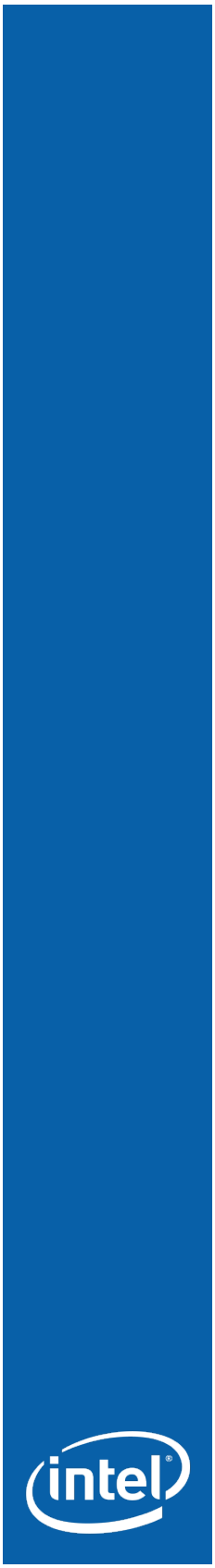


Intel® Software Guard Extensions (Intel® SGX) Data Center Attestation Primitives: ECDSA Quote Library API

Rev Production
August, 2018

Intel® Software Guard Extensions (Intel® SGX) Data Center Attestation Primitives: ECDSA Quote Library API

Table of Contents

1. Introduction

Attestation is a process of demonstrating that a software executable has been properly instantiated on a platform. The Intel® Software Guard Extensions (Intel® SGX) attestation allows a remote party to ensure that a particular software is securely running within an enclave on an Intel® SGX enabled platform.

This specification describes the API surface for a library that allows the software to generate an attestation evidence for an Intel® SGX enclave of an application. The Intel® Software Guard Extensions Data Center Attestation Primitives (Intel® SGX DCAP) version of the library generates the attestation evidence using an ECDSA Attestation Key to sign an identity Report of an Intel® SGX enclave of an application. The signed Report is called an attestation Quote. The ECDSA Attestation key is created and owned by the owner of the remote attestation infrastructure but is certified by an Intel® SGX rooted key whose certificate is distributed by Intel®. The Intel® SGX rooted certificate proves that the platform running the Intel® SGX enclave is valid and in good standing.

1.1. Terminology

SGX Quote	Data structure used to provide proof to an off-platform entity that an application enclave is running with Intel® SGX protections on a trusted Intel® SGX enabled platform.
Report (EREPORT)	Hardware report generated by the Intel® SGX HW that provides identity and measurement information of the enclave and the platform. It can be MAC'd with a key available to another enclave on the same platform.
Quoting Enclave (QE)	Signed enclave trusted by the attestation infrastructure owner to sign and issue Quotes or attestations about other enclaves.
Elliptic Curve Digital Signature Algorithm (ECDSA)	Signing cryptographic algorithm as described in FIPS 186-4.
Attestation Key (AK)	Key used by the Quoting Enclave (QE) to sign Quotes that describe the measurements and identity of an application enclave.
Provisioning Certification Enclave (PCE)	Intel® SGX architectural enclave that uses a Provisioning Certification Key (PCK) to sign QE REPORT structures for Provisioning or Quoting Enclaves. These signed REPORTS contain the ReportData indicating that attestation keys or provisioning protocol messages are created on genuine hardware.
Provisioning Certification Key (PCK)	Signing key available to the Provisioning Certification Enclave for signing certificate-like QE REPORT structures. The key is unique to a processor package or platform instance, the HW TCB, and the PCE version (PSVN).

Provisioning Certification Key Certificate (PCK Cert)	The x.509 Certificate chain signed and distributed by Intel for every Intel® SGX enabled platform. This certificate is used by Quote verifiers to verify that the QE generating quotes is valid and running on a trusted Intel® SGX platform at a particular PSVN. It matches the private key generated by the PCE.
Platform Provisioning ID (PPID)	Provisioning ID for a processor package or platform instance. PPID is not TCB dependent.
Security Version Number (SVN)	Version number that indicates when security relevant updates occurred. New versions can have increased functional versions without incrementing the SVN.
Platform Security Version Numbers (PSVN)	Set of SVNs for all components in the Intel® SGX attestation Trusted Computing Base (TCB) including the PCE SVN.
Enclave Page Cache (EPC)	Amount of memory on the platform allocated to enclave code and data storage.
Intel® SGX Provisioning TCB	Trusted Computing Base of Intel® SGX provisioning that includes the platform HW TCB and the PCE SVN.
PCEID	Identifies the version of the PCE used to generate the PPID and PCK signing key.
Intel® SGX DCAP	Intel® Software Guard Extensions Data Center Attestation Primitives
LE	Launch Enclave. Generates the launch token needed to load and initialize another enclave. The LE does not need a launch token to load but its signing key (MRSIGNER) must match the CPU configuration. See more in the Launch Control documents.
FLC	Flexible Launch Control. An Intel® SGX feature that allows arbitrary LE to generate Launch Tokens. The default Launch Control Policy adheres Intel® SGX client based whitelisting. FLC exposes a set of MSRs that allow a platform owner to change the default LE MRSINGER to a different MRSIGNER to enable LE to generate Launch Tokens. Not all platforms or all BIOSs support FLC.

Table 1-1: Terminology

2. Overview

Before an application enclave can be trusted by an off-platform entity, the application must prove that its enclave is running with Intel® SGX protections on an Intel® SGX platform in good standing. Once trusted, the off-platform entity or a relying party can provide secrets or trusted services. Each enclave can generate a hardware rooted identity REPORT MAC'd with a symmetric key that another enclave on the same platform can then verify. This is called an Intel® SGX Report based local attestation. This REPORT can then be verified and signed by an asymmetric private key owned by a special enclave called the Quoting Enclave (QE). The QE is running on the same platform as the application enclave. The resulting data structure is called a Quote and the asymmetric signing key is called an attestation key. Any relying parties that have access to a public portion of the attestation key can check the Quote signature, the application enclave identity and the TCB of the platform to establish trust in the application enclave.

Intel will develop a library for the Linux* OS based software that will generate quotes for application enclaves. This library will not depend on any specific platform software, such as the Intel® SGX PSW, but will rely on a set of APIs provided by the environment in which the library runs. This will allow the library to load the Intel signed enclaves required to generate the quotes. This allows the library to be designed and distributed to work in different environments. For example, it can be linked into the Intel® SGX PSW AESM or it can exist in another system service. It can also be linked as a part of an application allowing it to run in the application process. See section Quote Library Dependent APIs for the dependent system APIs.

2.1. Intel® SGX ECDSA Quoting Library

The ECDSA Quoting library contains an ECDSA-based Quoting Enclave (QE) that uses a FIPS 186-4 and RFC 6090 compliant algorithm to generate a 256 bit ECC signing key. The key is on the p256 curve. The QE is developed and signed by Intel.

