WHITE PAPER

Intel® Software Guard Extensions (Intel® SGX)

Intel® Software Guard Extensions (Intel® SGX) Debug and Build Configurations

Scope

This article explains the debug and build configurations used to develop Intel® Software Guard Extensions (Intel® SGX) enclaves. The goal is to give the Intel SGX application developer the information they need to choose the correct build configuration at each stage of the application's development and release process. This article covers both the Intel SGX SDKs for Windows* and for Linux*. General information on Intel SGX is provided on the Intel SGX portal at: https://software.intel.com/en-us/sgx.

Introduction

To correctly select debug settings/compilation profiles for building and debugging enclaves for Intel SGX applications, developers must understand the following:

- Intel SGX architecture
- Enclave signing methods
- Intel SGX build configurations
- Intel SGX SDK compilation profiles

Intel SGX architecture

The Intel SGX architecture supports two modes of operation for enclaves: *Debug* mode and *Production* (non-debug) mode. Production mode enclaves have the full protection provided by the architecture. To support enclave debug, however, Debug mode enclaves differ from Production mode enclaves in four basic ways:

- Debug mode enclaves are created with the ATTRIBUTES.DEBUG bit set. This field appears in the output of the EREPORT instruction REPORT.ATTRIBUTES (see Chapter 38, *Enclave Access Control and Data Structures,* in the Intel x86 Software Developers Manual, 3D). The debug bit is not measured as part of the build process, so Debug and Production enclaves can have the same measurement. This ensures that reports generated by activities like attestation will look the same, which helps simplify the debugging process.
- Keys returned by the EGETKEY instruction leaf in debug enclaves are different for the same enclave in Debug vs. Production modes. Therefore, data sealed by a production enclave cannot be inspected by a debug enclave (and vice versa), which helps ensure protection of secrets.
- Debug mode enclaves can be debugged using the Intel SGX debugger (debuggers that do not support Intel SGX cannot debug enclaves under any circumstances).
- Performance counters are enabled inside Debug mode enclaves. These counters can be used

for performance tuning enclaves using a tool like the Intel® VTune™ Amplifier XE.

Signing methods

Two different signing methods are supported to address the needs of developers during the enclave development life cycle. Both methods use the **sgx_sign** signing tool included in the Intel SGX SDK. These methods are:

- **Single-step method** using the developer's temporary, private key; used with Debug and Pre-release applications:

  The signing tool supports a single-step signing process, which uses the signing key pair on the local build system. In this scenario, the developer manages the signing key pair, which could be generated by the developer using their own means. Single-step method is the default signing method for non-production enclave applications, which are created with the Intel SGX project debug, simulation, and pre-release profiles. Note, however, that the developer's temporary key stored in the build platform will not be white-listed and enclaves signed with this key can only be launched in debug or pre-release mode.

- **2-step method** using an external signing facility; used with Production applications:

  First step: At the end of the enclave build process, the signing tool generates the enclave signing material. The developer takes the enclave signing material file to an external signing platform/facility where the private key is stored, signs the signing material file, and takes the resulting signature file back to the build platform. (There is a requirement that any white-listed enclave signing key be managed in a Hardware Security Module.)

  Second step: The developer then runs the signing tool, providing the necessary information at the command line to add the hash of the public key and signature to the enclave's metadata section.

Intel SGX build configurations

There are typically two types of build configurations that a non-Intel SGX SW project defines: *Debug* and *Release*. These configurations are both supported for Intel SGX enclaves.

Developers may also want to experiment with a non-production enclave built with release compilation and linking flags, on a real hardware SGX-enabled platform. This would be an enclave built exactly as a production enclave, except for the signing process, which would be Single-step. To support the generation of this kind of enclave, Intel SGX supports a hardware non-debug build configuration known as *Prerelease*.

In addition, Intel SGX-enabled projects must support building and testing Intel SGX-enabled applications on non-Intel SGX platforms (or an emulator) using simulation libraries. This need is supported by a *Simulation* mode. Simulation mode can be thought of as an overlay that can be used with either Debug, Release, or Pre-Release enclaves. In actual practice, however, it is most often used with Debug enclaves.

