*Key Destruction SFRs for cPPs and EAs for SDs*

This whitepaper contains a set Security Functional Requirements (SFRs) for key destruction that an international Technical Community (iTC) should use when writing a cPP. There is a corresponding set of Evaluation Activities (EAs) that should be used when drafting the Supporting Document (SD). This set of requirements is intended to include all possible use cases ranging from a TOE that is composed of software and hardware and where the developer has a complete understanding of how memory is managed in their product, to a software application where the developer has no idea what memory media the underlying hardware platform is using.

In addition to the SFRs that describe how a key is to be destroyed, there are two further SFRs that an iTC may have to consider. One SFR, FCS\_CKM.4.1(x), will more than likely be an optional SFR in the cPP. The x is used as a place holder for iteration number the iTC uses when writing the cPP. It allows an ST Author to “destroy” (render the key unusable) a key, by destroying a key in the key chain that wraps or encrypts a key. For example, if a product uses a Key Encryption Key (KEK) to encrypt a Data Encryption Key (DEK), destroying the KEK using one of the methods in FCS\_CKM.4 is sufficient, since the DEK would no longer be usable (of course, presumes the DEK is still encrypted and the KEK cannot be recovered or re-derived.

The other SFR, FCS\_CKM\_EXT.6, serves as a mechanism to allow the cPP to offer a choice to the ST Author of which iteration(s) of FCS\_CKM.4 they want to specify in the ST. If the iTC feels that only one of the iterations is appropriate, this SFR is not necessary and the iTC simply puts the iteration they feel is appropriate into the cPP.

It is important to note that an iTC can also exclude items in a selection if they feel it is inappropriate for their technology type. For example, for a software application, the iTC may feel that the selection option for the TOE logically addressing storage is not viable, and therefore they simply remove that item from the selection. In this case, care must be taken in the EA, since the testing for this SFR iteration differs based on the selection.

One final consideration is that the iTC should consider specifying in the cPP when the keys (or possibly key material) are destroyed. It should also define what keys are intended to be destroyed, often not all keys need destruction such as the public key in a key pair. This set of SFRs only address *how* the keys are to be destroyed.

Please send comments and feedback to [jmcdan3@nsa.gov](mailto:jmcdan3@nsa.gov) or open an issue on the GitHub page.

**Optional SFRs**

**FCS\_CKM.4.1 (x)** The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [by using the appropriate method to destroy all key encryption key] that meets the following: [none].

***Application Note:*** *A key can be considered destroyed by destroying the key that protects the key. If a key is wrapped or encrypted it is not necessary to “overwrite” that key, overwriting the key that is used to wrap or encrypt the key used to encrypt/decrypt data, using the appropriate method for the memory type involved, will suffice.**For example, if a product uses a Key Encryption Key (KEK) to encrypt a Data Encryption Key (DEK), destroying the KEK using one of the methods in FCS\_CKM.4 is sufficient, since the DEK would no longer be usable (of course, presumes the DEK is still encrypted* ***and the KEK cannot be recovered or re-derived.).***

TSS/KMD: The evaluator shall examine the TOE’s keychain in the TSS/KMD and identify each instance when a key is destroyed by this method. In each instance the evaluator shall verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method.

There is no guidance associated with this requirement.

There is no test associated with this requirement.

**FCS\_CKM.EXT.6.1** The TSF shall use the [selection: *FCS\_CKM.4.1(a), FCS\_CKM.4.1(b), FCS\_CKM.4.1(c)*] key destruction methods

Application Note: If multiple selections are made, the TSS shall identify which keys are destroyed according to which selections.

TSS/KMD: The evaluator shall examine the TOE’s keychain in the TSS/KMD and verify all keys subject to destruction are destroyed according to one of the specified methods.

There is no guidance associated with this requirement.

There is no test associated with this requirement.

At least one of the following SFR instances should be specified in the cPP. If multiple instances are used (based on selection in the optional SFR FCS\_CKM\_EXT.6) then the iTC iterates the SFRs, as appropriate.

