Intel® Software Guard Extensions:
EPID Provisioning and Attestation Services

Simon Johnson, Vinnie Scarlata, Carlos Rozas, Ernie Brickell, Frank Mckeen
Intel Corporation
{ simon.p.johnson, vincent.r.scarlata, carlos.v.rozas, ernie.brickell, frank.mckeen }@intel.com

ABSTRACT

Intel® Software Guard Extensions (SGX) has an attestation capability that can be used to remotely provision secrets to an enclave. Use of Intel® SGX attestation and sealing has been described in [1]. This paper describes how the SGX attestation key are remotely provisioned to Intel® SGX enabled platforms, the hardware primitives used to support the process, and the Intel Verification Service that simplifies the verification of an SGX attestation. The paper also includes a short primer on the Intel® Enhanced Privacy Identifier which is signature algorithm used by Intel® SGX attestation architecture.

1 Introduction

Intel® SGX is a set of processor extensions for establishing a protected execution environment inside an application [2]. This execution environment is called an enclave. SGX enclaves are created without secrets. Secrets are delivered after the enclave has been instantiated on the platform. The process of proving that the enclave has been established in a secure hardware environment is referred to as remote attestation. To perform an attestation the platform must have access to an attestation key. In [1] Intel outlined an architecture for how the platform produces the attestation but did not outline how the attestation key was delivered to the platform. It also did not outline the responsibilities of a relying party when verifying the attestations.

This paper outlines the services infrastructure Intel has constructed to support the initial implementations of the Intel® SGX technology. In Section 1, we recap the attestation flow outlined in [1] and identify requirements for a corresponding attestation key provisioning architecture. In particular, we highlight the need for recovery mechanism when the Trusted Computing Base (TCB) is modified.

Section 2 provides a high level overview of how initial implementations of Intel® SGX provide a recovery mechanism that allows the platform to be updated, and attest to this update.

Section 3 outlines the Intel® Enhanced Privacy Identifier (EPID) architecture used to support Intel® SGX architecture. EPID is the algorithm of choice for SGX attestations.

Section 4 details the provisioning and attestation verification services Intel has established to support SGX.

1.1 Attestation Primer

Attestation is the process of demonstrating that a software executable has been properly instantiated on a platform. The Intel® SGX attestation allows a remote party to gain confidence that the intended software is securely running within an enclave on an Intel® SGX enabled platform. The attestation conveys the following information in an assertion:

- Identities of software being attested.
- Details of unmeasured state (e.g. the mode software is running in).
- Data which software associates with itself.

SGX uses an asymmetric attestation key, representing the SGX TCB, to sign an assertion with the information listed above.

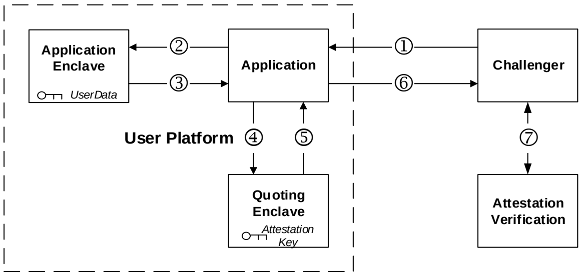

Figure 1: Attestation Flow

1.2 Attestation Flow

When an application receives an attestation request from an off-platform challenger (1), the application requests that its enclave produce an attestation (2). What follows is a two-part process involving the application sending a local-attestation (3 & 4) from its enclave to a Quoting Enclave. The Quoting Enclave verifies the local-attestation and converts it into a remote attestation by signing the local attestation using its asymmetric attestation key. The Quote (a remote attestation) is returned to the application (5) and then to the challenger (6). Finally, the Challenger can use an Attestation Verification Service (7) to perform Quote verification.

1.3 When to use Attestation

Intel recommends that remote attestation be performed during the set-up phase of a secure application. During this phase the service provider and the enclave will agree on an authentication token (e.g. a private/public key pair the enclave generates). To store this authentication token, the enclave can then encrypt the token using a key that is specific to the enclave, the platform and it's TCB – known as the Seal Key.

