Remote Attestation End-to-End Sample Code

Installation and User Guide

June, 2016

Revision 1.0

World Wide Web: http://www.intel.com

Notices

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm.

Intel, the Intel logo, and Intel Core 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.

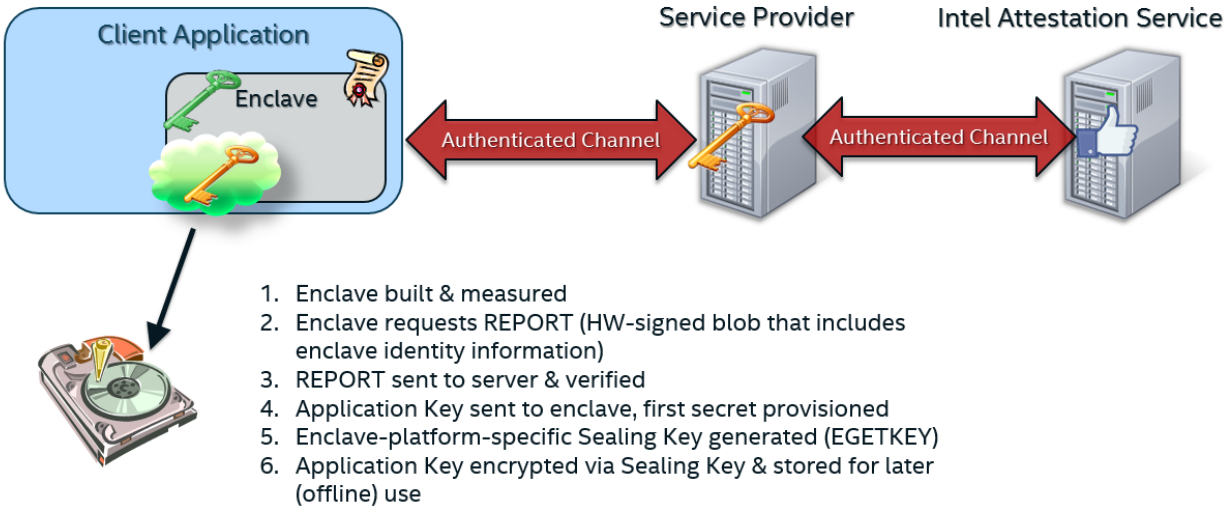
Contents

Overview

The purpose of the Remote Attestation (RA) Server Sample Code is to demonstrate how an SGX-enabled client can be initialized, attested, and securely provisioned with secrets. The sample is provided only as a proof of concept and not recommended for actual deployment without security reviews and other enhancements. Remote attestation involves three different elements:

1. **Client application**, running on the end-user SGX-enabled device, which includes an enclave initially deployed without secrets. In order for the secrets to be provisioned, the client application connects to its Service Provider and sends report information about its enclave to prove it is secure.
2. **Remote Attestation Server,** hosted by the Service Provider (an Independent Software Vendor (ISV) who developed the Client Application or is hosted by a third party by agreement with the ISV). It has the secrets which it will provision to the valid client enclave once it completes the attestation process. In order to verify that the client enclave is trustworthy, the Service Provider sends the client enclave report information to the IAS.
3. **Intel Attestation Service (IAS)**, available to be connected to via the internet, with its own dedicated URL. IAS checks the client enclave report, verifies that neither the client nor RA Server has been blacklisted, checks whether all components are up to date, and returns its findings to the Service Provider, who, if all was successful, is now able to trust the client and begin provisioning encrypted secrets.

The RA End-to-End Sample includes sample Client and Service Provider applications, and is able to connect to IAS to complete the attestation process. It completes all six of the below steps, including the final step of sealing the secret in an encrypted file for future use.



