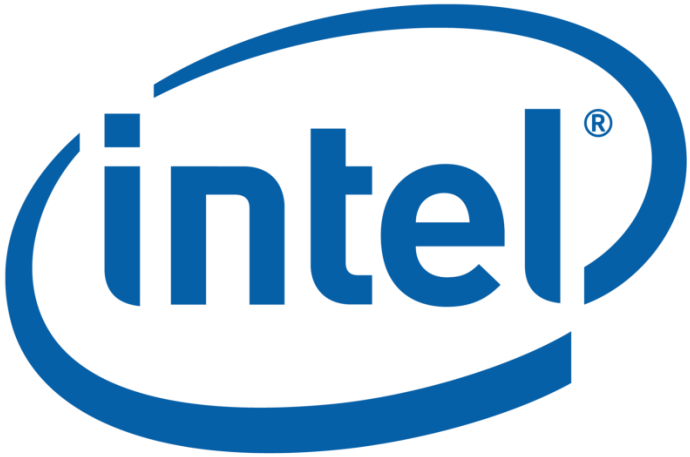

Software and Services Group

Intel® SgxSSL Library

User Guide

Revision History

Rev.	Description	Date
1.0	SgxSSL for the SGX SDK Windows 1.7 release	October 2016
1.1	Some fixes to the needed libraries and headers	November 2016
1.2	SgxSSL for the SGX SDK Windows 1.9 release. Remove duplicated symbols.	November 2017
1.2.1	Update OpenSSL version to 1.0.2o.	March 2018
1.2.1	Update OpenSSL version to 1.0.2p.	August 2018
1.2.2	Update OpenSSL version to 1.0.2q.	November 2018
1.2.3	Update OpenSSL version to 1.0.2r.	February 2019
1.2.4	Update OpenSSL version to 1.0.2s.	June 2019
1.2.5	Update OpenSSL version to 1.0.2t.	October 2019

Table of Contents

1. INTRODUCTION 4

1.1. General 4
1.2. Terminology 4
1.3. Legal considerations 4
1.4. Architecture Overview 5
1.5. Security Recommendations 6

2. RELEASE INFORMATION 7

2.1. Package Content 7
2.2. Usage 7
2.3. Additional SgxSSL APIs 9

3. SUPPORTED APIS 11

3.1. List of OpenSSL supported APIs 11
3.2. List of supported TLS1.2 cipher suites 13

4. RELEASE NOTES 15

4.1. Release Version "Intel® SgxSSL 1.9" 15

1. Introduction

1.1. General

The Intel® SgxSSL library is intended to provide cryptographic services, TLS, and certificate verification and identification services for SGX enclave applications.

The Intel® SgxSSL library is based on the underlying OpenSSL Open Source project, implementing the TLS protocol, certificate related services and providing a full-strength general purpose cryptography library.

The API exposed by the Intel® SgxSSL library is fully compliant with unmodified OpenSSL APIs. Note that only a specific subset of APIs available in OpenSSL is supported by the Intel® SgxSSL library. OpenSSL APIs that are not supported are nonetheless included in the Intel® SgxSSL library, but they are not validated and, it is not recommended to use them. The entire list of supported OpenSSL APIs can be found in the appropriate chapter below.

In addition, the Intel® SgxSSL library exposes a closed set of manageability APIs, a list of which is provided elsewhere in this document.

1.2. Terminology

Term	Description
SGX	Software Guard Extensions
TLS	Transport Layer Security
EDL	Enclave Definition Language. EDL files define the enclave interface for trusted-to-untrusted and untrusted-to-trusted transitions.
FIPS	Federal Information Processing Standards developed by NIST for use in computer systems government-wide
FIPS 140-2	Standard that defines security requirements for cryptographic modules and is required for sales to the Federal Governments.

1.3. Legal considerations

The Intel® SgxSSL Library is based on the OpenSSL* open source libraries licensed under a dual license, i.e. both the conditions of the OpenSSL license and the original SSLeay license apply.

More information regarding the OpenSSL license is available at https://www.openssl.org/source/license.html

You MUST be aware of license requirements and/or limitations of the underlying OpenSSL library and fully conform to it.

NOTE:

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (https://www/openssl.org)

1.4. Architecture Overview

The Intel® SgxSSL library consists of the following components:

- Intel® SgxSSL cryptographic and TLS libraries representing OpenSSL* libraries built to run inside an enclave.
- A trusted library providing implementation for missing system APIs inside an enclave.
- An untrusted library providing implementation of missing system APIs outside an enclave.

The following figure shows how Intel(R) SgxSSL library is used in an SGX application.

