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

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

**# Document**: UEFI Specification Version 2.10 (<https://uefi.org/specs/UEFI/2.10>)

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

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

**# Summary of the change**

**[Background]**

1. **PQC Progress**

In 2020, NIST Stateful Hash Based Signature (HBS) project [1] published SP 200-208 [2] and announced 2 HBS algorithm XMSS [3] and LMS [4] for firmware secure boot and firmware image update use case.

In 2022, NIST Post-Quantum Cryptography (PQC) project [5] announced PQC Selected Algorithm 2022 [6] for general purpose Public-Key Encryption and Key-Establishment Algorithms (KEM) and Digital Signature Algorithms (SIG). NIST is continuing round 4 for KEM and calling for additional digital signature proposals [7].

In September 2022, NSA announced Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) Cybersecurity Advisory (CSA) [8] as an update for CNSA 1.0 [9]. The CNSA 2.0 chose KYBER [10] and DILITHIUM [11] in Selected Algorithm 2022.

Table 1 shows the CNSA 2.0 algorithms. The new PQC algorithms are highlighted.

Table 1. CNSA 2.0 algorithms summary (Source: [8])

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Algorithm** | **Standard** | **Parameter** |
| Firmware Signing  (**Stateful HBS**) | Leighton-Micali Signature (LMS) | NIST SP 800-208 [2]  RFC 8554 [4] | All approved.  SHA-256/192 recommended. |
| eXtended Merkle Signature Scheme (XMSS) | NIST SP 800-208 [2]  RFC 8391 [3] | All approved. |
| General Use Public-Key Algorithms - Key Establishment  (**PQC KEM**) | CRYSTALS-Kyber | CRYSTALS-KYBER [10] | Level V parameters. |
| General Use Public-Key Algorithms -Digital Signature (**PQC SIG**) | CRYSTALS-Dilithium | CRYSTALS-DILITHIUM [11] | Level V parameters. |
| Symmetric-Key Algorithms | Advanced Encryption Standard (AES) | FIPS PUB 197 [12] | 256-bit keys |
| Secure Hash Algorithm (SHA) | FIPS PUB 180-4 [13] | SHA-384 or SHA-512 |

Table 2 shows the CNSA 2.0 timing requirement. NSA recommend the software and firmware signing begin transitioning immediately.

Table 2. CNSA 2.0 timing requirement (Source: [8])

|  |  |  |
| --- | --- | --- |
| **Category** | **Support and preferred** | **Exclusively used** |
| Software and firmware signing | 2025 | 2030 |
| Web browsers/servers and cloud services | 2025 | 2033 |
| Traditional networking equipment | 2026 | 2030 |
| Operating systems | 2027 | 2033 |
| Niche equipment | 2030 | 2033 |
| Custom applications and legacy equipment | 2033 | 2033 |

Both NIST and NSA allows the hybrid mode for digital signature and key establishment in the transition and migration phase [14]. The hybrid mode should be supported.

1. **UEFI Touchpoint**

Table 3 shows the potential PQC asymmetric algorithms touch point of a UEFI firmware. Please be aware that the table is not a full list. An OEM firmware may only adopt part of features or have other features.

Table 3. UEFI firmware potential asymmetric algorithm touchpoint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Use Case** | **Standard** | **Algorithm** | **Comment** |
| Firmware Image Verification | UEFI Secure Boot – Image Verification | UEFI,  Authenticode [15], [17] | Stateful HBS | Stateful HBS Algo ID WIP [18]. |
| UEFI FMP Signed Capsule | UEFI,  PKCS7 [16] | Stateful HBS | Stateful HBS Algo ID WIP [18]. |
| PI Signed FV/Section | PI | Stateful HBS | Need add stateful HBS. |
| Firmware Data Verification | UEFI Secure Boot – Auth Variable Update | UEFI,  PKCS7 [16] | Stateful HBS / PQC SIG | Stateful HBS Algo ID WIP [18]. |
| Network Secure Communication | TLS  (HTTPS boot, RedFish) | IETF TLS [21], [22] | PQC KEM + PQC SIG | PQC Support WIP [23].  PQC SIG Algo ID WIP [24]. |
| IPSec | IETF IPSec [25] | PQC KEM + PQC SIG | PQC Support WIP [26]. |
| Device Secure Communication | SPDM  (Device Authentication, Measurement, Session) | DMTF SPDM [27] | PQC KEM + PQC SIG | PQC Support WIP. |

Table 4 shows more detail on the algorithm usage in verification.

