Unified Extensible Firmware Interface  
Engineering Change Request (ECR)

*Draft for Review*

Title: Device Authentication Signature Database

Document: UEFI Specification

Sponsor: Intel

*Submission for Review Date:*

Review Approval Date: *x/xx/200x*

Submission for Technical Editing Date: *x/xx/200x*

Submission for Draft Review Date: *x/xx/200x*

Verification Date: *x/xx/200x*

Verifier: *firstname lastname, company*

Summary

## Summary of Change

**[Background]**

Today, there is new requirement to not only verify an executable image, but a device on the system. The entity to verify the device might be a standalone platform Root-of-Trust, or the system firmware once the system firmware becomes part of Chain-of-Trust. The Distributed Management Task Force (DMTF) defines Secure Protocol and Data Model (SPDM) specification. The hardware device standard group (such as PCI-SIG, USB, etc) defines the interface to transport the SPDM message for device authentication and measurement. Trusted Computing Group (TCG) also defines the event log for SPDM measurement.

As such, the system firmware needs an architecture way to verify a device on the platform by the device driver based upon a device authentication signature database.

**[Proposal]**

This ECR adds a new device authentication signature database for SPDM device. The device authentication signature database is similar to the UEFI image authentication signature databased used for UEFI secure boot.

For the platform that enables UEFI secure boot, it may optionally enroll the device authentication signature database to authenticate the device besides the UEFI image.

**[Tech Background – SPDM Certificate Chain]**

The SPDM specification defined the Requester and Responder. The Requester could be system firmware. The Responder could be the device to be authenticated.

There are 3 messages are designed for that purpose. See figure 1.

1. GET\_DIGESTS: The system firmware sends this message to get the DIGEST of the SPDM Certificate Chain from the device.
2. GET\_CERTIFICATE: The system firmware sends this message to get the whole SPDM Certificate Chain from the device.
3. CHALLENGE: This is the final step. The system firmware sends this message to challenge the device and expects the device sign the challenge with the device private key.

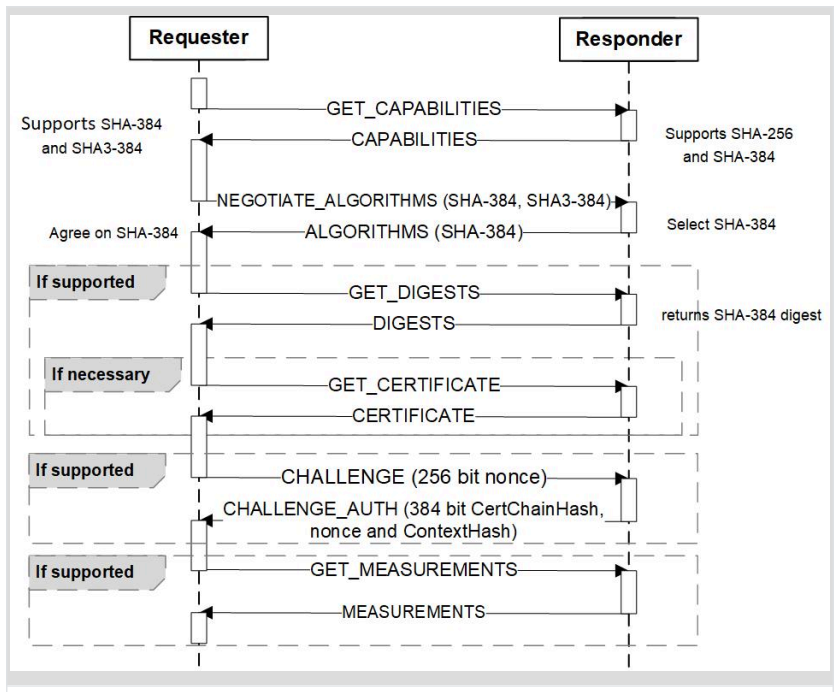
****

Figure 1 – *SPDM Message (Source: SPDM Specification)*

As such, there are at least 2 ways for the verification. See figure 2.

1. The firmware stores the DIGEST of the SPDM Certificate Chain. The firmware must send the GET\_CERTIFICATE message to get the SPDM Certificate Chain then verify the SPDM Certificate Chain. Then the firmware uses the SPDM Certificate Chain to verify the CHALLENG\_AUTH response.
2. The firmware stores the whole SPDM Certificate Chain. The firmware may send GET\_DIFESTS message then verify the SPDM Certificate Chain. Then the firmware uses the SPDM Certificate Chain to verify the CHALLENG\_AUTH response.

Option A consumes small UEFI variable storage but may need more time to get the whole SPDM Certificate Chain at runtime.

Option B consumes large UEFI variable storage but take less time at runtime.

A platform may choose either option based upon the variable storage size and firmware boot time.