When the application is restarted, the enclave can load and decrypt the authentication token with the enclave's Seal Key. This act of encrypting and decrypting the token using the Seal Key allows the enclave to assert the TCB is the same as it was when the enclave registered with the service provider.

1.4 Upgrading the TCB

The architecture of SGX was designed so that if certain classes of vulnerabilities are discovered in SGX, they can be removed by an upgrade to the platform. This is process is referred to as TCB Recovery. It is desirable in those cases that the new TCB be reflected in the platform's attestations. The updated attestation allows a relying party to verify that the vulnerability has been removed. The new TCB will be reflected in attestations that occur following the replacement of the attestation key. Should an individual platform become permanently revoked due to an irrecoverable attack, the platform should not receive a new attestation key.

1.5 Intel Attestation Requirements

Based on the description above of the attestation flow and upgrading the TCB the following requirements were identified:

1) Provide a platform attestation key that represents the hardware & software Trusted Computing Base (TCB) of Intel® SGX;
2) Replace an attestation key if the platform is upgraded to fix an identified security issue with SGX;
3) Prevent compromised parts from getting additional attestation keys;
4) Ensure user privacy is not eroded by attestation.

These requirements are met as follows: Requirement 1 is covered in Section 2 which describes how to provide a cryptographic binding between the TCB and its keys; the use of Intel® Enhanced Privacy Identifier as described in Section 3 covers requirement 4; Section 4 covers how the EPID key is provisioned and verified fulfilling requirements 2) and 3).

2 TCB Key Binding

The Intel® SGX attestation requirements not only require a mechanism to identify the components of the TCB, through the use of an attestation key, but to allow for the replacement of that key should one of the components need updating. This section details how different keys are created when the TCB is modified.

SGX has two basic components that comprise its Trusted Computing Base: the processor hardware and the software components used in Attestation. The TCB specific key creation process that binds both components together has two stages. The first stage accounts for the update to the hardware. The second stage binds the software components of the TCB to the hardware TCB key.

2.1 Hardware TCB Derivation

Each update to processor logic is signed and is assigned a Security Version Number (SVN). The SVN is an incrementing value that represents the security revision of the component. To bind a specific component version to the platform, the Root of Key Transformation mechanism derives a TCB key based on the SVN of the component, shown in Figure 2.

The transformation is a cryptographic pseudo-random function (PRF) that prevents the original hardware key from being compromised if the derived key is leaked.

For the processor logic, a microcode update can be released to remedy certain security issues. The process for performing such an update is described in the Intel IA32 Software Developers Manual [2]. The microcode update applied out of reset identifies the base SGX hardware SVN for the platform.

To allow the easy migration of data secured from previous SVN's the platform provide access to previous TCB keys. However, the platform prevents access to future TCB keys.

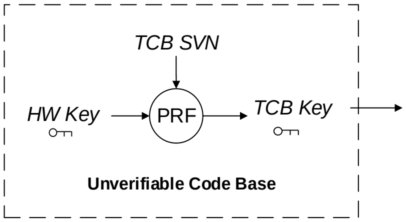

Figure 2: Basic TCB Derivation

To achieve easy data migration, the hardware uses a PRF looping mechanism, shown in Figure 3. The outcome of this process is a sequence of TCB key values which are in reverse-SVN order, shown in Figure 4. Keys which have a lower SVN are referred as being 'older' while keys from a higher SVN being 'future' keys. The advantage of this approach is the processor can dynamically generate previous TCB keys.