Table 4. UEFI firmware image/data verification

|  |  |  |  |
| --- | --- | --- | --- |
| **Category** | **Use Case** | **Verification Chain** | **Algorithm** |
| Firmware Image Verification | UEFI Secure Boot – Image Verification | db/dbx/dbt -> UEFI Image | Stateful HBS |
| PK/KEK/dbr/dbx -> Recovery Image | Stateful HBS |
| UEFI FMP Signed Capsule | FMP Cert -> FMP image | Stateful HBS |
| PI Signed FV/Section | FV/Section pub key -> FV/Section | Stateful HBS |
| Firmware Data Verification | UEFI Secure Boot – Auth Variable Update | PK -> PK/KEK | Stateful HBS / PQC SIG (?) |
| PK/KEK -> db/dbx/dbt/dbr | Stateful HBS / PQC SIG (?) |
| PK/KEK/dbr -> OsRecoveryOrder/ OsRecovery#### | Stateful HBS / PQC SIG (?) |
| Var Cert -> Private Auth Variable | Stateful HBS / PQC SIG (?) |

***Open for discussion****:*

*The UEFI image verification only requires stateful HBS. However, the authenticated variable is not UEFI image. We need determine if we only allow stateful HBS or we should allow general PQC SIG. Especially, the “dbr” will be used to verify both recovery image and auth variable data.*

*Using stateful HBS for auth variable will bring limitation for the number of signings. It seems OK because we do not see frequent update for those variables.*

*Using PQC SIG for auth variable (such as PK, KEK) will cause mixing stateful HBS and PQC SIG. The firmware image size may be increased, because the firmware need include both stateful HBS and PQC SIG.*

Table 5 shows the potential symmetric algorithms touch point of a UEFI firmware. Please be aware that the table is not a full list. An OEM firmware may only adopt part of features or have other features.

Table 5. UEFI potential symmetric algorithm touchpoint

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Category** | **Use Case** | **Standard** | **Algorithm** | **Comment** |
| Measured Boot | TCG Measured Boot | TCG PFP [28] | SHA | Already supported. |
| Network | iSCSI CHAP | IETF iSCSI [29] | SHA | open-iscsi added SHA256 [30].  Need move to SHA384. |
| Link Encryption | PCI/CXL Integrity and Data Encryption (IDE) | PCI IDE [31], CXL IDE [32] | AES | Already supported. |
| Storage | RPMC SPI Variable | RPMC [33] | SHA | Only support SHA256. |
| NVMe/eMMC/UFS RPMB Storage | NVMe [34],  eMMC [35], UFS [36] | SHA | Only support SHA256. |
| Password | HDD ATA [37] Password | N/A (EDKII) | SHA | Store password hash. [38]  (Implementation)  Need move to SHA384. |
| User Password (Authentication) | N/A (EDKII) | SHA | Store password hash. [39]  (Implementation)  Need move to SHA384. |

**[Proposal]**

1. Add PQC support in UEFI spec.
   1. Support PQC symmetric algorithm. Already done in UEFI 2.10 via crypto agile.
   2. **Support stateful HBS in Authenticode/PKCS7** for signed image, capsule, or auth variable (?). (In this proposal).
   3. **Support PQC SIG in PKCS7** for auth variable (?). (In this proposal).
   4. **Support hybrid mode**. (In this proposal).
2. Add PQC support in PI spec.
   1. **Support stateful HBS in raw format** for signed FV/Section. (In PI Post-Quantum Cryptography proposal).
3. Wait for other specification standardize:
   1. PQC algorithm (stateful HBS and PQC SIG) OID in PKCS7 and X.509 by IETF lamps WG [40].
   2. Crypto protocols with PQC algorithm, such as TLS protocol by IETF tls WG [41], IPSec protocol by IETF ipsecme WG [42], SPDM protocol by DMTF SPDM WG [43].