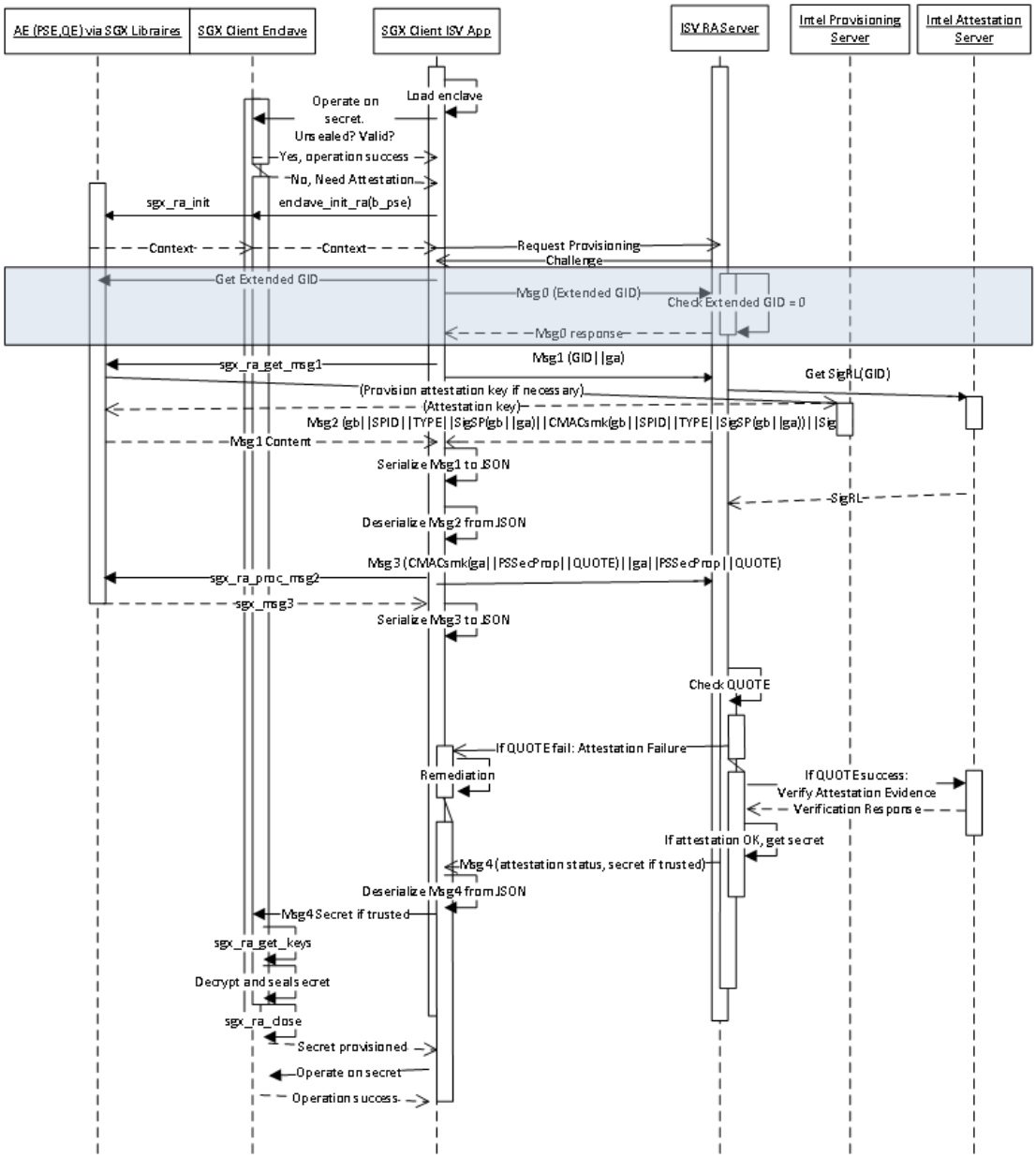
In real-world practice the Service Provider and client applications will run on different systems which connect via the internet. If the server is being run through Visual Studio with a fully-qualified domain name (instead of “localhost”) it must be started with administrator privileges, otherwise it may be run with standard privileges (see SPUri Setting below). The sample code is also designed to optionally run

both applications on the same system. If Service Provider and client are to run on the same system, dependencies for both applications are required to be on the same system:

- Service Provider: Intel® Performance Primitives (IPP)
- Client: SGX enabled system, Visual Studio 2012 or 2013 runtime libraries

The RA Server sample code has been tested on Windows 10. There are no known limitations to run on Windows 8.1.

The remote attestation process is based on the SIGMA protocol and follows the flowchart below. The Service Provider and client source code refers to this diagram when it references Msg0, Msg1, Msg2, Msg3, Msg4, etc. For more details, please refer to the SGX SDK User's Guide section on "Remote Attestation".



If critical errors are detected in the attestation process, exceptions are returned to the client that should be handled appropriately. Benign unsuccessful attestation sequences are completed with an appropriate response in Msg2 or Msg4.

Ultimately, the prime purpose of attestation is to determine whether a client enclave is trustworthy. If the client enclave also makes use of the Platform Services Enclave (PSE) for trusted services such as Trusted Time or Monotonic Counter, attestation can also be used to inform the client enclave whether the PSE is trustworthy.

Msg4 attestation status consists of 4 bytes: 1 byte for attestation result, and 3 bytes for lease duration (time that the secret will be valid – see LeaseDuration in Settings below). Attestation status has four potential statuses:

- `raTrustAll` = Enclave (and PSE if present) is trusted
- `raTrustEnclaveOnly` = The enclave but not the PSE is trusted
- `raTrustNone` = The enclave (and PSE if present) is not trusted
- `raTrustRetry` = Neither is trusted but a retry may still establish trust

The second purpose of attestation is to provision secret(s) if trust is established. If attestation is successful, Msg4 contains the payload which may include encrypted as well as clear content fields. Note that the clear content is used as Additional Authenticated Data (AAD) (i.e. not encrypted but authenticated) to encrypt the secret (if included in the payload) and later to decrypt the payload in the client.

