# Protocols Security Test

## HASH Protocol Test

Reference Document:

*UEFI Specification*, EFI\_HASH\_PROTOCOL Section.

Configuration

* Call “EFI\_HASH\_SERVICE\_BINDING\_PROTOCOL.CreateChild()” before testing.
* Call “EFI\_HASH\_SERVICE\_BINDING\_PROTOCOL.DestoryChild” after testing.
* Execute testing of 25.4.1.1~25.4.1.3 and 25.4.2.1.1~25.4.2.5.2 for every hash protocol(SHA-x/MD5).

Required Elements

### GetHashSize()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.4.1.0.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xae | check input parameters of testing EFI\_HASH\_PROTOCOL.GetHashSize(). | Check interface/environment valid. |
| 25.4.1.1.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa2 | EFI\_HASH\_PROTOCOL.GetHashSize() HashSize invalid checking test. | Call GetHashSize() with (Hashsize=NULL) should return EFI\_INVALID\_PARAMETER. |
| 25.4.1.2.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa3 | EFI\_HASH\_PROTOCOL.GetHashSize() HashAlgorithm invalid checking test A. | Call GetHashSize() with (HashAlgorithm=NULL) should return EFI\_INVALID\_PARAMETER. |
| 25.4.1.2.2 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa4 | EFI\_HASH\_PROTOCOL.GetHashSize() HashAlg invalid checking test B. | Call GetHashSize() with (HashAlgorithm invalid) should return EFI\_INVALID\_PARAMETER. |
| 25.4.1.3.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa5 | EFI\_HASH\_PROTOCOL.GetHashSize() get HashSize of the special HashAlgorithm. | Call GetHashSize() with (HashAlgorithm =SHA-x/MD5) should return EFI\_SUCCESS.  Exit testing when error occurred. |

### Hash()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.4.2.0.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xae | check input parameters of testing EFI\_HASH\_PROTOCOL. Hash(). | Check interface/environment valid. |
| 25.4.2.1.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa6 | EFI\_HASH\_PROTOCOL. Hash() Message invalid checking test. | Call Hash() with (Message=NULL) should return EFI\_INVALID\_PARAMETER. |
| 25.4.2.1.2 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa7 | EFI\_HASH\_PROTOCOL. Hash() Hash invalid checking test. | Call Hash() with (Hash=NULL) should return EFI\_INVALID\_PARAMETER. |
| 25.4.2.2.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa8 | EFI\_HASH\_PROTOCOL. Hash() HashAlgorithm invalid checking test. | Call Hash() with (HashAlgorithm=NULL) should return EFI\_INVALID\_PARAMETER. |
| 25.4.2.2.2 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa9 | EFI\_HASH\_PROTOCOL. Hash()HashAlgorithm invalid checking test. | Call Hash() with invalid HashAlgorithm should return EFI\_INVALID\_PARAMETER. |
| 25.4.2.3.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xaa | EFI\_HASH\_PROTOCOL. Hash() Extend invalid checking test. | Call Hash() with (HashAlgorithm=NULL and Extend=TRUE) should return EFI\_INVALID\_PARAMETER. |
| 25.4.2.4.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xab | EFI\_HASH\_PROTOCOL. Hash() hash some testing data. | Call Hash() with (Extend=FALSE) should return EFI\_SUCCESS.  Exit testing when error occurred. |
| 25.4.2.4.2 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xac | Verify hash result getting from EFI\_HASH\_PROTOCOL. Hash() (25.4.2.4.1) | check hash result getting from (25.4.2.4.1) correct or not. |
| 25.4.2.5.1 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xad | EFI\_HASH\_PROTOCOL. Hash() hash some extend testing data. | Call Hash() with (Extend=TRUE) should return EFI\_SUCCESS.  Exit testing when error occurred. |
| 25.4.2.5.2 | 0xf2db2578, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xae | Verify hash result getting from EFI\_HASH\_PROTOCOL. Hash() (25.4.2.5.1) | check extend hash result getting from (25.4.2.5.1) correct or not. |

## AUTHENTICATION\_INFO Protocol Test

Reference Document:

*UEFI* *Specification*, EFI\_AUTHENTICATION\_INFO\_PROTOCOL Section.