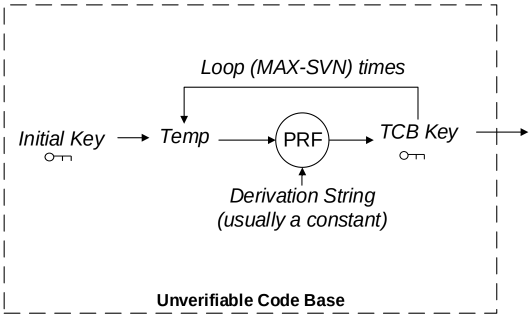

Figure 3: TCB PRF Loop

The looping scheme forms the basis of the Key Recovery Transformation process. The PRF loop takes place in the processor very early during the processor boot phase, before the first instruction is fetched, and the SVN is then fixed.

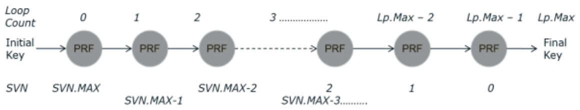

Figure 4: TCB Key Timeline

The Root of Key Transformation of the update loader calculates the number of loops that are performed, by subtracting the SVN of the update from a loop count maximum.

2.2 Root Keys

Each SGX enabled processor supports two statistically-unique values stored in fuses. These are known as the Root Provisioning Key and the Root Seal Key.

The key transformation process operates on the Root Provisioning Key, seen here in Figure 5. The Root Provisioning Key is randomly created and retained by Intel. It is the basis for how the processor demonstrates that it is a genuine Intel® SGX CPU at a specific TCB. This Root Provisioning Key is generated by a special purpose offline key generation facility, and is then delivered to Intel's factory network.

The Root Seal Key is created during processor manufacturing and is not retained by Intel. As shown in Figure 5 and Table 3 all keys except the Provisioning Key include the Root Seal Key in their derivations. This renders those keys unknown to Intel.

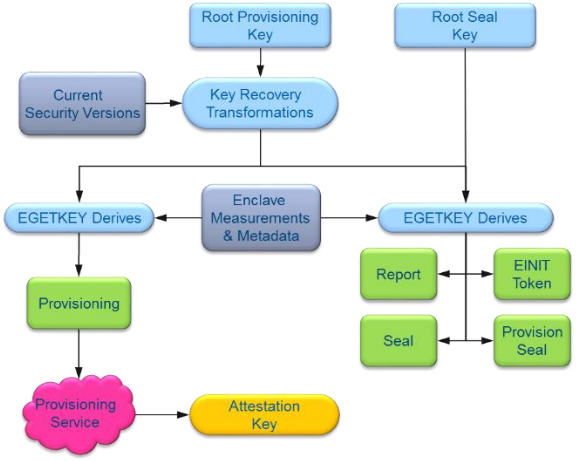

Figure 5: SGX Key Hierarchy

2.3 Software TCB Derivation

The hardware TCB transformed key is then used as the basis for the second derivation that brings the software elements into the derived key. This second derivation takes place in the EGETKEY instruction. Table 1 identifies the properties of the software environment that can go into the second key derivation.

The EGETKEY instruction provides a mechanism to derive a number of different keys. Table 2 identifies the keys that can be created by calling the EGETKEY instruction. The EGETKEY instruction combines the values from Table 1 using another PRF mechanism.

Field	Purpose
MRENCLAVE	the SHA256 hash measurement of the enclave computed during enclave build
MRSIGNER	the SHA256 hash of the public key used to sign the enclave's SIGSTRUCT
CPUSVN	a set of SVN of firmware components in the TCB. In the case where more than one updatable component is included in the TCB.
ISVSVN	the SVN of the software component in the TCB assigned by the enclave signer [through SIGSTRUCT].
ISVPRODID	a product identifier, assigned by the enclave signer [through SIGSTRUCT], used for dividing the key space up.
OwnerEpoch	a value provided by the platform, created when a new owner takes possession of the platform.

Table 1: Software Properties used in Key Derivation

The properties of each of these keys is shown in Table 3, where 'Yes'/'No' indicates whether the field is included using values associated with the hardware, and 'Req' means the values included is filtered by a user request through the KEYREQUEST structure. Consult the EGETKEY instruction description in the Software Developers Manual [2] for further details.