As documented in IAS 1.0 API spec v1.1 section 5.2, a PIB (Platform Information Block) will be sent in Msg4 in the following scenarios:

1. isvEnclaveQuoteStatus is equal to GROUP_REVOKED or GROUP_OUT_OF_DATE **OR**
2. pseManifestStatus contains one of the following values: PSE_ISVSVN_OUT_OF_DATE, PS_HW_GID_REVOKED, PS_HW_PSDA_SVN_OUT_OF_DATE, PS_HW_SIG_Rlver_MISMATCH, PS_HW_PrivKey_Rlver_MISMATCH.

The PIB will be sent if the MSb of Byte 0 of Attestation Status is set. If the PIB is sent it should be checked in the client using sgx_report_attestation_status().

Remote Attestation Server

Installation

To install and run the Remote Attestation Server (RaSpRef), follow these steps:

1. Install Visual Studio 2012 or 2013. Note that if the system does not plan to run the client code, a newer version of Visual Studio should work fine as long as .NET Framework 4.6.1 or above is installed, in which case skip step 2.
2. Install .NET Framework 4.6.1 Developer Pack.
    a. Get it from: https://www.microsoft.com/en-us/download/details.aspx?id=49978
3. Install NuGet for Visual Studio to automatically detect required packages and download the dependencies from within the IDE.

   a. For Visual Studio 2012 get it from: https://visualstudiogallery.msdn.microsoft.com/27077b70-9dad-4c64-adcf-c7cf6bc9970c
   b. For Visual Studio 2013, get it from: https://visualstudiogallery.msdn.microsoft.com/4ec1526c-4a8c-4a84-b702-b21a8f5293ca
4. Install IPP 9.0.x
   a. To get a community license and to download, go to: https://registrationcenter.intel.com/en/forms/?productid=2558&licensetype=2
   b. Select "Community Licensing for Intel® Performance Libraries for Windows".
   c. Download w_ipp_9.0.x.yyy.exe.
   d. Run exe to install.
5. Install IPP-CP (crypto library).
   a. To get the package follow steps in http://registrationcenter-download.intel.com/akdlm/irc_nas/8292/get-ipp-90-crypto-library.htm
   b. Make sure that the build number matches the IPP version above. File name will be w_crypto_ipp_p_9.0.x.yyy.exe
   c. Run exe to install.
   d. More details about this process are here: https://software.intel.com/en-us/articles/download-ipp-cryptography-libraries
6. Set System Environment Variables to below values to enable IPP. Append the values to already-existing corresponding environment variables; otherwise create the variables and values.

Variable:	Value:
IPPROOT:	%ProgramFiles(x86)%\IntelSWTools\compilers_and_libraries\windows\ipp
CPATH:	%IPPROOT%\include
INCLUDE:	%IPPROOT%\include
MIC_LD_LIBRARY_PATH:	%IPPROOT%\lib\mic;
PATH	%IPPROOT%\..\redist\intel64\ipp;%IPPROOT%\..\redist\intel64\compiler;

7. Install IAS Certificate that was created and submitted when registering to use IAS. Be sure to install .pfx file – it contains the requisite private key, whereas the .cer file only contains a public key.
   a. If the .pfx file is available, double click on the file to run Certificate Import Wizard. Follow steps to complete installation. The Certificate should be installed in the Current User store, not the Local Machine store.
   b. If the .pfx file is not available, before doing the above step create a .pfx file from the .cer and .key files by using tools.
      i. One option is to install Microsoft Windows Driver Kit and use Pvk2Pfx (https://msdn.microsoft.com/en-us/library/windows/hardware/ff552958(v=vs.85).aspx)
      ii. Another option is to install OpenSSL (openssl.org) and run the following command:

      "openssl pkcs12 -export -out domain.name.pfx -inkey domain.name.key -in domain.name.crt"

Compilation

1. Unzip source code.
2. Open the Server solution by double-clicking on RaSpRef.sln in the Server directory.
3. Download Nuget packages:
    a. Right click “Solution ‘RaSpRef’”, then click “Enable Nuget Package Restore”.
    b. Right click Solution, then click “Manage NuGet Packages for Solution”. If this message appears: “Some NuGet packages are missing from this solution”, click on “Restore” to restore.
    c. Check that all packages are available by going to Solution Explorer, and expanding RaSpRef -> References. There should be no yellow warning signs.
4. Select Debug or Release, x64 configuration.
5. Build the code. Output will be in Server\SGX_Output\RaSpRef\bin\x64\Debug [or Release]
6. Troubleshooting:
    a. Compilation errors about 'RSASignaturePadding' or 'HashAlgorithmName': .NET 4.6.1 or greater needs to be installed before compilation. If .NET 4.6.1 or greater was installed later, select .NET 4.6.1 (or greater) by selecting RaSpRef -> right click -> Properties -> Application tab -> Target Framework
        i. If .NET 4.6.1 is not seen as an option after installation, install Visual Studio 2015 Community Edition with Update 1 instead. Get it here: https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx. Once the installation is complete make sure that the non-Community Edition of Visual Studio is what is selected for below compilation and execution.

Configuration

Application-Level Settings

Open the RaSpRef settings file by selecting, within Visual Studio, the RaSpRef project (not RaSpRef solution), Properties, and then Settings.settings. If running the executable without Visual Studio then modify the file RaSpRef.exe.config instead.

