

Microsoft SQL Server

Integration Guide

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: 007-011108-001, Rev. U

Release Date: March 2016

Contents

Preface	5
Scope	5
Document Conventions	5
Command Syntax and Typeface Conventions	6
Support Contacts	7
1 Introduction	8
Overview	8
3rd Party Application Details	8
Supported Platforms	9
Prerequisites	12
SafeNet Network HSM Setup	12
SafeNet PCI HSM Setup	12
SafeNet Luna HSM Configuration Settings	13
Luna EKM Setup	13
SQL Server Setup	14
2 Integrating SafeNet Luna HSM with SQL Server	15
Setting up SafeNet Luna HSM with SQL Server	15
Enabling EKM Provider option	15
Registering Luna EKM Provider	16
Setting up Credential for Luna EKM Provider	16
Using Luna EKM Provider	17
Migration from SQL EKM to Luna EKM	22
Using Extensible Key Management on a SQL Server Failover Cluster	25
References	25
3 Integrating SafeNet Luna HSM with SQL Server High Availability (Always On) Group ..	26
SQL Server Setup	26
Enabling EKM Provider Option	26
Registering Luna EKM Provider	27
Setting up Credential for Luna EKM Provider	27
Creating the Always On Availability Group	28
Creating the Encryption Keys for Availability Group Database	29
Add the encrypted database in to the availability group	34
4 Integrating SafeNet Luna HSM with SQL Server Always Encrypted	36
SafeNet Luna HSM with SQL Server “Always Encrypted”	36
SQL Server Setup	37
Configure SafeNet CSP	37

Setup “Always Encrypted” using SafeNet Luna HSM	38
Remarks	51
5 Troubleshooting Tips.....	53

Preface

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

Scope

This guide provides instructions for setting up a small test lab with Microsoft SQL Server running with SafeNet Luna HSM for securing the Master Keys. It explains how to install and configure the software required for setting up Microsoft SQL Server while storing master key on SafeNet Luna HSM.

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

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

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.

Contact Method	Contact Information	
Address	Gemalto, Inc. 4690 Millennium Drive Belcamp, Maryland 21017, USA	
Phone	US	1-800-545-6608
	International	1-410-931-7520
Technical Support Customer Portal	https://serviceportal.safenet-inc.com 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.	

1

Introduction

Overview

SQL Server enables use of HSM devices for storage of keys and cryptographic operations such as key creation, deletion, encryption, decryption etc. by using the Extensible Key Management (EKM) feature. This is a more secure solution because the encryption keys do not reside with encryption data. Data can be encrypted by using encryption keys that only the database user has access to on the external EKM/HSM module. SafeNet provides Luna EKM which includes the EKM Provider Library for SafeNet Luna HSM that can be used to setup Extensible Key Management (EKM) for SQL Server and facilitate the integration with SafeNet Luna HSM.

This document provides low-level details of how the SafeNet Luna Hardware Security Modules (HSM) can be made to work with SQL Server. You must have basic knowledge of using SQL Server and HSM concepts to make full use of the recommendations in this document. This document is intended for:

- Developers and enterprise IT professionals who are planning or implementing a HSM deployment. This includes IT security administrators and IT personnel.

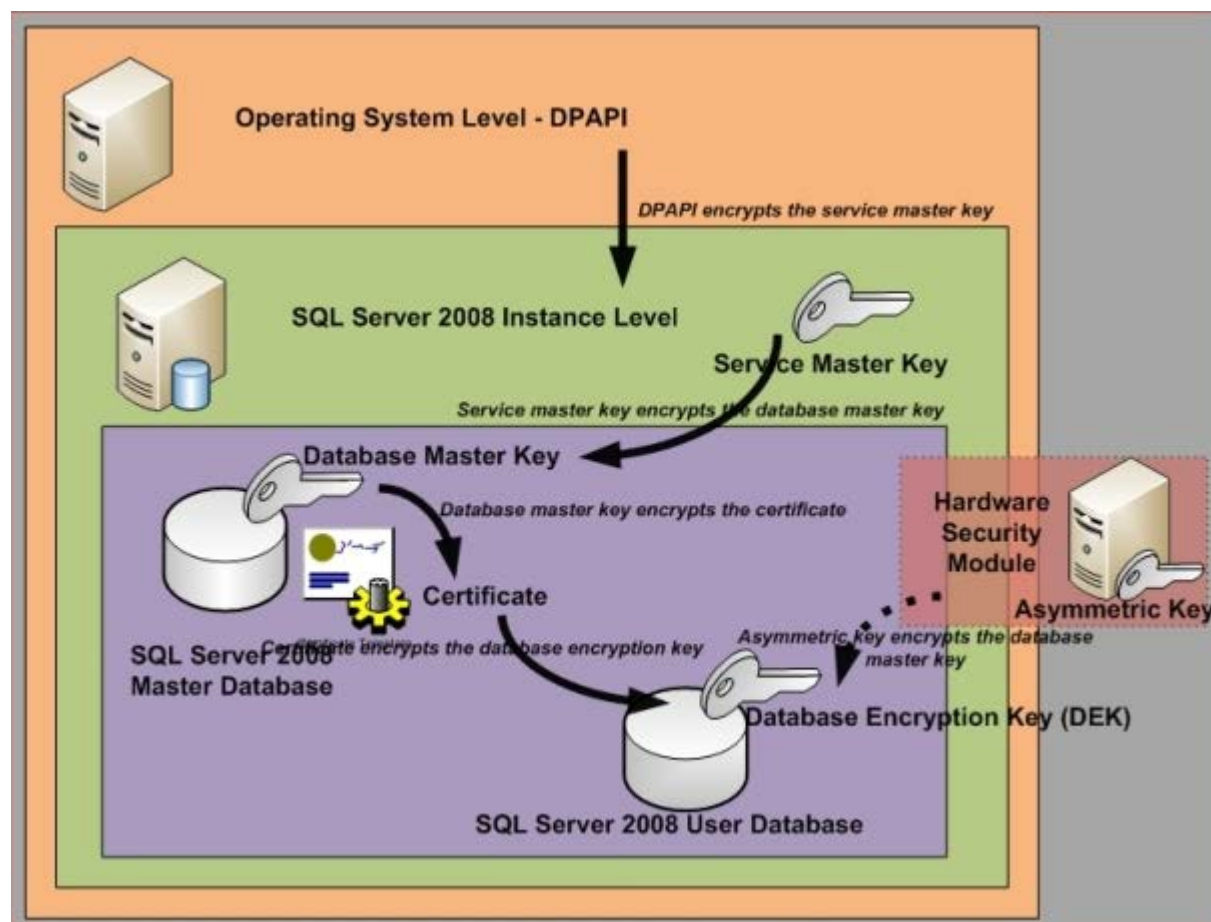
The installation is performed in several steps:

- Install and configure SafeNet Luna HSM.
- Install and configure SafeNet Luna EKM.

3rd Party Application Details

Microsoft SQL Server is a database platform for large-scale online transaction processing (OLTP), data warehousing, and e-commerce applications; it is also a business intelligence platform for data integration, analysis, and reporting solutions.

The following diagram shows the relationships between the database master key and Hardware Security Modules.



Supported Platforms

Microsoft® SQL Server® 2016 CTP3

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
Windows Server 2012 R2	Luna Client 6.2	Luna SA v6.2.0	6.10.9	EKM v1.2
Windows Server 2012 R2	Luna Client 6.1	Luna SA v6.1.0	6.10.9	EKM v1.2

Microsoft® SQL Server® 2014

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
Windows Server 2012 R2	Luna Client 6.1	Luna SA v6.1.0	6.23.0	EKM v1.2
		Luna SA v6.0	6.22.0	EKM v1.1
		Luna SA 5.4.1	6.21.0	EKM v1.1
	Luna Client 5.4.1			
Windows Server 2008 R2 SP1	Luna Client 5.4.1	Luna SA v5.4.1	6.21.0	EKM v1.1

Microsoft® SQL Server® 2014 CTP2

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
Windows Server 2012 Standard	Luna Client 5.2.1	Luna SA v5.2.1	6.10.1	EKM v1.1

Microsoft® SQL Server® 2014 CTP1

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
Windows Server 2008 R2 SP1	Luna Client 5.2.1	Luna SA v5.2.1	6.10.1	EKM v1.1

Microsoft® SQL Server® 2012

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
Windows Server 2012 R2	Luna Client 5.4.1	Luna SA v5.4.1	6.21.0	EKM v1.1
Windows Server 2012 Standard	Luna Client 5.4.1	Luna SA v5.4.1	6.21.0	EKM v1.1
	Luna Client 5.2.1	Luna SA v5.2.1	6.10.1	EKM v1.1
Windows Server 2008 R2	Luna Client 5.4.1	Luna SA v5.4.1	6.21.0	EKM v1.1

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
	Luna Client 5.2.1	Luna SA v5.2.1	6.10.1	EKM v1.1
	Luna SA CS v5.1.1	Luna SA v5.1.1	6.2.3	EKM v1.0.2
	Luna SA CS v5.1	Luna SA v5.1	6.2.1	EKM v1.0.2
	Luna PCI CS v5.0	Luna PCI v5.0	6.1.3	EKM v1.0.2
Windows Server 2008 (32-bit)	Luna SA CS v5.1.1	Luna SA v5.1.1	6.2.3	EKM v1.0.2
	Luna PCI CS v5.0	Luna PCI v5.0	6.1.3	EKM v1.0.2

Microsoft® SQL Server® 2008 R2 SP2

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
Windows Server 2008 R2	Luna Client 5.4.1	Luna SA v5.4.1	6.21.0	EKM v1.1

