that are zeroized.

[8] Since the operation does not require authentication, it is available to someone with crypto officer, MDP Reset Officer or user authority.

[9] With the web-based password authentication and secure text, the user's credentials are validated by a server that is external to the *SAFE*-Key device. The user does not have to be authenticated BY the SAFE-Key device itself in order to run this protocol

Service Name / Description	Crypto Officer	MDP Reset Officer	User	No Auth	CSP Access
Web-based cryptographic device identification	Yes[8]	Yes[8]	Yes[8]	Yes	Use Signing Private Key
Web domain removal	Yes	No	No	No	Use MDP; Write Domain Key
Compress Database	Yes	No	No	No	Read / Write User Passwords, Domain Keys, KEKs, MDP Reset Officer Entries
Load firmware update[10]	Yes	No	No	No	Use MDP; Use Firmware Update Key; Use Update Password; Use Firmware Protection Key; Use Integrity Obfuscation Bytes
Set cryptographic library error mode	Yes[8]	Yes[8]	Yes[8]	Yes	None
Rerun self-tests	Yes[8]	Yes[8]	Yes[8]	Yes	None
Test the KAT implementations	Yes[8]	Yes[8]	Yes[8]	Yes	None
PEM state machine reset	Yes[8]	Yes[8]	Yes[8]	Yes	None
Complete reset, i.e. reboot the MCU	Yes[8]	Yes[8]	Yes[8]	Yes	None

Table 7 - Authentication Required to Perform Services in the PEM Mode

[10] Crypto officer authentication is required to load a firmware update. The update will be automatically installed when the device reboots. Firmware updates are encrypted and digitally signed by BiObex.

Table 8 shows the operations that a SAFE-Key device can perform while in the BIOS mode, the authenticated roles that can access each operation and the CSPs that the operation accesses. See Table 9 for descriptions of the CSPs. The only authenticated operation in the BIOS mode is the *Load Firmware Update* service, which requires the Crypto Officer role. The other operations that are available in the BIOS mode are status and self-test services, which do not require authentication.

Service Name / Description	Crypto Officer	MDP Reset Officer	User	No Auth	CSP Access
Read SAFE-Key device status	Yes[8]	Yes[8]	Yes[8]	Yes	None
Load firmware update[10]	Yes	No	No	No	Use MDP; Use Firmware Update Key; Use Update Password; Use Firmware Protection Key; Use Integrity Obfuscation Bytes
Rerun self-tests	Yes[8]	Yes[8]	Yes[8]	Yes	None
Test the KAT implementations	Yes[8]	Yes[8]	Yes[8]	Yes	None
BIOS state machine reset	Yes[8]	Yes[8]	Yes[8]	Yes	None
Complete reset, i.e. reboot the MCU	Yes[8]	Yes[8]	Yes[8]	Yes	None

Table 8 – Authentication Required to Perform Services in the BIOS Mode

Operator Required Actions

To prevent man-in-the-middle attacks, Web-based password authentication and Web-based secure text must only be used with known good entities. Verification of the entity is outside of the scope of the SAFE-Key device and therefore outside of the scope of the device's certification. However, technologies exist, e.g. SSL certificates, direct connection, etc. that can ensure that the SAFE-Key device only authenticates the user to known good entities. Such assurances can be performed programmatically or by the user, for instance by examining the SSL status bar of a browser.

Servers that implement the SAFE-Key password and secure text protocols should continue to use additional security measures for tracking user login locations. For instance, many servers require users to answer security questions when logging in from a new IP address. Such measures also help to prevent man-in-the-middle, as well as other, attacks.

SAFE-Key devices and servers should exchange new domain keys periodically, the periodicity should be determined by the threat level and security requirements of the authentication environment.

A SAFE-Key device's owner shall report a lost device to anyone or any service relying on the device. The device shall be removed from any relying login authentication services. The crypto officer of any SAFE-key device that relied on the lost device to authenticate the MDP Reset Officer shall remove the MDP Reset Officer entry from the relying device. Relying parties shall remove the lost device from their list of devices to ensure that no new SPF file are generated that can be decrypted by the lost device.