**# Benefits of the change**

NSA CNSA 2.0 Compliance

**# Impact of the change**

**Reference:**

[1] NIST, Stateful Hash Based Signatures Project, <https://csrc.nist.gov/Projects/stateful-hash-based-signatures>

[2] NIST, SP 800-208 - Recommendation for Stateful Hash-Based Signature Schemes, October 2020, <https://csrc.nist.gov/publications/detail/sp/800-208/final>

[3] IETF, RFC 8391 - XMSS: eXtended Merkel Signature Scheme, May 2018, <https://datatracker.ietf.org/doc/rfc8391/>

[4] IETF, RFC 8554 - Leighton-Micali Hash-Based Signature, April 2019, <https://datatracker.ietf.org/doc/rfc8554/>

[5] NIST, Post-Quantum Cryptography Project, <https://csrc.nist.gov/Projects/post-quantum-cryptography>

[6] NIST, PQC Selected Algorithms 2022, <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>

[7] NIST, Post-Quantum Cryptography Digital Signature Schemes Project <https://csrc.nist.gov/Projects/pqc-dig-sig>

[8] NSA, Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) Cybersecurity Advisory (CSA), September 2022, <https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF>

[9] NSA, Commercial National Security Algorithm (CNSA) Suite 1.0, <https://media.defense.gov/2021/Sep/27/2002862527/-1/-1/0/CNSS%20WORKSHEET.PDF>

[10] CRYSTALS-KYBER, <https://pq-crystals.org/kyber/index.shtml>

[11] CRYSTALS-DILITHIUM, <https://pq-crystals.org/dilithium/index.shtml>

[12] NIST, FIPS 197, November 2001, <https://csrc.nist.gov/publications/detail/fips/197/final>

[13] NIST, FIPS 180-4, August 2015, <https://csrc.nist.gov/publications/detail/fips/180/4/final>

[14] NIST, PQC FAQs, <https://csrc.nist.gov/Projects/post-quantum-cryptography/faqs>

[15] Microsoft, PE Format (Authenticode), <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format> , <https://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/Authenticode_PE.docx>

[16] IETF, RFC 2315 - PKCS #7: Cryptographic Message Syntax Version 1.5, <https://datatracker.ietf.org/doc/rfc2315/>

[17] IETF, RFC 8708, Use of the HSS/LMS Hash-Based Signature Algorithm in the Cryptographic Message Syntax (CMS), <https://datatracker.ietf.org/doc/rfc8708/>

[18] IETF, Algorithm Identifiers for HSS and XMSS for Use in the Internet X.509 Public Key Infrastructure, <https://datatracker.ietf.org/doc/draft-vangeest-x509-hash-sigs/>

[19] IANA, Leighton-Micali Signatures (LMS), <https://www.iana.org/assignments/leighton-micali-signatures/leighton-micali-signatures.xhtml>,

[20] IANA, XMSS: Extended Hash-Based Signatures, <https://www.iana.org/assignments/xmss-extended-hash-based-signatures/xmss-extended-hash-based-signatures.xhtml>

[21] IETF, RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3, <https://datatracker.ietf.org/doc/rfc8446/>

[22] IETF, RFC 9147 - The Datagram Transport Layer Security (DTLS) Protocol Version 1.3, <https://datatracker.ietf.org/doc/rfc9147/>

[23] IETF, Hybrid key exchange in TLS 1.3, <https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/>

[24] IETF, Internet X.509 Public Key Infrastructure: Algorithm Identifiers for Dilithium, <https://datatracker.ietf.org/doc/draft-ietf-lamps-dilithium-certificates/>

[25] IETF, RFC 7296 - Internet Key Exchange Protocol Version 2 (IKEv2), <https://datatracker.ietf.org/doc/rfc7296/>

[26] IETF, Multiple Key Exchanges in IKEv2, <https://datatracker.ietf.org/doc/draft-ietf-ipsecme-ikev2-multiple-ke/>

