**# Title**: PI Post Quantum Cryptography (PQC) support

**# Status**: Submitted to industry standard forum

**# Document**: PI Specification Version 1.7 (<https://uefi.org/sites/default/files/resources/PI_Spec_1_7_A_final_May1.pdf>)

**# License**: SPDX-License-Identifier: CC-BY-4.0

**# Submitter**: [TianoCore Community](<https://www.tianocore.org>)

**# Summary of the change**

**[Background]**

Please refer to UEFI Post-Quantum Cryptography Support (<https://bugzilla.tianocore.org/show_bug.cgi?id=4087>)

**[Proposal]**

Add PQC support in PI spec for signed FV/Section.

**# Benefits of the change**

NSA CNSA 2.0 Compliance

**# Impact of the change**

**Reference:**

[1] UEFI Post-Quantum Cryptography Support (<https://bugzilla.tianocore.org/show_bug.cgi?id=4087>)

**# Detailed description of the change [normative updates]**

ADD means ADD, DELETE means DELETE

**PI Specification Vol 3.**

**3.2.1.1 EFI Signed Firmware Volumes**There may be one or more headers with a *FormatType* of value  
**EFI\_FIRMWARE\_CONTENTS\_SIGNED\_GUID.**A *signed firmware volume* is a cryptographic signature across the entire volume. To process the  
contents and verify the integrity of the volume, the  
**EFI\_FIRMWARE\_VOLUME\_EXT\_ENTRY\_GUID\_TYPE** *Data[]*shall contain an instance of  
**WIN\_CERTIFICATE\_UEFI\_GUID** where the *CertType* **=  
EFI\_CERT\_TYPE\_PKCS7\_GUID**or **EFI\_CERT\_TYPE\_RSA2048\_SHA256\_GUID**, **EFI\_CERT\_TYPE\_RSA3072\_SHA384\_GUID**, **EFI\_CERT\_TYPE\_RSA4096\_SHA512\_GUID**, **EFI\_CERT\_TYPE\_RSAPSS3072\_SHA512\_GUID**, **EFI\_CERT\_TYPE\_RSAPSS4096\_SHA512\_GUID**, **EFI\_CERT\_TYPE\_ECDSA\_ECC\_NIST\_P256\_SHA256\_GUID**, **EFI\_CERT\_TYPE\_ECDSA\_ECC\_NIST\_P384\_SHA384\_GUID**, **EFI\_CERT\_TYPE\_LMS\_GUID**, **EFI\_CERT\_TYPE\_XMSS\_GUID**, **EFI\_CERT\_TYPE\_MLDSA\_GUID**, **EFI\_CERT\_TYPE\_HYBRID\_GUID**.

**Related Definitions**

**#define EFI\_CERT\_TYPE\_LMS\_GUID  
{0x7c650d3e, 0x6112, 0x4ead, \**

**{0x83, 0x1c, 0xbf, 0xa3, 0x73, 0xe8, 0xdf, 0xf0}}**

**#define EFI\_CERT\_TYPE\_XMSS\_GUID  
{0x39116961, 0x5d7f, 0x4f69, \**

**{0xb0, 0x50, 0x2c, 0x1a, 0xe9, 0x63, 0x57, 0xc2}}**

**#define EFI\_CERT\_TYPE\_MLDSA\_GUID  
{0x50bf8d2e, 0x81f5, 0x40ea, \**

**{0x8b, 0xd5, 0xa7, 0xb0, 0xe, 0x72, 0x62, 0xc7}}**

**#define EFI\_CERT\_TYPE\_HYBRID\_GUID  
{0xa54d7ea, 0x7fd4, 0x4e25, \**

**{0xad, 0x15, 0x4, 0x4c, 0x43, 0xe5, 0x1, 0x88}}**

**typedef struct \_EFI\_CERT\_BLOCK\_HSS {  
 EFI\_GUID** *HashType***;  
 UINT8** *PublicKey***[];  
 UINT8** *Signature***[];  
} EFI\_CERT\_BLOCK\_HSS;**

*PublicKey* The public key block is “HSS public key” in RFC 8554. Byte [7:4] LMS algorithm type (numeric identifier) is defined in NIST SP 800-208.

*Signature* This signature block is “HSS signature” in RFC 8554.

**typedef struct \_EFI\_CERT\_BLOCK\_XMSS {  
 EFI\_GUID** *HashType***;  
 UINT8** *PublicKey***[];  
 UINT8** *Signature***[];  
} EFI\_CERT\_BLOCK\_XMSS;**

*PublicKey* The public key block is “XMSS public key” in RFC 8391. Byte [3:0] XMSS algorithm type (numeric identifier) is defined in NIST SP 800-208.

*Signature* This signature block is “XMSS signature” in RFC 8391.

