### BOOT\_ChooseAndLoadSW\_CLSW

#### BOOT\_CLSW\_LoadSW\_Load\_AESdecrypt

This function deciphers the RAM copy using the AES Key and verifies that the result is not corrupted using the CRC16 from the SUP file.

Prototype:

void BOOT\_CLSW\_LoadSW\_Load\_AESdecrypt(const TS\_FileInfo\* const p\_FileInfo,

const uint32\_t p\_SIFAddress,

boolean\_t\* p\_FlashReadFailed,

ts\_LoadStatus\* p\_LoadStatus)

Parameters:

Function return : Not used

p\_FileInfo, (R) : File Info

p\_SIFAddress (R) : SIF Header address

p\_FlashReadFailed(W) : Read failure status

p\_LoadStatus(W) : Load status

Calls:

LIBBSP\_MRAM\_Open

LIBBSP\_MRAM\_Read

LIBBSP\_MRAM\_Close

LIBBSP\_RMM\_Open

LIBBSP\_RMM\_Read

LIBBSP\_RMM\_Close

LIBUTI\_MEM\_MemSet

LIBBSP\_ENVM\_Open

LIBBSP\_ENVM\_Read

LIBBSP\_ENVM\_Close

LIBUTI\_AES\_decrypt\_in\_placet

LIBUTI\_CRC\_ComputeCRC16

Preconditions:

None

##### Input Data

Data:

AES\_KeyBuffer[C\_BOOT\_AES\_KEY\_MAX\_LENGTH]: uint8\_t

AES\_MaskBuffer[C\_BOOT\_AES\_KEY\_MAX\_LENGTH]: uint8\_t

Preconditions:

None.

##### Output Data

Data:

AES\_KeyBuffer[C\_BOOT\_AES\_KEY\_MAX\_LENGTH]: uint8\_t

##### Requirements

REQ\_SDDD\_BOOT\_BOOT\_ChooseAndLoadSW\_CLSW000\_xx-0x

*[COV.REQ\_BOOT\_SRD-00087; COV.REQ\_BOOT\_SRD-00089; COV.REQ\_BOOT\_SRD-00096; COV.REQ\_BOOT\_SRD-00170]*

BOOT\_CLSW\_LoadSW\_Load\_AESdecrypt

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Causes | | | |
| Effects | [SIF Header address is stored in FLASH] | | [SIF Header address is not stored in FLASH] | |
| **[**Open access to MRAM**]** | | **[**Open access to RMM**]** | |
| [MRAM access is successful] | [MRAM access is not successful] | [RMM access is successful] | [RMM access is not successful] |
| **[**Retrieve the masked AES key from the MRAM**]**  **[**Close the access to MRAM**]** | No effect | **[**Retrieve the masked AES key from the RMM**]**  **[**Close the access to RMM**]** | No effect |
| [MRAM access is not successful] OR  [RMM access is not successful] | | [MRAM access is successful] AND  [RMM access is successful] | |
| Set {p\_FlashReadFailed} to TRUE  Set {p\_LoadStatus->OutOfOperationalCondition} to TRUE  **[**Delete the AES key from RAM**]** | | No effect | |
| **[**Open the access to eNVM of PLD Compagnon**]** | | | |
| [eNVM access is successful] | | [eNVM access is not successful] | |
| **[**Retrieve the AES Mask from PLD Compagnon**]**  **[**Close the access to eNVM**]** | | Set {\*p\_FlashReadFailed} to TRUE  Set {p\_LoadStatus->OutOfOperationalCondition} to TRUE  **[**Delete the AES key from the RAM**]**  **[**Delete the AES Mask from the RAM**]** | |
| [Get the corresponding masked AES Key from MRAM and XOR it with its corresponding AES Mask from PLD Compagnon]  **[**Remove the AES Mask**]**  **[**Decipher the RAM copy**]** | | | |
| [AES Key decode is not successful] | | [AES Key decode is successful] | |
| Causes | | | |
| Set {p\_LoadStatus->Corrupted} to TRUE  and exit the function  Effects | | No effect | |
| **[**Delete the AES key from the RAM**]**  **[**Compute an ARINC665 compliant CRC16 checksum of the memory**]** | | | |
| [CRC16 checksum is different from the expected value] | | [CRC16 checksum is equal to the expected value] | |
| Set {p\_LoadStatus->Corrupted} to TRUE  and exit the function | | No effect | |

[SIF Header address is stored in FLASH]: bit0 of {p\_SIFAddress} is equal to 0.

**[**Open access to MRAM**]** corresponds to the following call:

**LIBBSP\_MRAM\_Open**

* **Function return:** {MRAM access status}

**[**Retrieve the masked AES key from the MRAM**]** corresponds to the following call:

**LIBBSP\_MRAM\_Read**

* **Function return:** Not used
* **IN:** (C\_BOOT\_MRAM\_AESKEY\_ZONE\_OFFSET + (({p\_SIFAddress} & 0x1FFFF)/ 0x10))
* **IN:** C\_BOOT\_AES\_KEY\_MAX\_LENGTH
* **OUT:** {AES\_KeyBuffer}
* **OUT:** {MRAM read status}

**[**Close the access to MRAM**]** corresponds to the following call:

**LIBBSP\_MRAM\_Close**

* **Function return:** Not used

[SIF Header address is not stored in FLASH]: bit0 of {p\_SIFAddress} is different from 0.