**Use Case 1: TOE includes hardware and developer is expected and/or required to understand the low-level details of how the memory is managed and controlled.**

#### FCS\_CKM.4 Cryptographic Key Destruction

**FCS\_CKM.4 Cryptographic Key Destruction**

**FCS\_CKM.4.1** The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [selection:

* *For volatile memory, the destruction shall be executed by a [selection: single overwrite consisting of [selection: a pseudo-random pattern using the TSF’s RBG, zeroes, ones, a new value of a key, [assignment: any value that does not contain any CSP]], removal of power to the memory, destruction of reference to the key directly followed by a request for garbage collection];*
* *For non-volatile memory [selection:*
  + *that employs a wear-leveling algorithm, the destruction shall be executed by a [selection: single overwrite consisting of [selection: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [assignment: some value that does not contain any CSP]], block erase];*
  + *that does not employ a wear-leveling algorithm, the destruction shall be executed by a [selection: [selection: single, [assignment: ST author defined multi-pass]] overwrite consisting of [selection: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [assignment: any value that does not contain any CSP] followed by a read-verify], block erase]*
    - *If the read-verification of the overwritten data fails, the process shall be repeated again up to [assignment: number of times to attempt overwrite] times, whereupon an error is returned.*

*]*

]

that meets the following: *No Standard*.

***Application Note***: *In the first selection, the ST Author is presented options for destroying a key based on the* memory *or storage technology where keys are stored within the TOE.*

*If non-volatile memory is used to store keys, the ST Author selects whether the memory storage algorithm uses wear-leveling or not. Storage technologies or memory types that use wear-leveling are not required to perform a read verify. The selection for destruction includes block erase as an option, and this option applies only to flash memory. A block erase does not require a read verify,* since the mappings of logical addresses to the erased memory locations are erased as well as the data itself*.*

*Within the selections is the option to overwrite a disused key with a new value of a key. The intent is that a new value of a key (as specified in another SFR within the PP) can be used to “replace” an existing key.*

*If a selection for read verify is chosen, it should generate an audit record upon failures.*

*Several selections allow assignment of a ‘value that does not contain any CSP’. This means that the TOE uses some other specified data not drawn from a source that may contain key material or reveal information about key material, and not being any of the particular values listed as other selection options. The point of the phrase ‘does not contain any CSP’ is to ensure that the overwritten data is carefully selected, and not taken from a general ‘pool’ that might contain current or residual data that itself requires confidentiality protection.*

**Use Case 2: This use case is for a TOE that includes the hardware (including the memory media). This use case is intended to address circumstances where the TOE developer has limited insight into how the memory media is managed – examples include a Network Device where the developer is using a third party memory controller, in any case, the developer knows the interface(s) that are invoked to destroy any keys that are created and stored, and has the ability to supply the data that is used to overwrite the key.**

**FCS\_CKM.4 Cryptographic Key Destruction**

**FCS\_CKM.4.1** The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [selection:

* *For volatile memory, the destruction shall be executed by a [selection: single overwrite consisting of [selection: a pseudo-random pattern using the TSF’s RBG, zeroes, ones, a new value of a key, [assignment: any value that does not contain any CSP]], removal of power to the memory, destruction of reference to the key directly followed by a request for garbage collection];*
* *For non-volatile memory the destruction shall be executed by a [selection: [selection: single, [assignment: ST author defined multi-pass]] overwrite consisting of [selection: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [assignment: any value that does not contain any CSP]], block erase];*

]

that meets the following: *No Standard*.

***Application Note:*** *In the first selection, the ST Author is presented options for destroying disused cryptographic keys based on whether they are in volatile memory or non-volatile storage within the TOE.*

*The selection of block erase for non-volatile storage applies only to flash memory.*

*Within the selections is the option to overwrite the memory location with a new value of a key. The intent is that a new value of a key (as specified in another SFR within the PP) can be used to “replace” an existing key.*

*Several selections allow assignment of a ‘value that does not contain any CSP’. This means that the TOE uses some other specified data not drawn from a source that may contain key material or reveal information about key material, and not being any of the particular values listed as other selection options. The point of the phrase ‘does not contain any CSP’ is to ensure that the overwritten data is carefully selected, and not taken from a general ‘pool’ that might contain current or residual data that itself requires confidentiality protection.*

===============================================================

**Use Case 3: This use case is for a software TOE (e.g., Operating System, Virtualization Server, application) where the developer has limited knowledge of the storage media types used by the underlying platform, and may not know where the keys are actually stored. Here, the developer simply knows the interface(s) that are invoked to destroy any keys that are created and stored.**

**FCS\_CKM.4 Cryptographic Key Destruction**

**FCS\_CKM.4.1** The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method [selection:

* *For volatile memory, the destruction shall be executed by a [selection: single overwrite consisting of [selection: a pseudo-random pattern using the TSF’s RBG, zeroes, ones, a new value of a key, [assignment: any value that does not contain any CSP]], removal of power to the memory, destruction of reference to the key directly followed by a request for garbage collection];*
* For non-volatile memory [*that consists of the invocation of an interface provided by the underlying platform that [selection:*

1. *logically addresses the storage location of the key and performs a [selection: single, [assignment: ST author defined multi-pass]] overwrite consisting of [selection: zeroes, ones, pseudo-random pattern, a new value of a key of the same size, [assignment: any value that does not contain any CSP]];*
2. *instructs the underlying platform to destroy the abstraction that represents the key]*

*]]*

that meets the following: *No Standard*.