Key	Purpose
EINIT Token	EINIT Token creation Key
Report	EREPORT verification key.
Seal	Protects enclave secrets that need to be exposed outside the enclave for long term retention.
Provisioning Seal	Attestation key provisioning enclave uses for protecting attestation keys for long term retention outside the enclave.
Provisioning	Attestation key provisioning enclave's uses for proving the platform is at the TCB it is claiming in the provisioning protocol.

Table 2: SGX Keys

The Provisioning Key and the Provisioning Seal Keys, do not contain OwnerEpoch to allow Intel to identify the platform as a genuine and secure attestation keys regardless of the owner. The Provisioning Seal key includes the Root Seal Key which renders the key unknown to Intel.

For each TCB that Intel issues EPID keys for, the Key Generation facility derives the Provisioning Key that EGETKEY will produce on each part. This key is used in the EPID provisioning protocol with Intel's provisioning service

	Attributes	Seal Fuses	Owner Epoch	CPU SVN	ISV SVN	ISV PRODID	MRENCLAVE	MRSIGNER	RAND
EINIT Token	Req	Yes	Yes	Req	Req	Yes	No	No	Req
Report	Yes	Yes	Yes	Yes	No	No	Yes	No	Req
Seal	Req	Yes	Yes	Req	Req	Yes	Req	Req	Req
Provisioning	Req	No	No	Req	Req	Yes	No	Yes	Yes
Provisioning Seal	Req	Yes	No	Req	Req	Yes	No	Yes	Yes

Table 3: SGX Key Properties

3 Intel® Enhanced Privacy ID

3.1 Intel® Enhanced Privacy ID

Attestation using standard asymmetric signing schemes has drawn some privacy concerns [3], even when used in a conjunction with a trusted third party (PrivacyCA). To address some of these privacy concerns Intel uses an extension to the Direct Anonymous Attestation scheme called Intel® Enhanced Privacy ID (EPID) [4]. EPID is a type of group signature scheme that allows a platform to sign objects without uniquely identifying the platform or linking different signatures. Each EPID signer belongs to a "group", and verifiers use the group's public key to verify signatures. For SGX, EPID is used by the Quoting Enclave to sign enclave attestations.

In SGX terms Intel defines a group to be a set of CPUs of the same type (e.g. core i3, i5 or i7) that are all at the same TCB (i.e. same CPUSVN and quoting enclave ISVSVN). A typical size for a fully populated group is a million to a few million platforms.

3.2 EPID Privacy Enhancing Properties

EPID is an extension to the Direct Anonymous Attestation (DAA) algorithm that was implemented in the TPM 1.2 [5]. It shares the privacy enhancing properties of DAA, and has an additional revocation mode, signature based revocation, as described below.

EPID signatures are anonymous. Given a signature, no one, including the group issuer can determine which key was used to create the signature. This is different from most group signature schemes in which signatures are anonymous to verifiers, but the issuer of the keys has the ability to open any signature and determine which key was used to sign.

3.3 EPID Signature Modes

EPID has two signature modes with different linkability properties. With any EPID signature, a base is chosen for that signature. If the base is different for two signatures, those signatures are unlinkable. That means that no one, including the issuer, can determine if the two signatures were generated by the same key or different keys. But if the base is the same for two signatures, it is easy to determine if the two signatures were generated by the same key or not. It is not possible to determine what key generated the signatures, just whether or not it was the same key.

The two EPID signature modes are called Random Base Mode and Name Based Mode. In Random Base Mode the signer picks a different random base for every signature and thus the signatures are unlinkable. In Name Base mode, a verifier picks a name for the base to be used for a signature. In this mode, two signatures generated with that same verifier name would be linkable. But two signatures generated using different names would not be linkable. Most verifiers will want to use the Name Base Mode to protect against compromise.