Microsoft® SQL Server® 2008 R2

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
Windows Server 2012 Standard	Luna Client 5.4.1	Luna SA v5.4.1	6.21.0	EKM v1.1
	Luna Client 5.2.1	Luna SA v5.2.1	6.10.1	EKM v1.1
Windows Server 2008 R2	Luna Client 5.4.1	Luna SA v5.4.1	6.21.0	EKM v1.1
	Luna Client 5.2.1	Luna SA v5.2.1	6.10.1	EKM v1.1
	Luna SA CS v5.1	Luna SA v5.1	6.2.1	EKM v1.0.2
	Luna SA CS v5.0	Luna SA v5.0	6.0.6	EKM v1.0.2
	Luna SA CS v4.4.1	Luna SA v4.4.1	4.6.8	EKM v1.0.1
	Luna PCI CS v5.0	Luna PCI v5.0	6.1.3	EKM v1.0.2
	Luna PCI CS v3.0	Luna PCI v3.0	4.7.1	EKM v1.0.1
Windows Server 2008 (32-bit / 64-bit)	Luna SA CS v5.1	Luna SA v5.1	6.2.1	EKM v1.0.1
	Luna SA CS v4.4.1	Luna SA v4.4.1	4.6.8	EKM v1.0.1
Windows Server 2003 R2 SP2 (32-bit)	Luna SA CS v5.0	Luna SA v5.0	6.0.6	EKM v1.0.1

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
Windows Server 2003 SP2 (32-bit / 64-bit)	Luna SA CS v4.4.1	Luna SA v4.4.1	4.6.8	EKM v1.0.1

Microsoft® SQL Server® 2008 SP3

Platforms Tested	Luna Client Software Version	SafeNet Luna HSM Appliance Software Version	Appliance Firmware Version	EKM Software Version
Windows Server 2012 Standard	Luna Client 5.2.1	Luna SA v5.2.1	6.10.1	EKM v1.1
Windows Server 2008 R2	Luna Client 5.4.1	Luna SA v5.4.1	6.21.0	EKM v1.1

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 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 SQL Server.
- 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 PCI HSM Setup

Refer to the SafeNet PCI documentation for installation steps and details regarding configuring and setting up the box on Windows systems. Before you get started, ensure the following:

- Initialize the HSM on the SafeNet PCI HSM appliance
- Create a partition on the HSM that will be later used by the SQL Server.
- Enable Partition "Activation" and "Auto Activation" (Partition policy settings 22 and 23 (applies to SafeNet PCI HSM with Trusted Path Authentication [which is FIPS 140-2 level 3] only).
- Use the following setting in Chrystoki Configuration file:

```
[Misc]
PE1746Enabled = 1
```

SafeNet Luna HSM Configuration Settings

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

C:\Program Files\SafeNet\LunaClient\crystoki.ini

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 Luna 6.x 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 will redirect 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 6.x. All other SafeNet Luna HSM versions do not require these setting for FIPS and non-FIPS mode.

Luna EKM Setup

A Windows-based installation program is provided to make the installation of the Luna EKM quick and easy. The installation CD can be obtained from the SafeNet Customer Connection Center.

LunaEKMConfig Utility

LunaEKM includes a command line configuration utility "LunaEKMConfig" that is used to register the Luna EKM. This command line utility gets installed in LunaEKM installation folder. It provides command to register slots, view slots and to configure log settings.

Run the following commands provided in LunaEKMConfig.

1. RegisterSlot
Register/Edit the Slot for the LunaEKM to use.
2. ViewSlots
View List of the Slots/HSM configured with this client.
3. LogSettings
Configure log settings for LunaEKM.
`LogLevel (NONE=0,INFO=1,DEBUG=2): <LogLevel>`
`LogFile name: <Name and location of LogFile>`

SQL Server Setup

SQL Server must be installed on the target machine to carry out the integration process. For a detailed installation procedure of SQL Server, refer to the Microsoft SQL Server online documentation.

2

Integrating SafeNet Luna HSM with SQL Server

Setting up SafeNet Luna HSM with SQL Server

To perform SafeNet Luna HSM integration with SQL Server, Luna EKM software provides Luna EKM Provider in the form of EKM Library (i.e. LunaEKM.dll). The Luna EKM Provider can be used if the EKM Provider option is enabled in the SQL Server. This feature is available only on the Enterprise, Developer, and Evaluation editions of SQL Server. By default, Extensible Key Management is off.

Enabling EKM Provider option

To enable this feature, use the `sp_configure` command that has the following option and value, as in the following example:

To enable the Extensible Key Management option:

1. Open the SQL Server Management Studio.
2. Connect to the SQL Server.
3. Open a query window, and then run the following command:

```
sp_configure 'show advanced', 1
GO
RECONFIGURE
GO
sp_configure 'EKM provider enabled', 1
GO
RECONFIGURE
GO
```



NOTE: If `sp_configure` command used in editions other than Enterprise, Developer or Evaluation editions, an error is received.

Registering Luna EKM Provider

To setup the Luna EKM provider, Luna EKM Software must be installed and needs to be registered with the SQL Server. Follow the below steps to create/register the provider:

To create/register the Luna EKM Provider:

1. Open the SQL Server Management Studio.
2. Connect to the SQL Server.
3. Open a query window, and then run the following command:

```
CREATE CRYPTOGRAPHIC PROVIDER <Name of Cryptographic Provider>
FROM FILE = '<Location of Luna EKM Provider Library>'
```

 where CRYPTOGRAPHIC PROVIDER can be any user defined unique name.
4. To view the list of EKM providers:

```
SELECT [provider_id]
[name]
,[guid]
,[version]
,[dll_path]
,[is_enabled]
FROM [model].[sys].[cryptographic_providers]
```

5. To view the provider properties:

```
SELECT [provider_id],[guid],[provider_version]
,[sqlcrypt_version]
,[friendly_name]
,[authentication_type]
,[symmetric_key_support]
,[symmetric_key_persistence]
,[symmetric_key_export]
,[symmetric_key_import]
,[asymmetric_key_support]
,[asymmetric_key_persistence]
,[asymmetric_key_export]
,[asymmetric_key_import]
FROM [master].[sys].[dm_cryptographic_provider_properties]
```

Setting up Credential for Luna EKM Provider

The next step is to create a CREDENTIAL for the Luna EKM Provider. Then the CREDENTIAL must be mapped to SQL User or Login to be able to use the Luna EKM Provider. A CREDENTIAL is basically used to access any external SQL Server resource such as SafeNet Luna HSM. Follow the below steps to create/map credential for the provider:

To create/map the CREDENTIAL for Luna EKM Provider:

1. Open a query window, and then run the following command:

```
CREATE CREDENTIAL <Name of credential>
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
FOR CRYPTOGRAPHIC PROVIDER LunaEKMPProvider
```

 Where CREDENTIAL and IDENTITY can be any user defined unique name.



NOTE: If the partition on a PED based SafeNet PCI HSM not having a partition challenge, then the value for SECRET should be a space character (e.g., ' '), else enter the partition challenge.

2. To map the LunaEKMcred with SQL User or Login:

```
ALTER LOGIN [Domain\Login Name]
ADD CREDENTIAL <Name of Credential created>
```



NOTE: EKM session needs to be re-opened in case the user changes the HSM slot or the client machine is deleted from SafeNet Luna HSM and registered again or network disconnection.

Using Luna EKM Provider

The Luna EKM provider is now ready to use, it can be used to create/drop symmetric and asymmetric keys to/from the Luna partition and can perform encryption/decryption using these keys. Follow the below steps to exercise the cryptographic capabilities of SafeNet Luna HSM from the SQL Server:

Creating Symmetric Keys on SafeNet Luna HSM

Following types of symmetric key can be created on SafeNet Luna HSM from the SQL Server:

- RC2
- RC4*
- RC4_128*
- DES
- Triple_DES
- Triple_DES_3KEY
- AES_128
- AES_192
- AES_256

* *Depreciated in SQL Server 2012.*

In the examples below, AES algorithm will be used for the symmetric key operation. In order to test other algorithms, AES ALGORITHM tag can be replaced with any of the other tags from the above list.

To create the symmetric key using Luna EKM Provider:

1. Execute the following command from the SQL query window:

```
CREATE SYMMETRIC KEY SQL_EKM_AES_256_Key
FROM Provider LunaEKMPProvider
WITH ALGORITHM = AES_256,
PROVIDER_KEY_NAME = 'EKM_AES_256_Key',
CREATION_DISPOSITION=CREATE_NEW
```



NOTE: Once a key is created on the SafeNet Luna HSM, it can be used or referred by its name from the SQL Server, for example in the above said test case, `SQL_EKM_AES_256_Key` is the unique name of the key in the SQL Server which can be used to perform crypto operation (encrypt/decrypt) using the key on the SafeNet Luna HSM.

Viewing Symmetric Keys

To view the symmetric keys for Luna EKM Provider:

1. Execute the following command from the SQL query window:

```
SELECT * FROM [master].[sys].[symmetric_keys]
```

Encryption using Symmetric Keys

To encrypt using symmetric key:

1. Create a test Table in the MASTER database with fields.

```
Create Table test(
id numeric(10),
name varchar (50),
data varchar (max),)
```

2. Execute the following command from the SQL query window:

```
INSERT INTO dbo.test
values( 1,'some text',
EncryptByKey(Key_GUID('SQL_EKM_AES_256_Key'), 'text to be encrypted'))
```

Decryption using Symmetric Keys

To decrypt using symmetric key:

1. Execute the following command from the SQL query window:

```
SELECT id,name,CONVERT(varchar(MAX),
DecryptByKey(data))
FROM dbo.test where id =1
```

Dropping Symmetric Keys

To drop the symmetric key:

1. Execute the following command from the SQL query window:

```
DROP SYMMETRIC KEY SQL_EKM_AES_256_Key REMOVE PROVIDER KEY
```

This command will drop the key from the SQL Server as well as from the SafeNet Luna HSM.

Creating Asymmetric Keys on SafeNet Luna HSM

Following types of asymmetric key can be created on SafeNet Luna HSM from the SQL Server:

- RSA_512
- RSA_1024
- RSA_2048

In the examples below, RSA_2048 algorithm will be used for the asymmetric key operation. In order to test other algorithms, RSA_2048 ALGORITHM tag can be replaced with any of the other tags from the above list.