The internal database of a SAFE-Key device that has been used for SPF will contain both at least one user account and at least one KEK associated with that user. If the crypto office removes a user account, the crypto officer should also remove all KEKs associated with that user.

Interrupting the power while the SAFE-Key device is compressing its internal database can lead to lost or unusable data entries. (The Crypto Officer password is stored redundantly so that it will not be lost by an interrupted compress operation.) Before starting and authenticating the operation, the Crypto Officer should ensure that the computer that the device is plugged into has sufficient power and/or a UPS (uninterruptable power supply) (The operation takes less than a minute.)

The only plaintext CSP that can be input to a SAFE-Key device is a password, whether it's the password for the Crypto Officer, an MDP Reset Officer or an ordinary user. A user must establish and maintain a trusted path for the password characters. This includes but is not limited to maintaining the PC's antivirus software and keeping the PC's Operating System (OS) software up to date. The user SHALL NOT use a PC that isn't trusted to operate the SAFE-Key device.

The SAFE-Key USB device driver intercepts keystrokes at the keyboard driver level and passes the characters to the device. Using this scheme, the plaintext password never reaches the user application level of the PC. The SAFE-Key USB device driver will NOT accept keystrokes

other than keystrokes observed at the driver level. This path is protected by the PC's digital signature requirements for device drivers. The digital signature of the device driver can be viewed using features built into the OS or the SAFE-Key console program.

Physical Security Policy

Physical Security Mechanisms

The cryptographic boundary of the SAFE-Key device is the entire USB stick. The SAFE-Key device is implemented in a single chip, an ST Micro STM32F415, inside the USB stick. The STM32F415 is commercially available, is based on the ARM Cortex M4 processor core and operates over standard commercial temperature range of -40° to +85° C.

During the SAFE-Key production process, the STM32F415's *Read Protection* option byte is set to Level 2. This change is permanent and irreversible. It disables the programming interface and forces the MCU to boot from its internal FLASH memory. It also disables all debugging ports on the chip, which means that access to the SAFE-Key device's internal circuit board provides ***exactly*** the same access to the chip internals and CSPs that can be achieved through the USB channels prior to accessing the circuit board.

Removing the MCU from the SAFE-Key circuit board and placing it on a different circuit board requires highly specialized tools. Accessing the MCU's pins without removing the chip from the board is virtually impossible because of the physical dimensions of the MCU chip package and the pins. There is a high likelihood that attempting to remove the MCU's case in order to access the bare silicon wafer will damage the chip and render it non-functional.

The SAFE-Key device's circuit board is completely encased in a poured plastic enclosure. This provides a high likelihood that any attempt at tampering will leave detectable evidence. It is also likely that any attempt to remove the coating will render the device inoperable before the circuit board can be accessed. MCU hardness testing was performed at room temperature (roughly 72° F or roughly 22° C).

Operator Required Actions

The SAFE-Key device's operator (both crypto officers and users) should:

- Check the outside appearance of the SAFE-Key device for evidence of tampering or attempted tampering
- Unplug the SAFE-Key device when not in use
- Store the SAFE-Key device in a location other than where it is used. For instance, it would be a violation of this policy to store a SAFE-Key device in a laptop bag where loss of the bag represents loss of both the laptop (with its protected data) and the SAFE-Key device.

Operational Environment

The SAFE-Key device has a limited operational environment; it is limited to the firmware within the device. The device is a USB device; all commands are received via the USB bus. It processes USB command and control messages in compliance with the USB protocol standard.

The device firmware can be updated. Only the crypto officer can install a firmware update. The SAFE-Key device also cryptographically verifies that an update came from BiObex before accepting the update and overwriting an operational module.