The ECDSA attestation key generated by the QE needs to be certified by an Intel® SGX key rooted to the platform HW fuses. Intel develops and signs an enclave called the Provisioning Certification Enclave (PCE). The key generated by the PCE to certify (sign) attestation keys is rooted to the CPU HW fuses. This key is called the Provisioning Certification Key (PCK) private key. Intel will also generate and publish a public key that matches the signing key (PCK) generated by the PCE. The public key is published as an X.509 certificate format called the Provision Certification Key Certificate (PCK Cert). The PCE will provide an interface to retrieve the PCK Certificate identifier (EncPPID+TCB+PCEID) used by a verifier to find the matching PCK Cert. The PCE also provides a mechanism to sign another enclave (i.e. QE) REPORT using the PCK private key. For Intel® SGX DCAP, the QE will generate the ECDSA Attestation Key (AK) and include a hash of the AK in the QE.REPORT.ReportData. Only the PCE can produce the PCK private key. This PCE certification data will ultimately be embedded in the ECDSA Quote generated by the QE. The AK is then used to signed application enclave Reports to prove that the enclave is running with Intel® SGX protections at a given TCB. This is called the ECDSA Quote. The Attestation infrastructure owner can verify the ECDSA attestation key using the PCK Certificate. The Intel® SGX DCAP ECDSA Quoting Library described in this document will be shipped with the PCE library and will use the PCE APIs internally. The applications will use the APIs described in this document to generate Quotes for its enclave.

3. Intel® SGX ECDSA Quote Generation APIs

3.1. Intel® SGX DCAP Quote Library Wrapping API's

This chapter presents a set of C-like APIs that allow applications to request an ECDSA Quote. The Intel® SGX DCAP usage exposes a set of quote generation APIs that simplify the quoting interface to support a single ECDSA attestation key specific to that platform.

This library is delivered as a dynamically linked library (.so).

3.1.1. Set Enclave Load Policy

When the Quoting Library is linked to a process, it needs to know the proper enclave loading policy. The library may be linked with a long lived process, such as a service, where it can load the enclaves and leave them loaded (persistent). This better ensures that the enclave interfaces are available upon quote requests and not subject to Intel® SGX memory (EPC) limitations when loaded on demand. However, if the Quoting library is linked within an application process, there may be many applications with the Quoting library and a better utilization of EPC is to load and unload the enclaves on demand (ephemeral). The library will be shipped with a default policy of loading enclaves and leaving them loaded until the library is unloaded (SGX_QL_PERSISTENT).

If the policy is set to SGX_QL_EPHEMERAL, then the QE and PCE are loaded and unloaded on demand. If an enclave is already loaded when the policy is changed to SGX_QL_EPHEMERAL, the enclaves are unloaded before returning.

Syntax

```
quote3_error_t sgx_qe_set_enclave_load_policy(sgx_ql_request_policy_t policy);
```

Parameters

policy[In]

Sets the requested enclave loading policy to SGX_QL_PERSISTENT, SGX_QL_EPHEMERAL, or SGX_QL_DEFAULT.

Return Values

SGX_QL_SUCCESS:

Successfully set the enclave loading policy for the quoting library's enclaves.

SGX_QL_UNSUPPORTED_LOADING_POLICY:

Selected policy is not supported by the quoting library.

SGX_QL_ERROR_UNEXPECTED:

Unexpected error occurred.

3.1.2. Get QE Target Info

Description

This API allows the calling code to retrieve the target info of the QE. The loading of the QE and the PCE follows the selected loading policy. The application enclave uses the returned QE target info when generating its Report.

During this API execution, the Quoting Library generates and certifies the attestation key. The key and certification data is stored in process memory for the sgx_qe_get_quote_size() and sgx_qe_get_quote() APIs to use. Generating and certifying the keys at this point make the following APIs more efficient. If the following APIs return the SGX_QL_ATT_KEY_NOT_INITIALIZED error, this API needs to be called again to regenerate and recertify the key.

Syntax

```
quote3_error_t sgx_qe_get_target_info(sgx_target_info_t *p_target_info);
```

Parameters

p_target_info [Out]

Pointer to the buffer that contains the QE target information. This is used by an application enclave to generate a REPORT verifiable by the QE. Must not be NULL.

Return Values

SGX_QL_SUCCESS:
Retrieved the p_target_info.

SGX_QL_ERROR_INVALID_PARAMETER:
p_target_info must not be NULL.

SGX_QL_ERROR_UNEXPECTED:
Unexpected internal error occurred.

SGX_QL_ENCLAVE_LOAD_ERROR:
Unable to load the enclaves required to initialize the attestation key. Could be due to file I/O error or some other loading infrastructure errors.

SGX_QL_OUT_OF_MEMORY:
Heap memory allocation error occurred in a library or an enclave.

SGX_QL_ERROR_OUT_OF_EPC:
Not enough EPC memory to load one of the enclaves needed to complete this operation.

SGX_QL_ATTESTATION_KEY_CERTIFCATION_ERROR:
Failed to generate and certify the attestation key. Typically, this may happen if the TCB used to request PCE signing is higher than the platform TCB.

SGX_QL_ENCLAVE_LOST:
Enclave is lost after power transition or used in a child process created by linux:fork().

3.1.3. Get Quote Size

The application needs to call this API before generating a quote. The quote size varies depending on the type of certification data used to describe how the ECDSA AK is certified. Once the application calls this API, it uses the returned `p_quote_size` in bytes to allocate a buffer to hold the quote. A pointer to this allocated buffer is provided to the `sgx_qe_get_quote`() API.

If the key is not available, this API returns an error (SGX_QL_ATT_KEY_NOT_INITIALIZED). In this case, you must call `sgx_qe_get_target_info`() to re-generate and re-certify the attestation key.

The size returned in this API indicates the size of the quote buffer required in the `sgx_qe_get_quote`() API.

Syntax

```
quote3_error_t sgx_qe_get_quote_size(
                    uint32_t *p_quote_size)
```

Parameters

p_quote_size[Out]:

Pointer to the size of the buffer in bytes required to contain the full quote. This value is passed in to the `sgx_qe_get_quote`() API. You need to allocate a buffer large enough to contain the quote.

Return Values

SGX_QL_SUCCESS:

Successfully calculated the required quote size. The required size in bytes is returned in the memory pointed to by p_quote_size.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error occurred.

SGX_QL_ERROR_INVALID_PARAMETER:

Invalid parameter. `p_quote_size` must not be NULL.

SGX_QL_ATT_KEY_NOT_INITIALIZED:

Platform quoting infrastructure does not have the attestation key available to generate quotes. Call `sgx_qe_get_target_info`() again.

SGX_QL_ATT_KEY_CERT_DATA_INVALID:

Data returned by the platform provider library `sgx_ql_get_quote_config`() is invalid (see section Platform Quote Provider Library).