Settings are as follows:

Network configuration settings:

- SPUri (string): Http location of this RA Service Provider, including port number. (“localhost” may be used if client and server are both running on the same machine, otherwise use fully qualified domain name [e.g. http://<domain>:<port>/]). Pick a port number not in use.
- UseIAS (bool): Select “True” if actually connecting to IAS, otherwise this connection will be faked. Service Provider will always return with successful attestation and always share the secret data with the client enclave if set to “False”.
- IASUri (string): Http location of IAS server as provided to customer. (May be left blank if UseIAS = “False”.) As-is, this Service Provider sample code should only be used with the Development Services Environment IAS.
- IASProxyUri (string): The Proxy URI to use if a company firewall must be crossed in order to connect to IAS.

IAS connection parameters (all established at registration time):

- SPID (string): Service Provider ID for IAS access, obtained during IAS registration. Note that this is a 32-character string representing a hexadecimal number. This MUST have a valid value in order to connect to IAS. If UseIAS=False, a fake SPID value will be used if this is left blank.
- IASCertSubject (string): Certificate subject for linkable quotes. Locate this by clicking **Start**, typing **certmgr.msc** in the **Search bar** to see Certificates for Current User. You should find the certificate installed above by clicking to open these folders: "Certificates - Current User", "Personal", "Certificates". If the certificate is not here, reinstall it. Otherwise, double-click it to see more information, selecting Details tab, clicking on Subject. The full text of subject appears below, starting with "CN = ". Copy this string, place this string into the IASCertSubject field, and remove "CN = " and all leading whitespace. This MUST have a valid value in order to connect to IAS. If UseIAS=False, this may be left blank.
- LinkableQuotes (bool): Select "True" if linkable quotes will be sent to IAS using this SPID, or select "False" for unlinkable quotes. (Linkable quotes can be used to uniquely identify client enclaves via EPID B and EPID K values which will both always be the same for a particular client – note that this requires client opt-in privacy measures.)
- RKMod (string): Set to the modulus of the Report Key (RK). Note that this is a string representing a hexadecimal number. (Not expected to change, but if it does, this value can be obtained by using OpenSSL command: "openssl.exe rsa -in RK_pub.txt **-pubin** -inform PEM -text"
- RKExp (string): Set to the exponent of the Report Key. Note that this is a string with even number of characters representing a hexadecimal number. See RKMod for OpenSSL command.

Debug settings:

- LoggingEnabled (bool): Set "True" to turn on logging. Logging parameters are set in log4net.config.
- LoggingExceptionDetails (bool): Set "True" to log details when exceptions are hit.
- LogFileName (string): Log filename and path (relative to SGX_Output\RaSpRef\bin\x64\Debug [or Release]). Note that this can be overridden by command line parameter –lfn

Enclave Type-Specific Settings

One or more enclave types (each used by multiple enclaves on one or more computers) will be supported by this RA Server. The enclave type to be used for each RA session is denoted by a unique MRSIGNER and ISVPRODID. The full set of parameters for the enclave type is described in the JSON format, in EnclaveTypeList.json. Each supported enclave type is displayed here in a list, with the following values supported:

Client/enclave settings:

- Name (string): User-friendly name for the enclave.
- MRSIGNER (string): Unique string which identifies the enclave. Note that this is a string representing a hexadecimal number. To obtain the value for this enclave, copy and paste the hex value reported in run-time output line starting with "Quote came from enclave with MRSIGNER:" into the EnclaveTypeList.json settings file. MRSIGNER only changes when the signing key changes, for example when changing from a Debug enclave to Release enclave. The signing key file is Enclave_private.pem in the Client enclave build.
- ISVPRODID (ushort): ISV product ID of the Client enclave to be paired with. This value is used together with MRSIGNER to uniquely identify the enclave type. Set this value when building the Client enclave. Multiple enclave types can be signed with the same signing key by setting different ISVPRODID.
- IsProductionEnclave (bool): Set "True" if a production enclave (i.e. Non-debug) is being attested. If this expectation is incorrect, the sequence will be stopped. If this value is set "False" and the enclave is proven to be a debug enclave, a debug version of ISV key can be sent instead of a production version (Set the debug value to a different value than production ISV key by modifying isvDebugKey in ProcessMessage.cs).
- MRENCLAVE (string): Measurement of enclave expected to be reported in Msg3. Note that this is a string representing a hexadecimal number. This string will change every time the enclave source code is changed and rebuilt. To obtain the current value, copy and paste the hex value reported in run-time output line starting with "m3 quote measurement" into the settings file. To override and allow RA sequence to proceed regardless of measurement, leave this field blank or omit it from settings.
- ISVSVNMinLevel (ushort): Minimum acceptable level of ISV Security Version Number (SVN) as set in the paired enclave. This number should increase each time a critical security change is made to the enclave. The intention of this parameter is to disallow attacks made by hackers who become aware of vulnerabilities in older versions (by refusing attestation). If ISV SVN is a lower level than required, the sequence will be stopped. Set this value when building the client enclave.
- LeaseDuration (int): Time duration (in time scale denoted by LeaseDurationTimeUnit) that the encrypted payload sent in Msg4 (in case of a successful attestation) will be valid. This is to be enforced by the client enclave, which will refuse to use the secret after this time has expired. If the client enclave has no access to trusted time, it is expected that attestation must be repeated every time the secret is accessed. However, a lease duration of 0 will imply that the secret never expires or that a monotonic counter is to be used instead.
- LeaseDurationTimeUnit (string): Time units agreed upon by RA server and client for LeaseDuration. Acceptable units include "seconds", "minutes", "hours", "days", "months", and "years"