To create the asymmetric key using Luna EKM Provider:

1. Execute the following command from the SQL query window:

```
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key
FROM Provider LunaEKMProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key',
CREATION_DISPOSITION=CREATE_NEW
```



NOTE: Once a key is created on the SafeNet Luna HSM, it can be used or referred by its name from the SQL Server, for example in the above said test case, SQL_EKM_RSA_2048_Key is the unique name of the key in the SQL Server which can be used to perform crypto operation (encrypt/decrypt) using the key on the SafeNet Luna HSM.

Viewing Asymmetric Keys

To view the asymmetric keys for Luna EKM Provider:

1. Execute the following command from the SQL query window:

```
SELECT * FROM [master].[sys].[asymmetric_keys]
```

Encryption using Asymmetric Keys

To encrypt using asymmetric key:

1. Create a test Table in the MASTER database with fields:

```
Create Table test(
id numeric(10),
name varchar (50),
data varchar (max),)
```

2. Execute the following command from the SQL query window:

```
INSERT INTO dbo.test
values ( 1,'some text',
EncryptByAsymKey (AsymKey_Id ('SQL_EKM_RSA_2048_Key'), 'text to be encrypted'))
```

Decryption using Asymmetric Keys

To decrypt using asymmetric key:

1. Execute the following command from the SQL query window:

```
SELECT id, name, CONVERT (varchar (MAX),
```

```
DecryptByAsymKey (AsymKey_Id ('SQL_EKM_RSA_2048_Key'), data))
FROM dbo.test where id =1
```

Dropping Asymmetric Keys

To drop the asymmetric key:

1. Execute the following command from the SQL query window:

```
DROP ASYMMETRIC KEY SQL_EKM_RSA_2048_Key REMOVE PROVIDER KEY
```

This command will drop the key from the SQL Server as well as from the SafeNet Luna HSM.

Creating Symmetric Key Encrypted by Asymmetric Key on SafeNet Luna HSM

To create a symmetric Key encrypted by an asymmetric Key on HSM

1. Execute the following command from SQL query window

```
Create SYMMETRIC KEY key1
WITH ALGORITHM = AES_256
ENCRYPTION BY Asymmetric Key SQL_EKM_RSA_2048_Key;
```

SQL_EKM_RSA_2048_Key is an existing asymmetric key on SafeNet Luna HSM. To generate the key, refer Creating Asymmetric Keys on SafeNet Luna HSM.

2. Before using the key, you need to open the key. Execute the following command to open the symmetric key.

```
OPEN SYMMETRIC KEY key1 DECRYPTION BY Asymmetric Key SQL_EKM_RSA_2048_Key;
```

3. Create a test Table in the MASTER database with fields:

```
Create Table test(
id numeric(10),
name varchar (50),
data varchar (max),)
```

4. To encrypt using symmetric key, execute the following command from the SQL query window:

```
INSERT INTO dbo.test
values ( 1,'some text',
Encryptbykey(KEY_GUID('key1'),'text to be encrypted'))
```

5. To decrypt using symmetric key, execute the following command from the SQL query window:

```
SELECT id,name,CONVERT(varchar(MAX),
DecryptByKey(data))
FROM dbo.test where id =1
```

6. Close the symmetric key by executing command

```
CLOSE SYMMETRIC KEY key1
```

Enable Transparent Database Encryption using Asymmetric key on SafeNet Luna HSM

To enable Transparent Database Encryption using asymmetric key on HSM

1. Create an asymmetric key using Luna EKM Provider.

```
Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE
```

```
FROM Provider LunaEKMPProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE',
CREATION_DISPOSITION=CREATE_NEW
```

2. Create a credential for Luna EKM Provider.

```
CREATE CREDENTIAL <Name of credential>
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
FOR CRYPTOGRAPHIC PROVIDER LunaEKMPProvider
```

3. Create a login based on the asymmetric key created above.

```
CREATE LOGIN <Name of login>
FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE;
```

4. Map the credential created above to the login created above.

```
ALTER LOGIN <Name of Login>
ADD CREDENTIAL <Name of credential>;
```

5. Create a Database Encryption Key.



NOTE: Database encryption operations cannot be performed for 'master', 'model', 'tempdb', 'msdb', or 'resource' databases.

```
CREATE DATABASE TDE;
Use tde;
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE;
```

6. Enable Transparent Database Encryption.

```
ALTER DATABASE TDE
SET ENCRYPTION ON;
```

7. To query the status of database encryption and its percentage completion.

```
SELECT DB_NAME (e.database_id) AS DatabaseName,
e.database_id,
e.encryption_state,
CASE e.encryption_state
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc,
c.name,
e.percent_complete
FROM sys.dm_database_encryption_keys AS e
LEFT JOIN master.sys.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint
```

Transparent Database Encryption (TDE) Key Rotation

1. Create an asymmetric key using Luna EKM Provider.

```

Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot
FROM Provider LunaEKMPProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE_Rot',
CREATION_DISPOSITION=CREATE_NEW

```

2. Create a credential for Luna EKM Provider.

```

CREATE CREDENTIAL <Name of credential>
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
FOR CRYPTOGRAPHIC PROVIDER LunaEKMPProvider

```

3. Create a login based on the asymmetric key created above.

```

CREATE LOGIN <Name of login>
FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot;

```

4. Map the credential created above to the login created above.

```

ALTER LOGIN <Name of Login>
ADD CREDENTIAL <Name of credential>;

```

5. Enable Transparent Database Encryption Key Rotation

```

Use tde;
ALTER DATABASE ENCRYPTION KEY
REGENERATE
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot;
go
SELECT * FROM sys.dm_database_encryption_keys
go

```

6. To query the status of database encryption, TDE key change and its percentage completion.

```

SELECT DB_NAME (e.database_id) AS DatabaseName,
e.database_id,
e.encryption_state,
CASE e.encryption_state
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc,
c.name,
e.percent_complete
FROM sys.dm_database_encryption_keys AS e
LEFT JOIN master.sys.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint

```

Migration from SQL EKM to Luna EKM

If you have enabled the Transparent Data Encryption for any database (let AdventureWorks) using the SQL EKM and now want to migrate TDE from SQL EKM to Luna EKM. Previously your database master key is encrypted by either Certificate or Asymmetric Key which was generated in SQL and after enabling the TDE with

Luna EKM. Now you want to generate a new database master key encrypted by asymmetric key generated on Luna HSM. To do this you need to perform the following steps:



NOTE: We have tested these steps in SQL Server 2014 with Luna Client 5.4.1.

1. Decrypt the database (let AdventureWorks)

```
USE master;
ALTER DATABASE AdventureWorks
SET ENCRYPTION OFF;
GO
```

2. Take the backup of database and transaction logs. When the backup completed restart the SQL database.

3. Create an asymmetric key using Luna EKM Provider.

```
Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_AW
FROM Provider LunaEKMProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_AW',
CREATION_DISPOSITION=CREATE_NEW
```

4. Create a credential for Luna EKM Provider.

```
CREATE CREDENTIAL <Name of credential>
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
```

5. Create a login based on the asymmetric key created above.

```
CREATE LOGIN <Name of login>
FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_AW;
```

6. Map the credential created above to the login created above.

```
ALTER LOGIN <Name of Login>
ADD CREDENTIAL <Name of credential>;
```

7. Create or Regenerate a Database Encryption Key.

```
USE AdventureWorks;
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_AW;
```

Or

```
USE AdventureWorks;
ALTER DATABASE ENCRYPTION KEY
REGENERATE WITH ALGORITHM = AES_192
ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_AW;
```

8. Enable Transparent Database Encryption:

```
ALTER DATABASE AdventureWorks
SET ENCRYPTION ON;
```

9. To query the status of database encryption and its percentage completion.

```
SELECT DB_NAME(e.database_id) AS DatabaseName,
e.database_id,
e.encryption_state,
```

```

CASE e.encryption_state
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc,
c.name,
e.percent_complete
FROM sys.dm_database_encryption_keys AS e
LEFT JOIN master.sys.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint

```

SQLQuery1.sql - SQL.master (SQL\Administrator (53)) - Microsoft SQL Server Management Studio (Administrator)

Object Explorer: SQL (SQL Server 12.0.2000 - SQL\Administrator)

Query: `SELECT DB_NAME(e.database_id) AS DatabaseName, e.database_id, e.encryption_state, CASE e.encryption_state WHEN 0 THEN 'No database encryption key present, no encryption' WHEN 1 THEN 'Unencrypted' WHEN 2 THEN 'Encryption in progress' WHEN 3 THEN 'Encrypted' WHEN 4 THEN 'Key change in progress' WHEN 5 THEN 'Decryption in progress' END AS encryption_state_desc, c.name, e.percent_complete FROM sys.dm_database_encryption_keys AS e LEFT JOIN master.sys.asymmetric_keys AS c ON e.encryptor_thumbprint = c.thumbprint`

Results:

	DatabaseName	database_id	encryption_state	encryption_state_desc	name	percent
1	tempdb	2	3	Encrypted	NULL	0
2	AdventureWorks	7	3	Encrypted	SQL_EKM_RSA_2048_Key_AW	0
3	tde	8	3	Encrypted	SQL_EKM_RSA_2048_Key_TDE_Rot	0

Messages: Query executed successfully. SQL (12.0 RTM) SQL\Administrator (53) master 00:00:00 3 rows

Properties: Current connection parameters

- Aggregate Status
 - Connection failed
 - Elapsed time: 00:00:00.093
 - Finish time: 8/27/2014 1:24:04 PM
 - Name: SQL
 - Rows returned: 3
 - Start time: 8/27/2014 1:24:04 PM
 - State: Open
- Connection
 - Connection name: SQL (SQL\Administrator)
- Connection Details
 - Connection elapsed: 00:00:00.093
 - Connection finish: 8/27/2014 1:24:04 PM
 - Connection rows: 3
 - Connection start: 8/27/2014 1:24:04 PM
 - Connection state: Open
 - Display name: SQL
 - Login name: SQL\Administrator
 - Server name: SQL
 - Server version: 12.0.2000
 - Session Tracing ID: 53
- Name
 - The name of the connection.