SGX_QL_ERROR_OUT_OF_EPC:

Not enough EPC memory to load one of the quote library enclaves needed to complete this operation.

SGX_QL_OUT_OF_MEMORY:

Heap memory allocation error occurred in a library or an enclave.

SGX_QL_ENCLAVE_LOAD_ERROR:

Unable to load one of the quote library enclaves required to initialize the attestation key. Could be due to file I/O error or some other loading infrastructure errors.

SGX_QL_ENCLAVE_LOST:
Enclave is lost after power transition or used in a child process created by linux:fork().

SGX_QL_ATT_KEY_CERT_DATA_INVALID:
Certification data retrieved from the platform provider library is invalid.

3.1.4. Get Quote

Description

Finally, the application calls this API to generate a quote. The function takes the application enclave REPORT as input and converts it into a quote once the QE verifies the REPORT. Once verified, it signs it with the ECDSA AK of the Intel® SGX DCAP QE. If the key is not available, this API returns an error (SGX_QL_ATT_KEY_NOT_INITIALIZED). In this case, call `sgx_qe_get_target_info()` to re-generate and re-certify the attestation key.

For Intel® SGX DCAP, the Quote.Header.UserData[0..15] (see Quote Format) contains the 128bit platform identifier (QE_ID) based on the QE Seal Key at TCB 0 (see QE ID Derivation). This allows the attestation infrastructure to link a quote generated on the platform with the platform PCK Cert.

To allow the application to remain agnostic to the type of the attestation key used generate the quote, the application should not try to parse the quote.

Syntax

```
quote3_error_t sgx_qe_get_quote(
                    const sgx_report_t *p_app_report,
                    uint32_t quote_size
                    uint8_t *p_quote);
```

Parameters

p_app_report [In]
Pointer to the application enclave REPORT that requires a quote. The report needs to be generated using the QE target info returned by the sgx_qe_get_target_info() API. Must not be NULL.

quote_size [In]
Size of the buffer that p_quote points to (in bytes).

p_quote [Out]
Pointer to the buffer that will contain the generated quote. Must not be NULL.

Return Values

SGX_QL_SUCCESS:
Successfully generated the quote.

SGX_QL_ERROR_UNEXPECTED:
Unexpected internal error occurred.

SGX_QL_ERROR_INVALID_PARAMETER:
Invalid parameter.

SGX_QL_ATT_KEY_NOT_INITIALIZED:
Platform quoting infrastructure does not have the attestation key available to generate quotes. Call init_quote() again.

SGX_QL_ATT_KEY_CERT_DATA_INVALID:
Data returned by the platform provider library sgx_ql_get_quote_config() is invalid.

SGX_QL_ERROR_OUT_OF_EPC:
Not enough EPC memory to load one of the Architecture Enclaves needed to complete this operation.

SGX_QL_OUT_OF_MEMORY:
Heap memory allocation error occurred in a library or an enclave.

SGX_QL_ENCLAVE_LOAD_ERROR:
Unable to load the enclaves required to initialize the attestation key. Could be due to file I/O error or some other loading infrastructure errors.

SGX_QL_ENCLAVE_LOST:
Enclave was lost after power transition or used in a child process created by linux:fork().

SGX_QL_INVALID_REPORT:
Report MAC check failed on a application report.

3.1.5. Cleanup Enclaves by Policy

Description

This method is primarily a hint for the Quote library that it can release the QE and the PCE it cached for efficiency. In the mainline case, sgx_qe_get_targetinfo(), sgx_qe_get_quote_size(), and sgx_qe_get_quote() are called in succession. If the Quote library keeps the enclaves loaded between sgx_qe_get_targetinfo() and sgx_qe_get_quote_size(), they may not be unloaded if the process using the quote library fails prior to sgx_qe_get_quote(). sgx_cleanup_qe_by_policy() informs the Quote Library that it should clean up the QE and the PCE since it cannot depend on the sgx_qe_get_quote() to unload them. If SGX_QE_PERSISTENT is the default policy, it can choose to no-op.

Syntax

```
quote3_error_t sgx_qe_cleanup_by_policy();
```

Parameters

None

Return Values

SGX_QL_SUCCESS:
Successfully completed.

3.2. Enclave Loading

The Quote library loads and unloads the Intel signed and formatted QE and PCE enclaves as needed and as specified by the enclave loading policy. The Quote Library loads the QE and the PCE using a library called the modified URTS (Untrusted Runtime Service) exposing APIs that are compatible with the enclave loading APIs exposed by the Intel® SGX SDK. The modified URTS library is shipped with the Quote Library or has it statically linked. The modified URTS library uses the Intel® Enclave Common Abstraction Layer Library (see 'Enclave Common Loader API Reference' document for API descriptions).

3.2.1. Enclave Launch Policy Implications

To use a Quoting Library that supports ECDSA attestation, the platform that runs the Quote Library must support Flexible Launch Control (FLC). FLC allows the platform owner to choose which Launch Enclave (LE) can generate launch tokens and enforce the enclave launch control policy supported by that LE. FLC is not available on all platforms and FLC must be supported by the BIOS.

Some environments may whitelist the PCE and the QE based on the enclave MRSIGNER (hash of the enclave signing key) and the attribute.ProvBit. For example, the Intel® SGX DCAP driver only allows the Intel signed PCE and QE enclaves to launch with the attribute.ProvBit. Any other enclaves must launch with this bit set to zero. Also, the Intel signed PCE does not provide certification information or Report signing to enclaves with the attribute.ProvBit to 0.

3.3. Quote Library Dependent APIs

The Quoting library looks for these APIs when needed and expects them to be available from a library dynamically linked with the Quoting Library.

3.3.1. Platform Quote Provider Library

The Platform Provider Library provides a set of APIs that allow the Quote Library to get platform specific services. They are not required for the Quote Library to function but they may be required to properly generate quotes in a given platform environment.

The Quote Library looks for a library named ***libdcap_quoteprov.so*** using dlopen during runtime. The Quote Library does not require the provider library to generate Quotes but the generated quotes may not be verifiable.

3.3.1.1.Get PCK Certification Information

Description

For ECDSA quote generation, the Quote Library by default generates an attestation key certified by the PCE using the raw Intel® SGX TCB of the platform. This may not work for all attestation environments. The TCB used to generate the PCE signature over the ECDSA AK needs a matching Intel generated x.509 PCK Certificate for that TCB. This is problematic for E3 and client platforms that do not have Intel Generated PCK Certs for all TCB levels. It also causes problems when the attestation infrastructure caches PCK Certs and may not have certs for all platforms due to restrictions on contacting Intel hosted services during runtime. In these cases, the Quote Library needs to get the TCB from the platform software to generate the proper PCE signature. The Quote Library first requests the TCB information from the Provider Library if it is available, and submits that value for PCE signing. In addition, the

provider library responds with the associated Certification Data to append to the ECDSA Quote. See the definitions for `sgx_ql_pck_cert_id_t and sgx_ql_config_t.`

If the provider library cannot be found, the sgx_ql_get_quote_config() symbol is not found within the provider library or it returns an error, the Quote Library uses the raw-TCB of the platform to certify the key and use the certification type PPID_RSA3072_ENCRYPTED as the Quote's Certification Data Type to identify platform. If the API is found, the API returns 2 pieces of information:

1. The TCB to use when requesting the PCE to certify the attestation key. This matches the TCB of the PCK Certificate that the quote verifier uses to certify the attestation key.
2. The certification data for the associated PCK Cert to be added to the quote when the quote is generated.

