WHITE PAPER

Intel® Software Guard Extensions (Intel® SGX)

Overview on Signing and Whitelisting for Intel® Software Guard Extension (Intel® SGX) Enclaves

Scope

This paper provides an overview on signing and whitelisting for Intel® Software Guard Extension (Intel® SGX) enclaves for application release. The information presented here applies to Intel SGX enabled applications developed for the Microsoft* Windows* and Linux* OS. The paper assumes a basic understanding of Intel SGX. General information on Intel SGX can be found on the Intel SGX portal at: https://software.intel.com/sgx.

Introduction

A signed enclave is critical to enforcing security for Intel SGX enabled applications. Only signed enclaves are allowed to be loaded into EPC (Enclave Page Cache) memory, both during development and after release. An enclave's signed structure includes a measurement of enclave code and metadata made at build time. A duplicate measurement is made at enclave load time and the results are compared with the original signed structure to detect whether any changes were made to the released enclave. An additional check is performed to see if the enclave signer is whitelisted by Intel. If any change is detected, or if the enclave signer is not whitelisted, the enclave load is aborted.

The Intel SGX SDK for Microsoft Windows and the Intel SGX SDK for Linux both include a signing tool called **sgx_sign**. This tool is typically run automatically by Microsoft Visual Studio or by the Linux make file as part of the post build flows during development. **sgx_sign** can be run manually in both the Windows and Linux environments, if needed. **sgx_sign** is run manually to generate whitelisting materials to help get an enclave signer whitelisted.

During development, there are two different signing methods: a *one-step* method and a *two-step* method.

- The one-step method is used during application development/debugging. The enclave is built either in Debug or Pre-Release mode. Enclaves are then signed post build using a private test key. For Windows, you can choose to use a private test key provided by Visual Studio. For Linux, you can use the sample private key that comes with the sample enclaves, or use your own private key as described in the Intel SGX SDK Developer Reference. Because one-step signing is straightforward, it is not discussed further in this paper.
- The two-step method is used to securely sign an enclave built in Release mode. This signing makes use of a private test key stored in a secure key facility or platform. The enclave signature is then incorporated into the enclave prior to release.

For whitelisting an enclave signer, **sgx_sign** is run manually on a signed enclave (usually an empty/dummy enclave) to generate materials submission (SIGStruct file and MRSIGNER value) needed for whitelisting.

> **Note:** Whitelist material generation support in **sgx_sign** is being added for the V1.9.5 SDK release for Windows (and for the V2.2 SDK release for Linux). Previously, whitelist material generation was performed by a separate tool, **sign_helper**, which was provided with the Whitelist Registration Form.

The processes of signing an enclave and whitelisting an enclave signer includes technical and business steps. Figure 1 shows signing and whitelisting, which are described in the following sections.

Signing flow for application release

Commercial License Agreement

Every Intel SGX application signer must have a signed Commercial Use License Agreement in place before they are whitelisted. To smooth the release process, you should start the steps to complete the license agreement as soon as possible after deciding to develop an Intel SGX enabled application. To help you understand this part of the overall signing/whitelisting process, browse the following:

- Intel SGX Licensee Guide (PDF)—Defines key terms and documents key Intel expectations of developers and Intel SGX enabled applications.
- Intel SGX License Agreement (PDF)—Copy of the (unsigned) license agreement for reference.
- Commercial License Request Form (online form)—Used to apply for the license. (Shows what information you will need to apply for the license.)

This collateral and the license request form can be accessed here: https://software.intel.com/sgx/commercial-use-license-request. That page also summarizes the steps to get the signed commercial use license in place.