It is the crypto officer's responsibility to verify the origin of a firmware update by means outside of the scope of the SAFE-key device's certification. Methods include verification of the SSL certificate of the download site, and verification of the update through the SAFE-Key console's signature verification. (The console's signature verification uses a different set of signing keys than the firmware update itself. It establishes trust between BiObex and the console, whereas the *Firmware Update Key* pair (the public key is programmed into the device) establishes trust between BiObex and the SAFE-Key device's boot module.)

Key Management

Table of Keys

Table 9 shows the secret keys that are contained in a SAFE-Key device, along with their types, sizes and uses. None of these keys can be exported.

Name	Type	Size	Generated	Stored	Input and Export[11]	Zeroed[12]	Function
Decryption Private Key	RSA Private Key	2048	External	Sector 0 Plaintext	Entered when programmed; never exported[13]	Yes	Key exchange; SPF keying material decryption; Decryption-based device ID verification
Signing Private Key	RSA Private Key	2048	External	Sector 0 Plaintext	Entered when programmed; never exported	Yes	Packet signing for the web-based authentication protocol; Signature generation; Signature-based device ID verification; MDP Reset generation
Domain Key	AES Key	256	Key Exchange	FLASH Plaintext	Key Exchange; never exported	Yes	Packet encryption of the web-based authentication packet
Master Device Password	SHA-256 Digest	256	SHA-256[14]	FLASH Digest	User entered; never exported	Yes	Crypto officer authentication
Password Entry	SHA-256 Digest	256	SHA-256[14]	FLASH Digest	User entered; never exported	Yes	User authentication

[11] CSPs cannot be exported. Once inside the SAFE-Key device, they stay there until replaced or zeroed out.

[12] "Yes" indicates that the CSP is included in the zeroization process.

[13] The SAFE-Key device itself does not export the Decryption Private Key, but it does provide the means to unlock the self-encrypted file that contains the Decryption Private Key. Crypto officer authentication is required in order to decrypt the self-encrypted file.

[14] The user enters a plaintext password, which is digested using SHA-256 before being stored in the FLASH.

Name	Type	Size	Generated	Stored	Input and Export[11]	Zeroed[12]	Function
KEK Entry	Authenticated KEK	256	Key Exchange	FLASH Plaintext	Key Exchange; never exported	Yes	Authenticated SPF keying material export
MDP Reset Entry	RSA Public Key	2048	User entered	FLASH Plaintext	User entered; never exported	Yes	Authentication of the Master Device Password Reset officer
DRBG Entropy Input	Entropy bytes	1532 1024	Hardware circuit	RAM	Generated, consumed and zeroed	After use	Entropy input to the DRBG Seed
DRBG Seed	Pseudo-Random Data	440	From Entropy	RAM	Generated, consumed and zeroed	After use	Translates entropy into the DRBG internal state
DRBG C Vector	Pseudo-Random Data	440	From DRBG Seed	RAM	Generated; never exported	Reset, Crypto Error	Part of the DRBG internal state; Updated during random bit generation
DRBG V Vector	Pseudo-Random Data	440	From DRBG C Vector	RAM	Generated; never exported	Reset, Crypto Error	Part of the DRBG internal state; Updated during random bit generation
SPF Keying Material	Key Material	256	External	RAM	Imported encrypted; Exported in encrypted	After export	Input to KDF that produces an AES key
Firmware Update Key	RSA Public Key	2048	Fixed[15]	Sector 0 Plaintext	Entered when programmed; never exported	No	Firmware update verification

[15] These keys are common to all SAFE-Key devices. These keys control the firmware update process and therefore are tightly controlled by BiObex.

Name	Type	Size	Generated	Stored	Input and Export[11]	Zeroed[12]	Function
Update Password	PB KDF Password	256	Fixed[15]	Sector 0 Plaintext	Entered when programmed; never exported	Yes[16]	The secret key for a PB KDF
Firmware Protection Key	AES Key	256	From PB KDF	RAM	External[17]; never exported	After use	This key is used to decrypt the actual firmware during the update process.
Integrity Obfuscation Bytes	Random Data	256	Fixed	Sector 0 Plaintext	Entered when Programmed; never exported	Yes[16]	Device-specific salt for the SHA-256 digest used by the PEM and BIOS module integrity tests.

