All information herein is either public information or is the property of and owned solely by Gemalto NV. and/or its subsidiaries who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Gemalto’s information.

This document can be used for informational, non-commercial, internal and personal use only provided that:

- The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- This document shall not be posted on any network computer or broadcast in any media and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided “AS IS” without any warranty of any kind. Unless otherwise expressly agreed in writing, Gemalto makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Gemalto reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Gemalto hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Gemalto be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Gemalto does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms’ resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemalto be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemalto products. Gemalto disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

© 2016 Gemalto. All rights reserved. Gemalto and the Gemalto logo are trademarks and service marks of Gemalto N.V. and/or its subsidiaries and are registered in certain countries. All other trademarks and service marks, whether registered or not in specific countries, are the property of their respective owners.

**Document Part Number:** 007-012068-001, Rev. C

**Release Date:** April 2016
## Contents

Preface .................................................................................................................................. 4  
Scope .................................................................................................................................. 4  
Document Conventions ........................................................................................................ 4  
  Command Syntax and Typeface Conventions ................................................................ 4  
  Support Contacts .......................................................................................................... 6  

1 Introduction ...................................................................................................................... 7  
  Overview ....................................................................................................................... 7  
    3rd Party Application Details ....................................................................................... 7  
    Supported Platforms ................................................................................................. 7  
  Prerequisites .................................................................................................................. 8  
    SafeNet Network HSM Setup .................................................................................... 8  
    SafeNet Luna HSM Configuration Settings ............................................................... 8  

2 Integrating SafeNet Luna HSM with Open SSL Using LunaCA3 Engine ....................... 10  
  Before You Begin ......................................................................................................... 10  
  Open SSL Toolkit .......................................................................................................... 10  
  Installation and Configuration of Open SSL Toolkit to use Luna CA3 Engine on UNIX ................................................................................................................... 10  
    Install and Configure the Open SSL toolkit ............................................................ 10  
    Verify Open SSL and Luna CA3 engine Integration ................................................. 13  
  Installation and configuration of Open SSL toolkit to use Luna CA3 engine on Microsoft Windows ................................................................................................. 16  
    Install and Configure the Open SSL toolkit ............................................................ 16  
    Verify Open SSL and Luna CA3 engine Integration ................................................. 19  

3 Integrating SafeNet Luna HSM with Open SSL Using Gem Engine.............................. 28  
  Before You Begin ......................................................................................................... 28  
  Open SSL Toolkit .......................................................................................................... 28  
  Installation and Configuration of Open SSL Toolkit to use Gem Engine on UNIX ........... 28  
    Install and Configure the Open SSL toolkit ............................................................ 28  
    Verify Open SSL and Gem engine Integration ......................................................... 32  
  Installation and configuration of Open SSL toolkit to use Gem engine on Microsoft Windows ......................................................................................................... 35  
    Install and Configure the Open SSL toolkit ............................................................ 35  
    Verify Open SSL and Gem engine Integration ......................................................... 38  

4 Troubleshooting Tips ...................................................................................................... 47
Preface

This document is intended to guide security administrators to install, configure, and integrate Open SSL with SafeNet Luna Hardware Security Module (HSM).

Scope

This document outlines the steps to configure and integrate SafeNet Luna HSM with the Open SSL toolkit that have already supports the LunaCA3/Gem engine for cryptographic operations. It describes some of the commands of Open SSL to show how to use LunaCA3/Gem engine with Open SSL commands.

Document Conventions

This section provides information on the conventions used in this template.

Notes

Notes are used to alert you to important or helpful information. These elements use the following format:

NOTE: Take note. Contains important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss. These elements use the following format:

CAUTION: Exercise caution. Caution alerts contain important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury. These elements use the following format:

WARNING: Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury.
## Command Syntax and Typeface Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
</table>
| **bold**   | The bold attribute is used to indicate the following:  
Command-line commands and options (Type `dir /p`.)  
Button names (Click **Save As**.)  
Check box and radio button names (Select the **Print Duplex** check box.)  
Window titles (On the **Protect Document** window, click **Yes**.)  
Field names (**User Name**: Enter the name of the user.)  
Menu names (On the **File** menu, click **Save**.) (Click **Menu > Go To > Folders**.)  
User input (In the **Date** box, type **April 1**.) |
| **italic** | The italic attribute is used for emphasis or to indicate a related document. (See the **Installation Guide** for more information.) |
| Consolas   | Denotes syntax, prompts, and code examples. |
Support Contacts

If you encounter a problem while installing, registering or operating this product, please make sure that you have read the documentation. If you cannot resolve the issue, contact your supplier or Gemalto Customer Support. Gemalto Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Gemalto and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

<table>
<thead>
<tr>
<th>Contact Method</th>
<th>Contact Information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Address</strong></td>
<td>Gemalto, Inc.</td>
</tr>
<tr>
<td></td>
<td>4690 Millennium Drive</td>
</tr>
<tr>
<td></td>
<td>Belcamp, Maryland  21017, USA</td>
</tr>
<tr>
<td><strong>Phone</strong></td>
<td></td>
</tr>
<tr>
<td>US</td>
<td>1-800-545-6608</td>
</tr>
<tr>
<td>International</td>
<td>1-410-931-7520</td>
</tr>
<tr>
<td><strong>Technical Support</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Customer Portal</strong></td>
<td><a href="https://serviceportal.safenet-inc.com">https://serviceportal.safenet-inc.com</a></td>
</tr>
</tbody>
</table>
| Existing customers with a Technical Support Customer Portal account can log in to manage incidents, get the latest software upgrades, and access the Gemalto Knowledge Base.
Overview

The Open SSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library managed by a worldwide community of volunteers that use the Internet to communicate, plan, and develop the Open SSL toolkit and its related documentation. Open SSL is based on the excellent SSLeay library developed by Eric A. Young and Tim J. Hudson. The Open SSL toolkit is licensed under an Apache-style license, which basically means that you are free to get and use it for commercial and non-commercial purposes subject to some simple license conditions.

Open SSL will use the LunaCA3/Gem Engine to store and use the SSL keys on SafeNet Luna HSM.

3rd Party Application Details

Open SSL is a popular Open Source implementation of the SSL/TLS protocols. The project is managed by a worldwide community of volunteers. Open SSL is the only free, full-featured SSL implementation currently available for use with the C and C++ programming languages. It works across most major platforms, including Microsoft Windows and Unix operating systems.

Supported Platforms

Open SSL v1.0.1

<table>
<thead>
<tr>
<th>Platforms Tested</th>
<th>Luna Client Software Version</th>
<th>SafeNet Luna HSM Appliance Software Version</th>
<th>Appliance Firmware Version</th>
<th>Open SSL Toolkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red Hat Enterprise Linux 6.5 (64 bit)</td>
<td>Luna Client 6.2.0</td>
<td>SafeNet Luna HSM v6.2.0</td>
<td>6.10.9</td>
<td>610-012987-002_SW_OPENSSL_TOOLKIT_v1.1_RevA</td>
</tr>
<tr>
<td>Windows Server 2008 R2</td>
<td>Luna Client 6.2.0</td>
<td>SafeNet Luna HSM v6.2.0</td>
<td>6.10.9</td>
<td>610-012987-002_SW_OPENSSL_TOOLKIT_v1.1_RevA</td>
</tr>
<tr>
<td>Red Hat Enterprise Linux 6.4 (64 bit)</td>
<td>Luna Client 5.4.1</td>
<td>SafeNet Luna HSM v5.4.1</td>
<td>6.21.0</td>
<td>Apache Toolkit (630-010135-001_Alpha8)</td>
</tr>
</tbody>
</table>
Open SSL v1.0.0