Configuration

Required: prepare testing data by calling EFI\_AUTHENTICATION\_INFO\_PROTOCOL.Set() before testing of Get().

Required Elements

### Get()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.1.1.1.1 | 0xf2db2579, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa3 | EFI\_AUTHENTICATION\_INFO\_PROTOCOL.Get() get authentication\_info of the special ControllerHandle. | Call Get() with (valid ControllerHandle) should return EFI\_SUCCESS.  Exit testing when error occurred. |
| 25.1.1.2.1 | 0xf2db2579, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa4 | EFI\_AUTHENTICATION\_INFO\_PROTOCOL.Get() ControllerHandle invalid checking test. | Call Get() with (ControllerHandle=NULL) should return EFI\_INVALID\_PARAMETER. |
| 25.1.1.3.1 | 0xf2db2579, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa5 | EFI\_AUTHENTICATION\_INFO\_PROTOCOL.Get()parameter *Buffer* invalid checking test. | Call Get() with **(**Buffer=NULL) should return EFI\_INVALID\_PARAMETER. |

### Set()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.1.2.1.1 | 0xf2db2579, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa6 | EFI\_AUTHENTICATION\_INFO\_PROTOCOL.Set() set authentication\_info of the special ControllerHandle. | Call Set() with (valid ControllerHandleand Buffer**)** should return EFI\_SUCCESS.  Exit testing when error occurred. |
| 25.1.2.2.1 | 0xf2db2579, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa7 | EFI\_AUTHENTICATION\_INFO\_PROTOCOL.Set() ControllerHandle invalid checking test. | Call Set() with (ControllerHandle=NULL) should return EFI\_INVALID\_PARAMETER. |
| 25.1.2.3.1 | 0xf2db2579, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa8 | EFI\_AUTHENTICATION\_INFO\_PROTOCOL.Set()parameter *Buffer* invalid checking test. | Call Set() with (Buffer=NULL) should return EFI\_INVALID\_PARAMETER. |
| 25.1.2.4.1 | 0xf2db2579, 0xdc54, 0x4896, 0x83, 0x7f, 0x8d, 0xab, 0x41, 0xfb, 0xde, 0xa9 | EFI\_AUTHENTICATION\_INFO\_PROTOCOL.Set()parameter *length* invalid checking test. | Call Set() with (GenericAuthenticationNodeStruct.length<18) should return EFI\_INVALID\_PARAMETER. |

## EFI\_HASH2\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_HASH2\_PROTOCOL Section.

### GetHashSize ()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.2.1.1.1 | 0xf70cb8e0, 0x2c12, 0x4976, 0xaf, 0xc9, 0xac, 0x90, 0xda, 0xae, 0x6e, 0x20 | EFI\_HASH2\_PROTOCOL. GetHashSize() - GetHashSize()returns EFI\_SUCCESSwith valid parameters and HashSize match the HashAlgorithm. | 1. Call GetHashSize() with the valid parameters, the return status should be EFI\_SUCCESS and returned HashSize should match the HashAlgorithm. |
| 25.2.1.1.2 | 0xb86858d8, 0xcb57, 0x4978, 0x9d, 0xed, 0xe7, 0xc7, 0xb1, 0x6, 0x75, 0xd7 | EFI\_HASH2\_PROTOCOL. GetHashSize() - GetHashSize()returns EFI\_UNSUPPORTEDwith unsupportedHashAlgorithm or HashAlgorithm being NULL. | 1. Call GetHashSize() with unsupported HashAlgorithm or HashAlgorithm being NULL, the return status should be EFI\_UNSUPPORTED. |
| 25.2.1.1.3 | 0x9a001932, 0x3abd, 0x4cca, 0x88, 0xb5, 0xdb, 0xa1, 0x58, 0xc5, 0xdb, 0xef | EFI\_HASH2\_PROTOCOL. GetHashSize() - GetHashSize()returns EFI\_INVALID\_PARAMETER when HashSize is NULL. | 1. Call GetHashSize() when HashSize is NULL, the return status should be EFI\_INVALID\_PARAMETER. |
|  |  |  |  |