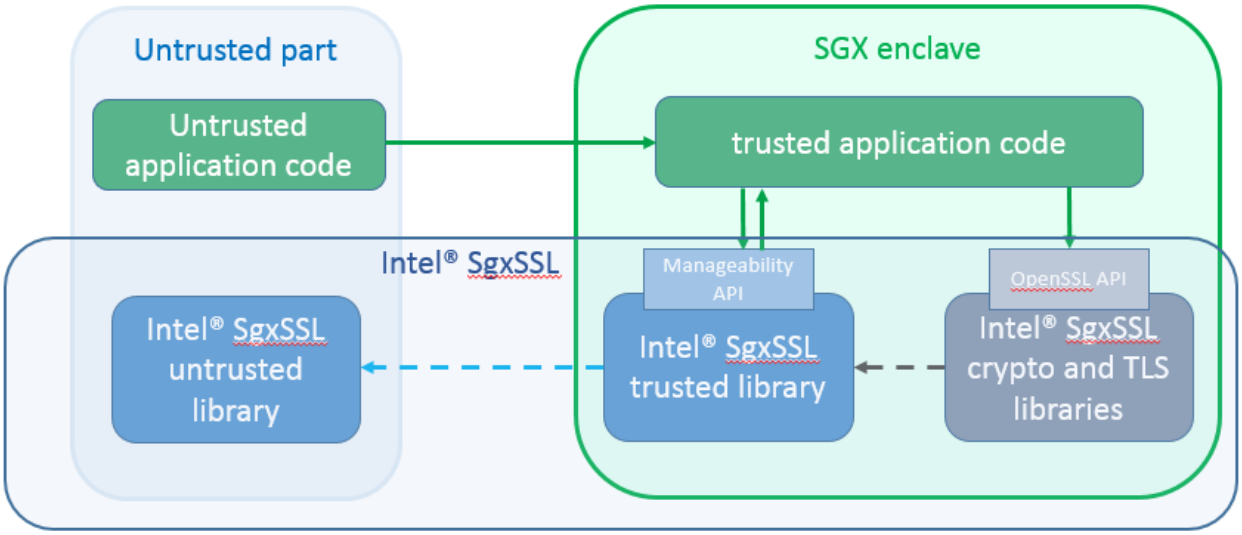

Figure 1 Intel® SgxSSL library Usage

Here is the flow of execution as illustrated in Figure 1:

1. The user's untrusted application code calls the trusted call with a function declared in the EDL file
2. The user's trusted code may use manageability API for different purposes, like to get Intel® SgxSSL library version, to register a callback function to intercept the Intel® SgxSSL cryptographic or TLS libraries printout messages, and so on.
3. The user's trusted code continues execution and at a certain point calls an API supported by the Intel® SgxSSL cryptographic or TLS libraries (the supported API is a subset of the unmodified OpenSSL API).
4. The call is passed to the Intel® SgxSSL cryptographic or TLS library. Some functions are internal and don't rely on system APIs (e.g. SHA256) so the functions complete and return.
5. Other functions require some system APIs, so the execution passes to the Intel® SgxSSL trusted library code that implements them. If the system API is simple and can be implemented internally (e.g. __gmtime64) it returns after completion without leaving the enclave.
6. Other APIs must leave the trusted code and are executed in the untrusted area (for example, send, recv, and so on.)

On an S3/S4 power event the internal state of the operation (for instance, during BASE64 encode/decode API usage) executed by the Intel(R) SGX SgxSSL library will be lost as part of the entire enclave loss. The Intel® SgxSSL library doesn't manage saving/restoring the state on suspend/resume operations. It is the customer's application responsibility to save an internal Intel® SgxSSL state on suspend and to restore it on resume when applicable.

1.5. Security Recommendations.

Intel® SgxSSL provides support for OpenSSL* inside an enclave. Security assets, like cryptographic keys, client certificate private keys, plain data (both network traffic and cryptographic payload) don't need to leave an enclave and thus they are protected by the Intel® SGX technology. The main security objective of the Intel® SgxSSL library is that integrity and confidentiality of security assets are protected from both malicious software and a simple hardware attack.

Intel® SgxSSL library relies on an implementation of OpenSSL and Intel® SGX to handle side channel attacks. Our architecture is designed to not leak additional information through the OCALLs, but it doesn't protect against side channel attacks. So, in case of side channel attacks it is as secure OpenSSL without Intel® SGX.