To understand why Name Base Mode is preferred, consider a bank using attestation for registration of authentication tokens. Now suppose an EPID key is compromised and an adversary creates malware which uses the compromised EPID key. The adversary could get many users to unknowingly use the malware during registration. If random base mode is used, the bank would not know that the attestations were all generated from a single EPID key because of the unlinkability property of Random Base Mode. On the other hand, if Name Base Mode is used the bank can notice many accounts are being registered with the same key. Then the bank would stop accepting attestations with this key, and could notify customers that had used this key that they had probably been infected with malware and should clean their platforms and re-register.

With Name Base Mode, the scheme implementer must ensure a particular name is not used too much. See section 4.5.2 on how the Intel Attestation service enforces this separation.

3.4 EPID Revocation Schemes

EPID supports four different revocation schemes

- Private Key Revocation
- Verifier Local Revocation
- Signature Based Revocation
- Group Based Revocation

3.4.1 Private Key Revocation

Should Intel ever receive a private EPID key in the clear, the offending key can be placed on a revocation list that is processed by the attestation service during signature verification. Since the private key should only exist in SGX protection on a single platform, it should clearly be revoked.

3.4.2 Verifier Local Revocation

With traditional signing algorithms (e.g. RSA or ECDSA), if a relying party notices behavior by a particular key that indicates that the key has been compromised, the relying party can revoke the key locally using some key identifier in the public key certificate (e.g. Issuer and Serial Number or public key hash).

When a verifier is using EPID with name base mode, this same type of local revocation is available. This is possible since the signatures are linked in Name Base Mode.

3.4.3 Signature Based Revocation

With traditional signing algorithms (e.g. RSA or ECDSA), if a relying party notices behavior by a particular key that is indicative of a particular key, the relying party can provide this evidence to a revocation authority, and if the revocation authority is convinced that the key has been compromised, then the revocation authority can add the corresponding certificate to the revocation list, so that it would be effectively revoked for all relying parties, not just the original one who noticed the behavior.

The corresponding EPID mechanism is known as signature revocation. It is available whether a relying party is using random base or name base. The method of checking for revocation is different for EPID than it is for the traditional RSA or ECDSA algorithms.

When a relying party notices odd behavior linked to a signature provided to it, the relying party can inform a revocation authority, and the revocation authority can determine if there is enough evidence to revoke the key that provided that signature. If it does so it places that signature on a revocation list. This will have the effect that any verifier who is using that revocation list would no longer accept a signature from that key.

When an attesting platform creates a signature, the attesting platform computes a non-revoked proof for each item on the signature revocation list (sigRL). The non-revoked proof is a set of mathematical values that demonstrates to the verifier that a signature on the sigRL was not generated by the attesting platform.

Allowing a relying party to create their own sigRL could eliminate a user's privacy, e.g. creating a sigRL with only one platform on it would allow an attacker to create a test for a specific platform. This is why the sigRL is signed by the revocation authority.

Signature based revocation is the revocation method that is available in EPID and is not available in DAA.

3.4.4 Group Based Revocation

Group based revocation allows the relying party to identify that the Group for is no longer current for some reason, e.g. the EPID Group Master Issuer Key has been compromised.

4 SGX Infrastructure Services

4.1 Overview

Figure 6 illustrates the wider picture of how the key provisioning system and the services Intel provides integrate together. The Offline Key Generation facility is responsible for the production of the random Root Provisioning Key for each part. These are transmitted securely to the High Volume Manufacturing system that integrates the Root Provisioning Key into the part.