<table>
<thead>
<tr>
<th>Platforms Tested</th>
<th>Luna Client Software Version</th>
<th>SafeNet Luna HSM Appliance Software Version</th>
<th>Appliance Firmware Version</th>
<th>Open SSL Toolkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows Server 2008 R2</td>
<td>Luna Client 5.1</td>
<td>SafeNet Luna HSM v5.1</td>
<td>6.2.1</td>
<td>Apache Toolkit (630-010135-001_Alpha3)</td>
</tr>
</tbody>
</table>

Prerequisites

SafeNet Network HSM Setup

Refer to the SafeNet Network HSM documentation for installation steps and details regarding the configuration and setup of the box on Windows/Unix systems. Before you get started, ensure the following:

- SafeNet Network HSM appliance and a secure admin password.
- SafeNet Network HSM, and a hostname, suitable for your network.
- SafeNet Network HSM network parameters are set to work with your network.
- Initialize the HSM on the SafeNet Network HSM appliance.
- Create and exchange certificates between the SafeNet Network HSM and your Client system.
- Create a partition on the HSM, remember the partition password that will be later used by Open SSL engine.
- Register the Client with the partition. And run the "vtl verify" command on the client system to display a partition from SafeNet Network HSM. The general form of command is "C:\Program Files\SafeNet\LunaClient> vtl verify".
- Enabled Partition "Activation" and "Auto Activation" (Partition policy settings 22 and 23 (applies to SafeNet Network HSM with Trusted Path Authentication [which is FIPS 140-2 level 3] only).

SafeNet Luna HSM Configuration Settings

The Luna Client configuration file located at the following path needs to be changed for Luna Client v6.x:

/etc/Chrystoki.conf

This configuration file needs to be edited for slot id because by default it is set to 0. Set the slot id to 1 by making the following changes in the configuration file:

```
Presentation = {
    OneBaseSlotId = 1;
}
```

If using SafeNet Luna HSM f/w 6.22.0 or above is in FIPS mode:

Under FIPS 186-3/4, the RSA methods permitted for generating keys are 186-3 with primes and 186-3 with aux primes. This means that RSA PKCS and X9.31 key generation is no longer approved for operation in a FIPS-
compliant HSM. If you are using the SafeNet Luna HSM in FIPS mode, you have to make the following change in configuration file:

```
Misc = {
    RSAKeyGenMechRemap = 1;
}
```

The above setting redirects the older calling mechanism to a new approved mechanism when SafeNet Luna HSM is in FIPS mode.

**NOTE:** SafeNet Luna HSM Configuration Settings are required only for Luna HSM f/w 6.22.0 or above. All other SafeNet Luna HSM versions do not require these setting for FIPS and non-FIPS mode.
Integrating SafeNet Luna HSM with Open SSL Using LunaCA3 Engine

Before You Begin

It is recommended that you should familiarize yourself with Open SSL. Refer to the appropriate documents for Open SSL commands at the following location:

http://www.openssl.org/docs/

Open SSL provides the support of engine (basically hardware devices) to store the keys on hardware devices to make keys more secure. SafeNet provides the Open SSL toolkit having support of Luna CA3 Engine that is used to communicate with the SafeNet Luna HSM.

Open SSL Toolkit

The Open SSL toolkit is provided with the Apache Toolkit to make the installation quick and easy. The installation CD can be obtained from the SafeNet Customer Support.

Installation and Configuration of Open SSL Toolkit to use Luna CA3 Engine on UNIX

Install and Configure the Open SSL toolkit

Case 1: If Open SSL not installed on the system.

1. Obtain latest Apache Toolkit from SafeNet Support.
2. Traverse to the toolkit: /root/_cdrom_apache, run the configuration script (abuild-dynamic) to install openssl for SafeNet Luna HSM.

   Usage:
   
   LUNA_CONFIG_BITS=[32|64|LEGACY] ./abuild-dynamic [option]

   Options:

   --openssl-without-fips   build and install openssl without fips.
   --fips-module            build and install the openssl fips module only.
   --openssl-with-fips      build and install openssl for use with fips module.
Example:

Build OpenSSL without FIPS mode (64bit Linux).

    # LUNA_CONFIG_BITS=64. /abuild-dynamic --openssl-without-fips

For further information, refer to the README-ABUILD-DYNAMIC under the APACHE toolkit.

3. Verify the luna engine is present.

    # /usr/local/ssl/bin/openssl engine -v LunaCA3
    (LunaCA3) Luna CA3 engine support
      enginearg, openSession, closeSession, login, logout, engineinit, CONF_PATH, ENGINE_INIT,
      ENGINE2_INIT, engine2init, DisableCheckFinalize, SO_PATH, GET_HA_STATE, SET_FINALIZE_PENDING,
      SKIP_C_INITIALIZE

Case 2: If Open SSL already installed. We need to load the dynamically-loadable engine module.

1. Obtain latest Apache Toolkit from SafeNet Support. It is assumed that open ssl is installed at the location "/usr/local/ssl/bin/openssl".
2. Traverse to toolkit: /root/_cdrom_apache, extract the following file openssl-x.x.x.tar.gz (where x.x.x is patched open ssl version)

   Example:

   # tar xzvf openssl-1.0.1i.tar.gz

3. Ensure that the following files are present at the extracted location openssl-1.0.1i/engine/:

   e_lunaca3.c
   e_lunaca3_err.h
   e_lunaca3_err.c
   e_lunaca3.ec
   vendor_defns/e_lunaca3.h

4. Run the makefile shown below; e.g., linux 64-bit:

   #!/usr/bin/make
   SSL=/usr/local/ssl
   CC=gcc
   CFLAGS=-fPIC -m64 -O3
   INCLUDES=-I$(SSL)/include
   DEFINES=-DOPENSSL_PIC -DOPENSSL_THREADS -D_REENTRANT -DSSO_DLF CN -DHAVE_DLF CN_H -DTERMIO
   LD=gcc
   LDFLAGS=-fPIC -m64 -shared
   LDEXTA=-ldl -L$(SSL)/lib -lcrypto
   default:
   $(CC) $(CFLAGS) $(INCLUDES) $(DEFINES) -c -o e_lunaca3.o e_lunaca3.c
Case 3: Apply the luna engine patch to Open SSL source code.

1. Obtain latest Apache Toolkit from SafeNet Support.
2. Download Open SSL source code from [www.openssl.org](http://www.openssl.org) and extract the original (unpatched) open ssl in temporary directory.
   ```
   # cd /tmp
   # tar xzvf /root/openssl-1.0.1j.tar.gz
   ```

   **NOTE:** The patch file (e.g., 'openssl-lunaca3-patch-1.0.1i.tar') may apply successfully to a more recent version of Open SSL (e.g., 1.0.1j). Otherwise or for any other version of Open SSL, contact SafeNet support for a more recent version of the luna engine.