The data returned by this API is used to determine the ultimate size of the quote. The currentrelease of the Quote Library only supports the sgx_ql_config_t.version of SGX_QL_CONFIG_VERSION_1 (0x0001). For this version of the sgx_ql_config_t data structure returned by the sgx_ql_get_quote_config(), the sgx_ql_config_t.p_cert_data is expected to point to the PCK Cert chain as defined by the certification type PCK_CERT_CHAIN (5) for the Quote Certification Data Type. The Quote Library uses this data to replace the default certification data type generated by the QE. Because of this, the 'Quote Signature Data Len' and the 'QE Certification Data' fields in the Quote are not signed by the AK.

Future versions of the sgx_ql_config_t data structure may support more Certification Data Types.

The functionality of the API is not limited to just ECDSA attestation. The data inputted and outputted from this function only pertains to the TCB to use for generating the PCE signature and the data needed to locate the associated PCK Certificate. It can be considered independent of the type of AK used for quote signing and may apply to other attestation environments.

Syntax

```
quote3_error_t sgx_ql_get_quote_config(
                        const sgx_ql_pck_cert_id_t *p_pck_cert_id,
                        sgx_ql_config_t **pp_cert_config);
```

Parameters

p_pck_cert_id [In]

The Quoting Library passes a pointer to the PCK Certificate ID structure. The Provider Library will use this information to find the proper TCB and Quote Certification Data. If the Quoting Library does not support reporting the optional field, encrypted PPID, when this call is made, then p_encrypted_ppid is NULL, encrypted_ppid_size is 0 and crypto_suite is 0.

pp_cert_config [Out]

Pointer to a pointer to the PCK certification data needed for quote generation. The provider library allocates this buffer and it is expected that the Quote Library frees it with the provider library sgx_ql_free_quote_config() API. If the platform does not yet have the configuration data available, the SGX_QL_NO_PLATFORM_CERT_DATA error is returned and the PCK Signature is generated using the raw TCB of the platform. If the library does not support the data or there is a problem with the format, the Quoting library returns SGX_QL_ATT_KEY_CERT_DATA_INVALID.

Return Values

SGX_QL_SUCCESS:

Platform has the certification data available and returned it in the p_quote_config buffer.

SGX_QL_NO_PLATFORM_CERT_DATA:
Platform does not have the certification data available.

SGX_QL_ERROR_INVALID_PARAMETER:
Provider library rejected the input.

SGX_QL_PLATFORM_LIB_UNAVAILABLE:
Quote Library failed to locate the provider library. The platform does not have the certification data available.

3.3.1.2. Free PCK Certification Information

Description

This API frees the PCK Cert configuration data buffer allocated by the provider library `sgx_ql_get_quote_config()` API.

Syntax

```
quote3_error_t sgx_ql_free_quote_config(
                          sgx_ql_config_t *p_cert_config);
```

Parameters

p_cert_config [Out]
Pointer to the PCK certification that the `sgx_ql_get_quote_config()` API allocated.

Return Values

SGX_QL_SUCCESS:
Pointer is successfully freed or the input pointer is NULL.

3.3.1.3. Store Persistent Data

Not required and does not need to be implemented by the quote provider library. If implemented, it is expected that the provider library stores the data to the file specified in the input.

Syntax

```
quote3_error_t sgx_ql_write_persistent_data(
                              const uint8_t *p_buf,
                              uint32_t buf_size,
                              const char *p_label);
```

Parameters

p_buf [In]
Pointer to the data to be written. Must not be NULL.

buf_size [In]
Size of the data in bytes that p_buf points to.

p_label [In]
Pointer to the string label of the data to be stored. Must not be NULL and must be a valid string.

Return Values

SGX_QL_SUCCESS:
Data was written successfully.

SGX_QE_PLATFORM_LIB_UNAVAILABLE:
Provider library was not found.

SGX_QL_ERROR_UNEXPECTED:
Unexpected internal error occurred.

SGX_QL_ERROR_INVALID_PARAMETER:
One of the pointers is NULL

SGX_QL_FILE_ACCESS_ERROR:
Not able to find the 'label' or there was a problem writing the data.

3.3.1.4. Retrieve Persistent Data

Description

Not required and does not need to be implemented by the quote provider library. If implemented, it is expected that the provider library loads the data from the file specified in the input.

Syntax

```
quote3_error_t sgx_ql_read_persistent_data(
                          const uint8_t *p_buf,
                          uint32_t *p_buf_size,
                          const char *p_label);
```

Parameters

p_buf [In/Out]
Pointer to the buffer to store the data.

p_buf_size [In/Out]
Pointer to the size in the buffer. If the p_buff is NULL, the API returns the required size. Must not be NULL.

p_label[In]
Pointer to the string label of the data to be stored. Must not be NULL and must be a valid string.

Return Values

SGX_QL_SUCCESS:
Data was read successfully.

SGX_QE_PLATFORM_LIB_UNAVAILABLE:
Provider library was not found.

SGX_QL_ERROR_UNEXPECTED:

Unexpected internal error occurred.

SGX_QL_ERROR_INVALID_PARAMETER:
: If all pointer are not NULL, the size of the inputted buffer is too small. Otherwise, one of the mandatory input pointers is NULL

SGX_QL_FILE_ACCESS_ERROR:
: Not able to find the 'label' or there was a problem retrieving the data.

3.3.2. Intel® SGX Enclave Loading Library

Provided by the modified 'urts' library that uses the Intel® SGX Enclave Loading Abstraction Layer Library (see Enclave Common Loader API Reference document). This 'modified-sgx-urts' library exposes the same enclave loading API as provided by the legacy Intel® SGX SDK.

3.4. Deployment Tool for PCK Certificate Chain Retrieval for Intel® SGX DCAP

Some attestation environments do not provision each platform with its PCK Cert or the PCK Cert TCB at platform bring-up and then store it persistently for each VM when the VM starts up. Also, some environments do not permit access to the external Intel hosted PCK Certificate Service during runtime. These environments retrieve the PCK Cert Chain from Intel during platform bring-up (Deployment) and store the PCK Cert chain in a PCK Cert Proxy/PCK Cert Inventory hosted within their attestation infrastructure. Then, the PCK Cert chain is provided to the VM when the VM starts up (run-time) based on a request from the VM to the Proxy Service. Or, it provides the PCK Cert TCB at VM start up and then retrieve the PCK Cert when the Quote is verified. There is a need to identify the PCK Cert retrieved during deployment to download the TCB or PCK Cert to the VM at runtime.