**typedef struct \_EFI\_CERT\_BLOCK\_MLDSA {  
 EFI\_GUID** *HashType***;**

**EFI\_MLDSA\_TYPE** *AlgorithmType***;  
 UINT8** *PublicKey***[];  
 UINT8** *Signature***[];  
} EFI\_CERT\_BLOCK\_MLDSA;**

*AlgorithmType* The algorithm type of MLDSA defined in FISP 204.

*PublicKey* The public key block is “public key” of MLDSA in FISP 204.

*Signature* This signature block is “signature” of MLDSA in FISP 204.

**typedef UINT8 EFI\_MLDSA\_TYPE;**

**#define EFI\_MLDSA\_TYPE\_MLDSA\_44 0**

**#define EFI\_MLDSA\_TYPE\_MLDSA\_65 1**

**#define EFI\_MLDSA\_TYPE\_MLDSA\_87 2**

**typedef struct \_EFI\_CERT\_BLOCK\_HYBRID {  
 EFI\_GUID** *CertType1***;  
 EFI\_GUID** *CertType2***;  
 UINT8** *CertBlock1***[];  
 UINT8** *CertBlock2***[];  
} EFI\_CERT\_BLOCK\_HYBRID;**

*CertBlock1* The first cert block **EFI\_CERT\_BLOCK\_\*** determined by *CertType1*. It should be one of PQC digital signature algorithm, such as LMS, XMSS, MLDSA.

*CertBlock2* The second cert block **EFI\_CERT\_BLOCK\_\*** determined by *CertType2*. It should be one of classic digital signature algorithm, such as RSA, ECDSA.

**Description**

The **WIN\_CERTIFICATE\_UEFI\_GUID** certificate type allows new types of certificates to be developed  
for driver authentication without requiring a new certificate type. The *CertType* defines the format of  
the *CertData*, which length is defined by the size of the certificate less the fixed size of the  
**WIN\_CERTIFICATE\_UEFI\_GUID** structure. Besides UEFI defined **EFI\_CERT\_TYPE\_PKCS7\_GUID** or **EFI\_CERT\_TYPE\_RSA2048\_SHA256\_GUID**, this specification adds below *CertType*:

• If *CertType* is **EFI\_CERT\_TYPE\_LMS\_GUID** then the structure which follows has the format specified by **EFI\_CERT\_BLOCK\_HSS**.  
• If *CertType* is **EFI\_CERT\_TYPE\_XMSS\_GUID** then the structure which follows has the format specified by **EFI\_CERT\_BLOCK\_XMSS**.  
• If *CertType* is **EFI\_CERT\_TYPE\_MLDSA\_GUID** then the structure which follows has the format specified by **EFI\_CERT\_BLOCK\_MLDSA**.  
• If *CertType* is **EFI\_CERT\_TYPE\_HYBRID\_GUID** then the structure which follows has the format specified by **EFI\_CERT\_BLOCK\_HYBRID**.

**3.2.5 Firmware File Section Types**

**…**

**EFI Signed Sections**For **EFI\_GUID\_DEFINED\_SECTION** and **EFI\_GUID\_DEFINED\_SECTION2** there is a  
*SectionDefinitionGuid* of type **EFI\_FIRMWARE\_CONTENTS\_SIGNED\_GUID.**The *GuidSpecificHeaderFields* shall include an entry *SignatureInfo* of type  
**WIN\_CERTIFICATE\_UEFI\_GUID***.***#define EFI\_FIRMWARE\_CONTENTS\_SIGNED\_GUID \  
{ 0xf9d89e8, 0x9259, 0x4f76, \**   
**{ 0xa5, 0xaf, 0xc, 0x89, 0xe3, 0x40, 0x23, 0xdf } }**The *signed section* is an encapsulation section in which the section data is cryptographically signed.  
To process the contents and extract the enclosed section stream, the section data integrity must be  
accessed by evaluating the enclosed data via the cryptographic information in the  
*SignatureInfo*. The *CertType* **= EFI\_CERT\_TYPE\_PKCS7\_GUID** or  
**EFI\_CERT\_TYPE\_RSA2048\_SHA256\_GUID**, **EFI\_CERT\_TYPE\_RSA3072\_SHA384\_GUID**, **EFI\_CERT\_TYPE\_RSA4096\_SHA512\_GUID**, **EFI\_CERT\_TYPE\_RSAPSS3072\_SHA512\_GUID**, **EFI\_CERT\_TYPE\_RSAPSS4096\_SHA512\_GUID**, **EFI\_CERT\_TYPE\_ECDSA\_ECC\_NIST\_P256\_SHA256\_GUID**, **EFI\_CERT\_TYPE\_ECDSA\_ECC\_NIST\_P384\_SHA384\_GUID**, **EFI\_CERT\_TYPE\_LMS\_GUID**, **EFI\_CERT\_TYPE\_XMSS\_GUID**, **EFI\_CERT\_TYPE\_MLDSA\_GUID**, **EFI\_CERT\_TYPE\_HYBRID\_GUID**.

The signed image is then interpreted as a section stream. **EFI\_GUID\_DEFINED\_SECTION2** is  
used if the section is 16MB or larger.