[27] DMTF, DSP0274 - Security Protocol and Data Model, <https://www.dmtf.org/dsp/DSP0274>

[28] TCG, PC Client Platform Firmware Profile (PFP), <https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/>

[29] IETF, RFC 3720 - Internet Small Computer Systems Interface (iSCSI), <https://www.rfc-editor.org/rfc/rfc3720>

[30] open-iscsi, <https://github.com/open-iscsi/open-iscsi>

[31] PCISIG, PCI Express Base Specification Revision 6.0.1, <https://members.pcisig.com/wg/PCI-SIG/document/18363>

[32] CXL, Compute Express Link Specification 3.0, <https://www.computeexpresslink.org/download-the-specification>

[33] JEDEC, JESD260 - Replay Protected Monotonic Counter (RPMC) for Serial Flash Devices, <https://www.jedec.org/standards-documents/docs/jesd260>

[34] NVMe, NVM Express Base Specification 2.0, <https://nvmexpress.org/developers/nvme-specification/>

[35] JEDEC, JESD84-B51 - Embedded Multi-Media Card (eMMC), Electrical Standard (5.1), <https://www.jedec.org/standards-documents/docs/jesd84-b51>

[36] JEDEC, JESD220E - Universal Flash Storage (UFS) Version 3.1, <https://www.jedec.org/standards-documents/docs/jesd220e>

[37] T13, AT Attachment 8 - ATA/ATAPI Command Set, <https://www.t13.org/>

[38] EDKII, HDD Password, <https://github.com/tianocore/edk2/tree/master/SecurityPkg/HddPassword>

[39] EDKII, User Authentication, <https://github.com/tianocore/edk2-platforms/tree/master/Features/Intel/UserInterface/UserAuthFeaturePkg>

[40] IETF, Limited Additional Mechanisms for PKIX and SMIME (lamps) WG, <https://datatracker.ietf.org/wg/lamps/documents/>

[41] IETF, Transport Layer Security (tls) WG, <https://datatracker.ietf.org/wg/tls/documents/>

[42] IETF, IP Security Maintenance and Extensions (ipsecme) WG, <https://datatracker.ietf.org/wg/ipsecme/documents/>

[43] DMTF, Security Protocols and Data Models (SPDM) WG, <https://www.dmtf.org/standards/spdm>

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

ADD means ADD, DELETE means DELETE

**UEFI Specification**

**32.4 Firmware/OS Crypto Algorithm Exchange**

**typedef struct {**

**UINT32 Version;**

**UINT32 Length;**

**UINT64 HashAlgorithmBitmap;**

**UINT64 AsymAlgorithmBitmap;**

**} EFI\_CRYPTO\_INDICATION;**

**#define EFI\_CRYPTO\_INDICATION\_HASH\_SHA\_256 0x1**

**#define EFI\_CRYPTO\_INDICATION\_HASH\_SHA\_384 0x2**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_RSASSA\_2048 0x1**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_RSASSA\_3072 0x2**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_RSAPSS\_3072 0x10**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_ECDSA\_ECC\_NIST\_P256 0x40**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_ECDSA\_ECC\_NIST\_P384 0x80**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_RSASSA\_4096 0x4**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_RSAPSS\_4096 0x20**

The **EFI\_CRYPTO\_INDICATION\_HASH\_SHA\_256** bit means SHA-256 hash algorithm.

The **EFI\_CRYPTO\_INDICATION\_HASH\_SHA\_384** bit means SHA-384 hash algorithm.

The **EFI\_CRYPTO\_INDICATION\_ASYM\_RSASSA\_2048** bit means a signature algorithm defined in section 8.2 (RSASSAPKCS1-v1\_5) in RFC8017. The key size is 2048 bits.

The **EFI\_CRYPTO\_INDICATION\_ASYM\_RSASSA\_3072** bit means a signature algorithm defined in section 8.2 (RSASSAPKCS1-v1\_5) in RFC8017. The key size is 3072 bits.