Attestation Policy settings (ISV is expected to choose the policy that fits the security need):

- TrustEnclaveGroupOutOfDate (bool): Select "True" if client enclave attestation is permitted even if the EPID Group ID is out of date. In this scenario, the EPID signature

of the ISV enclave QUOTE has been verified correctly, but the TCB level of SGX platform is outdated. The platform has not been identified as compromised and thus it is not revoked. It is up to the Service Provider to decide whether or not to trust the content of the QUOTE. See IAS 1.0 API Specification section 5.2.

- TrustPSEOutOfDate (bool): Select "True" if PSE attestation is permitted even if the PSE ISVSVN or a specific SW component involved in the SGX Platform Service is out of date. In this scenario, the TCB level of SGX Platform Service is outdated but the Service has not been identified as compromised and thus it is not revoked. It is up to the Service Provider to decide whether or not to assume the SGX Platform Service utilized by the ISV enclave is valid. See IAS 1.0 API Specification section 5.2 for details about PSE Out of Date scenarios.

Payload settings:

Payload settings have their own subsection in the JSON file. The ISV payload will be determined by Encrypt and Clear settings. (If Encrypt and/or Clear settings are also missing, the Encrypted and/or Clear payload will not be sent.) If no settings are set, no payload will be sent.

To generate a payload, choose one or both of the following:

1. Encrypt (string): Set to the filename and path (relative to SGX_Output\RaSpRef\bin\x64\Debug [or Release]) of binary content to be encrypted, or for keyword "Random", a 32byte random value will be encrypted and sent.
2. Clear (string): Set to the filename and path (relative to SGX_Output\RaSpRef\bin\x64\Debug [or Release]) of binary content to be sent in the clear, or for keyword "Random", a 32byte random value will be sent in the clear. This value can be used with the encryption of the payload as Additional Authenticated Data (AAD) for integrity checks, enabling the enclave to trust these values have not been modified since the ISV RA Server sent them.

Logging Configuration

Log4net is used for logging and configured with the configuration file log4net.config. Consult log4net documentation on the internet for more details about the options specified in the log4net configuration file. At the time of this writing, online documentation can be found at http://logging.apache.org/log4net/. This site is not maintained by Intel Corporation and may be moved, modified, or removed without further notice.

For the -llc and -llf command line options to work, they assume that the root logger is configured with the appropriate Appenders. The -llf option requires at least one Appender of type FileAppender or its derivatives (e.g. RollingFileAppender). The -llc option requires at least one Appender that is one of ConsoleAppender, ColoredConsoleAppender, ManagedColoredConsoleAppender, or their derivatives. These options will adjust the corresponding log level of their corresponding Appenders configured within the root logger, by manipulating the root logger log level, and the threshold settings for the

various Appenders. An unusually complex or unusual log4net configuration might not be compatible with these command line options.

The -lfn command line option configures a property named “LogFileName”. The log4net configuration must use this property for this command line option to be effective.

An example log4net configuration pattern that is compatible with these command line options (-llc, -llf, and -lfn) is as follows:

```
<log4net>
   <appender name=”FileAppender” type=”log4net.Appender.FileAppender”>
      <threshold value=”ALL” />
      <file type="log4net.Util.PatternString" value="%property{LogFileName}" />
      <!-- Additional parameters for this appender may be included.  -->
   </appender>
   <appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">
      <threshold value=”ALL” />
      <!-- Additional parameters for this appender may be included.  -->
   </appender>
   <root>
      <level value=”INFO” />
      <appender-ref ref="FileAppender" />
      <appender-ref ref="ConsoleAppender" />
   </root>
</log4net>
```

The following parameters are made available to the log4net configuration. Consult log4net documentation related to log4net.Util.PatternString for information about how to use them.