Note that getting current system time is not supported inside an enclave and is therefore implemented by Intel® Intel® SgxSSL library as OCALL. This approach allows an attacker to manipulate the time values coming from an untrusted component. Time values are used by Intel® SgxSSL library for time related certificate verification checks as well as for TLS session expiration checks. Assuming that the customer's enclave application utilizing Intel® SgxSSL library uses server certificate pinning, the risk of a time-based attack is considered low. So currently there is no additional mitigation for the threat, but it is recommended for customer SGX enclave application to implement server certificates pinning.

An enclave application, built with Intel® SgxSSL library, is responsible for preserving the security features of Intel® SgxSSL library. The security expectations of and recommendations for the customer enclave application are as follows:

- The customer's application is responsible to build production enclave as non-debug enclave.
- The customer's application should utilize Intel® SGX architecture and Intel® SGX SW to protect security assets. For instance, the customer's application should use certificate pinning. Server certificate should be securely provisioned into an enclave and protected by enclave sealing capabilities.
- The customer's application shouldn't expose security assets via trusted to untrusted transitions
- The customer's application should sanitize input data coming from untrusted components
- The customer's application should configure Intel® SgxSSL not to support obsolete protocols and cryptographic suites
- The customer's application should use only the OpenSSL APIs that are supported by the Intel® SgxSSL library.
- The customer's application should use OpenSSL APIs correctly to verify server certificate.

2. Release Information

2.1. Package Content

Starting Windows SDK 1.7 release, Intel® SgxSSL library is released as a zip package separated from the SGX SDK.

Our release package contains relevant include files (both header and edl files), libraries and relevant documentation.

Following is the list of libraries provided by our release package:

Library Name	Description
sgx_tfipscanister.lib	FIPS object module library built for Intel® SgxSSL* cryptographic library
sgx_tlibeayfips32.lib	FIPS capable Intel® SgxSSL* cryptographic library, built based on OpenSSL crypto library
sgx_tssleay32.lib	Intel® SgxSSL SSL/TLS library, built based on OpenSSL SSL/TLS library
sgx_tssl.lib	Trusted library, providing implementation for missing system APIs required by Intel® SgxSSL cryptographic and TLS libraries
sgx_ussl.lib	Untrusted library, providing implementation for system calls outside an enclave required to resolve external dependencies of Intel® SgxSSL* cryptographic and TLS libraries.

All the libraries are built for Windows Win32 and X64 configuration.

Intel® SgxSSL* cryptographic and TLS libraries are OpenSSL libraries built with a few changes needed to work inside an enclave.

2.2. Usage

In visual studio you will need to do the following (assuming you already have basic App + Enclave project):

- As a start, you may extract SGXSSL package to solution's directory. (You may also extract it into SGX SDK directory, or any other location, as long as you refer to the right location in projects' settings)
- In your EDL file add:
  from "sgx_tssl.edl" import *;
- Before using any OpenSSL API, you must include OpenSSL header that declares it. (e.g. <openssl\crypto.h>, <openssl\sha.h>, <openssl\ripemd.h>)
- Add "#include <windows.h>" before the "#include <openssl\xxx>" statements. The "windows.h" file is found in the SGX SSL include directory, and contains several definitions required by many of the OpenSSL headers.
- In the **Enclave** project (do these steps to all of your build environments):
  - Select **Properties->Linker->Input->Additional Dependencies**:
    Add "`sgx_tlibeayfips32.lib; sgx_tssleay32.lib; sgx_tssl.lib`";
    "`sgx_tfipscanister.lib`"

- Select **Properties->Linker->General->Additional Library Directories**:
  Add the folder where you placed the libraries, (you better use the built in macros like $(SolutionDir)$(Platform)\$(Configuration)\ etc. so you can control the different builds)
- Also add the path to the `sgx_tfipscanister.lib`, for instance: <path to the package>\lib\fipsopenssl\$(Platform);
- To add the folder where you placed the EDL file, right click your EDL file, then select **Properties->Custom Build Tool->Command Line**:
  Add the EDL file path to the `'--search path'` separated with ';'
- Select **Properties**->**C/C++**->**General**->**Additional Include Directories** and add the folder where Intel® SgxSSL header files are located. (`<path to the package>\include`)
- Select **Properties->Linker->All Options->Image Has Save Exception Handlers**:

  Choose No (/SAFESEH:NO)

- In the **Application** project, use the following steps to set up the environment for the Intel(R) SgxSSL library:
  - Select **Properties->Linker->Input->Additional Dependencies**:
    Add "`sgx_ussl.lib;Ws2_32.lib`"
  - Select **Properties->Linker->General->Additional Library Directories**:
    Add the folder where you placed the libraries (you better use the built in macros like $(SolutionDir)$(Platform)\$(Configuration)\ etc. so you can control the different builds)
  - To add the folder where you placed the EDL file, right click your EDL file, then select **Properties->Custom Build Tool->Command Line**:
    Add the EDL file path to the `'--search path'` separated with ';'
  - If your project does not use Intel compiler, add the path to the Intel compiler libraries through **Properties->Linker->General->Additional Library Directories**