### Hash()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.2.1.2.1 | 0xf6905190, 0x3664, 0x4ff9, 0xac, 0x68, 0xce, 0x78, 0x24, 0x6b, 0x2a, 0x51 | EFI\_HASH2\_PROTOCOL. Hash() -  Hash() returns EFI\_SUCCESS with valid parameters and Hash2Out should be correct. | 1. Call GetHashSize() to get the  supported HashAlgorithm.  2. Call Hash() with the supported  HashAlgorithm. The return status should be EFI\_SUCCESS. Hash ourput should be correct. |
| 25.2.1.2.2 | 0x89690c0c, 0x63c1, 0x40ab, 0x9b, 0x91, 0xfe, 0xd2, 0x32, 0x1a, 0x3e, 0x99 | EFI\_HASH2\_PROTOCOL. Hash() - Hash()  returns EFI\_UNSUPPORTED with unsupported HashAlgorithm or HashAlgorithm being NULL. | 1. Call Hash() with unsupported HashAlgorithm or HashAlgorithm being NULL, the return status should be EFI\_UNSUPPORTED. |
| 25.2.1.2.3 | 0xb9cceaa1, 0x3b8f, 0x45e3, 0x8a, 0x27, 0x99, 0x45, 0x3e, 0xb4, 0xd1, 0xbb | EFI\_HASH2\_PROTOCOL. Hash() - Hash()  returns EFI\_INVALID\_PARAMETER when Hash is NULL. | 1. Call Hash() when Hash is NULL, the return status should be EFI\_INVALID\_PARAMETER. |
|  |  |  |  |

### HashInit()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.2.1.3.1 | 0x644e5fa7, 0x3d9b, 0x4a7b, 0xb1, 0x4e, 0x43, 0x34, 0x28, 0xf1, 0x60, 0xdb | EFI\_HASH2\_PROTOCOL. HashInit() - HashInit()returns EFI\_UNSUPPORTEDwith unsupported HashAlgorithm or HashAlgorithm being NULL. | 1. Call HashInit() with unsupported HashAlgorithm or HashAlgorithm being NULL, the return status should be EFI\_UNSUPPORTED. |
| 25.2.1.3.2 | 0x622e2357, 0xc5ff, 0x46b7, 0xab, 0xe7, 0xdb, 0x5e, 0x76, 0xbd, 0xca, 0xa9 | EFI\_HASH2\_PROTOCOL. HashInit() - HashInit()returns EFI\_ALREADY\_STARTED when it follows the call to HashInit(). | 1. Call HashInit() when it follows the call to HashInit(), the return status should be EFI\_ALREADY\_STARTED. |
| 25.2.1.3.3 | 0x69c8ed23, 0xf7fd, 0x4122, 0xb3, 0x1a, 0x46, 0xf8, 0x48, 0x11, 0xa5, 0x77 | EFI\_HASH2\_PROTOCOL. HashInit() - HashInit()returns EFI\_ALREADY\_STARTED when it follows the call to HashUpdate(). | 1. Call HashInit() when it follows the call to HashUpdate(), the return status should be EFI\_ALREADY\_STARTED. |
|  |  |  |  |