Using Extensible Key Management on a SQL Server Failover Cluster

This section focuses on the preparation of the environment for 2-node SQL Server Cluster in Windows Server 2008 R2.

1. Refer to the SQL Server documentation to install a failover cluster.

Setting up a Shared Storage

To set up a shared storage disk for SQL Server Cluster, refer to the configuration procedures that apply for shared storage solution. Plan the size of the shared storage depending on the number of certificates that are required to be enrolled.

2. Once the cluster is up and running, install SafeNet Network HSM client on both the nodes.
3. Configure and setup the appliance on both the nodes and register the same partition on both node of SQL Server Cluster.
4. Install Luna EKM client on both the nodes.
5. Configure the Luna EKM provider on both the nodes.
6. Open the SQL Server management studio to register the Luna EKM provider on the first node.
7. Setup the credential on the first node.
8. Create some keys using the Luna EKM provider on the first node.
9. Create a table and encrypt some column with the Luna EKM key with the first node.
10. Shutdown the first node.
11. Login to the second node and decrypt the data encrypted on the first node.
12. Data is decrypted successfully.

Extensible Key Management using Luna EKM is working fine on a SQL Server cluster.

References

1. Understanding Extensible Key Management (EKM) <http://technet.microsoft.com/en-us/library/bb895340.aspx>
2. Creating a cryptographic provider within SQL Server <http://technet.microsoft.com/en-us/library/bb677184.aspx>
3. EKM provider enabled Option <http://technet.microsoft.com/en-us/library/bb630320.aspx>
4. Generating a symmetric key <http://technet.microsoft.com/en-us/library/ms188357.aspx>
5. Choosing an Encryption Algorithm <http://technet.microsoft.com/en-us/library/ms345262.aspx>
6. How to: Enable TDE using EKM <http://technet.microsoft.com/en-us/library/cc645957.aspx>

Integrating SafeNet Luna HSM with SQL Server High Availability (Always On) Group

The Always On Availability Groups feature is a high-availability and disaster recovery solution that provides an enterprise-level alternative to database mirroring. Introduced in SQL Server 2012, Always On Availability Groups maximizes the availability of a set of user databases for an enterprise. An availability group supports a failover environment for a discrete set of user databases, known as availability databases that fail over together. An availability group supports a set of read-write primary databases and one to eight sets of corresponding secondary databases. Optionally, secondary databases can be made available for read-only access and/or some backup operations.

To perform SafeNet Luna HSM integration with SQL Server, Luna EKM software provides Luna EKM Provider in the form of EKM Library (i.e. LunaEKM.dll). The Luna EKM Provider can be used if the EKM Provider option is enabled in the SQL Server. This feature is available only on the Enterprise, Developer, and Evaluation editions of SQL Server. By default, Extensible Key Management is off.



NOTE: To setup Luna Client and Luna EKM, refer to Chapter 1. Luna Client and Luna EKM must be setup on all nodes of SQL Server which needs to be added in the “Always On” availability group. All nodes must be registered with the same partition of SafeNet Luna HSM.

SQL Server Setup

SQL Server must be installed on the target machines to carry out the integration process and all nodes have WFCS. For a detailed installation procedure of SQL Server, Always On group pre-requisites, and recommendations refer to the Microsoft SQL Server online documentation.

Enabling EKM Provider Option

To enable this feature, use the `sp_configure` command on both the SQL Server nodes that has the following option and value, as in the following example:

To enable the Extensible Key Management option:

1. Open the SQL Server Management Studio.
2. Connect to the SQL Server.
3. Open a query window, and then run the following command:

```
sp_configure 'show advanced', 1
GO
RECONFIGURE
```

```
GO
sp_configure 'EKM provider enabled', 1
GO
RECONFIGURE
GO
```



NOTE: If `sp_configure` command used in editions other than Enterprise, Developer or Evaluation editions, an error is received.

Registering Luna EKM Provider

To setup the Luna EKM provider, Luna EKM Software must be installed and needs to be registered with the SQL Server. Follow the below steps to create/register the provider on all SQL Server nodes:

To create/register the Luna EKM Provider:

1. Open the SQL Server Management Studio.
2. Connect to the SQL Server.
3. Open a query window, and then run the following command:

```
CREATE CRYPTOGRAPHIC PROVIDER <Name of Cryptographic Provider>
FROM FILE = '<Location of Luna EKM Provider Library>'
where CRYPTOGRAPHIC PROVIDER can be any user defined unique name.
```

4. To view the list of EKM providers:

```
SELECT [provider_id]
,[name]
,[guid]
,[version]
,[dll_path]
,[is_enabled]
FROM [model].[sys].[cryptographic_providers]
```

5. To view the provider properties:

```
SELECT [provider_id],[guid],[provider_version]
,[sqlcrypt_version]
,[friendly_name]
,[authentication_type]
,[symmetric_key_support]
,[symmetric_key_persistence]
,[symmetric_key_export]
,[symmetric_key_import]
,[asymmetric_key_support]
,[asymmetric_key_persistence]
,[asymmetric_key_export]
,[asymmetric_key_import]
FROM [master].[sys].[dm_cryptographic_provider_properties]
```

Setting up Credential for Luna EKM Provider

The next step is to create a CREDENTIAL for the Luna EKM Provider. The CREDENTIAL must be mapped to SQL User or Login to be able to use the Luna EKM Provider. A CREDENTIAL is basically used to access any

external SQL Server resource such as SafeNet Luna HSM. Follow the below steps to create\map credential for the provider on all SQL Server nodes:

To create\map the CREDENTIAL for Luna EKM Provider:

1. Open a query window, and run the following command:

```
CREATE CREDENTIAL <Name of credential>  
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'  
FOR CRYPTOGRAPHIC PROVIDER LunaEKMPProvider
```

Where CREDENTIAL and IDENTITY can be any user defined unique name.

2. To map the LunaEKMCred with SQL User or Login:

```
ALTER LOGIN [Domain\Login Name]  
ADD CREDENTIAL <Name of Credential created>
```



NOTE: It is assumed that a common user is used on all SQL Server nodes that become the part of “Always On” availability group. Above steps (listed under Enabling EKM Provider option, Registering Luna EKM Provider, and Setting up Credential for Luna EKM Provider) needs to be executed on all secondary nodes as well.



NOTE: EKM session needs to be re-opened in case where user changes the SafeNet Luna HSM slot or the client machine is deleted from SafeNet Luna HSM and registered again or network disconnection.

Creating the Always On Availability Group

Open the **Microsoft SQL Server Management Studio** on primary replica and create a database first and then take full backup of that database on a shared location that is accessible by all SQL Server nodes. Open the **Always On Availability Group Creation** wizard and follow the instructions to create the Always On Availability Group. For detailed steps and prerequisites, refer the Microsoft online documentation for creating the Always On Availability Group.

After the successful creation of the group, the dashboard displays all the participating nodes. An example of a dashboard is shown below. For demo purpose, two nodes are added: primary and secondary.

The screenshot displays the 'Always On Availability Groups' dashboard in SQL Server Management Studio. The main pane shows the following details:

- Availability group state:** Healthy
- Primary instance:** NODE1
- Failover mode:** Automatic
- Cluster state:** SQLCluster (Normal Quorum)

Below this, a table lists the availability replicas:

Name	Role	Failover Mode	Synchronization State	Issues
NODE1	Primary	Automatic	Synchronized	
NODE2	Secondary	Automatic	Synchronized	

At the bottom, a 'Group by' section shows details for each node's replicas:

Name	Replica	Synchronization State	Failover Readiness	Issues
NODE1	HSM	Synchronized	No Data Loss	
	tde	Synchronized	No Data Loss	
NODE2	HSM	Synchronized	No Data Loss	
	tde	Synchronized	No Data Loss	

The right-hand Properties pane shows 'Current connection parameters' for the selected connection, including details like 'Connection name', 'Elapsed time', 'Connection finish', 'Connection start', 'Connection state', 'Display name', 'Login name', 'Server name', 'Server version', 'Session Tracing ID', and 'SPID'.

Creating the Encryption Keys for Availability Group Database.

Creating Symmetric Keys on SafeNet Luna HSM

Following types of symmetric key can be created on SafeNet Luna HSM from the SQL Server:

- RC2
- RC4*
- RC4_128*
- DES
- Triple_DES
- Triple_DES_3KEY
- AES_128
- AES_192
- AES_256

* Depreciated in SQL Server 2012.

In the examples below, AES algorithm will be used for the symmetric key operation. In order to test other algorithms, AES ALGORITHM tag can be replaced with any of the other tags from the above list.

To create the symmetric key using Luna EKM Provider on availability database (e.g. HSMDB) open the SMS on primary replica:

1. Execute the following command from the SQL query window:

```
USE HSMDB;
```

2. Execute the following command from the SQL query window:

```
CREATE SYMMETRIC KEY SQL_EKM_AES_256_Key  
FROM Provider LunaEKMPProvider  
WITH ALGORITHM = AES_256,  
PROVIDER_KEY_NAME = 'EKM_AES_256_Key',  
CREATION_DISPOSITION=CREATE_NEW
```



NOTE: Once a key is created on the SafeNet Luna HSM, it can be used or referred by its name from the SQL Server, for example in the above said test case, SQL_EKM_AES_256_Key is the unique name of the key in the SQL Server that is used to perform crypto operation (encrypt/decrypt) using the key on the SafeNet Luna HSM.