Build with "FIPS capable OpenSSL"

- In the **Enclave** project (do these steps to all of your build environments):
  - In 32 bit configuration select **Properties->Linker->Advanced**
    - Set **Randomized Base Address** to No(/DYNAMICBASE:NO)
    - Set **Fixed Base Address** to Yes(/Fixed)
    - Set the **Base Address**. Note, SDK requires the Base Address to be enclave size aligned.

    Note, this step is mandatory for the FIPS runtime integrity check in Windows 32 bit configuration only. It is NOT mandatory in a 64-bit configuration, as 64-bit binaries do not contain any reallocations. So, the digest, calculated during the build time, will be the same as the digest calculated during the runtime.
  - Copy `<path_to_the_package>\lib\fipsopenssl\$(Platform)\fips_premain.c` into your enclave project and add the copied file into the enclave project files.
  - Build your project
  - Calculate FIPS module fingerprint and incorporate it in the enclave dll:

- Open VS Win32/X64 CMD:
  - Change directory to the `<path to the package>\util`
  - Calculate the hash value by running `` `perl <release_package>\util\msincore –dso <enclave_path>\<enclave_name>.dll ``

    Note, the hash value is calculated from FIPS module text and data areas. The calculated value is not impacted by the enclave signing.

  Note, instead of opening VS Win32/X64 CMD, you can calculate the HASH as Post-Build Event by adding an additional command to the **Properties->BuildEvent->CommandLine**. In this case the command will look like: `C:\Perl64\bin\perl.exe <path_to_the_package>\util\msincore -dso $(OutDir)\<enclave_name>.dll`

- Select **Properties->C/C++->Preprocessor**

  Add `HMAC_SHA1_SIG="<calculated_hash_value>"`

  For instance, if calculated hash value is 8fe5425f0c5b1d944b89c27fb4a9d1d19a2cb9cd then you should put it in “” to define HMAC_SHA1_SIG, like: HMAC_SHA1_SIG="8fe5425f0c5b1d944b89c27fb4a9d1d19a2cb9cd"

- Build your project again

- Please note that in the current SGX SDK, the ‘release’ mode does not generate the enclave.signed.dll but rather prepare a signing material (since it should be signed in a secure machine that protects the private key etc.). Enclaves signed with single-step signing method using ISV’s test private key can only be launched in’ debug’ or ‘prerelease’ modes.

2.3. Additional SgxSSL APIs

The Intel® SgxSSL Library exposes two different set of APIs:

1. Supported OpenSSL APIs - representing a subset of the OpenSSL APIs supported by the Intel® SgxOpenSSL library. They are fully compliant with unmodified OpenSSL APIs. Other APIs are neither validated, not filtered out. All supported OpenSSL APIs are listed below.
2. Manageability APIs are exposed by our trusted library to provide following services:

API	Description
`setPrintToStdoutStderrCB`	Set callback function to intercept printouts sent by Intel® SgxSSL cryptographic and TLS libraries to `stdout/stderr`. If not used, the printouts will be silently omitted.
`setProxyCertsPolicy`	Set proxy certificates policy. For regular OpenSSL this policy is provided via `OPENSSL_ALLOW_PROXY_CERTS` environment variable. Intel® SGX OpenSSL library will ignore value provided by this environment variable By default, proxy certificates are not allowed.
`getSgxSSLVersion`	Get the Intel® SgxSSL library version.

API	Description
`setUnreachableCodePolicy`	Set unreachable code policy. Unreachable code consists of functions and flows that under our implementation should never be reached. That is why, by default, reaching unreachable code will cause an enclave to be aborted.