Table 9 - Critical Security Parameters (CSPs)

[16] This key is zeroized by installation of a specific firmware update.

[17] A new salt is generated and a new key is calculated using the *Update Password* CSP. The firmware update is encrypted with this new key. The salt is passed to the SAFE-Key device in order for it to recover the key and decrypt the update. The purpose of the encryption is to protect BiObex's intellectual property contained within a firmware update.

Table 10 shows the public keys that are contained in a SAFE-Key device, along with their types, sizes and uses. These keys are generated externally and stored in the device during the SAFE-Key device's initial firmware load. These keys can be exported.

Name	Type	Size	Generated	Stored	Input & Export	Zeroed[12]	Function
Decryption Public Key	RSA Public Key	2048	External	Sector 0 Plaintext	Entered when programmed; Exported on demand	Yes	Key Exchange; SPF keying material encryption; Decryption-based device ID verification
Signing Public Key	RSA Public Key	2048	External	Sector 0 Plaintext	Entered when programmed; Exported on demand	Yes	Password packet verification; Signature verification; Signature-based device ID verification; MDP Reset Officer verification

Table 10 - SAFE-Key Device's Exportable Public Keys

A SAFE-Key device's RSA keys are generated externally and written into the device's flash memory during firmware programming as part of BiObex's SAFE-Key device manufacturing process. All other keys are stored in the device's FLASH memory in what's referred to as the Password / Domain database. This database is used to store passwords, domain login AES keys, KEK keys and MDP reset records; database records can only be accessed from inside the cryptographic boundary.

RSA Keys

A SAFE-Key device contains two device-specific RSA key-blobs: decryption and signature generation. A third RSA public key, the firmware update key, is used to verify firmware updates. The firmware update key is stored in each SAFE-Key device. Its value is kept secret in order to protect BiObex's intellectual property.

Domain and Authenticated Key Encryption Keys

Domain keys and KEKs (Key Encryption Keys) are generated by running a key exchange process, per SP 800-56B rev 1. Domain keys are used to encrypt SAFE-Key password protocol packets. KEKs are used by the SAFE-Key device for authenticated export, per SP 800-38F, of keying material that can be used to decrypt SPF files.

Each entity (one entity being the SAFE-Key device) transmits its public key, generates 256 bits of random data (key material), encrypts the key material with the other entity's public key and transmits the encrypted key material. Each entity then uses its private key to decrypt the received message and runs a SHA-256 based key derivation function (KDF). The output of the KDF is used as the 256-bit AES key or KEK. The SAFE-Key device stores the key in the password / domain database in its FLASH memory. The SAFE-Key device's copy of domain keys and KEK entries cannot be accessed from outside of the SAFE-Key device.

The SAFE-Key device's key material is taken directly from the DRBG. The DRBG is initialized with greater than 256 bits of entropy.

Passwords

Passwords can be entered and changed by users. They are first passed through SHA-256 and the digest is stored in the SAFE-Key device's password / domain database. When needed for authentication, the user authentication keystrokes are passed through SHA-256 and the digest result is compared to the stored value. Password entries cannot be accessed from outside of the SAFE-Key device.

Master Device Password Reset Protocol

The MDP Reset (MDPR) protocol is an opt-in protocol. If the crypto officer does not create an MDP Reset entry, then the MDP Reset Officer role does not exist for that device and the MDP cannot be reset; in that case the MDP can only be changed by using the normal *Set Password* procedure, which requires the crypto officer to know the current MDP.

The MDP Reset protocol allows an ordinary user account on a second SAFE-Key device to authenticate into the MDP Reset Officer role and reset the MDP on the first device. The MDP is always reset to "password". The crypto officer of the first device shall immediately run the *Set Password* procedure on the first device to create a more secure MDP.