Viewing Symmetric Keys

To view the symmetric keys for Luna EKM Provider:

1. Execute the following command from the SQL query window:

```
SELECT * FROM [hsmdb].[sys].[symmetric_keys]
```

Encryption using Symmetric Keys

To encrypt using symmetric key:

1. Create a test Table in the HSMDB database with fields.

```
Create Table test(  
id numeric(10),  
name varchar (50),  
data varchar (max),)
```

2. Execute the following command from the SQL query window:

```
INSERT INTO dbo.test  
values( 1,'some text',  
EncryptByKey(Key_GUID('SQL_EKM_AES_256_Key'), 'text to be encrypted'))
```

Decryption using Symmetric Keys

To decrypt using symmetric key:

1. Execute the following command from the SQL query window:

```
SELECT id,name,CONVERT(varchar(MAX),  
DecryptByKey(data))
```

```
FROM dbo.test where id =1
```

- Now execute the above command on secondary replica and verify that the output is same as primary replica.

Creating Asymmetric Keys on SafeNet Luna HSM

Following types of asymmetric key can be created on SafeNet Luna HSM from the SQL Server:

- RSA_512
- RSA_1024
- RSA_2048

In the examples below, RSA_2048 algorithm will be used for the asymmetric key operation. In order to test other algorithms, RSA_2048 ALGORITHM tag can be replaced with any of the other tags from the above list.

To create the asymmetric key using Luna EKM Provider open the SMS on primary replica:

- Execute the following command from the SQL query window:

```
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key
FROM Provider LunaEKMProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key',
CREATION_DISPOSITION=CREATE_NEW
```



NOTE: Once a key is created on the SafeNet Luna HSM, it can be used or referred by its name from the SQL Server, for example in the above said test case, SQL_EKM_RSA_2048_Key is the unique name of the key in the SQL Server which can be used to perform crypto operation (encrypt/decrypt) using the key on the SafeNet Luna HSM.

Viewing Asymmetric Keys

To view the asymmetric keys for Luna EKM Provider:

- Execute the following command from the SQL query window:

```
SELECT * FROM [hsmdb].[sys].[asymmetric_keys]
```

Encryption using Asymmetric Keys

To encrypt using asymmetric key:

- Execute the following command from the SQL query window:

```
INSERT INTO dbo.test
values ( 2,'some text',
EncryptByAsymKey (AsymKey_Id ('SQL_EKM_RSA_2048_Key'), 'text to be encrypted'))
```

Decryption using Asymmetric Keys

To decrypt using asymmetric key:

1. Execute the following command from the SQL query window:

```
SELECT id, name, CONVERT (varchar (MAX),
DecryptByAsymKey (AsymKey_Id ('SQL_EKM_RSA_2048_Key'), data))
FROM dbo.test where id =2
```

2. Now execute the above command on secondary replica and verify that the output is same as primary replica.

Creating Symmetric Key Encrypted by Asymmetric Key on SafeNet Luna HSM

To create a symmetric Key encrypted by an asymmetric Key on SafeNet Luna HSM, open the SMS on primary replica:

1. Execute the following command from SQL query window

```
Create SYMMETRIC KEY key1
WITH ALGORITHM = AES_256
ENCRYPTION BY Asymmetric Key SQL_EKM_RSA_2048_Key;
where "SQL_EKM_RSA_2048_Key" is an existing asymmetric key.
```

2. Before using the key you need to open the key. Following command can be executed to open the symmetric key.

```
OPEN SYMMETRIC KEY key1 DECRYPTION BY Asymmetric Key SQL_EKM_RSA_1024_Key;
```

3. Encrypt the data using the key1

```
INSERT INTO dbo.test
values ( 3,'some text',
Encryptbykey(KEY_GUID('Key1'), 'text to be encrypted'))
```

4. Decrypt the data using the key1

```
SELECT id,name,CONVERT(varchar(MAX),
DecryptByKey(data))
FROM dbo.test where id =3
```

5. Close the symmetric key by executing command

```
CLOSE SYMMETRIC KEY key1
```

6. Now execute the above steps 2-5 on secondary replica and verify that the output is same as primary replica.

Enable Transparent Database Encryption using Asymmetric key on SafeNet Luna HSM

To enable Transparent Database Encryption using asymmetric key on SafeNet Luna HSM, open the SMS on primary replica:

1. Create an asymmetric key using Luna EKM Provider on primary replica.

```
Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE
FROM Provider LunaEKMPProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE',
CREATION_DISPOSITION=CREATE_NEW
```

2. Create same asymmetric key using Luna EKM Provider on secondary replica.

```
Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE
```



```
FROM Provider LunaEKMPProvider
WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE',
CREATION_DISPOSITION=OPEN_EXISTING
```

3. Create a credential for Luna EKM Provider.

```
CREATE CREDENTIAL <Name of credential>
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
FOR CRYPTOGRAPHIC PROVIDER LunaEKMPProvider
```

4. Create a login based on the asymmetric key created above.

```
CREATE LOGIN <Name of login>
FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE;
```

5. Map the credential created above to the login created above.

```
ALTER LOGIN <Name of Login>
ADD CREDENTIAL <Name of credential>;
```

6. Execute steps 2-5 for all secondary nodes. It is required because TDE encryption key, credential, and login are created in the master database and it is not a part of **Availability Groups**. Therefore, you need to create the same key, credential and login in the master database of all the secondary nodes to access the encrypted tables created in the database.

7. Create a Database Encryption Key on primary replica.



NOTE: Database encryption operations cannot be performed for 'master', 'model', 'tempdb', 'msdb', or 'resource' databases.

```
CREATE DATABASE TDE;
Use tde;
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE;
```

8. Enable Transparent Database Encryption:

```
ALTER DATABASE TDE
SET ENCRYPTION ON;
```

9. To query the status of database encryption and its percentage completion.

```
SELECT DB_NAME (e.database_id) AS DatabaseName,
e.database_id,
e.encryption_state,
CASE e.encryption_state
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc,
c.name,
e.percent_complete
FROM sys.dm_database_encryption_keys AS e
LEFT JOIN master.sys.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint
```

Add the encrypted database in to the availability group

Before adding the already encrypted database into availability group, take the full backup of the database on shared location that is accessible by all secondary nodes.

To add the encrypted database into the availability group, open the SMS on primary replica:

1. Add the database (e.g. TDE) into the availability group (e.g. AGroup).

```
use master;
ALTER AVAILABILITY GROUP AGroup ADD DATABASE tde;
GO
```

Above command add the database into the availability group but it will not available on secondary replica. To synchronize the database you need to restore the database on secondary replica.

2. Restore the database on secondary replica from the location where you have backed up with “RESTORE WITH NORECOVERY” option.
3. Add the database on secondary replica using the following SQL command:

```
use master;
ALTER DATABASE tde SET HADR AVAILABILITY GROUP = AGroup;
```

4. To query the status of database encryption and its percentage completion on secondary node.

```
SELECT DB_NAME (e.database_id) AS DatabaseName,
e.database_id,
e.encryption_state,
CASE e.encryption_state
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc,
c.name,
e.percent_complete
FROM sys.dm_database_encryption_keys AS e
LEFT JOIN master.sys.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint
```

Transparent Database Encryption (TDE) Key Rotation

1. Create an asymmetric key using Luna EKM Provider on primary replica.

```
Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot
FROM Provider LunaEKMProvider
WITH ALGORITHM = RSA_2048,
PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE_Rot',
CREATION_DISPOSITION=CREATE_NEW
```

2. Create same asymmetric key using Luna EKM Provider on secondary replica.

```
Use master;
CREATE ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot
FROM Provider LunaEKMProvider
WITH PROVIDER_KEY_NAME = 'EKM_RSA_2048_Key_TDE_Rot',
```

```
CREATION_DISPOSITION=OPEN_EXISTING
```

3. Create a credential for Luna EKM Provider.

```
CREATE CREDENTIAL <Name of credential>
WITH IDENTITY='<Name of EKM User>', SECRET='<HSM partition password>'
FOR CRYPTOGRAPHIC PROVIDER LunaEKMProvider
```

4. Create a login based on the asymmetric key created above.

```
CREATE LOGIN <Name of login>
FROM ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot;
```

5. Map the credential created above to the login created above.

```
ALTER LOGIN <Name of Login>
ADD CREDENTIAL <Name of credential>;
```

6. Execute steps 2-5 for all secondary nodes. It is required because TDE encryption key, credential, and login are created in the master database and it is not a part of **Availability Groups**. Therefore, you need to create the same key, credential, and login in the master database of all the secondary nodes to access the encrypted tables created in the database.

7. Enable Transparent Database Encryption Key Rotation on the primary replica.

```
Use tde;
ALTER DATABASE ENCRYPTION KEY
REGENERATE
WITH ALGORITHM = AES_128
ENCRYPTION BY SERVER ASYMMETRIC KEY SQL_EKM_RSA_2048_Key_TDE_Rot;
go
SELECT * FROM sys.dm_database_encryption_keys
go
```