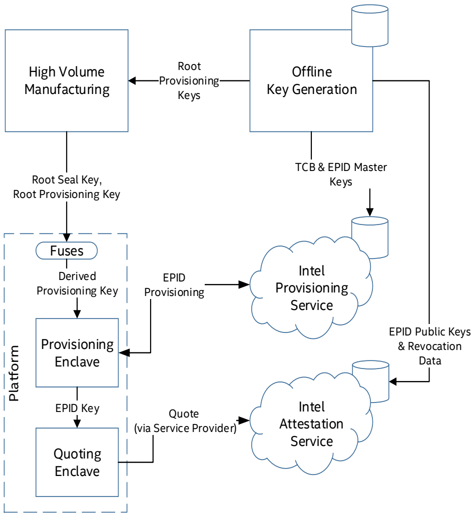
High Volume Manufacturing
Root Provisioning Keys
Offline Key Generation
Root Seal Key, Root Provisioning Key
TCB & EPID Master Keys
Fuses
Derived Provisioning Key
EPID Provisioning
Intel Provisioning Service
Provisioning Enclave
EPID Public Keys & Revocation Data
Platform
EPID Key
Quote (via Service Provider)
Intel Attestation Service
Quoting Enclave

Figure 6: SGX infrastructure Services

The offline Key Generation facility also issues derived TCB specific provisioning keys and EPID Master Keys to the Provisioning Server. This is performed on new part creation and during TCB recovery. It also allows for provisioning service incidents to be handled via a TCB recovery process. Finally EPID public keys and revocation information are issued to the Intel Attestation Verification Service.

4.2 Keys and Revocation Data

As well as being the creator of the Root Provisioning Key, the offline key generation is responsible for the creation of the data used to support the EPID Attestation architecture. Figure 7 shows how all the different elements in the EPID system relate to one another.

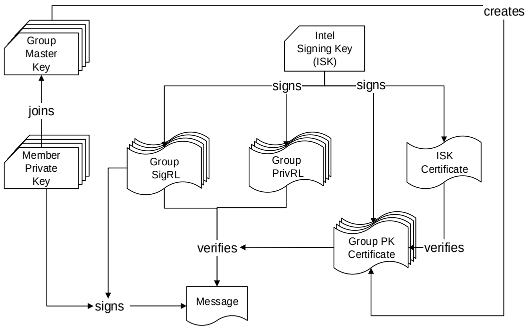

Figure 7: EPID Infrastructure Elements

Intel Signing Key (ISK) – is the main signing key used by Intel to authenticate objects it produces. This serves a similar function to the root key of a Certification Authority.

Intel Signing Key Certificate – contains the public signature verification key that is paired with the Intel Signing Key. This is similar to the root certificate of a Certification Authority.

EPID Group Master Key – each group has a master issuing key. This secret key is used in the creation of the group public key and the member join process.

EPID Member Private Key – each member of an EPID group has their own private key. Intel never knows a members' private key. The key is randomly generated through a blinded join protocol between the provisioning enclave on the platform and Intel

EPID Private Key Revocation List (PrivRL) – is a list of EPID private keys that Intel knows have been compromised, because Intel has been given the EPID private keys.

EPID Signature Revocation List (SigRL) – is a list of EPID signatures which were signed by EPID keys that are suspected to have been compromised.

EPID Group Public Key Certificate – is the certified form of the EPID Group Public Key. EPID Group Certificates are signed by the Intel Signing Key.

4.3 Provisioning Scenarios

To support the requirements for attestation, the Intel EPID Provisioning service implements three scenarios:

i. First key generation
ii. New key generation after TCB upgrade
iii. Retrieving previous key

4.3.1 First Key Generation

When SGX software is first deployed, the platform needs to establish an attestation key. To establish an attestation key, the platform demonstrates that it is genuine Intel hardware running at the TCB it claims then joins the EPID group selected by the EPID Provisioning Server. The platform also encrypts a copy of its private EPID key for backup using the Provisioning Seal Key. The Provisioning Seal Key is known only to this platform.

4.3.2 New Key Generation after TCB Upgrade

Should the platform undergo a TCB recovery (e.g. by either receiving a BIOS upgrade or a new provisioning/quoting enclave), the platform may request a new EPID key for the upgraded TCB. Before allowing the platform to join a new EPID group, the provisioning process needs to guarantee that any key previously issued to the platform was not revoked.