An MDP Reset entry consists of the MDP Reset Officer's user name, as well as the serial number and signature modulus of the MDP Reset Officer's SAFE-Key device. A device will not accept an MDP Reset entry that uses itself as the MDP Reset Officer's authenticating device. This is enforced by comparing the MDP Reset entry's modulus with the device's own signature modulus. MDP Reset entries cannot be accessed from outside of the SAFE-Key device.

SPF Keying Material

The SAFE-Key device implements a Key Transport Scheme (KTS) that delivers keying material that can be used to decrypt SPF (SAFE Protected File) files. The header of an SPF file contains one or more RSA encrypted KTS packets. After authenticating the user, KTS packet is recovered by running an RSA decrypt using the device's Decryption RSA Private Key. Per SP 800-56B Rev 1, the plaintext of the KTS packet consists of a SHA-256 hash of the *Other* string and 48 bytes of random keying material. The packet is RSA-OAEP (RSA with Optimal Asymmetric Encryption Padding) encoded. The keying material is encrypted using a KEK prior to being exported from the SAFE-Key device. Only a KEK that is tied to the user who authenticated the RSA decryption can be used for the keying material export.

The *Other* string consists of the 6-byte string "AES256", the 8-byte binary representation of the SAFE-Key device's serial number, the name of the authenticating user and 32 application-specific bytes intended to strongly tie the KTS packet to its intended use. The first three fields in the *Other* string are known to the device. The application-specific bytes are passed in to the SAFE-Key device as part of the KTS decryption process. In the case of an SPF file, the 32-bytes consist of the SHA-256 digest of the plaintext SPF document that's being protected; these bytes are stored in the SPF header. The decryption operation will fail if the hash bytes in the KTS packet do not match the calculated *Hash of Other*.

Once exported, it is up to the application to decide how to use the keying material. In the case of SPF, 32 bytes of keying material are used as input to a single-step KDF (SP 800-56B Rev 1 Section 5.5.1). The other 16 bytes of the keying material are used as an AES Initialization Vector or a KEK Authentication vector.

The SAFE-Key device wraps the KTS keying material inside of an authenticated key encryption message using a KEK. Each KEK in the SAFE-Key device is associated with a particular user or the crypto-officer. Each KEK also has a unique ID, so that one user can have more than one KEK, presumably one for each keystore (e.g. one on their PC at home and one on their PC at work). The KEK must be generated and shared (between the device and the PC) prior to exporting the KTS. If the SAFE-Key device and the user have not exchanged a KEK prior to the KTS decode, the device will report a *No Key* error when the user requests the keying material.

Integrity Obfuscation Bytes

The *Integrity Obfuscation Bytes* are used to salt the SHA-256 algorithm prior to generating the digest of the PEM or BIOS modules. These bytes are device-specific random data and are generated during the device's initial firmware programming. These bytes add an additional layer to the firmware protection. Even is an attacker figures out how to rewrite the BIOS or PEM module, unless the attacker also figures out how to export the *Integrity Obfuscation Bytes* the module's integrity test will fail. It will therefore be deemed *invalid* by the Boot module and the attacker's module will not be allowed to run. These bytes are not zeroized by the normal zeroization process, since they help to protect the integrity of the firmware, even in a zeroized SAFE-Key device. They can be zeroized by installation of a task-specific firmware update. Zeroization of these bytes will cause the Boot integrity test for both the PEM and BIOS modules to fail, which will permanently leave the device stuck in the Boot error mode.

Update Password

The *Update Password* bytes are the secret "password" bytes for a Password-Based KDF as defined in SP 800-132. The *Salt* for the KDF is contained in the *Update Verification Header* bytes that are contained in the update.

The *Update Password* bytes contain 256 bits of randomly generated data. The odds of guessing this password are 1 in 1.16×10^{77}. Keys derived from the password are solely used to securely encrypt the bulk of a firmware update until it is decrypted inside of the SAFE-Key device.

Zeroization

All of the device- or user-specific cryptographic keys can be zeroed out by running the SAFE-Key *Sunset* protocol. This removes the RSA decryption and signature keys and erases the entire password / domain database, which includes the MDP.