8. To query the status of database encryption, TDE key change and its percentage completion.

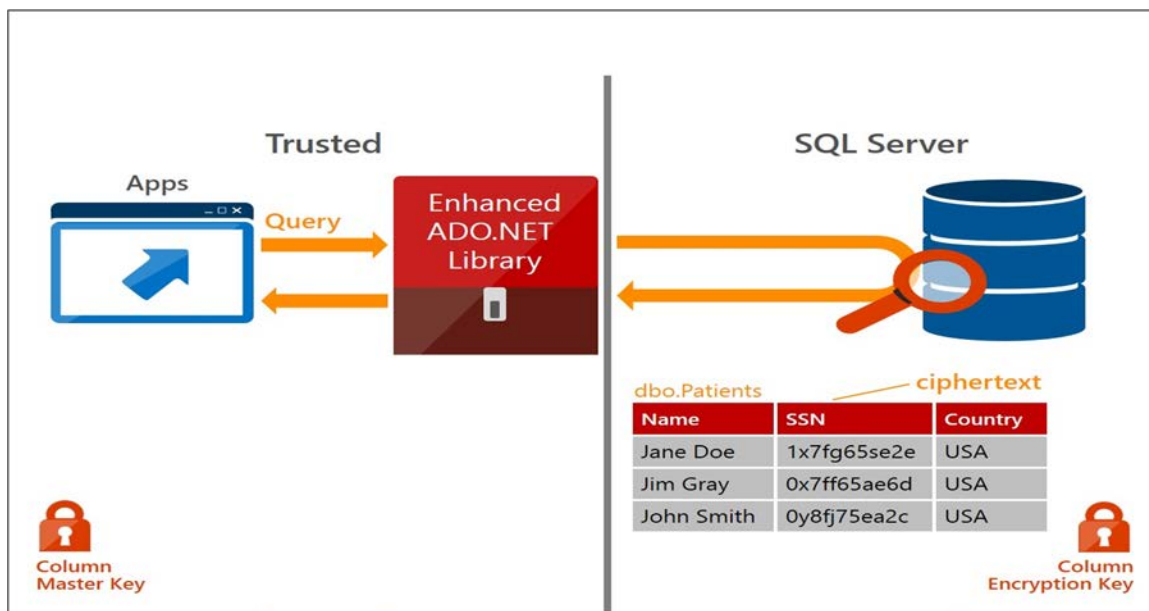
```
SELECT DB_NAME (e.database_id) AS DatabaseName,
e.database_id,
e.encryption_state,
CASE e.encryption_state
WHEN 0 THEN 'No database encryption key present, no encryption'
WHEN 1 THEN 'Unencrypted'
WHEN 2 THEN 'Encryption in progress'
WHEN 3 THEN 'Encrypted'
WHEN 4 THEN 'Key change in progress'
WHEN 5 THEN 'Decryption in progress'
END AS encryption_state_desc,
c.name,
e.percent_complete
FROM sys.dm_database_encryption_keys AS e
LEFT JOIN master.sys.asymmetric_keys AS c
ON e.encryptor_thumbprint = c.thumbprint
```

4

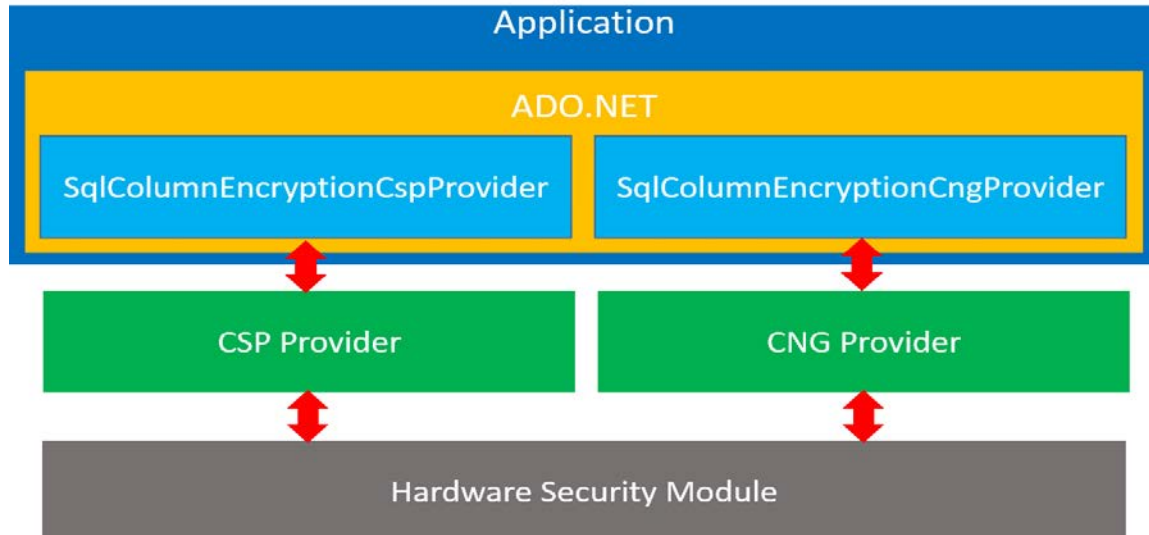
Integrating SafeNet Luna HSM with SQL Server Always Encrypted

SafeNet Luna HSM with SQL Server “Always Encrypted”

The new feature, called Always Encrypted, is available as a part of SQL Server 2016's first public preview. Always Encrypted adds an extra measure of security when the data is being used. That's the point at which data can be most susceptible to attack. The new security layer addresses that vulnerability by keeping the data encrypted during transactions and computations, and by only giving the client keys to decrypt it. It means that if anyone else, including a database or system administrator, tries to access that client's database, the credit card information or other sensitive data would just look like gibberish. The feature is depicted in the figure below:



.NET Framework 4.6.1 RC introduces two new column master key store provider classes, “SqlColumnEncryptionCspProvider” and “SqlColumnEncryptionCngProvider”, which use the CSP and CNG providers to interact with an HSM to encrypt and decrypt column encryption keys with column master keys stored in an HSM. The below diagram illustrates the relationships between the new column master key store provider classes and the CSP/CNG cryptographic providers.



SQL Server Setup

SQL Server must be installed on the target machine to carry out the integration process. For a detailed installation procedure of SQL Server, Always Encrypted pre-requisites, and recommendations refer to the Microsoft SQL Server online documentation.

Configure SafeNet CSP

To generate the Column Master Key on SafeNet Luna HSM either SafeNet CSP/KSP must be installed on the target machine. To demonstrate the feature SafeNet CSP is used in this guide, similar steps will be used for SafeNet KSP.

To use the CSP Provider, SafeNet CSP must be installed on the target machine. Use the following commands:

1. Run the utility, register.exe to register Luna CSP. The general form of command is

```
<Luna Client Installation Directory>\CSP>register.exe
```

For Example: C:\Program Files\SafeNet\LunaClient\CSP>register.exe

2. To list the SafeNet CSPs for Microsoft Windows. The general form of command is

```
<Luna Client Installation Directory>\CSP>register.exe /l
```

For Example: C:\Program Files\SafeNet\LunaClient\CSP>register.exe /l

Setup “Always Encrypted” using SafeNet Luna HSM

Before you get started, ensure the following:

- A database either SQL Server 2016 CTP3 or later is installed on a machine.
- On a client machine, hosting your client applications:
 - .NET Framework 4.6.1 RC.
 - SafeNet CSP must be configured.
 - Visual Studio is configured to use .NET Framework 4.6.1 RC.

Perform the following steps to configure “Always Encrypted” using SafeNet Luna HSM:

1. In a client machine, open the Visual Studio and create a **New Project...** and set the **Target framework** as **.NET Framework 4.6.1**.
2. Use the following C# code to build the project. The code is used to generate the Column Master Key (CMK) in SafeNet Luna HSM that encrypt the Column Encryption Key (CEK) and display the encrypted CEK. Use the following SafeNet CSP:

Luna enhanced RSA and AES provider for Microsoft Windows.

```
-----
using System;
using System.Text;
using System.Data.SqlClient;
using System.Security.Cryptography;
using Microsoft.Win32;

namespace AlwaysEncryptedKeysUsingCsp
{
    class Program
    {
        static void Main(string[] args)
        {
            // Using Luna CSP Provider
            string providerName = "Luna enhanced RSA and AES provider for Microsoft Windows";
            string keyContainerName = "AlwaysEncryptedCMK";

            // Create a CMK using CSP
            if (!CreateCmkUsingCsp(providerName, keyContainerName))
            {
                throw new Exception(@"Failed to create the column master key inside the HSM");
            }

            // Create and encrypt a CEK
            byte[] encryptedCek = CreateEncryptedCekUsingCsp(providerName, keyContainerName);
            if (encryptedCek == null)
            {
                throw new Exception(@"Failed to create th encrypted column encryption key");
            }
            Console.WriteLine(@"Encrypted CEK: {0}", ConvertBytesToHexString(encryptedCek, true));
            Console.ReadKey();
        }

        internal static byte[] CreateEncryptedCekUsingCsp(string providerName, string
keyContainerName)
        {
            try
```

```

    {
        // Create a random column encryption key of size 256 bits
        byte[] columnEncryptionKey = GenerateRandomBytes(32);
        Console.WriteLine(@"Plaintext CEK: {0}",
ConvertBytesToHexString(columnEncryptionKey, true));

        // Encrypt CEK with CMK stored in HSM
        string keyPath = String.Format(@"{0}/{1}", providerName, keyContainerName);
        SqlColumnEncryptionCspProvider cspProvider = new SqlColumnEncryptionCspProvider();

        return cspProvider.EncryptColumnEncryptionKey(keyPath, @"RSA_OAEP",
columnEncryptionKey);
    }
    catch (Exception e)
    {
        Console.WriteLine(@"FAILURE: Creating the encrypted column encryption key failed");
        Console.WriteLine(@"    {0}", e.Message);
        return null;
    }
}

/// <summary>
/// Creates an RSA 2048 key inside the specified CSP.
/// </summary>
/// <param name="providerName">CSP name</param>
/// <param name="containerName">Container name</param>
/// <returns></returns>
internal static bool CreateCmkUsingCsp(string providerName, string containerName)
{
    try
    {
        const int KEYSIZE = 2048;
        int providerType = GetCspProviderKey(providerName);

        // Create a new instance of CspParameters.
        CspParameters cspParams = new CspParameters(providerType, providerName,
containerName);

        //Create a new instance of RSACryptoServiceProvider to generate
        //a new key pair. Pass the CspParameters class to persist the key in the container.
        RSACryptoServiceProvider rsaAlg = new RSACryptoServiceProvider(KEYSIZE, cspParams);
        rsaAlg.PersistKeyInCsp = true;
    }
    catch (CryptographicException e)
    {
        Console.WriteLine(@"FAILURE: The RSA key was not persisted in the container,
\"{0}\".", containerName);
        Console.WriteLine(@"    {0}", e.Message);
        return false;
    }

    return true;
}

/// <summary>
/// Deletes the specified RSA key
/// </summary>
/// <param name="providerName">CSP name</param>
/// <param name="containerName">Container name to be deleted</param>
/// <returns></returns>
internal static bool DeleteCmkUsingCsp(string providerName, string containerName)