3. Create a symbolic link in the directory with name “b” (the patch expects path b).
   ```
   # ln -s openssl-1.0.1j b
   ```
4. Apply the patch (that is simple diff file) from Apache Toolkit to downloaded open ssl.
   ```
   # zcat openssl-lunaca3-patch-1.0.1i.tar.gz | patch -p0
   ```
5. Build openssl normally; e.g., on Linux:
   ```
   # cd b
   # ./config --prefix=/usr/local/ssl --openssldir=/usr/local/ssl shared
   # make
   # make test
   # make install
   ```
6. Create a soft link with name LunaCA3.
# ln -s /usr/local/ssl/lib/engines/liblunaca3.so /usr/local/ssl/lib/engines/libLunaCA3.so

7. Verify the luna engine is present.

   # /usr/local/ssl/bin/openssl engine -v LunaCA3

   (LunaCA3) Luna CA3 engine support
   
   enginearg, openSession, closeSession, login, logout, engineinit, CONF_PATH, ENGINE_INIT,
   ENGINE2_INIT, engine2init, DisableCheckFinalize, SO_PATH, GET_HA_STATE, SET_FINALIZE_PENDING,
   SKIP_C_INITIALIZE

**Case 4: Configure Open SSL to enable Luna Engine by default.**

1. It is assumed that Open SSL with Luna engine is installed at the location /usr/local/ssl/bin/openssl
2. Edit the openssl configuration file at the location "/usr/local/ssl/openssl.cnf" and make the following changes:

   # Insert near top of file openssl.cnf:
   
   openssl_conf = openssl_init
   
   # Insert at bottom of file openssl.cnf:
   
   [ openssl_init ]
   engines = engine_section
   [ engine_section ]
   LunaCA3 = LunaCA3_section
   [ LunaCA3_section ]
   dynamic_path = /usr/local/ssl/lib/engines/liblunaca3.so
   default_algorithms = ALL

3. Test the open ssl without explicitly mentioned the Luna CA3 engine in commands.

   # /usr/local/ssl/bin/openssl genrsa 2048
   
   # /usr/local/ssl/bin/openssl req -new -nodes -key tmpkey.pem -out tmpkey.req -days 30 -verify

   Where tmpkey.pem is output generated by first command. It will use the Luna CA3 engine by default without
mentioning it in the command line.

**Verify Open SSL and Luna CA3 engine Integration**

1. Open the /etc/Chrystoki.conf file and write a section for Luna CA3 engine as follows:

   EngineLunaCA3 = {
   LibPath = /usr/safenet/lunaclient/lib/libCryptoki2.so;
   LibPath64 = /usr/safenet/lunaclient/lib/libCryptoki2_64.so;
   EnableDsaGenKeyPair = 1;
   EnableRsaGenKeyPair = 1;
   DisablePublicCrypto = 1;
   EnableRsaSignVerify = 1;
   EnableLoadPubKey = 1;
EnableLoadPrivKey = 1;
DisableCheckFinalize = 1;
DisableEcdsa = 1;
DisableDsa = 0;
DisableRand = 0;
EngineInit = 1:10:11;
}

2. Run the sautil utility to open the session on the SafeNet Luna HSM slot.
   
   # /usr/local/sautil/bin/sautil -v -s 1 -i 10:11 -o -q

3. Open the /usr/local/ssl/opessl.cnf file and edit the [CA_default] section and make the following changes:
   
   dir              = /usr/local/ssl
   new_certs_dir    = $dir/certs


5. Open the /usr/local/ssl/serial file and write 01 at top and press Enter. Save the file.

6. First, create a 2048-bit private key on SafeNet Luna HSM using Luna CA3 engine that is used to create CA.
   
   # /usr/local/ssl/bin/openssl genrsa -engine LunaCA3 2048
   
   It generates the 2048 RSA keys on SafeNet Luna HSM.

7. Next, create a CA certificate based on this key, that is used for signing other certificates:
   
   # /usr/local/ssl/bin/openssl req -engine LunaCA3 -new -x509 -days 365 -key rsa-private-394e865fa875db0ff013205202ecb92b065303be -keyform engine -out /usr/local/ssl/certs/ca.cer
   
   Where “rsa-private-394e865fa875db0ff013205202ecb92b065303be” is the object label for the ca private key on the SafeNet Luna HSM.

8. Create a directory to generate the certificate request for sender and receiver (to validate the sign, verify, encrypt, and decrypt using open ssl)
   
   # mkdir /usr/local/ssl/certs/sender
   
   # mkdir /usr/local/ssl/certs/receiver

9. Generate the certificate request for sender.
   
   # /usr/local/ssl/bin/openssl req -engine LunaCA3 -newkey rsa:2048 -out /usr/local/ssl/certs/sender/sender.txt
   
   Sender request is used to generate the sender’s certificate signed by the generated Certificate Authority.

10. Generate the certificate request for receiver.
    
    # /usr/local/ssl/bin/openssl req -engine LunaCA3 -newkey rsa:2048 -out /usr/local/ssl/certs/receiver/receiver.txt
    
    Receiver request is used to generate the receiver’s certificate signed by the generated Certificate Authority.

11. Sign the certificate request for Sender and Receiver by CA (Certificate Authority).
    
    # /usr/local/ssl/bin/openssl ca -engine LunaCA3 -policy policy_anything -cert /usr/local/ssl/certs/ca.cer -in /usr/local/ssl/certs/sender/sender.txt -keyfile rsa-private-394e865fa875db0ff013205202ecb92b065303be -keyform engine -out /usr/local/ssl/certs/sender/sender.cer
Where “rsa-private-394e865fa875db0ff013205202ecb92b065303be” is the object label for the ca private key on the SafeNet Luna HSM.

Use the same command to sign the certificate request for Receiver, i.e.

```
# /usr/local/ssl/bin/openssl ca -engine LunaCA3 -policy policy_anything -cert
/usr/local/ssl/certs/ca.cer -in /usr/local/ssl/certs/receiver/receiver.txt -keyfile rsa-private-394e865fa875db0ff013205202ecb92b065303be -keyform engine -out
/usr/local/ssl/certs/receiver/receiver.cer
```

Where “rsa-private-394e865fa875db0ff013205202ecb92b065303be” is the object label for the ca private key on the SafeNet Luna HSM.

12. Now use Open SSL CMS command to sign, verify, encrypt and decrypt the message. To do this first create a text file message.txt and enter some text in it.

The sender is used to send this message to the receiver by signing the message with his private key and encrypt the message by receiver’s public key.

Receiver then decrypts the message using his private key and verifies the message using sender’s public key.

Sender’s and Receiver’s keys are stored on SafeNet Luna HSM so you have to use the object label of the keys to use those keys.

13. Sign the message.txt with the sender’s private key.

```
# /usr/local/ssl/bin/openssl cms -engine LunaCA3 -sign -in message.txt -signer
/usr/local/ssl/certs/sender/sender.cer -inkey rsa-private-2774dd1ee6ba6633141a83ad57e21b67a86cf623 -keyform engine -out sendmail.msg
```

Where “rsa-private-2774dd1ee6ba6633141a83ad57e21b67a86cf623” is the object label for the sender’s private key on the SafeNet Luna HSM.

14. Encrypt the sendmail.msg using receiver’s public key, supplied with the receiver’s certificate.

```
# /usr/local/ssl/bin/openssl cms -engine LunaCA3 -encrypt -in sendmail.msg -out sendmail_enc.msg
/usr/local/ssl/certs/receiver/receiver.cer
```

15. Decrypt the sendmail_enc.msg using receiver’s private key.

```
# /usr/local/ssl/bin/openssl cms -engine LunaCA3 -decrypt -in sendmail_enc.msg -inkey rsa-private-d459b652ec0a9e283d2963b2f3f1999bda12f8fb -keyform engine -out sendmail_dec.msg
```

Where “rsa-private-d459b652ec0a9e283d2963b2f3f1999bda12f8fb” is the object label for the receiver’s private key on the SafeNet Luna HSM.