Zeroization of the RSA decryption and signature keys can be confirmed by LED blinks, by the PKCWIPED flag in the *Serial Number / Version* and *FIPS Stats* reply messages, and by the public keys, which will both be all zeros ("AAA…AAA" in Base-64). Removal of the password / domain database entries can be confirmed by the *Domain Table Stats* in the *Serial Number / Version* reply message

Zeroization of the *Update Password* bytes and the PEM / BIOS *Integrity Obfuscator* bytes requires the installation of task-specific firmware updates, available from BiObex on request.

Key Removal

The crypto officer can reset or remove individual passwords, remove login domain AES keys, remove KEK entries and remove MDP Reset entries. Under no circumstances (other than complete zeroization) can the MDP be removed from the device's database.

With all database types, the new record is written before the old records are invalidated. An interrupted write operation might leave the old record and either a partial (probably invalid) new

record or a complete new records, in which case both the old and the new records exist in the database. If the device reports an error or in cases where bad CSPs are suspected because the device fails to correctly authenticate a user or perform a service, the *Crypto Officer* should remove suspected records and then create a new CSP.

Self-Tests

The device performs certain tests in order to verify that it and its libraries are operating correctly. These are performed primarily during device initialization, hence are sometimes referred to as the Power-On Self-Test (POST).

Each cryptographic implementation is tested using Known Answer Tests (KAT) or similar testing method. The Boot, PEM and BIOS modules contain independent implementations of some functions, which are tested individually.

Integrity Test - Firmware Validation

FLASH memory that contains firmware is validated during the boot process. Modules are validated individually by passing the entire FLASH segment, including the blank portions, through SHA-256. The boot code always validates the firmware within the boot sector. The boot sector validates the integrity of a module before entering that mode of operation. Once in the PEM or BIOS mode, only an MCU reset can cause the SAFE-Key device to enter the other mode or return control to the Boot module.

Entropy Generator Testing

The entropy generator is tested using the NIST S.P. 800-22 "Frequency (monobit)" and "Runs" tests. The number of ones, and the number of transitions from one to zero and zero to one are counted. The acceptance levels of these tests are set so that a healthy entropy generator will fail (false positive) only about two times in a billion. The point of this test is not to determine the quality of the entropy generator, but simply to prove that it is not so dysfunctional as to fail such a relaxed test. A false positive should be cleared by a device reset.

List of Power-On Self-Tests

BOOT Cryptographic Power-On Self-Tests

- Firmware Integrity Test
- SHA-256 KAT
- AES Decrypt KAT
- RSA Verify KAT
- HMAC KAT
- PBKDF KAT

PEM Cryptographic Power-On Self-Tests

- SHA-256 KAT

- AES Decrypt KAT
- AES Encrypt KAT
- Authenticated Key Wrapping KAT
- RSA Sign KAT
- RSA Verify KAT
- RSA Encrypt KAT
- RSA Decrypt KAT
- NDRNG Mono-bit and Runs tests
- NDRNG Test Failure Test
- DRBG Initialize, Generate and Reseed KAT
- Single-step KDF KAT

BIOS Cryptographic Power-On Self-Tests

- SHA-256 KAT
- NDRNG Mono-bit and Runs tests
- NDRNG Test Failure Test
- DRBG Initialize, Generate and Reseed KAT

List of Conditional Tests

Boot

- Firmware Load Test
- AES Engine Status Test (when loading a key into the engine)

PEM

- NDRNG Stuck Output Test (continuous)
- DRBG Stuck Output Test (continuous)
- AES Engine Status Test (when loading a key into the engine)

BIOS

- NDRNG Stuck Output Test (continuous)
- DRBG Stuck Output Test (continuous)

Security Design

The SAFE-Key device has been designed to meet or exceed FIPS 140-2 Level 3 protections. The following are SAFE-Key device properties that implement these protections.

- The SAFE-Key device provides four access roles: crypto officer, MDP Reset officer, user and No Auth access.
- The SAFE-Key device enforces password authentication to perform the operations reserved for the crypto officer and users.