```

```

    {
        try
        {
            int providerType = GetCspProviderKey(providerName);

            CspParameters cspParams = new CspParameters(providerType, providerName,
containerName);

            //Create a new instance of RSACryptoServiceProvider.
            //Pass the CspParameters class to use the key in the container.
            RSACryptoServiceProvider rsaAlg = new RSACryptoServiceProvider(cspParams);
            // Delete the key entry in the container.
            rsaAlg.PersistKeyInCsp = false;
            rsaAlg.Clear();
        }
        catch (CryptographicException e)
        {
            Console.WriteLine("\tFAILURE: The RSA key was not deleted from the container,
\"{0}\".", containerName);
            Console.WriteLine("\t{0}", e.Message);
            return false;
        }

        return true;
    }

    internal static int GetCspProviderKey(string providerName)
    {
        RegistryKey key =
Microsoft.Win32.Registry.LocalMachine.OpenSubKey(@"SOFTWARE\Microsoft\Cryptography\Defaults\Provider
");
        RegistryKey providerKey = key.OpenSubKey(providerName);
        return (int)providerKey.GetValue(@"Type");
    }

    internal static byte[] GenerateRandomBytes(int length)
    {
        // Generate random bytes cryptographically.
        byte[] randomBytes = new byte[length];
        RNGCryptoServiceProvider rngCsp = new RNGCryptoServiceProvider();
        rngCsp.GetBytes(randomBytes);

        return randomBytes;
    }

    /// <summary>
    /// Gets hex representation of byte array.
    /// <param name="input">input byte array</param>
    /// <param name="addLeadingZeroX">Add leading 0x</param>
    /// </summary>
    internal static string ConvertBytesToHexString(byte[] input, bool addLeadingZeroX = false)
    {
        StringBuilder str = new StringBuilder();
        if (addLeadingZeroX)
        {
            str.Append(@"0x");
        }

        foreach (byte b in input)
        {
            str.AppendFormat(b.ToString(@"X2"));
        }
    }

```



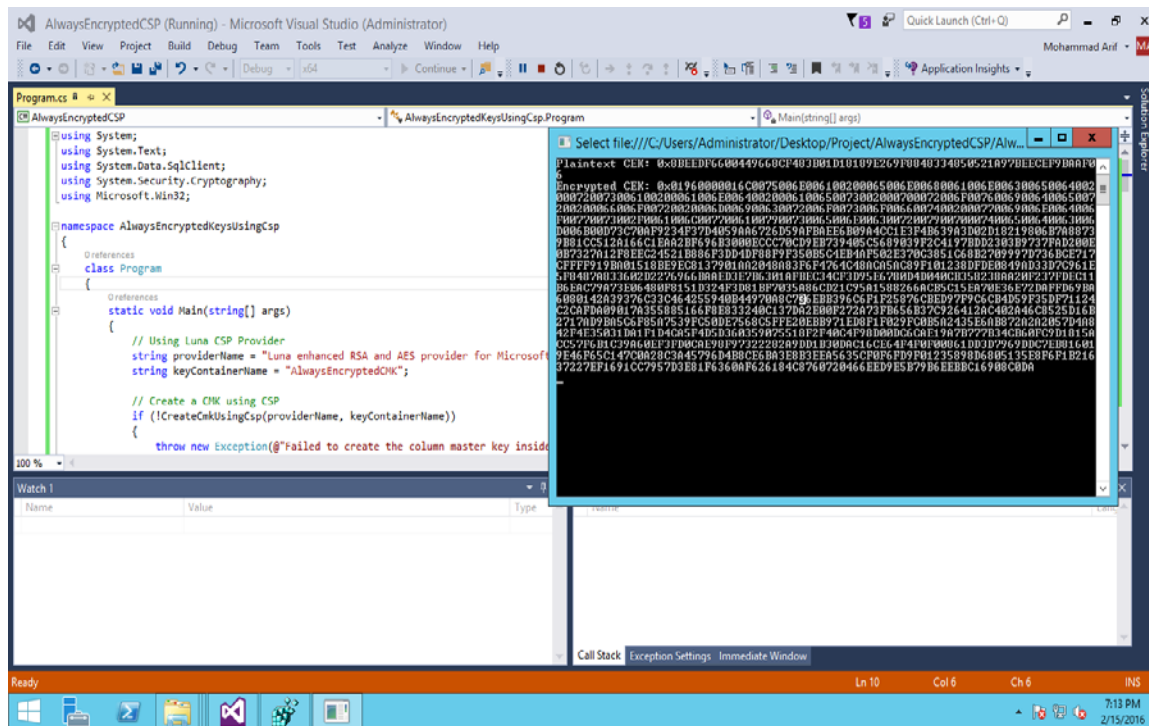
```

    }

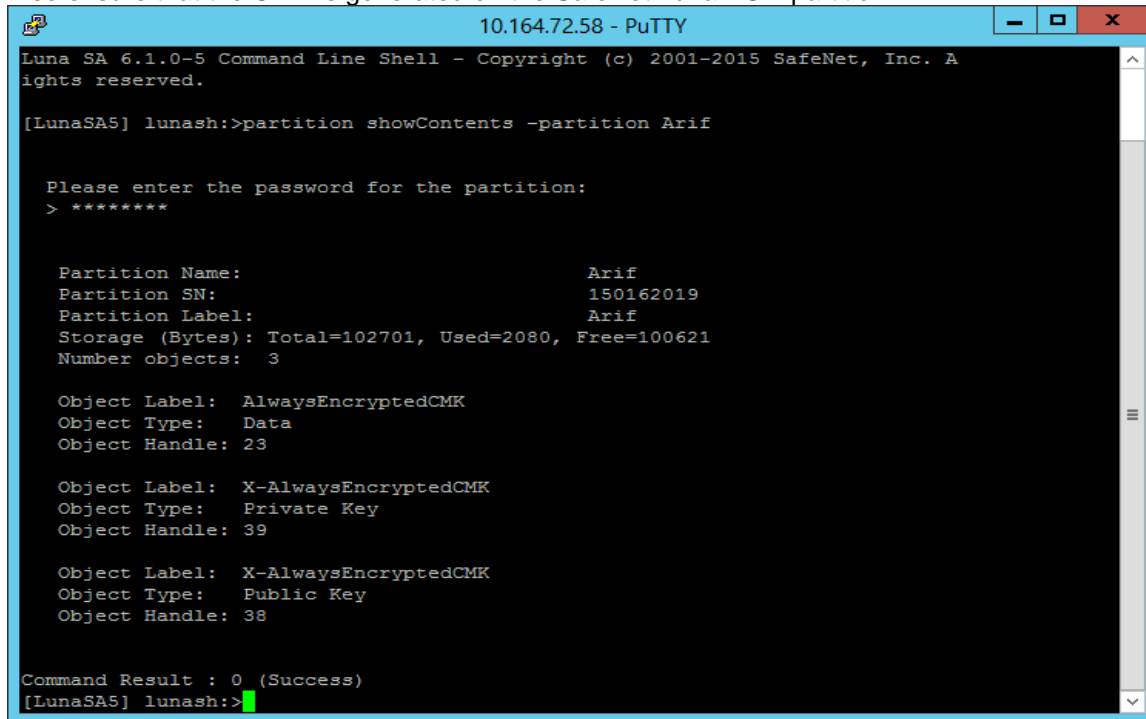
    return str.ToString();
}
}
}
}

```

- When the code build successfully, execute it on the command prompt and note down the value of Encrypted CEK, it will be used at the time of defining the CEK in the database.



Also ensure that the CMK is generated on the SafeNet Luna HSM partition.



```

10.164.72.58 - PuTTY
Luna SA 6.1.0-5 Command Line Shell - Copyright (c) 2001-2015 SafeNet, Inc. All rights reserved.

[LunaSA5] lunash:>partition showContents -partition Arif

Please enter the password for the partition:
> *****

Partition Name:                Arif
Partition SN:                  150162019
Partition Label:               Arif
Storage (Bytes): Total=102701, Used=2080, Free=100621
Number objects: 3

Object Label: AlwaysEncryptedCMK
Object Type: Data
Object Handle: 23

Object Label: X-AlwaysEncryptedCMK
Object Type: Private Key
Object Handle: 39

Object Label: X-AlwaysEncryptedCMK
Object Type: Public Key
Object Handle: 38

Command Result : 0 (Success)
[LunaSA5] lunash:>

```



NOTE: Above code generates a 2048-bit RSA key named AlwaysEncryptedCMK, which is actually a public-private key pair. Both keys are stored in a container inside the SafeNet Luna HSM. It is a sample code to demonstrate the Always Encrypted feature, to know more about it, refer Microsoft online documentation for Always Encrypted Client Driver development.