***Application Note:*** *The interface referenced in the requirement could take different forms, the most likely of which is an application programming interface to an OS kernel. There may be various levels of abstraction visible. For instance, in a given implementation, selection a, the application may have access to the file system details and may be able to logically address specific memory locations. In another implementation, selection b, the application may simply have a handle to a resource and can only ask the platform to delete the resource, as may be the case with a platforms secure key store. Selection b should only be used for the most restricted access. The level of detail to which the TOE has access will be reflected in the TSS section of the ST.*

*Several selections allow assignment of a ‘value that does not contain any CSP’. This means that the TOE uses some other specified data not drawn from a source that may contain key material or reveal information about key material, and not being any of the particular values listed as other selection options. The point of the phrase ‘does not contain any CSP’ is to ensure that the overwritten data is carefully selected, and not taken from a general ‘pool’ that might contain current or residual data that itself requires confidentiality protection.*

# Evaluation Activity for Supporting Document

**For Use Case 1**

### FCS\_CKM.4 Cryptographic Key Destruction

#### TSS (Key Management Description may be used if necessary details describe proprietary information)

1. The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.
2. The evaluator shall check to ensure the TSS lists each type of key that is stored in non-volatile memory, and identifies the memory type where key material is stored. When listing the type of memory employed, the TSS will list each type of memory selected in the FCS\_CKM.4.1 SFR, as well as any memory types that employ a different memory controller or storage algorithm. For example, if a TOE uses NOR flash and NAND flash, both types are to be listed.
3. The evaluator shall examine the TSS to ensure it describes the method that is used by the memory controller to write and read memory from each type of memory listed. The purpose here is to provide a description of how the memory controller works so one can determine exactly how keys are written to memory. The description would include how the data is written to and read from memory (e.g., block level, cell level), mechanisms for copies of the key that could potentially exist (e.g., a copy with parity bits, a copy without parity bits, any mechanisms that are used for redundancy).
4. The evaluator shall examine the TSS to ensure it describes the destruction procedure for each key that has been identified. If different types of memory are used to store the key(s), the evaluator shall check to ensure that the TSS identifies the destruction procedure for each memory type where keys are stored (e.g., key X stored in flash memory is destroyed by overwriting once with zeros, key X’ stored in EEPROM is destroyed by an overwrite consisting of a pseudo random pattern – the EEPROM used in the TOE uses a wear-leveling scheme as described).
5. If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.
6. The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.
7. Upon completion of the TSS examination, the evaluator understands how all the keys (and potential copies) are destroyed.

#### Operational Guidance

1. There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.
2. Some examples of what is expected to be in the documentation are provided here.
3. When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, to mitigate this the drive should support the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.
4. Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.
5. The drive should be healthy and contains minimal corrupted data and should be end of lifed before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.
6. For destruction on wear-leveled memory, if a time period is required before is processed destruction the ST author shall provide an estimated range.

#### Test

1. For these tests the evaluator shall utilize appropriate development environment (e.g. a Virtual Machine) and development tools (debuggers, simulators, etc.) to test that keys are cleared, including all copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.
2. For destruction on wear-leveled memory, if a time period is required before is evaluator shall wait that amount of time after clearing the key in tests 2 and 3.
3. Test 1: Applied to each key held as in volatile memory and subject to destruction by overwrite by the TOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:
4. Record the value of the key in the TOE subject to clearing.
5. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
6. Cause the TOE to clear the key.
7. Cause the TOE to stop the execution but not exit.
8. Cause the TOE to dump the entire memory of the TOE into a binary file.
9. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

Test 2: Applied to each key help in non-volatile memory and subject to destruction by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.

1. Identify the purpose of the key and what access should fail when it is deleted. (e.g. the data encryption key being deleted would cause data decryption to fail.)
2. Cause the TOE to clear the key.
3. Have the TOE attempt the functionality that the cleared key would be necessary for.

The test succeeds if step 3 fails.