- The SAFE-Key device enforces two-factor authentication using a second SAFE-Key device for the MDP Reset officer role.
- The SAFE-Key device accumulates the number of failed password or decrypt operations. More than 9 failures result in the device setting a Boot KAT failure test mode and a device reset. The KAT failure can only be cleared by a power cycle.
- The SAFE-Key device performs one operation at a time. This:
  - Prevents concurrent operations from entering unanticipated states
  - Prevents multiple operators from concurrently accessing the device
- Authentication is required on a per-operation basis, which:
  - Blocks unauthenticated users from performing protected actions
  - Prevents an authenticated user from changing roles
  - Prevents authenticated roles from persisting at power off
  - The two operations in which an authenticated role persists are when a firmware update is loaded and a domain key is exchanged, in which case the crypto officer authentication persists for 60 or 120 seconds, respectively. The firmware update unlock blocks all other secured operations. The domain registration unlock is terminated by the completion of the operation or any other non-status command.
- All of the RAM is zeroed following reset / power-up, which:
  - Prevents authenticated roles from persisting at power on
  - Prevents any other data, keys or secrets from persist at power on
  - Guarantees that the firmware does not access any uninitialized variables
  - Guarantees that the state machines are not powered up into invalid states
- Intermediate values are zeroed after each operation is completed, which:
  - Prevents authenticated roles from persisting across operations
  - Prevents any other data, keys or secrets from persist across operations
- The firmware integrity is checked during the power-up sequence
- Firmware updates are signed and encrypted. The signature allows an update to be verified both before and after it is decrypted. Both verifications must pass before the new module can run.
- Self-tests are performed following power-up, prior to allowing cryptographic operations. The tests can be repeated on demand.
- The USB protocol allows any user to command the SAFE-Key device to perform a cryptographic library test at any time that it is otherwise idle.
- Operations that require use of CSPs are inhibited if:
  - The cryptographic library is in the untested, testing or error states
  - The SAFE-Key device is being zeroized
  - The SAFE-Key device is accepting a firmware update
  - The SAFE-Key device is performing an unrelated cryptographic operation
  - The SAFE-Key device is compressing its internal database
- The SAFE-Key device uses the FIPS SP 800-56B Key Agreement protocol for key entry (exchange).

- The SAFE-Key device uses the FIPS SP 800-56B Key Transport Scheme to deliver keying material to a SAFE-Key device user.
- SAFE Protected File keying material cannot be exported unless wrapped using a previously established key encryption key
    - The user role is required to create a KEK; that same user role is required to use that KEK
    - The crypto officer role is required to remove a KEK
- The Crypto Officer Role (a.k.a. the Master Device Password) is a singular and unique entry in the SAFE-Key devices internal storage. No user password entry can be created, modified or elevated to assume the Crypto Officer role.
- The Cryptographic Office's password is stored redundantly to prevent corruption or deletion of any given copy from permanently locking the account or from ever unlocking the account.
    - The Compress Database operation requires the Crypto Officer's password.
- The SAFE-Key device does NOT support:
    - Output of intermediate key generation values
    - Entry of seed keys
- All cryptographic secrets can be erased from the device using the SAFE-Key *Sunset* protocol.
    - Only the Crypto Officer can authorize the zeroization of a SAFE-Key device
- When zeroizing, the RSA keys are deleted before the password database in order to prevent an interrupted zeroization process from acting as a form of password reset. In other words, the secrets disappear before the authentication that protects the secrets disappears.
- The contents of RAM are periodically checked.
    - Stack growth is checked. Growth past an extremely generous boundary results in a device reset.
    - A band of memory separates the grow-down stack from the static variables allocated by the compiler. Any changes to (i.e. anything that writes into) this band result in a device reset.
- A watchdog timer resets the device after 8 seconds if it gets stuck in an infinite loop.

Mitigation of Other Attacks Policy

The SAFE-Key device has not been specifically designed to mitigate attacks beyond the scope of the FIPS 140-2 Level 3 requirements.