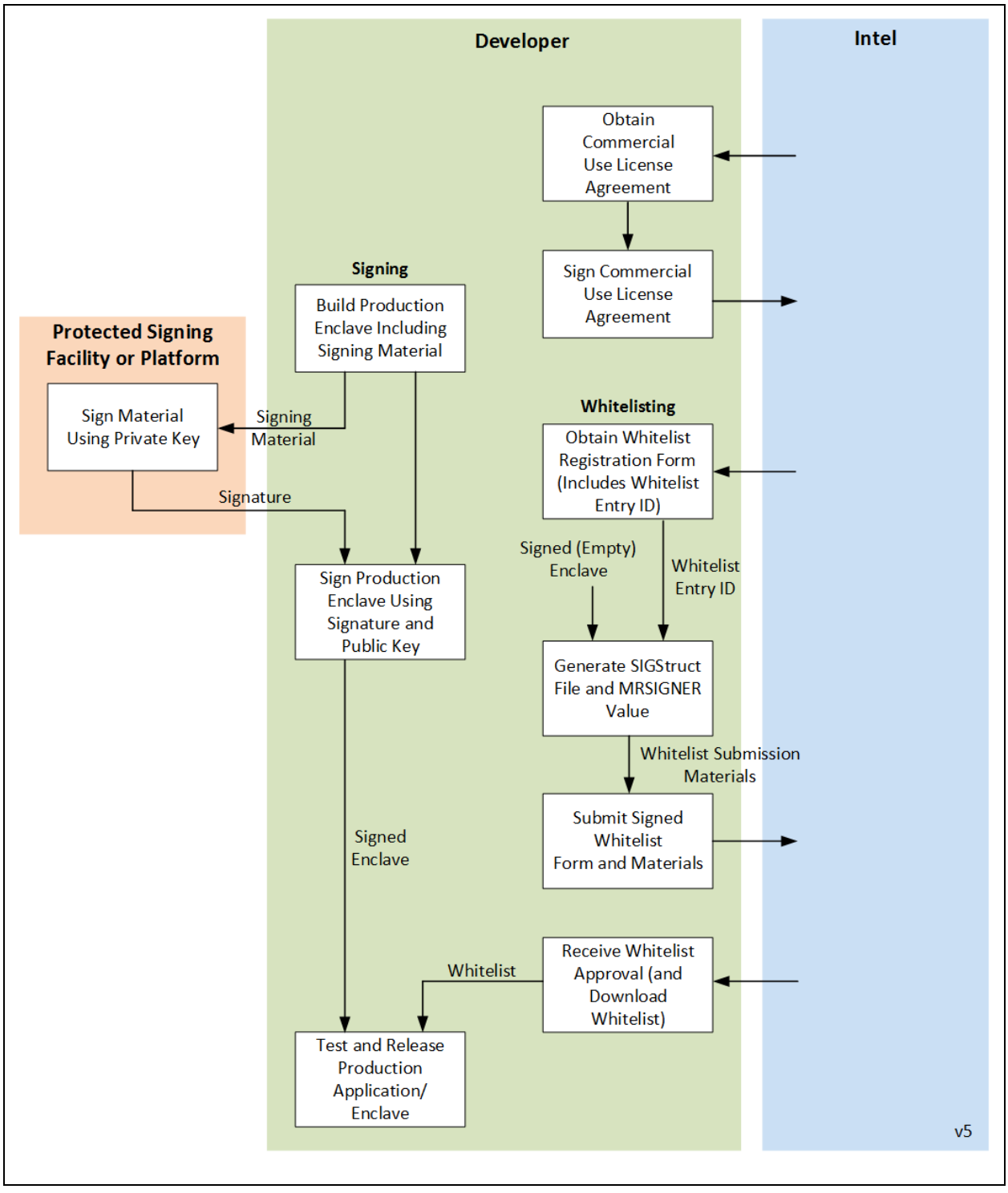

Figure 1. Signing and whitelisting flows for releasing Intel SGX enabled applications

Signing a Production mode enclave

Signing a production mode enclave for release involves:

1. Building the enclave in **Release** (non-debug) mode, which generates signing material at the end of the build. This is the first step of the two-step signing method.
   - For the Intel SGX SDK for Windows, this is done in Visual Studio by selecting **Release** as the build configuration.
   - For the Intel SGX SDK for Linux, this is done by including `SGX_MODE=HW` and `SGX_DEBUG=0` in the make command.
   - If using the Intel SGX Eclipse Plugin with Linux, select SGX **Hardware Release** as the build configuration.
   - The **sgx_sign** tool automatically runs at the end of the build to generate the signing material in the form of a file with a `.hex` file extension. The default name is the same as the enclave filename. The `.hex` file is written to the same folder as the enclave output `.dll`/`.so` file.
2. Taking the signing material to an external signing facility or platform to be signed with the ISVs private signing key (in place of the private test key used during development). See the discussions on *External signing facility/platform* and *HSM and alternatives* below.
3. Taking the signature from the signing by the external facility/platform back to the development platform and integrating it into the enclave using **sgx_sign** with your public key. This is the second step of the two-step signing method.

For details on the use of **sgx_sign**, see the *Intel SGX SDK Developers Reference* for your OS, or the whitepaper titled: Enclave Signing Tool for Intel Software Guard Extensions (Intel SGX).

External signing facility/platform

Protection of the ISV's private signing key is extremely important. The basics of a highly-secure external signing facility/platform include:

- Separate the development/test key from the production key. This is supported by the SDK switching from the one-step to the two-step signing method for Release mode enclaves.
- Cryptographic key protection. It is recommended that private signing keys be protected in a secure, tamperproof, cryptographic manner, such as with a Hardware Security Module (HSM).
- Physical security. This includes limiting the personal who have physically access to the signing facility/platform and who can log into the cryptographic key protection technology.

This link takes you to a paper documenting actual situations that occurred when a malicious party gains access to an ISVs private signing key: https://www.thawte.com/code-signing/whitepaper/best-practices-for-code-signing-certificates.pdf. (Excellent reading when spinning up.)

HSM and alternatives

Intel recommends use of a protected environment such as an HSM-managed enclave signing system for production signing. An alternative is to use another Intel SGX enclave to store the private signing key and to perform enclave signing.