1. Test 3: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:
2. Record the value of the key in the TOE subject to clearing.
3. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
4. Cause the TOE to clear the key.
5. Search the non-volatile memory the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.
6. Test 4: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:
7. Record the storage location of the key in the TOE subject to clearing.
8. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
9. Cause the TOE to clear the key.
10. Read the storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

**For Use Case 2**

### FCS\_CKM.4 Cryptographic Key Destruction

#### TSS (Key Management Description may be used if necessary details describe proprietary information)

1. The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.
2. The evaluator shall check to ensure the TSS lists each type of key that is stored in non-volatile memory, and identifies the memory type (volatile or non-volatile) where key material is stored.
3. The TSS identifies and describes the interface(s) that is used to service commands to read/write memory. The evaluator examines the interface description for each different media type to ensure that the interface supports the selection(s) made by the ST Author.
4. If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.
5. The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

#### Operational Guidance

1. There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.
2. Some examples of what is expected to be in the documentation are provided here.
3. When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, to mitigate this the drive should support the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.
4. Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.
5. The drive should be healthy and contains minimal corrupted data and should be end of lifed before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.

#### Test

1. For these tests the evaluator shall utilize appropriate development environment (e.g. a Virtual Machine) and development tools (debuggers, simulators, etc.) to test that keys are cleared, including all copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.
2. Test 1: Applied to each key held as in volatile memory and subject to destruction by overwrite by the TOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:
3. Record the value of the key in the TOE subject to clearing.
4. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
5. Cause the TOE to clear the key.
6. Cause the TOE to stop the execution but not exit.
7. Cause the TOE to dump the entire memory of the TOE into a binary file.
8. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

Test 2: Applied to each key help in non-volatile memory and subject to destruction by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.

1. Identify the purpose of the key and what access should fail when it is deleted. (e.g. the data encryption key being deleted would cause data decryption to fail.)
2. Cause the TOE to clear the key.
3. Have the TOE attempt the functionality that the cleared key would be necessary for.

The test succeeds if step 3 fails.

1. Test 3: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:
2. Record the value of the key in the TOE subject to clearing.
3. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
4. Cause the TOE to clear the key.
5. Search the non-volatile memory the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.
6. Test 4: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to view the key storage location:
7. Record the storage location of the key in the TOE subject to clearing.
8. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
9. Cause the TOE to clear the key.
10. Search the storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.
11. The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

**For Use Case 3**

### FCS\_CKM.4 Cryptographic Key Destruction

1. TSS (Key Management Description may be used if necessary details describe proprietary information)
2. The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.
3. The evaluator shall check to ensure the TSS lists each type of key that is stored in in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).
4. The evaluator examines the interface description for each different media type to ensure that the interface supports the selection(s) and description in the TSS.
5. If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.
6. The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

#### Operational Guidance

1. There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.
2. Some examples of what is expected to be in the documentation are provided here.
3. When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, to mitigate this the drive should support the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.
4. Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.
5. The drive should be healthy and contains minimal corrupted data and should be end of lifed before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.

#### Test

1. Test 1: Applied to each key held as in volatile memory and subject to destruction by overwrite by the TOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:
2. Record the value of the key in the TOE subject to clearing.
3. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
4. Cause the TOE to clear the key.
5. Cause the TOE to stop the execution but not exit.
6. Cause the TOE to dump the entire memory of the TOE into a binary file.
7. Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.

Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.

Test 2: Applied to each key help in non-volatile memory and subject to destruction by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.

1. Identify the purpose of the key and what access should fail when it is deleted. (e.g. the data encryption key being deleted would cause data decryption to fail.)
2. Cause the TOE to clear the key.
3. Have the TOE attempt the functionality that the cleared key would be necessary for.

The test succeeds if step 3 fails.

1. *The following tests apply only to selection a), since the TOE in this instance has more visibility into what is happening within the underlying platform (e.g., a logical view of the media). In selection b), the TOE has no visibility into the inner workings and completely relies on the underlying platform, so there is no reason to test the TOE beyond test 1.*
2. *For selection a), the following tests are used to determine the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.*
3. Test 3: Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g., MBR file system):
4. Record the value of the key in the TOE subject to clearing.
5. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
6. Cause the TOE to clear the key.
7. Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.
8. Test 4: Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media:
9. Record the logical storage location of the key in the TOE subject to clearing.
10. Cause the TOE to perform a normal cryptographic processing with the key from Step #1.
11. Cause the TOE to clear the key.
12. Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.

The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.