16. Now verify the signature of sendmail_dec.msg using sender’s public key supplied with the sender’s certificate.

```
# /usr/local/ssl/bin/openssl cms -engine LunaCA3 -verify -in sendmail_dec.msg -CAfile
/usr/local/ssl/certs/ca.cer -out out.txt /usr/local/ssl/certs/sender/sender.cer
```

Open the out.txt and verify the message which you have typed in the message.txt

17. Close the session when the work has been done using sautil.

```
# /usr/local/sautil/bin/sautil -c -s 1 -i 10:11 -q
```
Installation and configuration of Open SSL toolkit to use Luna CA3 engine on Microsoft Windows

Install and Configure the Open SSL toolkit

1. Traverse the apache toolkit and extract the files sautil-win64-openssl-1.0.0e.tar and sautil-win64-openssl-0.9.8n.tar in a folder C:\OpenSSL.
2. Add C:\OpenSSL\bin to your system path (Control Panel -> System -> Change Settings -> Advanced -> Environment Variables -> System Variables). This is not mandatory but it makes the process easy.
3. Create a directory C:\ssl. Now, it is used as a working folder.
4. Create a openssl.conf file under the working directory. Copy and paste the following and save it as C:\ssl\openssl.conf:

```plaintext
# SSLeay example configuration file.
# This is mostly being used for generation of certificate requests.
#
RANDFILE = .rnd

[ ca ]
default_ca = CA_default  # The default ca section

[ CA_default ]
certs = certs  # Where the issued certs are kept
crl_dir = crl  # Where the issued crl are kept
database = database.txt  # database index file.
new_certs_dir = certs  # default place for new certs.

certificate = cacert.pem  # The CA certificate
serial = serial.txt  # The current serial number
crl = crl.pem  # The current CRL
private_key = private\cakey.pem  # The private key
RANDFILE = private\private.rnd  # private random number file
x509_extensions = x509v3_extensions  # The extentions to add to the cert
default_days = 365  # how long to certify for
default_crl_days= 30  # how long before next CRL
default_md = md5  # which md to use.
```
preserve = no    # keep passed DN ordering

# A few different ways of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)  
policy = policy_match

# For the CA policy
[ policy_match ]
countryName = match  
stateOrProvinceName = match  
organizationName = match  
organizationalUnitName = match  
commonName = supplied  
emailAddress = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName = optional  
stateOrProvinceName = optional  
localityName = optional  
organizationName = optional  
organizationalUnitName = optional  
commonName = supplied  
emailAddress = optional

#...........................................................................
[ req ]
default_bits = 1024  
default_keyfile = privkey.pem  
distinguished_name = req_distinguished_name  
attributes = req_attributes

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_min = 2
countryName_max = 2
stateOrProvinceName = State or Province Name (full name)
localityName = Locality Name (eg, city)
0.organizationName = Organization Name (eg, company)
organizationalUnitName = Organizational Unit Name (eg, section)
commonName = Common Name (eg, your website's domain name)
commonName_max = 64
emailAddress = Email Address
emailAddress_max = 40

[ req_attributes ]
challengePassword = A challenge password
challengePassword_min = 4
challengePassword_max = 20

[ x509v3_extensions ]
# under ASN.1, the 0 bit would be encoded as 80
# nsCertType = 0x40
# nsBaseUrl
# nsRevocationUrl
# nsRenewalUrl
# nsCaPolicyUrl
# nsSslServerName
# nsCertSequence
# nsCertExt
# nsDataType

NOTE: Toolkit provided with the openssl.cnf file which can be used for configuration file as well but requires some changes when used on Windows for certificate generation.

5. Run CMD and execute the following command to set the openssl configuration file path:
   set OPENSSL_CONF=C:\ssl\openssl.conf

6. Open the crystoki.ini file and write a section for Luna CA3 engine as follows:
   [EngineLunaCA3]
   LibPath = C:\Program Files\LunaSA\cryptoki.dll
   LibPath64 = C:\Program Files\LunaSA\cryptoki.dll
EnableDsaGenKeyPair = 1
EnableRsaGenKeyPair = 1
DisablePublicCrypto = 1
EnableRsaSignVerify = 1
EnableLoadPubKey = 1
EnableLoadPrivKey = 1
DisableCheckFinalize = 1
DisableEcdsa = 1
DisableDsa = 0
DisableRand = 0
EngineInit = 1:10:11

7. Set the crystoki.ini path to use with openssl using command prompt:
   set CONF_PATH=C:\Program Files\LunaSA\crystoki.ini

**Verify Open SSL and Luna CA3 engine Integration**

This part is used to generate keys on SafeNet Luna HSM using Luna CA3 engine in Open SSL command and use those keys to generate certificate request, encrypt and decrypt the message as well as sign and verify the message through Open SSL commands.

1. Set up the directory structure and files required by Open SSL using the command prompt:
   C:\ssl>mkdir keys
   C:\ssl>mkdir requests
   C:\ssl>mkdir certs

2. Create the file database.txt – an empty (zero-byte) text file under C:\ssl\database.txt

3. Create the serial number file serial.txt. This is a plain ASCII file containing the string "01" on the first line, followed by a new line under C:ssl\serial.txt.

4. Open the session on SafeNet Luna HSM slot 1 for application id 10:11
   C:\OpenSSL\sautil\bin>sautil -o -s 1 -i 10:11 -q
   Enter the token password, it displays the session for application id 10:11
5. First, create a 2048-bit private key on SafeNet Luna HSM using Luna CA3 engine that is used to create CA.
   
   ```bash
   C:\openssl genrsa -engine LunaCA3 2048
   ```
   
   RSA 2048 bit key for CA is generated on SafeNet Luna HSM.

6. Next, create CA certificate based on generated key that is used to sign other certificates:
2 – Integrating SafeNet Luna HSM with OpenSSL Using LunaCA3 Engine

C:\ssl\openssl req -engine LunaCA3 -new -x509 -days 365 -key rsa-private-13ea213a27eaa0ad2528f50821adda9e3dde1bf -keyform engine -out certs/ca.cer

Where “rsa-private-13ea213a27eaa0ad2528f50821adda9e3dde1bf” is the object label for the CA private key on the SafeNet Luna HSM.

7. Create a certificate request for sender.

C:\ssl\openssl req -engine LunaCA3 -newkey rsa:1024 -out requests/sender.txt

Sender request is used to generate the sender’s certificate signed by generated Certificate Authority.

8. Create a certificate request for receiver.

C:\ssl\openssl req -engine LunaCA3 -newkey rsa:1024 -out requests/receiver.txt
Receiver request is used to generate the receiver’s certificate signed by generated Certificate Authority.

You can see that sender and receiver keys are generated on SafeNet Luna HSM:

<table>
<thead>
<tr>
<th>Partition Name</th>
<th>Part2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition SN</td>
<td>150207009</td>
</tr>
<tr>
<td>Storage (Bytes)</td>
<td>Total=102701, Used=3932, Free=98769</td>
</tr>
<tr>
<td>Number objects</td>
<td>5</td>
</tr>
<tr>
<td>Object Label</td>
<td>rsa-public-13ea213a27eeaa0ad2528f50621adda9e3dde1bf</td>
</tr>
<tr>
<td>Object Type</td>
<td>Public Key</td>
</tr>
<tr>
<td>Object Label</td>
<td>rsa-private-13ea213a27eeaa0ad2528f50621adda9e3dde1bf</td>
</tr>
<tr>
<td>Object Type</td>
<td>Private Key</td>
</tr>
<tr>
<td>Object Label</td>
<td>rsa-public-71d14b4ee1a52636d3cdcf38c8d452590463404b</td>
</tr>
<tr>
<td>Object Type</td>
<td>Public Key</td>
</tr>
<tr>
<td>Object Label</td>
<td>rsa-private-71d14b4ee1a52636d3cdcf38c8d452590463404b</td>
</tr>
<tr>
<td>Object Type</td>
<td>Private Key</td>
</tr>
<tr>
<td>Object Label</td>
<td>rsa-public-af007ac8c333fc1c6cd67f461124a2de25229d1</td>
</tr>
<tr>
<td>Object Type</td>
<td>Public Key</td>
</tr>
<tr>
<td>Object Label</td>
<td>rsa-private-af007ac8c333fc1c6cd67f461124a2de25229d1</td>
</tr>
<tr>
<td>Object Type</td>
<td>Private Key</td>
</tr>
</tbody>
</table>

9. Sign the certificate request for Sender and Receiver by CA (Certificate Authority).

C:\ssl\openssl req -engine LunaCA3 -newkey rsa:1024 -out requests/receiver.txt engine "LunaCA3" set.
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter , the field will be left blank.