- FooterRule: a horizontal rule that might be useful in a <footer>.
- HeaderArgs: a list of command line options passed to the application.
- HeaderCopyright: the application’s copyright message.
- HeaderFrameworkName: the .NET framework name and version
- HeaderParameters: a multi-line list of various application parameters.
- HeaderProductName: the application’s product name
- HeaderRule: a horizontal rule that might be useful in a <header>
- HeaderSimpleName: the application’s simple name
- HeaderVersion: the applications version number
- LogFileName: the full log file path and name

Execution

1. Executable is found from the root directory in: SGX_Output\RaSpRef\bin\x64\Debug [or Release]
2. RaSpRef can be run either from Command Line or from Visual Studio console.
3. Command line options – note that these are all optional and are case sensitive:
    a. “–l” on or “–l” off: enable/disable logging (can be used to override setting LoggingEnabled)

   b. "–lxd" on or "–lxd" off: enable/disable detailed log for exception (can be used to override setting LoggingExceptionDetails)
   c. "–lfn <path>": filename and path to log file (e.g. "-lfn `logs\RaSpRef.log`"). This overrides the setting LogFileName if it exists.
   d. "–llc <level>": set log level for console output, overriding default. By default, if logging is enabled, INFO level output will be sent to the console, otherwise no output will appear on the console. Default can be changed in log4net.config.
   e. "–llf <level>": set log level for file output appended to `logfile`, overriding the default. By default, if logging is enabled, all output will be sent to the log file. Defaults can be changed in log4net.config.
   f. For "d" and "e", <level> can be one of the following. Setting the level enables all output below this level (e.g. –llc info will log information, warnings and all errors, but not debug messages):
      - all - enable all logging
      - debug - debug messages
      - info - informational messages
      - warn - warnings
      - error - general errors
      - fatal - fatal errors
      - off - disable all logging
4. Start RaSpRef before running the client application.
5. Notes:
   a. Program works with a single client at a time and resets when the sequence ends. It should not be necessary to stop RaSpRef in order to run Client code multiple times.
   b. It is recommended to use a two-step process to get the full interaction working:
      i. Set UseIAS to "False" to first test the interaction between Client and Service Provider, before testing with IAS server. The attestation process will be faked and always succeed. Be sure to set SPUri correctly for a successful connection to your client.
      ii. Once this is working, set UseIAS to "True" in order to actually connect to IAS. Note that LinkableQuotes, IASCertSubject, SPID, InFirewall, and ProxyUri settings need to be set correctly for this second step. IASUri, RKMod, and RKExp also need to be set – these are pre-set to default Development Services Environment IAS settings.
   c. Items for the ISV to consider in making a product-ready implementation are marked with "NOTE" in the code.
6. Troubleshooting:
   a. Program hangs after receiving Msg1 but before Msg2 is sent to client - likely causes:
      i. Server issues:
         1. IPP environment variables (or IPP itself) were not set up correctly. Test IPP setup by running IPP sample apps.
         2. Debug DLLs copied to test device without dependent libraries. Resolution: build the executables on the test machine itself.

3. Bad Request response from IAS: certificate may not be valid or installed correctly. Try using a different certificate/SPID.

ii. Client issues:

1. BIOS or drivers are not set correctly for SGX. Check that other SGX sample applications work correctly.
2. Usage of PSE requires the Intel® Management Engine to be installed, specifically the presence of iCLS driver. If iCLS is installed the following directory will be present: "C:\Program Files\Intel\iCLS Client". If it is missing, install it or set "b_pseFlag: false" in client settings file App.config.txt.
3. EPID provisioning was unsuccessful. This may occur if the SGX system is not connected to the internet or if computer proxy is not set up correctly. If all is well, the "Remote Attestation" SDK sample should run without failure. Note that this is a rare event and should not be an issue after it is done successfully once.

b. Connectivity: make sure that client and server successfully complete attestation flow when running on the same system (default setting) before running on separate systems.
c. Most other issues will be resolved by setting all Settings correctly. Defaults have been set to values that should work out of the box, but these will need to be changed to meet needs. (Check LinkableQuotes is correct for your SPID. Check EnclaveTypeList.json settings for your enclave type being used.)
d. Verify application dependencies are installed: NET Framework 4.6.1, IPP & IPP-CP. Also check client dependencies (may be running on different PC).
e. Verify environment variables for IPP: see Installation, section 6 above.

Remote Attestation Client

Details

The Client demonstrates how attestation might be implemented in an actual client. It focuses on the enclave which securely obtains and uses a secret which is provisioned by the RA server. If the secret is available it is used to encrypt and decrypt any phrase (limited to 100 characters) as typed by the user.

The secret as well as a "Secret Expiration Time" are obtained when completing the attestation process. The "Secret Expiration Time" is computed by adding to the current time, the LeaseDuration (time in seconds that the secret is valid) obtained from the server. A secure timestamp is obtained from the PSE's Trusted Time, which is insured to be not tampered with. The secret and secret expiration time are then sealed and stored in a file for future use and should still be available the next time the program is run. However, when the lease time is expired, the secret is no longer valid, and the next time the secret needs to be operated upon, the enclave requires that the client undergo remote attestation again in order to obtain a new lease (and potentially a new secret as well). If remote attestation fails, the program exits since it is unable to operate on the secret.

The enclave can either be:

1. A service that is initialized at program startup and always runs. Select by setting keepEnclave to true. Note that a permanent resident enclave will take up system memory resources, so in an actual implementation, this option should be carefully considered.