Whitelist registration flow

An Intel SGX enclave signer must be added to the Intel SGX Whitelist before the application is released. The whitelist gives an application the authorization to create and sign enclaves and use the signed enclave `.DLL/.so` file as part of an application. The process to get your signing authority whitelisted is as follows:

1. Create an enclave (empty/dummy) and sign it at your signing facility/platform. This signed enclave is used in step 3.
2. Obtain the Whitelist Registration Form by contacting sgx_program@intel.com. Intel provides Whitelist Registration Form, which includes the Whitelist Entry ID for the enclave signer.
3. Run the **sgx_sign** tool on the signed (empty/dummy) enclave, using your public key, to generate whitelist submission materials. **sgx_sign** generates the SIGStruct file needed for whitelist submission and displays the MRSIGNER value for the SIGStruct on the screen. See the *Intel SGX SDK Developer's Reference* for your OS for information for details about the SIGStruct file.
4. Write the MSB and LSB of the MRSIGNER value on the Whitelist Registration Form.

   Note: For details on the use of **sgx_sign**, see the *Intel SGX SDK Developers Reference* for your OS, or the whitepaper titled: Enclave Signing Tool for Intel Software Guard Extensions (Intel SGX).

   Note: See *Legacy sign_helper information* below if you as using an Intel SGX SDK version earlier than 1.9.5 for Windows or 2.2 for Linux.
5. Submit the completed Whitelist Form and whitelist materials for approval to sgx_program@intel.com. For whitelist submission, the SIGStruct filename must follow this naming structure:

   <ISVName>_<EnclaveName>_<WhitelistEntryID>_sigstruct.bin
6. After receiving the Whitelist Approval Letter from Intel, download a current copy of the Intel SGX Whitelist for final testing and distribution.

Final testing and distribution

A valid whitelist must exist on the target system to load/create an enclave. The Intel SGX Plat Software (PSW) that will ship with your Intel SGX enabled application includes an Untrusted Run Time System (uRTS). Each time the target system is reset, the uRTS attempts to download

a current whitelist. If successful, the current whitelist is used. If for any reason the download does not succeed, the uRTS uses the default whitelist provided with your application/enclave. This means that an appropriate level of final testing must be done with your application, enclave, default whitelist, and PSW as installed by your application installer.

You can release your application package once it passes your final testing.

Legacy sign_helper information

Whitelist material generation prior to the Intel SGX SDK for Linux OS (V2.2) or for Microsoft Windows OS (1.9.5) was performed by the **sign_helper** tool, available apart from the SDK. This capability is now supported by **sgx_sign** in current SDKs. The following discussion documents the two command lines used with **sign_helper** if you are using an older SDK.

The command to generate the SIGStruct file from the signed (empty/dummy) enclave looks like this:

```
sign_helper get-css -in enclave_signed.dll -out sigstruct.bin
```

For whitelist submission, the SIGStruct filename must follow this naming structure:

<ISVName>_<EnclaveName>_<WhitelistEntryID>_sigstruct.bin

The command to generate/display the MRSIGNER value from the SIGStruct file looks like this:

```
sign_helper.exe print-css -in sigstruct.bin
```

The MSB and LSB of the resulting MRSIGNER value must be included on the Whitelist Registration Form.

Summary

At enclave load time, signed enclaves are checked to see if they have been changed and to see if the enclave signer is whitelisted. If any change is detected, or if the enclave signer is not whitelisted, the enclave load is aborted.

The **sgx_sign** tool, included in current versions of the Intel SGX SDKs for Linux and Windows, is used to sign a Release mode (non-debug mode) enclave with the ISVs private key. The tool is also used to generate whitelisting materials for the enclave signer to complete the whitelisting step. Whitelisting includes submission of the Whitelist Registration Form and whitelisting materials and receiving approval from Intel.

Once an enclave signer has been included in the Intel SGX Whitelist, a copy of that whitelist is downloaded for integration into the application package for final testing. When final testing is complete, the application package can be released.

References

1. Intel Software Guard Extensions SDK for Windows OS V1.9 (or later) — 2017 Intel Corporation. https://software.intel.com/sgx-sdk/download.
2. Intel Software Guard Extensions SDK Developers Reference for Windows OS — 2017 Intel Corporation. https://software.intel.com/sgx-sdk/documentation.
3. Intel Software Guard Extensions SDK V2.0 for Linux OS (or later) — 2017 Intel Corporation. https://software.intel.com/sgx-sdk/download.
4. Intel Software Guard Extensions SDK Developer Guide for Linux OS — 2017 Intel Corporation. https://software.intel.com/sgx-sdk/documentation.
5. Enclave Signing Tool for Intel SGX Enclaves — 2017 Intel Corporation. https://software.intel.com/sites/default/files/managed/ae/2e/Enclave-Signing-Tool-for-Intel-SGX.pdf.
6. Intel Software Guard Extensions Forums. https://software.intel.com/forums/intel-software-guard-extensions-intel-sgx.