Country Name (2 letter code) [IN]:IN
State or Province Name (full name) []:UP
Locality Name (eg, city) []:Noida
Organization Name (eg, company) []:Receiver
Organizational Unit Name (eg, section) []:Test
Common Name (eg, your websites domain name) []:Receiver
Email Address []:receiver@luna.com

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:

Receiver request is used to generate the receiver's certificate signed by generated Certificate Authority.
Use the same command to sign the certificate request for Receiver, i.e.

C:\ssl>openssl ca -engine LunaCA3 -policy policy_anything -cert certs/ca.cer -in requests/receiver.txt -keyfile rsa-private-13ea213a27eea00ad2528f50821adda9e3dde1bf -keyform engine -out certs/receiver.cer

Where "rsa-private-13ea213a27eea00ad2528f50821adda9e3dde1bf" is the object label for the CA private key on the SafeNet Luna HSM.

Below is the both receiver’s and sender’s certificate which are signed by CA (Certificate Authority created in Step 6).
10. Now Open SSL CMS command is used to sign, verify, encrypt, and decrypt the message. To do this, create a text file message.txt in the C:\ssl directory.

```
message.txt
Hi Receiver,
good morning.
Regards,
sender
```

The sender sends the above message to the receiver by signing the message with own private key and encrypt the message by receiver’s public key.

Receiver then decrypt the message using own private key and verify the message using sender’s public key.

Sender’s and Receiver’s key are stored on SafeNet Luna HSM, use the object label of the keys to use those keys.

11. Sign the message.txt with the sender’s private key.

```
C:\ssl>openssl cms -engine LunaCA3 -sign -in message.txt -signer certs\sender.cer -inkey rsa-private-71d14b4ee1a52636d3ccdf38c8d452590463404b -keyform engine -out sendmail.msg
```

Where “rsa-private-71d14b4ee1a52636d3ccdf38c8d452590463404b” is the object label for the sender’s private key on the SafeNet Luna HSM.
12. Encrypt the sendmail.msg using receiver's public key, supplied with the receiver's certificate.

C:\ssl>openssl cms -engine LunaCA3 -encrypt -in sendmail.msg -out sendmail_enc.msg
certs\receiver.cer

13. Decrypt the sendmail_enc.msg using receiver's private key.

C:\ssl>openssl cms -engine LunaCA3 -decrypt -in sendmail_enc.msg -inkey rsa-private-afd007ac833fc1c6cd67f461124a2de25229d1 -keyform engine -out sendmail_dec.msg

Where "rsa-private-afd007ac833fc1c6cd67f461124a2de25229d1" is the object label for the receiver's private key on the SafeNet Luna HSM.

C:\ssl>openssl cms -engine LunaCA3 -verify -in sendmail_dec.msg -CAfile certs\ca.cer -out out.txt certs\sender.cer

15. Open the out.txt and verify the message which you have typed in the message.txt
16. Close the session with sautil when you are done.

   `C:\OpenSSL\sautil\bin>sautil -c -s 1 -i 10:11 -q`

Integration of Open SSL with SafeNet Luna HSM using Luna CA3 engine is completed successfully. All keys are generated on SafeNet Luna HSM and accessible while using Open SSL commands for sign, verify, encrypt, and decrypt the data.
Before You Begin

It is recommended that you should familiarize yourself with Open SSL. Refer to the appropriate documents for Open SSL commands at the following location:

http://www.openssl.org/docs/

Open SSL provides the support of engine (basically hardware devices) to store the keys on hardware devices to make keys more secure. SafeNet provides the Open SSL toolkit having support of GEM Engine that is used to communicate with the SafeNet Luna HSM.

Open SSL Toolkit

The Open SSL toolkit is provided to make the installation quick and easy. The installation CD can be obtained from the SafeNet Customer Support.

Installation and Configuration of Open SSL Toolkit to use Gem Engine on UNIX

Install and Configure the Open SSL toolkit

The gembuild script is provided with the toolkit that is centered on generating/integrating SafeNet's Open SSL dynamic engine with a working installation of Open SSL. There are various scenarios for using gembuild depending on exactly what you are trying to accomplish. The gembuild script has a set of commands that it accepts as well as various options. Run "/gembuild --help" for usage.

NOTE: The supported Open SSL versions are 1.0.0x, 1.0.1x and 1.0.2x (x means any of the lettered versions).

SCENARIO A – Integrating pre-built dynamic engine with an existing installation of Open SSL on Linux.

1. Obtain the latest Open SSL toolkit from SafeNet Support. Extract it and traverse to the gemengine directory.
2. Locate the libgem.so and sautil files inside of builds/linux/<flavour>/<bits>/<stream> where selecting the flavor of Linux, how many bits the architecture is and the Open SSL stream.
   An example would be builds/linux/rhel/64/1.0.1 for RHEL 64 bit where the Open SSL version is a variant of 1.0.1.

3. Locate the location of the Open SSL engines directory using gembuild. It is available in the gem engine directory that is downloaded and extracted.
   
   ```bash
   # ./gembuild locate-engines
   ```
   
   The openssl engines directory is located at:
   
   /usr/local/lib64/engines
   
   This will display the engines directory for the openssl that is in PATH.

4. Copy the libgem.so to the engines directory and test engine.
   
   Example:
   
   ```bash
   # cp builds/linux/rhel/64/1.0.1/libgem.so /usr/local/lib64/engines/
   ```

5. Verify the gem engine is present.
   
   Example:
   
   ```bash
   # openssl engine gem -v
   ```
   
   ```
   (gem) Gem engine support
   enginearg, openSession, closeSession, login, logout, engineinit,
   CONF_PATH, ENGINE_INIT, ENGINE2_INIT, engine2init, DisableCheckFinalize,
   SO_PATH, GET_HA_STATE, SET_FINALIZE_PENDING, SKIP_C_INITIALIZE,
   IntermediateProcesses
   ```
   
   If the output looks as above then the gem engine is successfully installed.

6. Copy the sautil command to /usr/local/bin
   
   Example:
   
   ```bash
   # cp builds/linux/rhel/64/1.0.1/sautil /usr/local/bin
   # sautil
   ```
   
   The usage of sautil displays.

**SCENARIO B - Compiling of the dynamic engine to be installed with an existing installation of Open SSL.**

Ensure that the system has an appropriate C compiler installed such as gcc and the make utility.

1. Download and extract OpenSSL source tarball. Get the version that is close to your existing OpenSSL installation (run "openssl version").
   
   Example:
   
   Download openssl-1.0.1s.tar.gz from https://www.openssl.org/source/
   
   ```bash
   # tar xvfz openssl-1.0.1s.tar.gz
   ```

2. Locate the OpenSSL engines directory.
   
   ```bash
   # ./gembuild locate-engines
   ```

3. Run gembuild config.
Example:

# ./gembuild config --openssl-source=/home/openssl-1.0.1s --openssl-engines=/usr/lib64/openssl/engines --config-bits=64

**NOTE:** This assumes that the openssl headers directory is located in /usr/include. It may be necessary to install the openssl development package. If the header files are located in a different location, the --openssl-includes option can be used. As well, the --openssl-libs options can be used to specify the location of the lib directory with libcrypto.so. All paths need to be absolute.