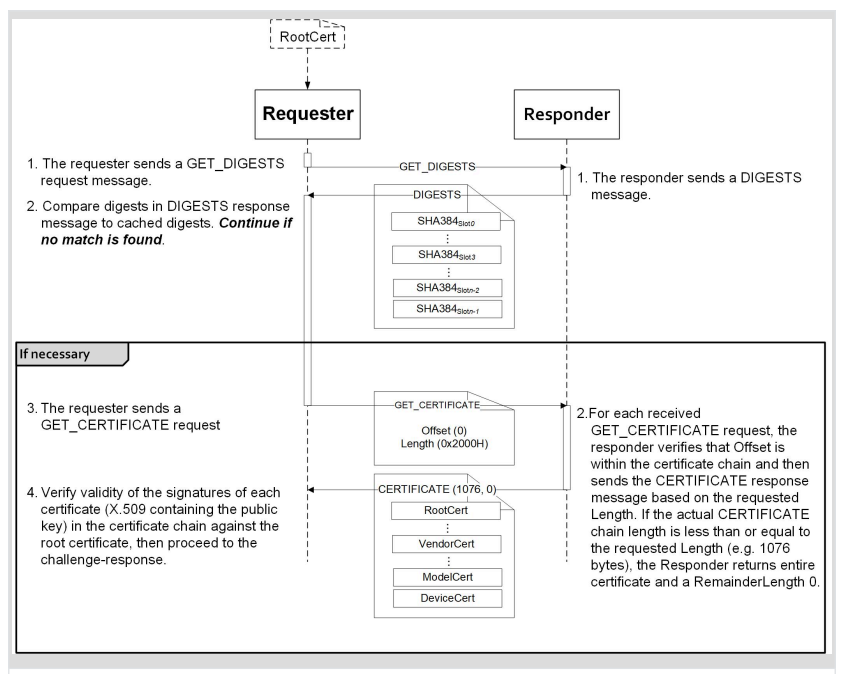


Figure 2 - *Responder authentication: (Source: SPDM Specification)*

The SPDM Certificate Chain format is also defined in SPDM specification.

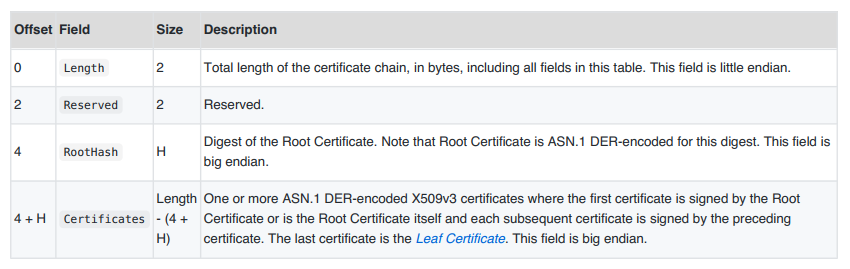


Figure 3 – *SPDM Certificate Chain: (Source: SPDM Specification)*

## Impact of Change

Add a new protocol in UEFI specification Chapter 32 – Secure Boot and Driver Signing.

No impact to the existing image authentication signature database.

## References

1. UEFI Specification 2.8 - www.uefi.org
2. DMTF “Security Protocol and Data Model Specification” - https://www.dmtf.org/standards/pmci
3. USB “USB Authentication Specification” - https://www.usb.org/documents
4. PCI-SIG “Component Measurement and Authentication (CMA) ECR” - https://pcisig.com/specifications/review-zone
5. TCG “TCG PC Client Platform firmware Profile Specification” - https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/, https://trustedcomputinggroup.org/wp-content/uploads/TCG\_PCClient\_PFP\_r1p05\_05\_3feb20.pdf
6. NIST SP800-193 “Platform Firmware Resiliency Guidelines” - https://csrc.nist.gov/publications/sp800

Detailed Description of the Change  
and Special Instructions

**32.4.1 Signature Database**

**……**

**#define EFI\_CERT\_X509\_SHA512\_GUID \  
 { 0x446dbf63, 0x2502, 0x4cda, \  
 { 0xbc, 0xfa, 0x24, 0x65, 0xd2, 0xb0, 0xfe, 0x9d } };  
Prototype  
 #pragma pack(1)  
 typedef struct \_EFI\_CERT\_X509\_SHA512 {  
 EFI\_SHA512\_HASH** *ToBeSignedHash***;  
 EFI\_TIME TimeOfRevocation;  
 } EFI\_CERT\_X509\_SHA512;  
 #pragma pack()  
Members** *ToBeSignedHash* The SHA512 hash of an X.509 certificate’s To-Be-Signed contents.  
 *TimeOfRevocation* The time that the certificate shall be considered to be revoked.  
This identifies a signature containing the SHA512 hash of an X.509 certificate’s To-Be-Signed contents,  
and a time of revocation. The *SignatureHeader* size shall always be 0. The *SignatureSize* shall  
always be 16 (size of the *SignatureOwner* component) + 80 bytes for an **EFI\_CERT\_X509\_SHA512** structure. If the *TimeOfRevocation* is non-zero, the certificate should be considered to be revoked  
from that time and onwards, and otherwise the certificate shall be considered to always be revoked.