The PPID in the PCK Cert could be used for this purpose but the Intel® SGX protects the PPID privacy by encrypting the PPID with a PCK Server owned public key. The RSA-OAEP algorithm used to encrypt the PPID changes the encrypted PPID value between successive requests to the PCE. The Enc(PPID) generated during deployment does not match the Enc(PPID) used during runtime.

The *PCKRetrievalTool* will be released along with the Quote Library release. The production version of the Quote Library encrypts the PPID with a 3072bit RSA-OAEP key owned by the Intel hosted PCK Certification Service. The PCKRetrievalTool output can be used to request a PCK Cert from the service.

The PCK Retrieval Tool will output a Base16 (Hex) encoded text file in CSV format:

```
EncryptedPPID(384 BE byte array),PCE_ID(LE 16 bit integer),CPUSVN(16 byte BE
byte array),PCE ISVSVN (LE 16 bit integer),QE_ID (16 byte BE byte array)
```

To request PCK Certs Online, follow the onboarding and RESTful API described in the PCK Service documentation.

Note: The EncPPID changes each time the tool is called while the QE_ID does not. The user of the tool should keep a link between the QE_ID and EncPPID to properly link the Platform to its PCK Cert Chain.

Note: You may get more than one PCK Cert Chain for each platform depending on the number of active TCB levels for that platform and the PCK Certificate Service API used.

3.5. Key Derivations

3.5.1. QE_ID Derivation

The QE_ID is a platform ID that is not associated with a particular SVN but is dependent on the Quoting Enclave (QE) MRSIGNER and its Seal Key. The QE_ID is designed to be dependent on the seal key, which depends on the platform OWNER_EPOCH value. The OWNER_EPOCH value is set by the platform owner in the BIOS configuration. If the BIOS non-volatile memory (FLASH) is wiped, then the QE_ID changes even if generated by the same QE. This prevents the QE_ID from being a true HW ID. A true HW ID cannot be modified by the platform owner.

1) QE_ID-Seed = EGETKEY(KEYNAME=SEAL_KEY,
   KEY_POLICY=MRSIGNER,
   KEY_ID = 0,
   CPUSVN=0,
   ISVSVN = 0)
2) QE_ID = AES128-CMAC(QE_ID-Seed, 16 bytes below)

Byte Position	Value
0	0x00
1-9	"QE_ID_DER" (ascii encoded)
10-13	0x00000000
14-15	0x0080 (Big Endian)

3.5.2. ECDSA Attestation Key Derivation

3.5.2.1. ECDSA Attestation Key Derivation using QE Seal Key (Intel® SGX DCAP Solution)

The ECDSA Attestation key is derived from the QE seal key at the current TCB level. This allows the QE to regenerate the same attestation key without requiring persistent storage. However, the key changes when any of the QE TCB components change (CPUSVN, PCE_ISVSVN or the QE_ ISVSVN). This extends the lifetime of the QE attestation key beyond the time the QE library exists in process memory.

The QE attestation key is used by the QE to sign reports from application enclaves. It is a 256 bit ECC signing key using NIST curve secp256r1.

The QE attestation key derivation is rooted in the HW key. EGETKEY does a series of AES-base derivations resulting in a Provisioning Key unique to the HW, the current TCB (CPUSVN), and the QE identify (MRSIGNER, ISVPRODID, ISVSVN). The AK private key is derived from 320 random bits (providing 128 bits of entropy) derived from the Seal Key using the following flow:

1) Sealing Key = EGETKEY(KEYNAME = SEAL_KEY,
   KEY_POLICY =MRSIGNER,

KEY_ID = 0,
Current CPUSVN,
Current ISVSVN)

2) Block 1 = AES-CMAC(Sealing Key, QE string with Counter = 0x01)
3) Block 2 = AES-CMAC(Sealing Key, QE string with Counter = 0x02)
4) Block 3 = AES-CMAC(Sealing Key, QE string with Counter = 0x03)
5) QE ATT Seed = most significant 320 bits of (Block 1 || Block 2 || Block 3).
6) QE ATT key pair is generated using NIST SP 186-4 section B 4.1 “Key Pair Generation Using Extra Random Bits.” AE ATT Seed are used for the random bits.

Byte Position	Value
0	Counter (See Description)
1-10	“QE_KEY_DER” (ascii encoded)
11-13	0x000000
14-15	0x0140 (Big Endian)

A. Data Structures

A.1. Quote Library Data Structures

```
typedef enum {
    SGX_QL_PERSISTENT, ///< AEs areinitialized on first use and reused until process
                           ends.
    SGX_QL_EPHEMERAL,  ///< AEs are initialized and terminated on every quote.
                       ///< If a previous QE exists, it is stopped & restarted before
                           quoting.
    SGX_QL_DEFAULT = SGX_QL_PERSISTENT
} sgx_ql_request_policy_t;


/** Identifies the type of certification data used in the Quote */
typedef struct _sgx_ql_certification_data_t {
    uint16_t cert_key_type;  ///< The type of certification key used to sign the QE3
                                Report and Att key hash (ECDSA_ID+Authentication
                                Data).
    uint32_t size;           ///< Size of the data structure for the cert_key_type
                                information.
    uint8_t certification_data[];  ///< Certification data associated with the
cert_key_type
} sgx_ql_certification_data_t;



/** Enumerates the different certification data types used to describe the signer of
the attestation key */
typedef enum {
    PPID_CLEARTEXT = 1,         ///< Clear PPID + CPU_SVN, PvE_SVN, PCE_SVN, PCE_ID
    PPID_RSA2048_ENCRYPTED = 2, ///< RSA-2048-OAEP Encrypted PPID + CPU_SVN, PvE_SVN,
                                   PCE_SVN, PCE_ID
    PPID_RSA3072_ENCRYPTED = 3, ///< RSA-3072-OAEP Encrypted PPID + CPU_SVN, PvE_SVN,
                                   PCE_SVN, PCE_ID
    PCK_CLEARTEXT = 4,          ///< Clear PCK Leaf Cert
    PCK_CERT_CHAIN = 5,         ///< Full PCK Cert chain
                                   (trustedRootCaCert||intermediateCa||pckCert)
    ECDSA_SIG_AUX_DATA = 6,     ///< Indicates the contents of the
                                   CERTIFICATION_INFO_DATA contains the
                                   ECDSA_SIG_AUX_DATA of another Quote.
} sgx_ql_cert_key_type_t;
```