4.3.3 Retrieving a Previous Key

If the platform loses its previous attestation key, the backup retrieval process will recover the ability for the platform to conduct attestations. Providing a recovery service also guarantees every platform has access to previous keys to demonstrate that it was not previously revoked.

4.4 Provisioning Flows

The EPID Provisioning Service protocol consists of four messages. Figure 8 outlines the high level message flow between the Provisioning Enclave and the Provisioning Server.

The Provisioning Server has two main components: a protocol engine that handles the main flow of the messages, and a secure processing environment that processes the cryptographic proofs and secures the EPID Master Key.

The secure processing environment can be implemented with a Hardware Security Module or SGX. The following sections describes each of the messages further.

4.4.1 Message 1: Enclave Hello

The first message contains a Platform Provisioning ID (PPID) and a claimed TCB version. The PPID is derived from the very first provisioning key available to the provisioning enclave. This message is encrypted using a public key of the Intel Provisioning Server as it contains the Platform Provisioning ID. To help guard privacy, only authorized enclaves can access their derived provisioning key.

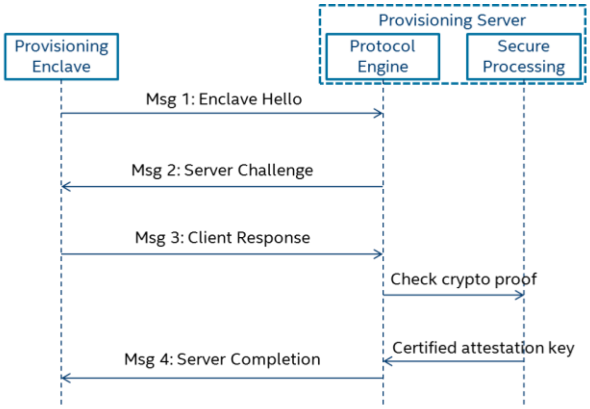

Figure 8: Provisioning Flow

4.4.2 Message 2: Server Challenge

In response to message 1, the provisioning server uses the PPID to determine whether the platform has been already provisioned. If the platform has not been previously provisioned, the provisioning server selects a group for the platform to join and returns the public key and a liveness challenge (e.g. a nonce).

4.4.3 Message 3: Client Response

The provisioning enclave now demonstrates that it is executing inside an enclave on an Intel platform with a particular TCB. Proof of platform and TCB can happen in a couple of ways:

a. The provisioning enclave is provided a pre-encrypted random value (computed by the offline key generation facility using the provisioning key) and demonstrates TCB level by decrypting the key with its provisioning key for the claimed TCB.

b. The provisioning enclave uses its provisioning key to seed an EC-DSA key creation process. The key generation facility generates the same EC-DSA key and provides the provisioning service with the corresponding public key.

Next, the provisioning enclave conducts the EPID blind join protocol with Intel, including the liveness challenge issued in message 2. At the completion of this protocol, the provisioning enclave will have a private EPID key, and Intel will not know what it is. As part of this flow the platform also provides an encrypted backup copy of the EPID private member key. This encryption is done with the Provisioning Seal key, which is unknown to Intel.

4.4.4 Message 4: Server Completion

In this step, the verification of the proof of platform TCB and the blind join are verified and the member's key is certified. As these operations are very security sensitive, the processing takes place inside a secure environment.

The results of the certification process are returned in Message 4 including a copy of the encrypted key backup.

4.4.5 Handling Revocation

The EPID Provisioning Service protocol prevents compromised parts from getting additional attestation keys. This is achieved by applying the current revocation list before issuing a new key. If the platform has been previously provisioned with an attestation key, message 2 will include the current revocation list, which must be satisfied via non-revoke proofs in message 3. See section 3.4 for further details on EPID revocation.

4.5 The Intel Attestation Service (IAS)

Intel provides two services, a "development" service for developers testing out their services, and a "production" service for services providers to use in production mode.