### HashUpdate()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.2.1.4.1 | 0xa6a79ffd, 0x7e93, 0x4302, 0xb5, 0xaf, 0xe5, 0x43, 0xc5, 0x16, 0x35, 0x95 | EFI\_HASH2\_PROTOCOL. HashUpdate() - HashUpdate()returns EFI\_NOT\_READY when it is not preceded by a call to HashInit(). | 1. Call HashUpdate() when it is not preceded by the call to HashInit(),the return status should be EFI\_NOT\_READY. |
| 25.2.1.4.2 | 0x4021bf59, 0x8fab, 0x4a5e, 0xa8, 0x6b, 0x3e, 0xad, 0xa2, 0x78, 0xb3, 0x72 | EFI\_HASH2\_PROTOCOL. HashUpdate() - HashUpdate()returns EFI\_NOT\_READY when it follows the call to Hash(). | 1. Call HashUpdate() when it follows the call to Hash(), the return status should be EFI\_NOT\_READY. |
| 25.2.1.4.3 | 0xf7cd2a58, 0x18f9, 0x4285, 0xb9, 0x2b, 0x22, 0x76, 0x7e, 0xff, 0xc8, 0xf5 | EFI\_HASH2\_PROTOCOL. HashUpdate() - HashUpdate()returns EFI\_NOT\_READY when it follows the call to HashFinal(). | 1. Call HashUpdate() when it follows the call to HashFinal(), the return status should be EFI\_NOT\_READY. |
|  |  |  |  |

### HashFinal()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.2.1.5.1 | 0xd66d9eb8, 0x52a9, 0x415d, 0xa9, 0x15, 0x7b, 0x50, 0xb8, 0x53, 0x34, 0x5a | EFI\_Hash2\_PROTOCOL.HashFinal() -HashFinal()returnsEFI\_SUCCESSwith valid parameters. | 1. Call GetHashSize()to get thesupported HashAlgorithm.2. Call HashInit() with the supported HashAlgorithm, the return status should be EFI\_SUCCESS.3. Call HashInit() with the supported HashAlgorithm, the return status should be EFI\_ALREADY\_STARTED.4. Call HashUpdate() with the updated message, the return status should be EFI\_SUCCESS.5. Call HashUpdate() with the updated message, the return status should be EFI\_SUCCESS.6. Call HashFinal() to get the Hash output. The return status should be EFI\_SUCCESS. Hash output should be correct. |
| 25.2.1.5.2 | 0x459f2e7e, 0x1a98, 0x44c6, 0x97, 0xe, 0x38, 0x92, 0x67, 0xdb, 0xe1, 0x57 | EFI\_Hash2\_PROTOCOL.HashFinal() -HashFinal()returnsEFI\_NOT\_READY when it is not preceded by the call to HashInit()/HashUpdate(). | 1. Call HashFinal() when it is not preceded by the call to HashInit()/HashUpdate(),the return status should be EFI\_NOT\_READY. |
| 25.2.1.5.3 | 0x57baa339, 0xab9b, 0x4cb7, 0x8e, 0xed, 0xeb, 0x97, 0x68, 0x82, 0xaf, 0x6b | EFI\_Hash2\_PROTOCOL.HashFinal() -HashFinal()returnsEFI\_NOT\_READY when it is not preceded by the call to HashUpdate(). | 1. Call HashFinal() when it is not preceded by the call to HashUpdate(),the return status should be EFI\_NOT\_READY. |
| 25.2.1.5.4 | 0x69af3be6, 0x3ac2, 0x467c, 0x8c, 0x41, 0x74, 0xd4, 0x53, 0x2f, 0x66, 0xa6 | EFI\_Hash2\_PROTOCOL.HashFinal() -HashFinal()returnsEFI\_NOT\_READY when it follows the call to Hash(). | 1. Call HashFinal() when it follows the call to Hash(), the return status should be EFI\_NOT\_READY. |
| 25.2.1.5.5 | 0x6022b449, 0x9fe1, 0x4bd9, 0x84, 0x9c, 0x67, 0x9e, 0x7f, 0x7, 0xa5, 0xfe | EFI\_Hash2\_PROTOCOL.HashFinal() -HashFinal()returnsEFI\_INVALID\_PARAMETER when Hash is NULL. | 1. Call HashFinal() when Hash is NULL, the return status should be EFI\_INVALID\_PARAMETER. |
| 25.2.1.5.6 | 0x2a6201e8, 0xe536, 0x4e92, 0xb6, 0x4e, 0x8e, 0xbd, 0xc6, 0xfe, 0xe0, 0x25 | EFI\_Hash2\_PROTOCOL.HashFinal() -HashFinal()returnsEFI\_NOT\_READY when it follows the call to HashFinal(). | 1. Call HashFinal() when it follows the call to HashFinal(), the return status should be EFI\_NOT\_READY. |