A.2. Core Generic Quote Wrapper Structures

```
/** Enumerates the different attestation key algorithms */
typedef enum {
    SGX_ALG_EPID = 0,       ///< EPID 2.0 - Anonymous
    SGX_ALG_RESERVED_1 = 1, ///< Reserved
    SGX_ALG_ECDSA_P256 = 2, ///< ECDSA-256-with-P-256 curve, Non - Anonymous
    SGX_ALG_ECDSA_P384 = 3, ///< ECDSA-384-with-P-384 curve, Non-Anonymous
    SGX_ALG_MAX = 4
} sgx_ql_attestation_algorithm_id_t;

/** Describes the header that contains the list of attestation keys supported by a
given verifier */
typedef struct _sgx_ql_att_key_id_list_header_t {
    uint16_t       id;          ///< Structure ID
    uint16_t       version;     ///< Structure version
    uint32_t       num_att_ids; ///< Number of 'Attestation Key Identifier' Elements
}sgx_ql_key_id_list_header_t;
```

```
/** Describes a single attestation key.  Contains both QE identity and the attestation
algorithm ID. */
typedef struct _sgx_ql_att_key_id_t {
    uint16_t    id;                                 ///< Structure ID
    uint16_t    version;                            ///< Structure version
    uint16_t    mrsigner_length;                    ///< Number of valid bytes in
                                                        MRSIGNER.
    uint8_t     mrsigner[48];                       ///< SHA256 or SHA384 hash of the
                                                        Public key that signed the QE.
                                                    ///< The lower bytes contain
                                                        MRSIGNER.  Bytes beyond
                                                        mrsigner_length '0'
    uint32_t    prod_id;                            ///< Legacy Product ID of the QE
    uint8_t     extended_prod_id[16];               ///< Extended Product ID or the QE.
                                                        All 0s for legacy format
                                                        enclaves.
    uint8_t     config_id[64];                      ///< Config ID of the QE.
    uint8_t     family_id[16];                      ///< Family ID of the QE.
    sgx_ql_attestation_algorithm_id_t algorithm_id; ///< Identity of the attestation
                                                         key algorithm.
}sgx_ql_att_key_id_t;

/** The full data structure passed to the platform by the verifier. It will list all
of the attestation algorithms and QE's supported by the verifier */
typedef struct _ sgx_ql_att_key_id_list_t {
    sgx_ql_att_key_id_list_header_t   header;      ///< Header for the attestation key
                                                        ID list provided by the quote
                                                        verifier.
    sgx_ql_att_key_id_t               id_list[];   ///< Place holder for the
                                                        attestation ID list.
}sgx_ql_att_key_id_list_t;


typedef struct _sgx_ql_qe_report_info_t {
    sgx_quote_nonce_t   nonce;                      ///<
    sgx_target_info_t   app_enclave_target_info;
    sgx_report_t        qe_report;
}sgx_ql_qe_report_info_t;
```

A.3. Intel® SGX DCAPQuote Wrapper Structures

```
/** Used to describe the PCK Cert for a platform */
typedef struct _sgx_ql_pck_cert_id_t
{
    uint8_t *p_qe3_id;                  ///< The QE_ID used to identify the platform
                                            for PCK Cert Retrieval
    uint32_t qe_id_size;
    sgx_cpu_svn_t *platform_cpu_svn;    ///< The Size of the QE_ID (currently 16
                                            bytes)
    sgx_cpu_svn_t *platform_pce_isv_svn; ///< Pointer to the platform raw CPUSVN
    uint8_t *p_encrypted_ppid;          ///< Pointer to the encrypted PPID (Optional)
    uint32_t encrypted_ppid_size;       ///< Size of encrypted PPID.
    uint8_t crypto_suite;               ///< Crypto algorithm used to encrypt the
                                            PPID (currently only
                                            PCE_ALG_RSA_OAEP_3072 = 1 supported)
    uint16_t pce_id;                    ///< Identifies the PCE-Version used to
                                            generate the encrypted PPID.
}sgx_ql_pck_cert_id_t;

/** Contains valid versions of the sgx_ql_config_t data structure. */
typedef enum _sgx_ql_config_version_t
{
    SGX_QL_CONFIG_VERSION_1 = 1,
}sgx_ql_config_version_t;
```

```
/** Contains the certification data used by the quoting library to certify the
attestation key and the certification data required to generate the final quote. */
typedef struct _sgx_ql_config_t
{
    sgx_ql_config_version_t version;
    sgx_cpu_svn_t cert_cpu_svn;        ///< The CPUSVN used to generate the PCK
                                            Signature that certifies the attestation
                                            key.
    sgx_isv_svn_t cert_pce_isv_svn;    ///< The PCE ISVSVN used to generate the PCK
                                           Signature that certifies the attestation
                                           key.
    uint32_t p_cert_data_size;         ///< The size of the buffer that
                                           p_cert_data points to
    uint8_t *p_cert_data;              ///< The certification data used for the quote.
                                               ///todo: It is the assumed to be the
                                                  PCK Cert Chain.  May want to change
                                                  this to specify the certification
                                                  type too.
}sgx_ql_config_t;
```

A.4. Quote Format

The new quote structure to support ECDSA will have a version number of 3. The existing EPID Quote structure with a version number of 2 will still exist. The version 3 quote does not specifically support EPID but was designed so that the header is compatible based on size and the first 5 fields of the header.

Endianess: *Little Endian (applies to all integer fields).*

Name	Size (bytes)	Type	Description
Quote Header	48	Quote Header	Header of *Quote* data structure. This field is ***transparent*** (the user knows its internal structure). Rest of the *Quote* data structure can be treated as ***opaque*** (hidden from the user).
ISV Enclave Report	384	Enclave Report Body	Report of the attested *ISV Enclave*. The CPUSVN and ISVSVN is the TCB when the quote is generated. The REPORT.ReportData is defined by the ISV but should provide quote replay protection if required.
Quote Signature Data Len	4	uint32_t	Size of the Quote Signature Data structure
Quote Signature Data	Variable	Signature Dependent	Variable-length data containing the signature and supporting data. E.g. ECDSA 256-bit Quote Signature Data Structure

Table 2: High-Level Quote Structure

Name	Size (bytes)	Type	Description
Version	2	Integer	Version of the *Quote* data structure. • *Value*: 3
Attestation Key Type	2	Integer	Type of the *Attestation Key* used by the *Quoting Enclave*. • *Supported values*: - 2 (ECDSA-256-with-P-256 curve) - 3 (ECDSA-384-with-P-384 curve) (*Note: currently not supported*) (Note: 0 and 1 are reserved, EPID is moved to version 3 quotes.)
Reserved	4	Byte Array	Reserved field. • *Value*: 0
QE SVN	2	Integer	Security Version of the Quoting Enclave currently loaded on the platform.
PCE SVN	2	Integer	Security Version of the Provisioning Certification Enclave currently loaded on the platform.
QE Vendor ID	16	UUID	Unique identifier of the QE Vendor. • *Value*: 939A7233F79C4CA9940A0DB3957F0607 (Intel® SGX QE Vendor) *Note: Each vendor that decides to provide a customized Quote data structure should have unique ID.*
User Data	20	Byte Array	Custom user-defined data. For the Intel® SGX DCAP library, the first 16 bytes contain a QE identifier that is used to link a PCK Cert to an Enc(PPID). This identifier is consistent for every quote generated with this QE on this platform.