OR

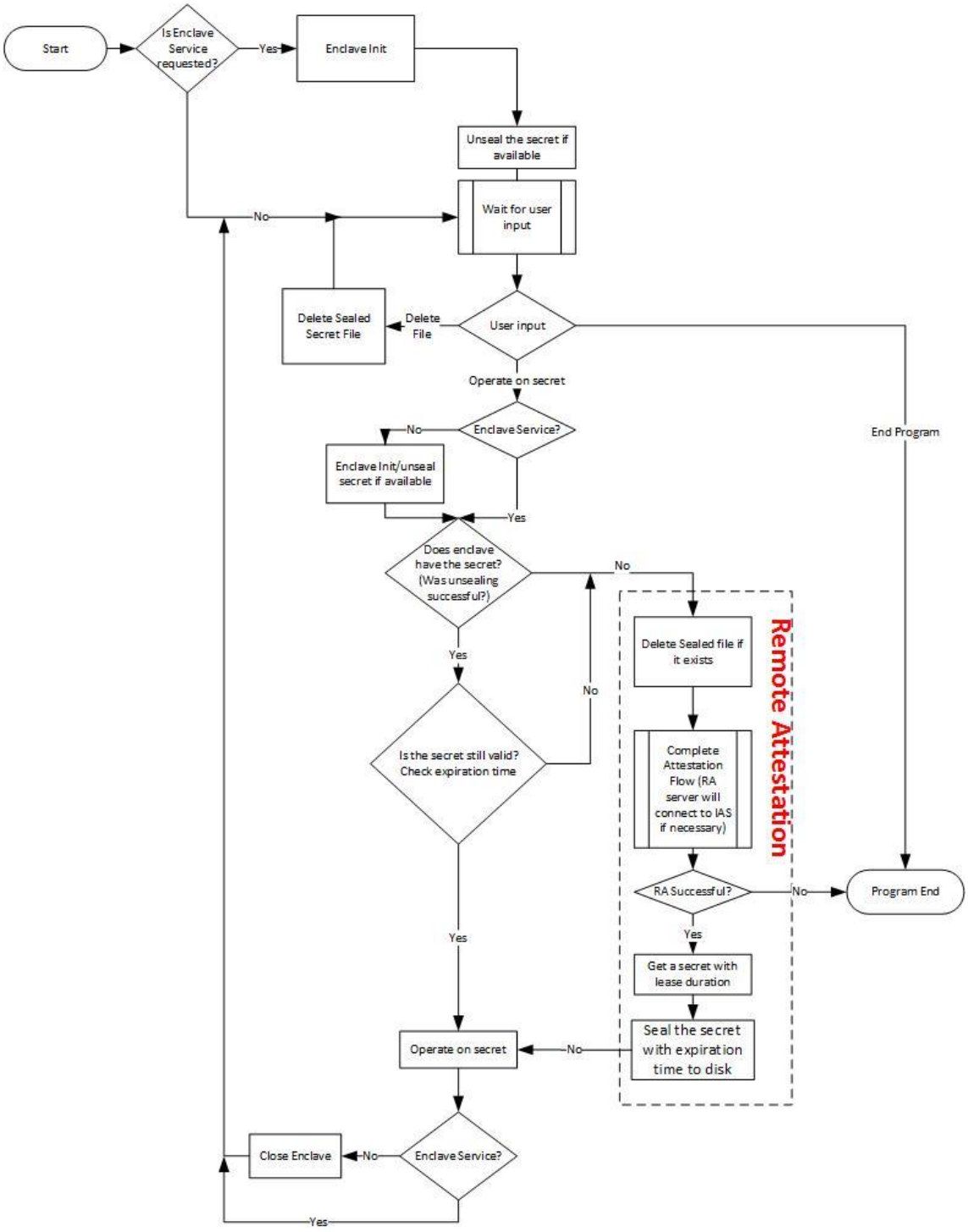
2. Enclave is initialized only when access to the secret is needed. It will be closed once the operation on the secret is completed. (As noted above, this may require attestation to take place if a valid secret is not currently available.)

This application uses Platform Services Enclave (PSE) to access Trusted Time to ensure that the secret's LeaseDuration requirement is met. If PSE is not present or is found to be not trusted, the secret will not be sealed (i.e. will be lost when the program ends) and Client Enclave will allow only one EncryptDecrypt call to be made. A second call to EncryptDecrypt will force Remote Attestation again.

The following measures are taken to protect the secret once obtained:

1. Sealing is done immediately after attestation to protect against program shutdown and power events.
2. The sealed secret is placed in a read-only protected file.

The flow below is followed by the Client in this RA End-to-End Sample:



Prerequisites

The Remote Attestation Client runs on a PC system that supports SGX. SGX supported platforms include 6th Gen Core processors and supported BIOS.

BIOS settings must be set to enable SGX. If you have questions about this, please contact your Intel representative.