## EFI\_PKCS7\_VERIFY\_PROTOCOL Test

Reference Document:

*UEFI Specification*, EFI\_PKCS7\_VERIFY\_PROTOCOL Section.

### VerifyBuffer()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 25.3.1.1.1 | 0x5c0eec50, 0xa6ea, 0x413c, 0x8a, 0x46, 0x4a, 0xd1, 0x4a, 0x77, 0x76, 0xf1 | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_SUCCESSwhen content signature was verified against hash of content, the signer's certificate was not found in RevokedDb, and was found in AllowedDb. | 1. Call VerifyBuffer() when content signature was verified against hash of content, the signer's certificate was not found in RevokedDb, and was found in AllowedDb, the return status should be EFI\_SUCCESS. |
| 25.3.1.1.2 | 0x6ea61fbd, 0x1e46, 0x4854, 0x83, 0xf8, 0x22, 0x93, 0x24, 0x1a, 0x38, 0x67 | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_BUFFER\_TOO\_SMALLwhen the size of buffer indicated by ContentSize is too small to hold the content. ContentSize should be updated to required size. | 1. Call VerifyBuffer() when the size of buffer indicated by ContentSize is too small to hold the content, the return status should be EFI\_BUFFER\_TOO\_SMALL. ContentSize should be updated to required size. |
| 25.3.1.1.3 | 0x51af2845, 0x1bfe, 0x4bc3, 0x90, 0x69, 0x7b, 0x29, 0xbc, 0x7c, 0xc3, 0xc6 | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_SUCCESSwhen the size of buffer indicated by ContentSize is big enough to hold the content, and retrive the correct content. | 1. Call VerifyBuffer() when the size of buffer indicated by ContentSize is big enough to hold the content, and retrive the correct content, the return status should be EFI\_SUCCESS. |
| 25.3.1.1.4 | 0x912e23ef, 0x299c, 0x41ab, 0xa0, 0xf5, 0xfc, 0xbc, 0xf6, 0xfd, 0xd3, 0x32 | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_SUCCESSwhen the content signature was verified against hash of content, signer is found in both AllowedDb and RevokedDb, the signing was allowed by reference to TimeStampDb. | 1. Call VerifyBuffer() when the content signature was verified against hash of content, signer is found in both AllowedDb and RevokedDb, the signing was allowed by reference to TimeStampDb, the return status should be EFI\_SUCCESS. |
| 25.3.1.1.5 | 0x5ccc7dff, 0xc397, 0x4733, 0xb6, 0xc7, 0x88, 0xc4, 0x3e, 0x80, 0x6a, 0x67 | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_UNSUPPORTEDwhen SignedData is NULL or SignedDataSize is 0 or AllowedDb is NULL or Content is not NULL and ContentSize is NULL. | 1. Call VerifyBuffer() when SignedData is NULL or SignedDataSize is 0 or AllowedDb is NULL or Content is not NULL and ContentSize is NULL, the return status should be EFI\_INVALID\_PARAMETER. |
| 25.3.1.1.6 | 0xb1f546c3, 0x4e, 0x4e33, 0xb1, 0x81, 0x76, 0xf3, 0xf8, 0xb1, 0xd6, 0x5b | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_UNSUPPORTED when SignedData buffer is not correctly formatted for processing. | 1. Call VerifyBuffer() when SignedData buffer is not correctly formatted for processing, the return status should be EFI\_UNSUPPORTED. |