Table 3: Quote Header

Name	Size (bytes)	Type	Description
ISV Enclave Report Signature	64	ECDSA P-256 Signature	ECDSA signature over the *Header* and the *Enclave Report* calculated using *ECDSA Attestation Key*.
ECDSA Attestation Key	64	ECDSA P-256 Public Key	Public part of the *ECDSA Attestation Key* generated by the *Quoting Enclave*.

QE Report	384	Enclave Report Body	Report of the *Quoting Enclave* that generated the *ECDSA Attestation Key*. • *Report Data*: SHA256(*ECDSA Attestation Key* || *QE Authentication Data*) || 32-0x00's Note: The QE Report is a report when the QE Report is certified. The CPUSVN and ISVSVN in this report may be older than the currently loaded QE.
QE Report Signature	64	ECDSA P-256 Signature	ECDSA signature over the *QE Report* calculated using the *Provisioning Certification Key*.
QE Authentication Data	Variable	QE Authentication Data	Variable-length data chosen by the *Quoting Enclave* and signed by the *Provisioning Certification Key* (as a part of the *Report Data* in the *QE Report*). It can be used by the *QE* to add additional context to the *ECDSA Attestation Key* utilized by the *QE*. For example, this may indicate the customer, geography, network, or anything pertinent to the identity of the Quoting Enclave. Size should be set to 0 if there is no additional data.
QE Certification Data	Variable	QE Certification Data	Data required to verify the QE Report Signature.

Table 4: ECDSA 256-bit Quote Signature Data Structure

Name	Size (bytes)	Type	Description
CPU SVN	16	Byte Array	Security Version of a CPU (raw value).
MISCSELECT	4	LE Integer	SSA Frame extended feature set.
Reserved	28	Byte Array	Reserved field.
Attributes	16	Byte Array	Set of flags describing attributes of the enclave.
MRENCLAVE	32	Byte Array	Enclave measurement.
Reserved	32	Byte Array	Reserved field.
MRSIGNER	32	Byte Array	Hash of the enclave signing key.
Reserved	96	Byte Array	Reserved field.

ISV ProdID	2	LE Integer	Enclave Product ID.
ISV SVN	2	LE Integer	Security Version of the enclave.
Reserved	60	Byte Array	Reserved field.
Report Data	64	Byte Array	Additional report data.

Table 5: Enclave Report Body

Name	Size (bytes)	Type	Description
Signature	64	Byte Array	ECDSA signature, the r component followed by the s component, 2 x 32 bytes.

Table 6: ECDSA P-256 Signature

Name	Size (bytes)	Type	Description
Public Key	64	Byte Array	EC KT-I Public Key, the x-coordinate followed by the y-coordinate (on the RFC 6090 P-256 curve), 2 x 32 bytes.

Table 7: ECDSA P-256 Public Key

Name	Size (bytes)	Type	Description
Size	2	Integer	Size of the 'Data' array. 0 is a valid value.
Data	Variable	Byte Array	Data that to be additionally 'signed' by the certification key.

Table 8: QE Authentication Data

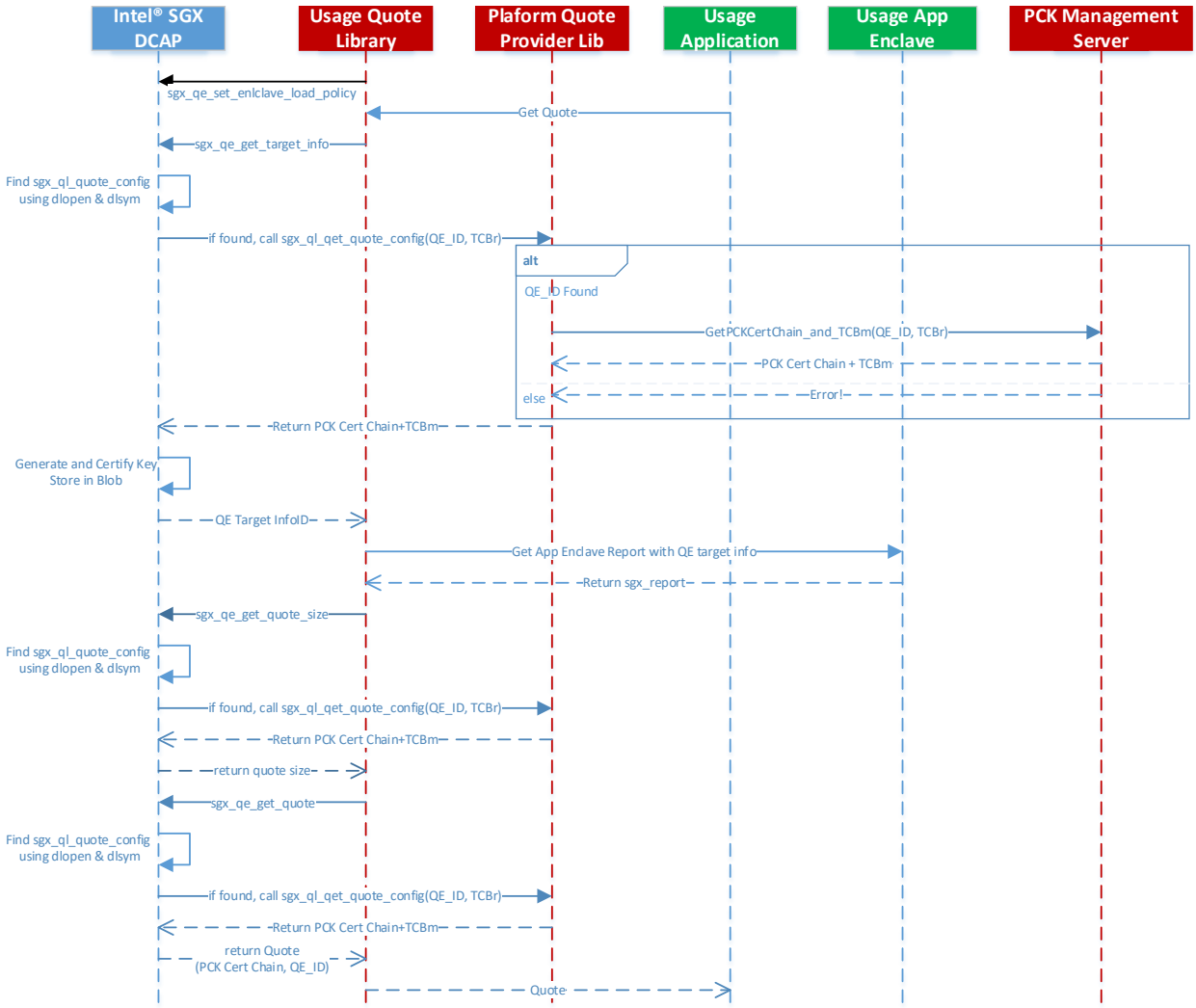
Name	Size (bytes)	Type	Description
Certification Data Type	2	Integer	Determines type of data required to verify the QE Report Signature in the Quote Signature Data structure. • *Supported values*: - 1 (PCK identifier: PPID in plain text, CPUSVN and PCESVN) - 2 (PCK identifier: PPID encrypted using RSA-2048-OAEP, CPUSVN and PCESVN) - 3 (*PCK* identifier: PPID encrypted using RSA-3072-OAEP, CPUSVN and PCESVN) - 4 (PCK Leaf Certificate in plain text, currently not supported) - 5 Concatenated PCK Cert Chain - 7 (PLATFORM_MANIFEST, currently not