The **EFI\_CRYPTO\_INDICATION\_ASYM\_RSAPSS\_3072** bit means a signature algorithm defined in section 8.1 (RSASSAPSS) in RFC8017. The key size is 3072 bits.

The **EFI\_CRYPTO\_INDICATION\_ASYM\_ECDSA\_ECC\_NIST\_P256** bit means a signature algorithm defined in section 6 (ECDSA) in FIPS 186-4. The key size is 256 bits.

The **EFI\_CRYPTO\_INDICATION\_ASYM\_ECDSA\_ECC\_NIST\_P384** bit means a signature algorithm defined in section 6 (ECDSA) in FIPS 186-4. The key size is 384 bits.

**32.4.1 Post-Quantum Cryptography**

The UEFI specification adds below post-quantum cryptography algorithms. The Leighton-Micali Signature (LMS) and eXtended Merkle Signature Scheme (XMSS) are stateful hash-based signature (HBS). CRYSTALS-Dilithium is a general use digital signature.

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_LMS\_SHA256\_M32 0x1000**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_LMS\_SHA256\_M24 0x2000**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_XMSS\_SHA2\_256 0x10000**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_XMSS\_SHA2\_192 0x20000**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_DILITHIUM5 0x100000**

The **EFI\_CRYPTO\_INDICATION\_ASYM\_LMS\_SHA256\_M32** bit means a signature algorithm defined in RFC8554 and parameter set defined in section 4.1 (LMS with SHA-256) in NIST SP 800-208.

The **EFI\_CRYPTO\_INDICATION\_ASYM\_LMS\_SHA256\_M24** bit means a signature algorithm defined in RFC8554 and parameter set defined in section 4.2 (LMS with SHA-256/192) in NIST SP 800-208.

The **EFI\_CRYPTO\_INDICATION\_ASYM\_XMSS\_SHA2\_256** bit means a signature algorithm defined in RFC8391 and parameter set defined in section 5.1 (XMSS with SHA-256) in NIST SP 800-208.

The **EFI\_CRYPTO\_INDICATION\_ASYM\_XMSS\_SHA2\_192** bit means a signature algorithm defined in RFC8391 and parameter set defined in section 5.2 (XMSS with SHA-256/192) in NIST SP 800-208.

The **EFI\_CRYPTO\_INDICATION\_ASYM\_DILITHIUM5** bit means a signature algorithm defined in CRYSTALS-DILITHIUM with parameter set Dilithium5 (level V).

**32.4.2 Hybrid Signature**

A hybrid signature algorithm including one post-quantum signature algorithm (such as LMS, XMSS, or Dilithium) and one classic signature algorithm (such as RSA or ECDSA). A hybrid signature is accepted only if both post-quantum signature and classic signature are accepted. The UEFI specification adds below hybrid signature algorithms.

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_LMS\_SHA256\_M32\_RSAPSS\_3072 0x100000000ull**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_LMS\_SHA256\_M32\_ECDSA\_ECC\_NIST\_P384 0x200000000ull**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_LMS\_SHA256\_M24\_RSAPSS\_3072 0x400000000ull**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_LMS\_SHA256\_M24\_ECDSA\_ECC\_NIST\_P384 0x800000000ull**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_XMSS\_SHA2\_256\_RSAPSS\_3072 0x1000000000ull**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_XMSS\_SHA2\_256\_ECDSA\_ECC\_NIST\_P384 0x2000000000ull**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_XMSS\_SHA2\_192\_RSAPSS\_3072 0x4000000000ull**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_XMSS\_SHA2\_192\_ECDSA\_ECC\_NIST\_P384 0x8000000000ull**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_DILITHIUM5\_RSAPSS\_3072 0x10000000000ull**

**#define EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_DILITHIUM5\_ECDSA\_ECC\_NIST\_P384 0x200000000000ull**

The **EFI\_CRYPTO\_INDICATION\_ASYM\_HYBRID\_\*** bit means a hybrid signature algorithm including one post-quantum signature algorithm and one classic signature algorithm. The post-quantum signature algorithms and classic signature algorithms are already defined in previous section.