Installation

To install and run Remote Attestation Client (RaClientRef) on a Win8/8.1 or Win10 PC, follow these steps:

1. Install Microsoft Visual Studio 2012 or 2013 with C/C++ support component
    - This build is tested with Visual Studio Premium and Professional editions
    - No special settings are required (all may be unchecked)
    - (Note: Visual Studio Express Edition does not work)
2. Install SGX SDK and SGX Platform Software (PSW)
    - Latest version of SGX SDK for Windows can be found at https://software.intel.com/sgx-sdk
    - Install Intel® SGX SDK from the SDK folder.
    - Install Intel® SGX PSW from the PSW folder. To install PSW, open command prompt in admin mode and run the command:
      msiexec /i psw.exe
3. Install C++ REST SDK (Casablanca)
    - Go to http://casablanca.codeplex.com/releases/view/114888
    - Download and install C++ REST SDK for Visual Studio 2012 or 2013 (version 1.3.1)

Compilation

1. Browse to Client directory.
2. Open the solution by double-clicking on RaClientRef.sln in the root directory.
3. Select Debug x64 configuration. Release mode should only be used later.
4. Set appropriate settings. Open the settings file App.config.txt and configure the required settings.
5. Build the code.
6. Troubleshooting:
    a. If you see "Microsoft.CppCommon.targets" error during compiling, try reinstalling the SGX SDK and PSW.
    b. If enclave creation fails with error code 0x200f: in the RaClientRef solution, go to ISV-APP Property Pages | Configuration Properties | Debugging, change the Working Directory from $(ProjectDir) to $(OutDir)
    c. pdb error failure: go to Tools | Options| Debugging| Symbols and in Symbol File(.pdb) Locations, check the Microsoft Symbol Servers checkbox
7. The code may also be built and run in Prerelease or Release mode. Differences between modes:
    a. "Debug": complier optimization disabled, debug symbols are available. Suitable for source level debugging. The enclave will be launched in enclave-debug mode
    b. "Release": compiler optimization enabled, no debug symbols, suitable for production built, for performance testing and final product release (typical for any SW development, standard terminology of common IDEs). The enclave will be launched in enclave-non-debug mode, which requires the enclave to be whitelisted in order to run.
    c. "Prerelease": same as "Release" with regard to optimization and debug symbols, but the enclave will be launched in enclave-debug mode, which doesn't require the enclave to be whitelisted and is suitable for performance testing. This is a new profile added by the SDK Visual Studio plugin.
        i. To build and run the project in Prerelease mode.

1. Go to C++ REST SDK installation directory, by default at "C:\Program Files (x86)\Microsoft Cpp REST SDK for VS 201[2/3] v1.3\SDK\bin\x64"
2. Create a folder name Prerelease
3. Copy all the contents of Release folder to Prerelease folder
4. Change VC++ directories in the properties (if necessary)

Configuration

To configure the parameters, open the App.config.txt file in the RaClientRef\ISV-APP and make the necessary changes if any:

- SPUri –URL of the RA Service Provider which the client should connect to. Ex: SPUri: http://localhost:25711/ (or http://<domain>:<port>/). Remember: If the server is being run through Visual Studio with a fully-qualified domain name (instead of "localhost"), it must be started with administrator privileges, otherwise it may be run with standard privileges.
- b_pseFlag – A Boolean flag indicating whether the app/enclave uses Platform Services Enclave. True indicates Platform Services are used and False indicates Platform Services are not used. Acceptable values are true, false. Ex: b_pseFlag: true. To use PSE, make sure Intel® Management Engine is installed, including the iCLS driver. The following directory should exist: C:\Program Files\Intel\iCLS Client
- verboseFlag – A Boolean flag to set the verbose option. A "true" will enable the verbose option and a "false" disables the verbose option. Acceptable values are true, false. Ex: verboseFlag: true
- keepEnclave - A Boolean flag indicating whether an enclave is to be initialized at program startup, not to be closed until program completion.

Next, go to the RA Server application and check the EnclaveTypeList.json settings describing the client enclave. (See details in Server Configuration section above). These have been pre-set to working values but may be changed to meet particular needs. In particular, note that MRSIGNER and ISVPRODID have been set to define this client enclave. LeaseDurationTimeUnit should be set to "seconds" for our client.

To lock down a particular enclave build even during debugging, set MRENCLAVE as noted above. (Typically this can be left blank or omitted to allow small changes in the enclave, especially for enclaves being debugged).

Execution

1. Run the Client code after the RA Server is started.
2. Type in a message (under 100 bytes) and press enter to see it encrypted and decrypted with the key provisioned from the server. The first time the program is run, the secret is not yet provisioned so Remote Attestation will take place. Then the encryption/decryption operation should be successful.
3. The secret will be sealed to a file (if PSE is enabled), so each time encryption/decryption operation is repeated, no remote attestation is needed until LeaseDuration expires. Even exiting and restarting the program will not require remote attestation. To invalidate the secret, press "/" to delete the sealed file or set LeaseDuration to a sufficiently small value in the server settings that the secret will

quickly become invalid. Then Remote Attestation will again take place and the secret once again sealed and stored in a file.

4. Press “*” to exit
5. Troubleshooting:
    a. If the client fails during Msg1 generation, check the b_pseFlag value in the app.config.txt. b_pseFlag=true causes failure in PSE instantiation if the SGX drivers are not up to date on the machine or Intel® ME is not installed