			supported)
Size	4	Integer	Size of Certification Data field.
Certification Data	Variable	Byte Array	Data required to verify the QE Report Signature depending on the value of the *Certification Data Type*: - 1: Byte array that contains concatenation of PPID, CPUSVN, PCESVN (LE), PCEID (LE). - 2: Byte array that contains concatenation of PPID encrypted using RSA-2048-OAEP, CPUSVN, PCESVN (LE), PCEID (LE). - 3: Byte array that contains concatenation of PPID encrypted using RSA-3072-OAEP, CPUSVN, PCESVN (LE), PCEID (LE). - 4: PCK Leaf Certificate - 5: Concatenated PCK Cert Chain (PEM formatted). PCK Leaf Cert||Intermediate CA Cert|| Root CA Cert - 6: Intel® SGX Quote (currently not supported) - 7: PLATFORM_MANIFEST (currently not supported)

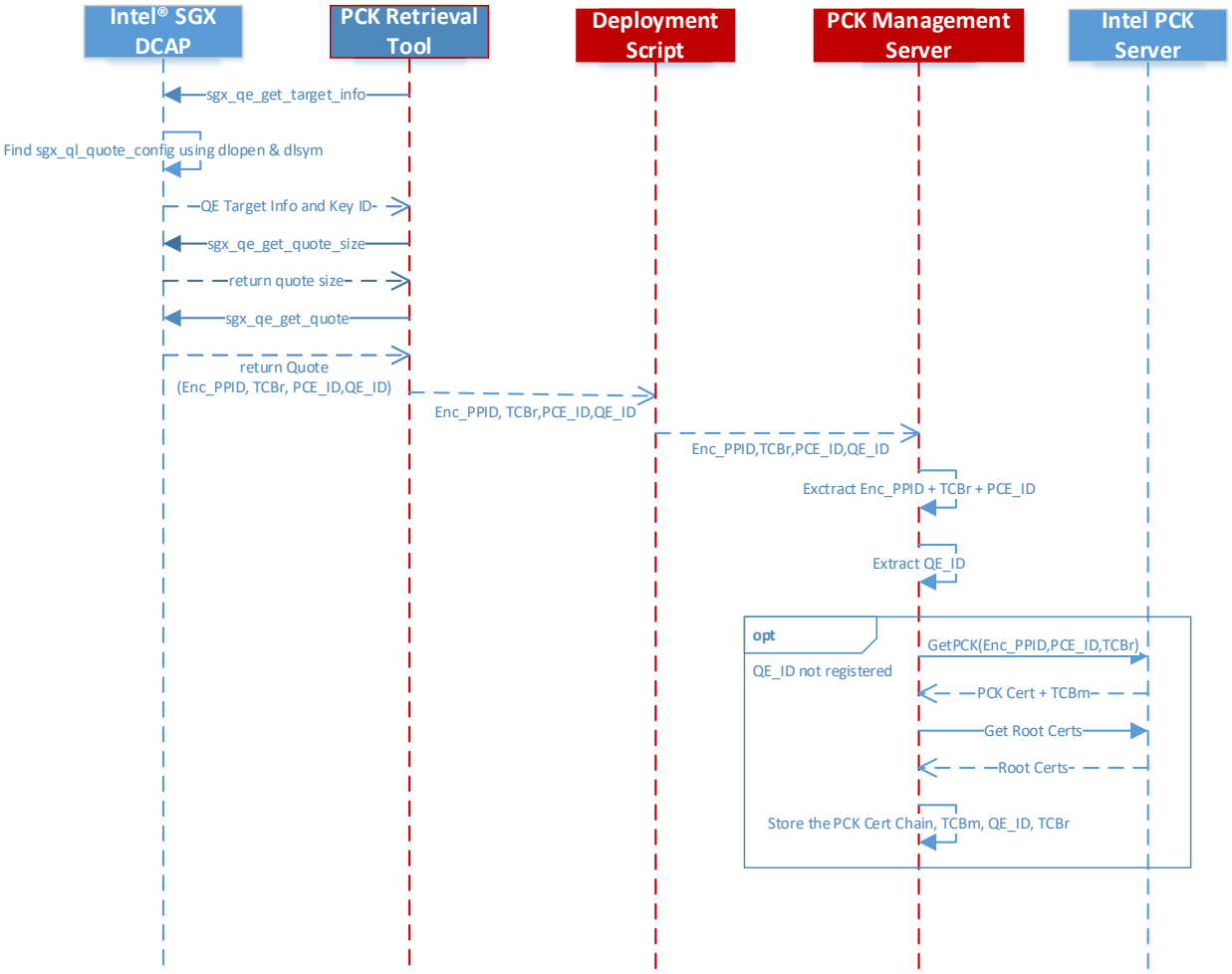
Table 9: QE Certification Data

B. Sample Sequence Diagrams

B.1. Sample Quote Generation Sequence Diagram for the Intel® SGX DCAP APIs



B.2. Deployment Phase PCK Retrieval Sequence Diagram


Intel® SGX DCAP
PCK Retrieval Tool
Deployment Script
PCK Management Server
Intel PCK Server
sgx_qe_get_target_info
Find sgx_ql_quote_config using dlopen & dlsym
QE Target Info and Key ID-
sgx_qe_get_quote_size
return quote size
sgx_qe_get_quote
return Quote
(Enc_PPID, TCBr, PCE_ID,QE_ID)
Enc_PPID, TCBr,PCE_ID,QE_ID
Enc_PPID,TCBr,PCE_ID,QE_ID
Exctract Enc_PPID + TCBr + PCE_ID
Extract QE_ID
opt
QE_ID not registered
GetPCK(Enc_PPID,PCE_ID,TCBr)
PCK Cert + TCBm
Get Root Certs
Root Certs
Store the PCK Cert Chain, TCBm, QE_ID, TCBr