Intel SGX SDK compilation profiles

The compilation profiles for the Intel SGX SDKs for Visual Studio on Windows and Eclipse on Linux, are identical as far as *Debug*, *Release*, and *Prerelease*. Characteristics of these profiles are as follows:

- *Debug*: compiler optimizations are disabled and symbol information is saved. This mode is suitable for source-level debugging. Enclaves built in debug mode are launched in enclave-debug mode.

- *Release*: compiler optimizations are enabled and no symbol information is saved. This mode is suitable for production build, performance testing, and final product release. Enclaves built in this mode are launched in enclave-production (non-debug) mode.

  Note: Enclaves compiled under the Release profile will not launch until the developer completes the production licensing process. To deliver a production-quality application using Intel SGX, contact the Intel SGX Program for information about a production license.

- *Pre-release*: same as Release with regard to compiler optimizations and symbol information. However, the enclaves are launched in enclave-debug mode, suitable for performance testing.

Simulation mode

Simulation mode links the application with libraries that simulate the Intel SGX instructions. This allows the enclave to be run on systems that do not support the Intel SGX instructions, but the application and enclave will not have any of the hardware protection that Intel SGX provides.

Note: The CPU must support SSE 4.1 in order to simulate Intel SGX instructions.

See the next subsections to understand how Visual Studio and Eclipse integrate Simulation mode.

SDK profiles for Visual Studio

The SDK for Microsoft Visual Studio supports the standard three build configuration profiles (**Debug**, **Pre-release**, and **Release**) and provides *one* **Simulation** Profile, which includes the Debug Profile (even though the Visual Studio menu option only says **Simulation**). Table 1 summarizes these SDK profiles for Visual Studio. Figure 1 shows the options on the Visual Studio **Configuration Manager** screen.

Table 1: Visual Studio SDK Compilation Profiles and Signing Options

Configuration Name	Simulation?	Debug Mode?	Signing Scheme to be Used
Debug	Hardware	Debug	Single Step
Pre-release	Hardware	Non-debug	Single Step
Release	Hardware	Non-debug	Two Step
Simulation	Simulation	Debug	Single Step