4. Install some needed EC header files.
   # ./gembuild openssl-ec-headers

5. Compile the engine.
   # ./gembuild engine-build

6. Install and test the engine.
   # ./gembuild engine-install
   
   # openssl engine gem -v
   
   (gem) Gem engine support
   
   enginearg, openSession, closeSession, login, logout, engineinit,
   CONF_PATH, ENGINE_INIT, ENGINE2_INIT, engine2init, DisableCheckFinalize,
   SO_PATH, GET_HA_STATE, SET_FINALIZE_PENDING, SKIP_C_INITIALIZE

7. Compile and install sautil
   
   # ./gembuild sautil-build
   
   # ./gembuild sautil-install

   By default this will install the sautil command to /usr/local/bin/sautil.

   If a different location is desired, use the --sautil-prefix option to specify the desired directory either by redoing STEP 3 with the option or by specifying the option as part of the "../gembuild sautil-install" command.

**SCENARIO C - Compiling and installing Open SSL from source and compiling and installing gem engine.**

1. Download and extract OpenSSL source tarball.
   
   Example:
   
   Download openssl-1.0.1s.tar.gz from https://www.openssl.org/source/
   
   # tar xvfz openssl-1.0.1s.tar.gz

2. Download and extract OpenSSL FIPS module. Ignore this step if FIPS module is not required.
   
   Example:
   
   Download openssl-fips-2.0.9.tar.gz from https://www.openssl.org/source/
   
   # tar xvfz openssl-fips-2.0.9.tar.gz

3. Run gembuild config using the --prefix option.
Example:
```bash
# ./gembuild config --openssl-source=/home/openssl-1.0.1s --prefix=/usr/local --config-bits=64
```
If FIPS module is required, add `--openssl-fips-source=/home/openssl-fips-2.0.9` to the `./gembuild config` command.

4. Compile and install FIPS module. Proceed to STEP 5 if FIPS module is not required.
   ```bash
   # ./gembuild openssl-fips-build
   # ./gembuild openssl-fips-install
   ```

5. Compile and install OpenSSL.
   ```bash
   # ./gembuild openssl-build
   # ./gembuild openssl-install
   ```

   ```bash
   # ./gembuild engine-build
   # ./gembuild engine-install
   # /usr/local/ssl/bin/openssl engine gem -v
   ```
   (gem) Gem engine support
   ```
   enginearg, openSession, closeSession, login, logout, engineinit, Conf_PATH, ENGINE_INIT, ENGINE2_INIT, engine2init, DisableCheckFinalize, 
   SO_PATH, GET_HA_STATE, SET_FINALIZE_PENDING, SKIP_C_INITIALIZE
   ```

7. Compile and install sautil command.
   ```bash
   # ./gembuild sautil-build
   # ./gembuild sautil-install
   ```
   By default this installs the sautil command to `<prefix>/sautil/bin/sautil` where `<prefix>` is the directory specified with `--prefix` option in the step 3.
   If a different location is desired, use the `--sautil-prefix` option to specify the desired directory either by redoing the step 3 with the option or by specifying the option as part of the `"./gembuild sautil-install"` command.

8. Add openssl and sautil to PATH
   Example:
   ```bash
   # export PATH=/usr/local/ssl/bin:/usr/local/sautil/bin:$PATH
   ```

**SCENARIO D: Configure Open SSL to enable Gem Engine by default.**

It is assumed that Open SSL with Luna engine is installed at the location `/usr/local/ssl/bin/openssl`

1. Locate the openssl.cnf and engines directory.
   ```bash
   # openssl version -d
   OPENSSLDIR: "/usr/local/ssl"
   ```
   The openssl.cnf file is located in this directory. Note that this is for the openssl that is in PATH. Run "which openssl" to ensure it is the right one.
   ```bash
   # ./gembuild locate-engines
   ```
This gives the directory where the libgem.so should be located. Note that this is for the openssl that is in PATH.

2. Edit the openssl.cnf file.
   Example:
   
   # Insert near top of file openssl.cnf:
   openssl_conf = openssl_init
   
   # Insert at bottom of file openssl.cnf:
   [ openssl_init ]
   engines = engine_section
   [ engine_section ]
   gem = gem_section
   [ gem_section ]
   dynamic_path = /usr/local/ssl/lib/engines/libgem.so
   default_algorithms = ALL

3. Verify that engine is loading without specifying it.
   
   # openssl engine -v
   
   (dynamic) Dynamic engine loading support
   SO_PATH, NO_VCHECK, ID, LIST_ADD, DIR_LOAD, DIR_ADD, LOAD
   
   (gem) Gem engine support
   enginearg, openSession, closeSession, login, logout, engineinit,
   CONF_PATH, ENGINE_INIT, ENGINE2_INIT, engine2init, DisableCheckFinalize,
   SO_PATH, GET_HA_STATE, SET_FINALIZE_PENDING, SKIP_C_INITIALIZATE,

4. Test the application. First generate the key and then generate certificate request.
   
   # /usr/local/ssl/bin/openssl genrsa 2048
   openssl command-line WITHOUT explicit engine:
   
   # openssl req -new -nodes -key tmpkey.pem -out tmpkey.req -days 30 –verify
   Where tmpkey.pem is output generated by “openssl genrsa” command. It will use the Gem engine by default without mentioning it in command line.

Verify Open SSL and Gem engine Integration

1. Open the /etc/Chrystoki.conf file and write a section for Gem engine as follows:
   
   GemEngine = {
   LibPath = /usr/safenet/lunaclient/lib/libCryptoki2.so;
   LibPath64 = /usr/safenet/lunaclient/lib/libCryptoki2_64.so;
   EnableDsaGenKeyPair = 1;
   EnableRsaGenKeyPair = 1;
   DisablePublicCrypto = 1;
EnableRsaSignVerify = 1;
EnableLoadPubKey = 1;
EnableLoadPrivKey = 1;
DisableCheckFinalize = 1;
DisableEcdsa = 1;
DisableDsa = 0;
DisableRand = 0;
EngineInit = 1:10:11;
}

2. Run the sautil utility to open the session on SafeNet Luna HSM slot.
   # /usr/local/sautil/bin/sautil -v -s 1 -i 10:11 -o -q

3. Open the /usr/local/ssl/opessl.cnf file and edit the [CA_default] section and make the following changes:
   dir      = /usr/local/ssl
   new_certs_dir = $dir/certs


5. Open the /usr/local/ssl/serial file and write 01 at top and press enter. Save the file.

6. First, create a 2048-bit private key on SafeNet Luna HSM using Gem engine that is used for creating our CA.
   # /usr/local/ssl/bin/openssl genrsa -engine gem 2048
   It generates RSA 2048 key on SafeNet Luna HSM.

7. Next, create a CA certificate based on this key that is used for signing other certificates:
   # /usr/local/ssl/bin/openssl req -engine gem -new -x509 -days 365 -key rsa-private-a55e015d94ee6c4a559dfab7c39a2069d4064bcd -keyform engine -out /usr/local/ssl/certs/ca.cer
   Where "rsa-private-a55e015d94ee6c4a559dfab7c39a2069d4064bcd" is the object label for the ca private key on the SafeNet Luna HSM.

8. Create a directory to generate the certificate request for sender and receiver.
   # mkdir /usr/local/ssl/certs/sender
   # mkdir /usr/local/ssl/certs/receiver

9. Generate a certificate request for sender.
   # /usr/local/ssl/bin/openssl req -engine gem -newkey rsa:2048 -out /usr/local/ssl/certs/sender/sender.txt
   Sender request is used to generate the sender’s certificate signed by generated Certificate Authority.

