**# Title**: UEFI Crypto Agile – support crypto OS indicator variable

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

**# Document**: UEFI Specification Version 2.9

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

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

**# Summary of the change**

**[Background]**

This is phase II proposal for UEFI crypto agile - support compatibility.

We need consider:

1. New OS + Old BIOS
2. New BIOS + Old OS

**[Proposal]**

Basic idea:

1. BIOS SHALL enable and only enable SHA256/RSA2048, to support old OS.
2. BIOS SHALL report the activated algo + supported algo to OS.
3. BIOS SHALL support get the preferred algo from OS. The new algo takes effect in next boot.

This idea is similar to TCG Physical Presence Interface specification, which is used to set TPM PCR banks (algorithms).

**# Benefits of the change**

1. New BIOS can work with old OS, which is still using SHA256/RSA2048 algorithm.
2. New OS can take one more step to activate new algorithm such as SHA384/RSA3072.

**# Impact of the change**

**Reference:**

1. UEFI Specification 2.9 - <https://uefi.org/specifications>
2. TCG Physical Presence Interface 1.3 - <https://trustedcomputinggroup.org/resource/tcg-physical-presence-interface-specification/>
3. Commercial National Security Algorithm Suite (CNSA Suite Guide) - <https://apps.nsa.gov/iaarchive/programs/iad-initiatives/cnsa-suite.cfm>

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

ADD means ADD, DELETE means DELETE

**3.3 Globally Defined Variables**

Table 3-1 Global Variable

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Attribute** | **Description** |
| ...... |  |  |
| *OsIndications* | NV, BS, RT | Allows the OS to request the firmware to enable certain features and to take certain action. |
| *OsIndicationsSupported* | BS, RT | Allows the firmware to indicate supported features and actions to OS. |
| *CryptoIndications* | NV, BS, RT | Allows the OS to request the crypto algorithm to BIOS. |
| *CryptoIndicationsSupported* | BS, RT | Allows the firmware to indicate supported crypto algorithm to OS. |
| *CryptoIndicationsActivated* | BS, RT | Allows the firmware to indicate activated crypto algorithm to OS. |
| …… |  |  |

…

The *OsIndicationsSupported* variable indicates which of the OS indication features and actions that the firmware supports. This variable is recreated by firmware every boot, and cannot be modified by the OS (see **SetVariable()**Attributes usage rules once **ExitBootServices()** is performed).

The *OsIndications* variable is used to indicate which features the OS wants firmware to enable or which actions the OS wants the firmware to take. The OS will supply this data with a **SetVariable()** call. See Section 8.5.4 for the variable definition.

The *CryptoIndicationsSupported* variable indicates which crypto algorithms the firmware supports. This variable is recreated by firmware every boot, and cannot be modified by the OS (see **SetVariable()**Attributes usage rules once **ExitBootServices()** is performed).

The *CryptoIndicationsActivated* variable indicates which crypto algorithms the firmware activates. This variable is recreated by firmware every boot, and cannot be modified by the OS (see **SetVariable()**Attributes usage rules once **ExitBootServices()** is performed). It must be a subset of *CryptoIndicationsSupported*.

The *CryptoIndications* variable is used to indicate which crypto algorithms the OS wants firmware to activate in the next boot. It must be a subset of *CryptoIndicationsSupported*. The OS will supply this data with a **SetVariable()** call. See Section 8.5.4 for the variable definition. Once the data is consulted by the firmware and synced to *CryptoIndicationsActivated*, the firmware must delete this variable.

…

**32.4 Firmware/OS Crypto Algorithm Exchange**

The firmware and an Operating System may exchange information through the *CryptoIndicationsSupported*, *CryptoIndicationsActivated* and the *CryptoIndications* variables as follows:

• The *CryptoIndications* variable returns an **EFI\_CRYPTO\_INDICATION** structure owned by the OS and is used to indicate which crypto algorithms the OS wants firmware to activate. The algorithm bitmap in *CryptoIndications* must be a subset of the algorithm bitmap in the *CryptoIndicationsSupported* variable. The OS will supply this data with a **SetVariable()**call. The OS should set one bit. If the OS sets more than one bit, then the firmware shall support all those algorithms. Because this variable can be written by any entity, the firmware shall validate the data before use it. For example, any bits beyond the supported bitmask shall be treated as illegal. The firmware may also combine other validation before accepting the new configuration, such as physical present user action, etc. If error is detected, the firmware may choose to notify end user, ignore this request, or reset to manufacture state, etc. The detailed validation process or error remediation is out of scope of this specification.

• The *CryptoIndicationsSupported* variable returns an **EFI\_CRYPTO\_INDICATION** structure owned by the firmware and indicates which crypto algorithms are supported by the firmware supports. This variable is recreated by firmware every boot and cannot be modified by the OS.

• The *CryptoIndicationsActivated* variable returns an **EFI\_CRYPTO\_INDICATION** structure owned by the firmware and indicates which crypto algorithms are supported by the firmware activates. The algorithm bitmap in *CryptoIndicationsActivated* must be a subset of the algorithm bitmap in the *CryptoIndicationsSupported* variable. This variable is recreated by firmware every boot and cannot be modified by the OS.

**Related Definitions**

**typedef struct {**

**UINT32 Version;**

**UINT32 Length;**

**UINT64 HashAlgorithmBitmap;**

**UINT64 AsymAlgorithmBitmap;**

**} EFI\_CRYPTO\_INDICATION;**

**#define EFI\_CRYPTO\_INDICATION\_VERSION\_1 0x00000001**

**#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**

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.