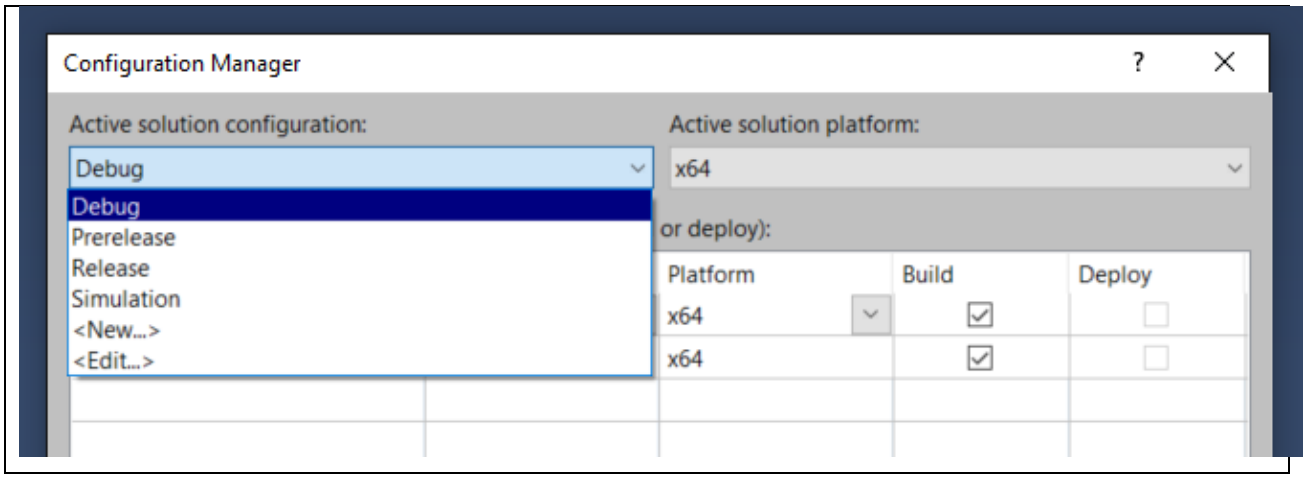

Figure 1: Intel SGX® SDK for Visual Studio Build Configurations

SDK profiles for Eclipse

The SDK for Eclipse supports the standard three build Configuration Profiles (**Debug**, **Pre-release**, and **Release**) and provides *two* Simulation profiles. The first is **Simulation**, which includes Simulation + Prerelease, and the second is **SGX Simulation Debug**, which includes Simulation + Debug. Table 2 summarizes these SDK profiles for Eclipse. Figure 2 shows the menu options in Eclipse.

Table 2: Eclipse SDK Compilation Profiles and Signing Options

Configuration Name	Simulation?	Debug Mode?	Signing Scheme to be Used
SGX Hardware Debug	Hardware	Debug	Single Step
SGX Hardware Pre-release	Hardware	Non-debug	Single Step
SGX Hardware Release	Hardware	Non-debug	Two Step
SGX Simulation*	Simulation	Non-debug	Single Step
SGX Simulation Debug	Simulation	Debug	Single Step
*Prerelease + Simulation			

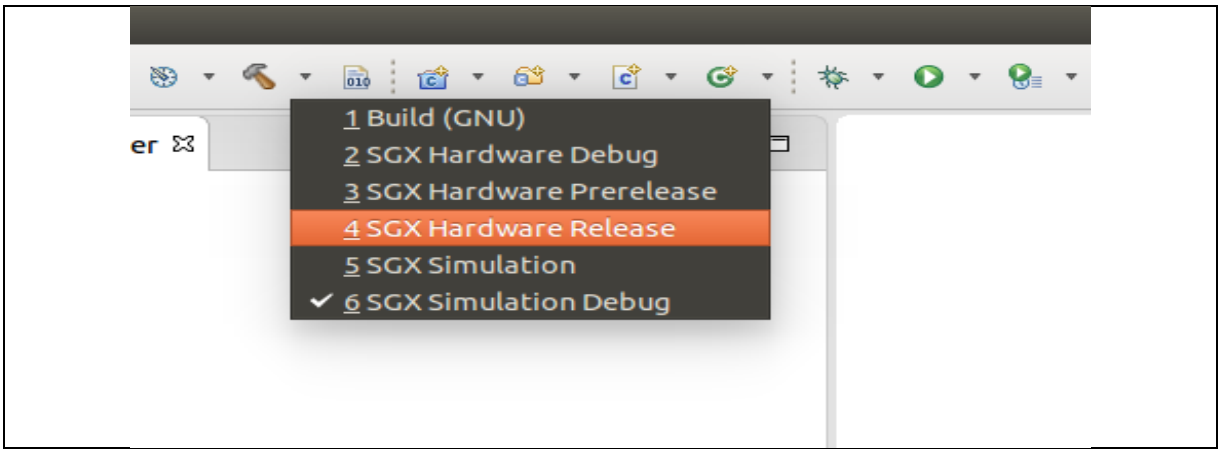

Figure 2: SGX® Build Configurations in Eclipse

WHITE PAPER

Intel® Software Guard Extensions (Intel® SGX)

Summary

Intel SGX architectural features, SDK build configurations, and SDK tools provide a robust set of capabilities to build Intel SGX enclaves that are appropriate for the debug, pre-release, and/or release stages of the application development cycle. Developers must understand how these capabilities and tools can be used together to speed the development, debug, tuning, and release of their Intel SGX-enabled applications.

References:

1. https://software.intel.com/en-us/blogs/2016/01/07/intel-sgx-debug-production-prelease-whats-the-difference — 2016 Intel Corporation
2. https://software.intel.com/en-us/documentation/sgx-developer-guide — 2016 Intel Corporation
3. https://software.intel.com/en-us/sgx-sdk/documentation — 2016/2017 Intel Corporation