4.5.1 Interfaces

To use IAS, a service provider must register a TLS certificate and are assigned a Service Provider ID (SPID). The service provider then has access to two main interfaces:

GetSigRL[GID] – Returns the up to date Signature Revocation List for the identified EPID group (GID).

VerifyQuote[QUOTE] – returns an indication of the successful nature of signature verification

The GetSigRL interface is used so that Relying Parties requesting attestations can easily obtain the current revocation list for the EPID group of the platform with which they are communicating.

4.5.2 Support for Linkable Signatures

The Intel Quoting Enclave supports both EPID signature scheme modes. The signature mode used is communicated from the relying party to the platform processing the attestation request.

The IAS ensures that two service providers using linkable quotes (i.e. EPID Name Based Mode) cannot collude and determine whether signatures that each possess came from the same platform. Specifically, the IAS ensures that two service providers use different names for their bases. In effect, in this mode, EPID operates like the platform has a key for each service provider that requests an attestation from the platform.

Use of linkable signatures allows recurring attestations from the same platform to the same relying party to be linked with previous attestation. This allows a relying party to detect behavior anomalies in the use of an EPID key.

The use of the TLS certificate is especially important when the service provider chooses the use of linkable quotes. The SPID is used by the quoting enclave as part of signature specialization process. The quote is encrypted to the Attestation Service. Intel will only grant the service provider access to the results if the SPID in the quote matches the service provider's SPID registered with TLS certificate.

4.5.3 Quote Verification

When a quote is submitted for verification, the IAS takes the following steps:

1) Decrypts the quote if necessary
2) Checks the QUOTE.SPID is the same SPID as user requesting verification. (Only for linkable signatures)
3) Verifies that QUOTE.GID is an up to date group
4) Retrieves from local storage Private Key Revocation List, Signature Revocation List and EPID Public Key for QUOTE.GID
5) Verifies EPID signature by:
    a. Verifying signature portion of quote is signed by a group member.
    b. Checks signature not issued using a revoked private key
    c. Verifies non-revoked portion of signature used up to date sigRL & checks each member of the sigRL has a correct entry.
6) Returns the status of checks to the service user.

5 Summary

To get the full benefit of Intel® Software Guard Extensions requires the use Remote Attestation. Intel provides a worldwide infrastructure to enable both the delivery and use of Intel Enhanced Privacy

Identifier for SGX. The infrastructure can issue new attestation keys if errors are found in the TCB of SGX (hardware or software). The Intel Attestation Verification Service to simplified verification that SGX attestations are genuine. Further information about SGX can be found at http://software.intel.com/sgx .

6 References

[1] I. Anati, S. Gueron, S. P. Johnson and V. R. Scarlata, "Innovative Instructions for Attestation and Sealing," 2013. [Online]. Available: https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing.

[2] Intel, "Intel(r) 64 and IA-32 Architectures Software Developers Reference Manual," November 2015. [Online]. Available: http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html.

[3] Article 29 Data Protection Working Party, "Working Document on Trusted Computing Platforms and in particular on the work," European Commission, Brussels, 2004.

[4] E. Brickell and J. Li, "Enhanced privacy ID from bilinear pairing for Hardware Authentication and Attestation," *International Journal for Information Privacy, Security and Integrity,* vol. 1, no. 1, pp. 3-33, 2011.

[5] Trusted Computing Group, "Trusted Platform Module Main Specification (TPM1.2)," March 2011. [Online]. Available: http://www.trustedcomputinggroup.org/resources/tpm_main_specification.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

No computer system can provide absolute security under all conditions. Built-in security features available on select Intel® processors may require additional software, hardware, services and/or an Internet connection. Results may vary depending upon configuration. Consult your system manufacturer for more details.

Intel®, the Intel® Logo, Intel® Inside, Intel® Core™ , Intel® Atom™, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and/or other countries. Other names and brands may be claimed as the property of others.