10. Generate a certificate request for receiver.
    # /usr/local/ssl/bin/openssl req -engine gem -newkey rsa:2048 -out /usr/local/ssl/certs/receiver/receiver.txt
    Receiver request is used to generate the receiver’s certificate signed by generated Certificate Authority.

11. Sign the certificate request for Sender and Receiver by CA (Certificate Authority).
    # /usr/local/ssl/bin/openssl ca -engine gem -policy policyAnything -cert /usr/local/ssl/certs/ca.cer -in /usr/local/ssl/certs/sender/sender.txt -keyfile rsa-private-
a55e015d94ee6c4a559dfab7c39a2069d4064bcd -keyform engine -out
/usr/local/ssl/certs/sender/sender.cer

Where “rsa-private-a55e015d94ee6c4a559dfab7c39a2069d4064bcd” is the object label for the ca private key on the SafeNet Luna HSM.

Use the same command to sign the certificate request for Receiver, i.e.

# /usr/local/ssl/bin/openssl ca -engine gem -policy policy_anything -cert
/usr/local/ssl/certs/ca.cer -in /usr/local/ssl/certs/receiver/receiver.txt -keyfile rsa-private-a55e015d94ee6c4a559dfab7c39a2069d4064bcd -keyform engine -out
/usr/local/ssl/certs/receiver/receiver.cer

Where “rsa-private-a55e015d94ee6c4a559dfab7c39a2069d4064bcd” is the object label for the ca private key on the SafeNet Luna HSM.

12. Now use the OpenSSL CMS command to sign, verify, encrypt and decrypt the message. To do these create a text file message.txt.

The sender will send the above message to the receiver by signing the message with own private key and encrypt the message by receiver’s public key.

Receiver then decrypt the message using own private key and verify the message using sender’s public key.

Sender’s and Receiver’s keys are stored on SafeNet Luna HSM, use the object label of the keys to use those keys.

13. Sign the message.txt with the sender’s private key

# /usr/local/ssl/bin/openssl cms -engine gem -sign -in message.txt -signer
/usr/local/ssl/certs/sender/sender.cer -inkey rsa-private-7dfdb3caaf25a63b3d8a81d2a3b51668decafe0f -keyform engine -out sendmail.msg

Where “rsa-private-7dfdb3caaf25a63b3d8a81d2a3b51668decafe0f” is the object label for the sender private key on the SafeNet Luna HSM.

14. Encrypt the sendmail.msg using receiver’s public key, supplied with the receiver’s certificate.

# /usr/local/ssl/bin/openssl cms -engine gem -encrypt -in sendmail.msg -out sendmail_enc.msg
/usr/local/ssl/certs/receiver/receiver.cer

15. Decrypt the sendmail_enc.msg using receiver’s private key.

# /usr/local/ssl/bin/openssl cms -engine gem -decrypt -in sendmail_enc.msg -inkey rsa-private-ec0fcb1ce9114662556caab35fc2ba6f0565752ec -keyform engine -out sendmail_dec.msg

Where “rsa-private-ec0fcb1ce9114662556caab35fc2ba6f0565752ec” is the object label for the receiver private key on the SafeNet Luna HSM.


# /usr/local/ssl/bin/openssl cms -engine gem -verify -in sendmail_dec.msg -CAfile
/usr/local/ssl/certs/ca.cer -out out.txt /usr/local/ssl/certs/sender/sender.cer

Open the out.txt and verify the message which you have typed in the message.txt

17. Close the session when you are done with sautil.

# /usr/local/sautil/bin/sautil -c -s 1 -i 10:11 -q
Installation and configuration of Open SSL toolkit to use Gem engine on Microsoft Windows

Install and Configure the Open SSL toolkit

1. Traverse the Open SSL toolkit “gemengine-1.1\builds\win” and extract the file sautil-win64-openssl-1.0.1q.tar.gz and ssl-win64-openssl-1.0.1q.tar.gz in a folder C:\OpenSSL
2. Add C:\OpenSSL\ssl\bin to your system path (Control Panel -> System -> Change Settings -> Advanced -> Environment Variables -> System Variables). This is not mandatory but it makes the process easy.
3. Create a directory C:\ssl. Now, it is used as a working folder.
4. Create an openssl.conf file under the working directory. Copy and paste the following and save it as C:\ssl\openssl.conf:

```
# SSLeay example configuration file.
# This is mostly being used for generation of certificate requests.
#
RANDFILE = .rnd

[ ca ]
default_ca = CA_default  # The default ca section

[ CA_default ]
certs = certs  # Where the issued certs are kept
crl_dir = crl  # Where the issued crl are kept
database = database.txt  # database index file.
new_certs_dir = certs  # default place for new certs.
certificate = cacert.pem  # The CA certificate
serial = serial.txt  # The current serial number
crl = crl.pem  # The current CRL
private_key = private\cakey.pem  # The private key
RANDFILE = private\private.rnd  # private random number file
x509_extensions = x509v3_extensions  # The extensions to add to the cert
default_days = 365  # how long to certify for
default_crl_days= 30  # how long before next CRL
default_md = md5  # which md to use.
```
preserve = no  # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy = policy_match

# For the CA policy
[ policy_match ]
countryName = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = match
countryName = supplied
commonName = supplied
eMailAddress = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
countryName = supplied
eMailAddress = optional

#**************************************************************************
[ req ]
default_bits = 1024
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_min = 2
countryName_max = 2
stateOrProvinceName = State or Province Name (full name)
localityName = Locality Name (eg, city)
organizationName = Organization Name (eg, company)
organizationalUnitName = Organizational Unit Name (eg, section)
commonName = Common Name (eg, your website's domain name)
commonName_max = 64
emailAddress = Email Address
emailAddress_max = 40

[ req_attributes ]
challengePassword = A challenge password
challengePassword_min = 4
challengePassword_max = 20

[ x509v3_extensions ]
# under ASN.1, the 0 bit would be encoded as 80
# nsCertType = 0x40
# nsBaseUrl
# nsRevocationUrl
# nsRenewalUrl
# nsCaPolicyUrl
# nsSslServerName
# nsCertSequence
# nsCertExt
# nsDataType

NOTE: Toolkit provided with the openssl.cnf file which can be used for configuration file as well but requires some changes when used on Windows for certificate generation.

5. Run CMD and execute the following command to set the openssl configuration file path:
   set OPENSSL_CONF=C:\ssl\openssl.conf

6. Open the crystoki.ini file and write a section for Gem engine as follows:
   [GemEngine]
   LibPath = <Luna Client installation Directory>\cryptoki.dll
   LibPath64 = <Luna Client installation Directory>\cryptoki.dll
EnableDsaGenKeyPair = 1
EnableRsaGenKeyPair = 1
DisablePublicCrypto = 1
EnableRsaSignVerify = 1
EnableLoadPubKey = 1
EnableLoadPrivKey = 1
DisableCheckFinalize = 1
DisableEcdsa = 1
DisableDsa = 0
DisableRand = 0
EngineInit = 1:10:11

7. Set the crystoki.ini path to use with openssl using command prompt:
   set CONF_PATH=<Luna Client installation Directory>\crystoki.ini

Verify Open SSL and Gem engine Integration

This part is used to generate keys on SafeNet Luna HSM using Gem engine in Open SSL command and use those keys to generate certificate request, encrypt and decrypt the message as well as sign and verify the message through Open SSL commands.

1. Set up the directory structure and files required by Open SSL using the command prompt:
   C:\ssl>mkdir keys
   C:\ssl>mkdir requests
   C:\ssl>mkdir certs

2. Create the file database.txt – an empty (zero-byte) text file under C:\ssl\database.txt

3. Create the serial number file serial.txt. This is a plain ASCII file containing the string "01" on the first line, followed by a new line under C:\ssl\serial.txt.