| 25.3.1.1.7 | 0xf9382c57, 0xd51d, 0x4ba9, 0x91, 0x41, 0x30, 0xc6, 0x28, 0x8b, 0xd3, 0x64 | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_ABORTED when AllowedDb is invalid format. | 1. Call VerifyBuffer() when AllowedDb is invalid format, the return status should be EFI\_ ABORTED. |
| 25.3.1.1.8 | 0x3b322e30, 0x8378, 0x441a, 0xba, 0x1d, 0xee, 0xe5, 0x53, 0xda, 0x21, 0x49 | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_ABORTED when RevokedDb is invalid format. | 1. Call VerifyBuffer() when RevokedDb is invalid format, the return status should be EFI\_ ABORTED. |
| 25.3.1.1.9 | 0xdfe02003, 0xb2ad, 0x46bc, 0xae, 0xe0, 0xf9, 0xb8, 0xd0, 0xec, 0xd3, 0x4a | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_ABORTED when TimeStampDb is invalid format. | 1. Call VerifyBuffer() when TimeStampDb is invalid format, the return status should be EFI\_ ABORTED. |
| 25.3.1.1.10 | 0x8de626c4, 0x7112, 0x4a57, 0xb2, 0xbb, 0x30, 0xc, 0x5f, 0x2a, 0xc1, 0x8e | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_SECURITY\_VIOLATION when Buffer is correctly formatted but signer is not in AllowedDb. | 1. Call VerifyBuffer() when Buffer is correctly formatted but signer is not in AllowedDb, the return status should be EFI\_SECURITY\_VIOLATION. |
| 25.3.1.1.11 | 0x399e1246, 0xd15a, 0x491a, 0xbb, 0x82, 0x99, 0xa4, 0xda, 0xb3, 0xac, 0x28 | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_SECURITY\_VIOLATION when Buffer is correctly formatted but signer is in RevokedDb. | 1. Call VerifyBuffer() when Buffer is correctly formatted but signer is in RevokedDb, the return status should be EFI\_SECURITY\_VIOLATION. |
| 25.3.1.1.12 | 0x670b4eab, 0xf28d, 0x42db, 0xa7, 0xbc, 0xad, 0xd, 0x59, 0x80, 0x49, 0xaf | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_SECURITY\_VIOLATION when Buffer is correctly formatted but the content hash is in RevokedDb. | 1. Call VerifyBuffer() when Buffer is correctly formatted but the content hash is in RevokedDb, the return status should be EFI\_SECURITY\_VIOLATION. |
| 25.3.1.1.13 | 0xfd98e4e5, 0xf8af, 0x4dcf, 0x81, 0x1a, 0x6c, 0xf4, 0x99, 0x8a, 0x3, 0x9d | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_UNSUPPORTEDwhen Signed data embedded in SignedData but InData is not NULL. | 1. Call VerifyBuffer() when Signed data embedded in SignedData but InData is not NULL, the return status should be EFI\_UNSUPPORTED. |
| 25.3.1.1.14 | 0xb136e016, 0x4f80, 0x44bd, 0xba, 0xb0, 0x1c, 0x34, 0x8a, 0x2d, 0xa1, 0x8a | EFI\_PKCS7\_VERIFY\_PROTOCOL.VerifyBuffer() - VerifyBuffer()returns EFI\_ NOT\_FOUNDwhen InData is NULL and no content embedded in SignedData. | 1. Call VerifyBuffer() when InData is NULL and no content embedded in SignedData, the return status should be EFI\_NOT\_FOUND. |