**[**Open access to RMM**]:** corresponds to the following call:

**LIBBSP\_RMM\_Open**

* **Function return:** {RMM access status}

**[**Retrieve the masked AES key from the RMM**]:** corresponds to the following call:

**LIBBSP\_RMM\_Read**

* **Function return:** Not used
* **IN:** (C\_BOOT\_RMM\_ADDR\_AES\_START + ((((({p\_SIFAddress} & 0x7FFFFFFF) – C\_BOOT\_RMM\_ADDR\_SIF\_START) & 0x1000) / 0x1000) \* 0x200)
* **IN:** C\_BOOT\_AES\_KEY\_MAX\_LENGTH
* **OUT:** {AES\_KeyBuffer}
* **OUT:** {RMM read status}

**[**Close the access to RMM**]** corresponds to the following call:

**LIBBSP\_RMM\_Close**

* **Function return:** Not used

[MRAM access is not successful]: {MRAM access status} is different from E\_LIBBSP\_MRAM\_OK

[RMM access is not successful]: {RMM access status} is different from E\_LIBBSP\_MRAM\_OK

**[**Delete the AES key from RAM**]** corresponds to the following call: Thenexit the function.

**LIBUTI\_MEM\_MemSet**

* **Function return:** Not used
* **IN/OUT:** {AES\_KeyBuffer}
* **IN:** 0 (is written on 1 byte)
* **IN:** C\_BOOT\_AES\_KEY\_MAX\_LENGTH

[MRAM access is successful]: {MRAM access status} is equal to E\_LIBBSP\_MRAM\_OK.

[RMM access is successful]: {RMM access status} is equal to E\_LIBBSP\_MRAM\_OK.

**[**Open the access to eNVM of PLD Compagnon**]** corresponds to the following call:

**LIBBSP\_ENVM\_Open**

* **Function return:** {eNVM access status}
* **IN:** E\_LIBBSP\_ENVM\_COMPAGNON

[eNVM access is successful]: {eNVM access status} is equal to E\_LIBBSP\_ENVM\_OK.

**[**Retrieve the AES Mask from PLD Compagnon**]** corresponds to the following call:

**LIBBSP\_ENVM\_Read**

* **Function return:** Not used
* **IN:** E\_LIBBSP\_ENVM\_COMPAGNON
* **IN:** C\_BOOT\_PLD\_ENVM\_ADDR\_AES\_START
* **IN:** C\_BOOT\_AES\_KEY\_MAX\_LENGTH
* **OUT:** {AES\_MaskBuffer}
* **OUT:** {eNVM read status}

**[**Close the access to eNVM**]** corresponds to the following call:

**LIBBSP\_ENVM\_Close**

* **Function return:** Not used
* **IN:** E\_LIBBSP\_ENVM\_COMPAGNON

[eNVM access is not successful]: {eNVM access status} is different from E\_LIBBSP\_ENVM\_OK.

**[**Delete the AES key from the RAM**]** corresponds to the following call:

**LIBUTI\_MEM\_MemSet**

* **Function return:** Not used
* **OUT:** {AES\_KeyBuffer}
* **IN:** 0
* **IN:** C\_BOOT\_AES\_KEY\_MAX\_LENGTH

Then exit the function.

**[**Delete the AES Mask from the RAM**]** corresponds to the following call: Then exit the function

**LIBUTI\_MEM\_MemSet**

* **Function return:** Not used
* **OUT:** {AES\_MaskBuffer}
* **IN:** 0 (written on 1 byte)
* **IN:** C\_BOOT\_AES\_KEY\_MAX\_LENGTH

[Get the corresponding masked AES Key from MRAM and XOR it with its corresponding AES Mask from PLD Compagnon]: For all elements in C\_BOOT\_AES\_KEY\_MAX\_LENGTH:  
set {AES\_KeyBuffer[index]} to {AES\_KeyBuffer[index]} XOR {AES\_MaskBuffer[index]}  
[index] is from 0 to C\_BOOT\_AES\_KEY\_MAX\_LENGTH - 1

**[**Remove the AES Mask**]** corresponds to the following call:

**LIBUTI\_MEM\_MemSet**

* **Function return:** Not used
* **OUT:** {AES\_MaskBuffer}
* **IN:** 0 (written on 1 byte)
* **IN:** C\_BOOT\_AES\_KEY\_MAX\_LENGTH

**[**Decipher the RAM copy**]** corresponds to the following call:

**LIBUTI\_AES\_decrypt\_in\_place**

* **Function return:** {AES Key decode status}
* **IN/ OUT:** {p\_FileInfo->Address}
* **IN:** {p\_FileInfo->ARINC\_665\_FileInfo->Length}
* **IN:** {AES\_KeyBuffer}
* **IN:** 256
* **OUT:** address of the {size of the decrypted AES Key}

[AES Key decode is successful]: {AES Key decode status} is different from FALSE.

**[**Compute an ARINC665 compliant CRC16 checksum of the memory**]** corresponds to the following call:

**LIBUTI\_CRC\_ComputeCRC16**

* **Function return:** {CRC16 buffer}
* **IN:** {p\_FileInfo->Address}
* **IN:** {size of the decrypted AES Key}

[AES Key decode is not successful]: {AES Key decode status} is equal to FALSE.

[CRC16 checksum is different from the expected value]: {CRC16 buffer} is different from {p\_FileInfo->ExpectedCRC16}.

[CRC16 checksum is equal to the expected value]: {CRC16 buffer}is equal to {p\_FileInfo->ExpectedCRC16}.

Traceability: Refined

Rationale:

Mean of verification: Test

[END\_REQ\_SDDD\_BOOT\_CLSW\_LoadSW\_Load\_AESdecrypt\_000xx-0x]