3. Supported APIs

3.1. List of OpenSSL supported APIs.

The following APIs are supported:

Purpose	Type	OpenSSL APIs	Comment
Digest	SHA1, SHA256	SHA1_Init SHA1_Update SHA1_Final SHA256_Init SHA256_Update SHA256_Final EVP_sha256	
Key generation	RSA 2048	RSA_new RSA_free RSA_generate_key EVP_PKEY_assign EVP_PKEY_new EVP_PKEY_free PEM_read_PrivateKey	
Asymmetric Encryption	RSA PKCS #1	RSA_generate_key_ex RSA_public_encrypt RSA_private_decrypt	
Symmetric Encryption	3DES, AES128 ECB	EVP_CIPHER_CTX_init EVP_CipherInit_ex EVP_CipherUpdate EVP_CipherFinal_ex	
Sign and Verification	X.509 certificate(PKCS#7)	EVP_MD_CTX_init EVP_MD_CTX_create EVP_MD_CTX_destroy EVP_SignInit_ex EVP_SignUpdate EVP_SignFinal EVP_VerifyInit_ex EVP_VerifyUpdate EVP_VerifyFinal PKCS7_sign PKCS7_sign_add_signer	

		PKCS7_get_signer_info PKCS7_verify()	
Certificate signing request	X.509 certificate	X509_free X509_NAME_add_entry_by_txt X509_REQ_set_pubkey X509_REQ_new X509_REQ_sign X509_REQ_free PEM_read_bio_X509 PEM_write_bio_X509_REQ	
Certificate Revocation Lists (CRL)		X509_CRL_new X509_STORE_add_crl X509_CRL_sign X509_CRL_verify X509_CRL_free X509_verify_cert i2d_X509_CRL d2i_X509_CRL	
Secure transport (TLS 1.2)	TLS1_CK_RSA_WITH_AES_128_SHA TLS1_CK_RSA_WITH_AES_256_SHA TLS1_CK_RSA_WITH_AES_128_SHA256 TLS1_CK_RSA_WITH_AES_256_SHA256 TLS1_CK_RSA_WITH_AES_128_GCM_SHA256 TLS1_CK_RSA_WITH_AES_256_GCM_SHA384 TLS1_TXT_DHE_RSA_WITH_AES_256_SHA	SSLeay_version SSL_get1_session SSL_set_session SSL_get_verify_result SSL_CTX_ctrl SSL_state_string_long SSL_get_error SSL_CIPHER_get_name SSL_new TLSv1_2_client_method SSL_pending SSL_CTX_set_default_passwd_cb SSL_shutdown ERR_load_crypto_strings SSL_get_current_cipher OPENSSL_add_all_algorithms_noconf SSL_CTX_set_default_passwd_cb_userdata SSL_CTX_new SSL_write	

		SSL_free SSL_CTX_free SSL_read SSL_library_init SSL_CTX_use_PrivateKey SSL_alert_desc_string_long SSL_CTX_set_cipher_list SSL_set_fd CRYPTO_THREADID_set_numeric ERR_get_error CRYPTO_num_locks CRYPTO_set_locking_callback CRYPTO_THREADID_set_callback SSL_load_error_strings SSL_alert_type_string_long SSL_CTX_set_info_callback SSL_get_cipher_list SSL_connect SSL_CTX_set_verify SSL_CTX_use_certificate X509_STORE_add_cert	
Versioning	View underlying OpenSSL version details	SSLeay_version	
Assembly Optimizations	View and modify processor capabilities bit vector	OPENSSL_ia32cap_loc	Available since 1.1 (Beta)
Encode/Decode	Base64	EVP_EncodeInit EVP_EncodeUpdate EVP_EncodeFinal EVP_EncodeBlock EVP_DecodeInit EVP_DecodeUpdate EVP_DecodeFinal EVP_DecodeBlock	Available since 1.3 (post-Gold)
Random Number Generator	Add entropy to Random Number Generator	RAND_add RAND_seed	Available since 1.3 (post-Gold)
FIPS mode	FIPS mode	FIPS_mode_set FIPS_mode	Available since 1.7.1 (post SDK 1.6 Gold)

3.2. List of supported TLS1.2 cipher suites.

Following is the list of cipher suites of TLS1.2 using RSA key-exchange that are supported:

- TLS1_CK_RSA_WITH_AES_128_SHA
- TLS1_CK_RSA_WITH_AES_256_SHA
- TLS1_CK_RSA_WITH_AES_128_SHA256
- TLS1_CK_RSA_WITH_AES_256_SHA256
- TLS1_CK_RSA_WITH_AES_128_GCM_SHA256
- TLS1_CK_RSA_WITH_AES_256_GCM_SHA384
- TLS1_TXT_DHE_RSA_WITH_AES_256_SHA

4. Release Notes

4.1. Release Version "Intel® SgxSSL 1.9"

Release Date: <02/10/2019>
The release was built and validated using the SGX SDK 1.9.

Main Release features:

1. Using OpenSSL version 1.0.2s.
   Note: See *Supported APIs* chapter.
2. Using OpenSSL FIPS version 2.0.16.
3. This version of the Intel® SgxSSL was tested using Microsoft Visual Studio* 2015 with the Intel® C++ Compiler 16.0.