**#define EFI\_CERT\_SPDM\_CERT\_CHAIN\_GUID \  
 { 0xa4ba9c9e, 0x1574, 0x4a2e, \  
 { 0x87, 0xc1, 0xc9, 0xbb, 0x87, 0x87, 0x66, 0xe6 } };**

This identifies a signature based on the SPDM defined certificate chain. This signature should only be used in the device authentication. If the signature is the SPDM defined certificate chain, the firmware should send SPDM GET\_DIGESTS message to verify the digest of the SPDM define certificate chain or send GET\_CERTIFICATE message to verify the entire SPDM define certificate chain. The *SignatureHeader* size shall always be 0. The *SignatureSize* may vary but shall always be 16 (size of the *SignatureOwner* component) + the size of the SPDM defined certificate chain itself.

**#define EFI\_CERT\_SPDM\_CERT\_CHAIN\_DIGEST\_GUID \  
 { 0x14749a83, 0x5c7, 0x4f79, \  
 { 0xab, 0xaf, 0x75, 0x92, 0x71, 0x73, 0xf7, 0x43 } };  
Prototype  
 #pragma pack(1)  
 typedef struct {  
 UINT32** *BaseHashAlgo***;  
 UINT32** *ExtHash***;  
 //UINT8** *Digest[HashSize]***;  
 } EFI\_CERT\_SPDM\_CERT\_CHAIN\_DIGEST;  
 #pragma pack()  
Members** *BaseHashAlgo* The SPDM BaseHashAlgo bit mask. No more than 1 bit should be set. If BaseHashAlgo is non-zero, then ExtHash shall be zero and ignored. If BaseHashAlgo is zero, then ExtHash should be referred.  
 *ExtHash* The SPDM ExtHash algorithm. This field should be referred only if the BaseHashAlgo is zero. *Digest* The digest of the SPDM Certificate Chain.

This identifies a signature containing a hash for the SPDM defined certificate chain. This signature should only be used in the device authentication. If the signature is the digest of SPDM defined certificate chain, the firmware should send SPDM GET\_CERTIFICATE message to verify the digest of the SPDM define certificate chain. The *SignatureHeader* size shall always be 0. The *SignatureSize* may vary but shall always be 16 (size of the *SignatureOwner* component) + 4 (size of the *BaseHashAlgo*) + 4 (size of the *ExtHash*) + the size of the digest based upon the hash type in *BaseHashAlgo* or *ExtHash* .

**#define EFI\_CERT\_EXTERNAL\_MANAGEMENT\_GUID \  
 { 0x452e8ced, 0xdfff, 0x4b8c, \  
 { 0xae, 0x01, 0x51, 0x18, 0x86, 0x2e, 0x68, 0x2c } };**This *SignatureType* describes a pseudo-signature which will not facilitate authentication. It is only  
meaningful within a signature list used for authenticating writes through **SetVariable()**, and is only  
effective if it is the only signature present in that signature list. It allows a signature list to be populated  
without providing any means for **SetVariable()** to succeed. This signature type is intended for use on  
a platform with an external out-of-band management agent (e.g. hypervisor or service processor). When  
a platform is configured such that only signatures of this *SignatureType* are available for  
authenticating writes to a variable, that variable may only be modified by the external management  
agent using a platform-specific interface.

32.6.2 UEFI Device Signature Variable GUID and Variable Name

**Summary**

Constants used for UEFI device signature database variable access..

**Prototype**

**#define EFI\_DEVICE\_SECURITY\_DATABASE\_GUID \**

**{0xb9c2b4f4, 0xbf5f, 0x462d, 0x8a, 0xdf, 0xc5, 0xc7, 0xa, 0xc3, 0x5d, 0xad}**

**#define EFI\_DEVICE\_SECURITY\_DATABASE L”devdb”  
#define EFI\_DEVICE\_SECURITY\_DATABASE1 L”devdbx”**

**#define EFI\_DEVICE\_SECURITY\_DATABASE2 L”devdbt”**

**Parameters**

• This GUID and name are used when calling the EFI Runtime Services **GetVariable()** and  
**SetVariable()**.

• The **EFI\_DEVICE\_SECURITY\_DATABASE\_GUID** and **EFI\_DEVICE\_SECURITY\_DATABASE** are  
used to retrieve and change the authorized device signature database.

• The **EFI\_DEVICE\_SECURITY\_DATABASE\_GUID** and **EFI\_DEVICE\_SECURITY\_DATABASE1** are  
used to retrieve and change the forbidden device signature database.

• The **EFI\_DEVICE\_SECURITY\_DATABASE\_GUID** and **EFI\_DEVICE\_SECURITY\_DATABASE2** are  
used to retrieve and change the authorized timestamp signature database.

Addition/Change to original request