4. Copy the libeay32.dll and the ssleay32.dll library and place in directory where sutil.exe is present.

5. Open the session on SafeNet Luna HSM slot 1 for application id 10:11
   C:\OpenSSL\sutil\bin>sutil -o -s 1 -i 10:11 -q
   Enter the token password, when prompt and it opens the session for application id 10:11
6. First, create a 2048-bit private key on SafeNet Luna HSM using Gem engine that is used to create CA.

   C:\ssl\openssl genrsa -engine gem 2048

   RSA 2048 bit key for CA is generated on SafeNet Luna HSM partition.

7. Create CA certificate based on generated key that is used to sign other certificates:

   C:\ssl\openssl req -engine gem -new -x509 -days 365 -key rsa-private-08bde9331fa3be515d2e7db9dd1e28b36b50632e -keyform engine -out certs/ca.cer

   Where "rsa-private-08bde9331fa3be515d2e7db9dd1e28b36b50632e" is the object label for the CA private key on the SafeNet Network HSM.
8. Create a certificate request for sender.

```
C:\ssl>openssl req -engine gem -newkey rsa:1024 -out requests/sender.txt
```

Sender request is used to generate the sender’s certificate signed by the generated Certificate Authority.

Create a certificate request for receiver.

```
C:\ssl>openssl req -engine gem -newkey rsa:1024 -out requests/receiver.txt
```

Receiver request is used to generate the receiver’s certificate signed by the generated Certificate Authority.
The sender and receiver keys are generated on the SafeNet Luna HSM partition:

```
9. Sign the certificate request for Sender and Receiver by CA (Certificate Authority).

C:\ssl\openssl ca -engine gem -policy policy_anything -cert certs/ca.cer -in requests/sender.txt -keyfile rsa-private-08bde9331fa3be515d2e7db9dd1e28b36b50632e -keyform engine -out certs/sender.cer

Use the same command to sign the certificate request for Receiver, i.e.

C:\ssl\openssl ca -engine gem -policy policy_anything -cert certs/ca.cer -in requests/receiver.txt -keyfile rsa-private-08bde9331fa3be515d2e7db9dd1e28b36b50632e -keyform engine -out certs/receiver.cer
```
Where “rsa-private-08bde9331fa3be515d27db9dd1e28b36b50632e” is the object label for the CA private key on the SafeNet Network HSM.

Below is the both receiver’s and sender’s certificate which are signed by CA (Certificate Authority created in the step 6).

10. Open SSL CMS command is used to sign, verify, encrypt, and decrypt the message. To perform this task, create a text file message.txt in the C:\ssl directory.

```
C:\ssl>openssl cms -engine gem -sign -in message.txt -signer certs\sender.cer -inkey rsa-private-605ddfab1e95bfac36ee44f291647e8a3ff5f64 -keyform engine -out sendmail.msg
```

Where “rsa-private-605ddfab1e95bfac36ee44f291647e8a3ff5f64” is the object label for the sender’s private key on the SafeNet Network HSM.

The sender sends the above message to the receiver by signing the message with own private key and encrypt the message by receiver’s public key.

Receiver then decrypt the message using own private key and verify the message using sender’s public key.

Sender’s and Receiver’s key are stored on SafeNet Luna HSM, use the object label of the keys to use those keys.

11. Sign the message.txt with the sender’s private key.

```
C:\ssl>openssl cms -engine gem -sign -in message.txt -signer certs\sender.cer -inkey rsa-private-605ddfab1e95bfac36ee44f291647e8a3ff5f64 -keyform engine -out sendmail.msg
```

Where “rsa-private-605ddfab1e95bfac36ee44f291647e8a3ff5f64” is the object label for the sender’s private key on the SafeNet Network HSM.
3 – Integrating SafeNet Luna HSM with Open SSL Using Gem Engine

C:\openssl cme -engine gen -sign -in message.txt -signer certs\sender.cer -inkey rsa-private-605d6a1e95bfae36 -keyform engine -out sendmail.msg

gem engine "gen" set.

Loading 'screen' into random state - done

C:\ssl
12. Encrypt the sendmail.msg using receiver’s public key, supplied with the receiver’s certificate.

```
C:\ssl>openssl cms -engine gem -encrypt -in sendmail.msg -out sendmail_enc.msg
certs\receiver.cer
```

13. Decrypt the sendmail_enc.msg using receiver’s private key.

```
C:\ssl>openssl cms -engine gem -decrypt -in sendmail_enc.msg -inkey rsa-private-a216db9276268459598f163ff7163aa30cbd3d0 -keyform engine -out sendmail_dec.msg
```

Where "rsa-private-a216db9276268459598f163ff7163aa30cbd3d0" is the object label for the sender’s private key on the SafeNet Network HSM.
3 – Integrating SafeNet Luna HSM with Open SSL Using Gem Engine

14. Now verify the signature of sendmail_dec.msg using sender’s public key supplied with sender’s certificate.

C:\ssl>openssl cms -engine gem -verify -in sendmail_dec.msg -CAfile certs\ca.cer -out out.txt
certs\sender.cer

C:\ssl>openssl cms -engine gem -verify -in sendmail_dec.msg -CAfile certs\ca.cer -out out.txt
certs\sender.cer

engine "gem" set.

Verification successful
15. Open the out.txt and verify the message which you have typed in the message.txt.

![Notepad window with message contents](image)

16. Close the session when you are done with sautil.

```
c:\OpenSSL\sautil\bin>sautil -c -s 1 -i 10:11 -q
C:\ssl\ed C:\openssl\sautil\bin
C:\openssl\sautil\bin>sautil -c -e -i 10:11 -q
Copyright (C) 2009 SafeNet, Inc. All rights reserved.
sautil is the property of SafeNet, Inc. and is provided to our customers for
the purpose of diagnostic and development only. Any re-distribution of this
program in whole or in part is a violation of the license agreement.

Close ok.
c:\openssl\sautil\bin>
```

Integration of Open SSL with SafeNet Luna HSM using Gem engine is completed successfully. All keys are generated on SafeNet Network HSM and accessible while using Open SSL commands for sign, verify, encrypt, and decrypt the data.
Troubleshooting Tips

Problem – 1

SAUTIL not found on your system. When the Open SSL is already installed or you have patched the luna engine in open ssl source code then SAUTIL utility will not installed on your system.

Solution:

SAUTIL installed by default with open ssl when you installed the open ssl from Apache Toolkit. In other cases when it is not installed you can follow the below steps to install it.

1. Traverse to the toolkit and untar the luna-samples file.
2. Under the luna-samples you will find sautil. Under sautil you will find the sautil.c file.
3. Open the sautil.c and search the “#define LUNA_OSSL_ECDSA (1)” and disable it.
4. Save and close the file and run the following commands in sautil directory under luna-samples.

   # ./configure.sh
   # make

Now check /usr/local/sautil/bin/sautil is installed on your system.

Problem – 2

Open SSL source installation when fail with the following error using gembuild script provided with Open SSL toolkit:

make[2]: *** No rule to make target `../../include/openssl/idea.h', needed by `e_idea.o'.  Stop.
make[2]: Leaving directory `/home/openssl-1.0.1s/crypto/evp'
make[1]: *** [subdirs] Error 1
make[1]: Leaving directory `/home/openssl-1.0.1s/crypto'
make: *** [build_crypto] Error 1
ERROR: There was an issue compiling OpenSSL. See /home/gemengine-1.1/logs/openssl-build.log for details.

Solution:

Add “make depend” to the gembuild script on line 453 right after the make clean. The error is due to the change in Open SSL that has happened recently which required “make depend” to be run before “make install”.