4. Create metadata objects for the generated **Column Master Key** in the database. Connect to the database using **SQL Server Management Studio** and execute the following T-SQL query.

```

CREATE COLUMN MASTER KEY [CMK1]
WITH
(
KEY_STORE_PROVIDER_NAME = N'MSSQL_CSP_PROVIDER',
KEY_PATH = N'Luna enhanced RSA and AES provider for Microsoft Windows/AlwaysEncryptedCMK'
)

```

5. Create metadata objects for encrypted **Column Encryption Key** in the database. Execute the following T-SQL query.

```

CREATE COLUMN ENCRYPTION KEY [CEK1]
WITH VALUES
(
COLUMN_MASTER_KEY = [CMK1],
ALGORITHM = 'RSA_OAEP',
ENCRYPTED_VALUE =
0x01960000016C0075006E006100200065006E00680061006E006300650064002000720073006100200061006E006400
20006100650073002000700072006F0076006900640065007200200066006F00720020006D006900630072006F007300
6F00660074002000770069006E0064006F00770073002F0061006C00770061007900730065006E006300720079007000
74006500640063006D006B00D73C70AF9234F37D4059AA6726D59AFBAEE6B09A4CC1E3F4B639A3D02D18219806B7A887
39B81CC512A166C1EAA2BF696B3000ECCC70CD9EB739405C5689039F2C4197BDD2303B9737FAD200E0B7327A12F8EEC2
4521B886F3DD4DF88F9F350B5C4EB4AF502E370C3851C68B2709997D736BCE717CFFFF919BA01518BE9EC8137901AA20

```

```

48A83F6F4764C48ACA5AC89F101238DFDE0849AD33D7C961E5F8487A833602D2276966BAAED3E7B6301AFBEC34CF3D95
E6780D4D040CB358238AA20F237FDEC11B6EAC79A73E06480F8151D324F3D81BF7035A86CD21C95A1588266ACB5C15EA
70E36E72DAFFD69BA6080142A39376C33C464255940B44970A8C736EBB396C6F1F25876CBED97F9C6CB4D59F35DF7112
4C2CAFDA09017A355885166F8E833240C137DA2E00F272A73FB656B37C926412AC402A46C8525D16B2717AD9BA5C6F85
A7539FC50DE7568C5FFE20EBB971ED8F1F029FC0B5A2435E6AB872A2A2057D4A42F4E35031DA1F1D4CA5F4D5D3603590
75518F2F40C4F98D00DC6CAE19A7B777B34CB60FC9D1815ACC57F6B1C39A60EF3FD0CAE98F97322282A9DD1B30DAC16C
E64F4F0F00861DD3D7969DDC7EB816019E46F65C147C0A28C3A45796D4B8CE6BA3E8B3EEA5635CF0F6FD9F01235898D6
805135E8F6F1B21637227EF1691CC7957D3E81F6360AF626184C8760720466EED9E5B79B6EEBBC16908C0DA
)

```



NOTE: ENCRYPTED_VALUE is the Encrypted CEK and CMK1 is Column Master Key. For ENCRYPTED_VALUE, you need to paste the exact cipher text of the column encryption key, generated in the step 3.

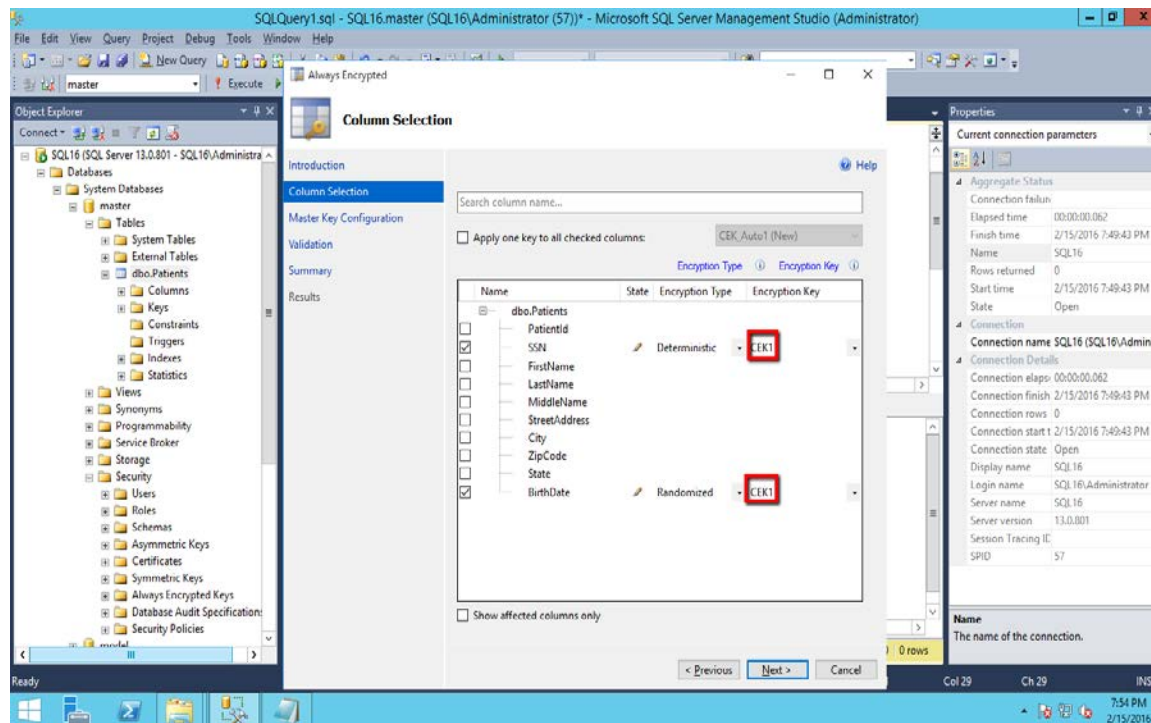
6. Create a table in the database and reference the defined column encryption key while setting up encrypted columns in the table. Execute the following T-SQL query:

```

CREATE TABLE [dbo].[Patients](
  [PatientId] [int] IDENTITY(1,1),
  [SSN] [char](11) COLLATE Latin1_General_BIN2
  ENCRYPTED WITH (ENCRYPTION_TYPE = DETERMINISTIC,
  ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',
  COLUMN_ENCRYPTION_KEY = CEK1) NOT NULL,
  [FirstName] [nvarchar](50) NULL,
  [LastName] [nvarchar](50) NULL,
  [MiddleName] [nvarchar](50) NULL,
  [StreetAddress] [nvarchar](50) NULL,
  [City] [nvarchar](50) NULL,
  [ZipCode] [char](5) NULL,
  [State] [char](2) NULL,
  [BirthDate] [date]
  ENCRYPTED WITH (ENCRYPTION_TYPE = RANDOMIZED,
  ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',
  COLUMN_ENCRYPTION_KEY = CEK1) NOT NULL
  PRIMARY KEY CLUSTERED ([PatientId] ASC) ON [PRIMARY]
)

```

As an alternative to creating a new table using T-SQL, if you have an existing table with column you want to encrypt, you can use the Always Encrypted wizard. You just need to ensure **SQL Server Management Studio** is running on a machine with **.NET Framework 4.6.1** installed, and then point the wizard to CEK1 (a metadata object referencing the column encryption key you generated in step 5).



7. In Visual Studio, create a new console application using C#. Since the SqlClient enhancements to support Always Encrypted were introduced in .Net Framework 4.6.1, ensure the application is using the right version of the framework.

Below is the sample code that connects to the database, inserts and selects data using SqlClient. The only change required to use Always Encrypted is including "ColumnEncryptionSetting=Enabled;" in the connection string.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;

namespace AlwaysEncryptedDemo
{
    /// <summary>
    /// A simple class to demonstrate insert and query sensitive information using Always Encrypted
    /// functionality.
    /// Apart from the connection string keyword ColumnEncryptionSetting, rest of the code looks
    /// very similar to
    /// a regular SqlClient application code.
    ///
    /// Pre-Requisites
    /// .NET 4.6.1

```

```

///      SQL Script to setup the schema for Patients table.
/// </summary>
class Program
{
    private static SqlConnection _sqlconn;

    /// <summary>
    /// Insert a row for a new patient.
    /// </summary>
    /// <param name="ssn">Patient's SSN.</param>
    /// <param name="firstName">Patient's First name</param>
    /// <param name="lastName">Patient's last name</param>
    /// <param name="birthdate">Patient's date of bith</param>
    private static void AddNewPatient(string ssn, string firstName, string lastName, DateTime
birthdate)
    {
        SqlCommand cmd = _sqlconn.CreateCommand();

        // Use parameterized SQL to insert the data.
        //
        cmd.CommandText = @"INSERT INTO [dbo].[Patients] ([SSN], [FirstName], [LastName],
[BirthDate]) VALUES (@SSN, @FirstName, @LastName, @BirthDate)";

        SqlParameter paramSSN = cmd.CreateParameter();
        paramSSN.ParameterName = @"@SSN";
        paramSSN.DbType = DbType.AnsiStringFixedLength;
        paramSSN.Direction = ParameterDirection.Input;
        paramSSN.Value = ssn;
        paramSSN.Size = 11;
        cmd.Parameters.Add(paramSSN);

        SqlParameter paramFirstName = cmd.CreateParameter();
        paramFirstName.ParameterName = @"@FirstName";
        paramFirstName.DbType = DbType.String;
        paramFirstName.Direction = ParameterDirection.Input;
        paramFirstName.Value = firstName;
        paramFirstName.Size = 50;
        cmd.Parameters.Add(paramFirstName);

        SqlParameter paramLastName = cmd.CreateParameter();
        paramLastName.ParameterName = @"@LastName";
        paramLastName.DbType = DbType.String;
        paramLastName.Direction = ParameterDirection.Input;
        paramLastName.Value = lastName;
        paramLastName.Size = 50;
        cmd.Parameters.Add(paramLastName);

        SqlParameter paramBirthdate = cmd.CreateParameter();
        paramBirthdate.ParameterName = @"@BirthDate";
        paramBirthdate.SqlDbType = SqlDbType.Date;
        paramBirthdate.Direction = ParameterDirection.Input;
        paramBirthdate.Value = birthdate;
        cmd.Parameters.Add(paramBirthdate);

        cmd.ExecuteNonQuery();
    }

    /// <summary>
    /// Query the DB to find the patient with the desired SSN, and print the data in the console
    /// </summary>
    /// <param name="ssn">Patient's SSN</param>

```

```

private static void FindAndPrintPatientInformation(string ssn)
{
    SqlDataReader reader = null;
    try
    {
        reader = (ssn == null) ? FindAndPrintPatientInformationAll() :
FindAndPrintPatientInformationSpecific(ssn);

        PrintPatientInformation(reader);
    }
    finally
    {
        if (reader != null)
        {
            reader.Close();
        }
    }
}

/// <summary>
/// Query the DB to find all the patients and print the data in the console
/// </summary>
private static void FindAndPrintPatientInformation()
{
    FindAndPrintPatientInformation(null);
}

/// <summary>
/// Implementation for querying all patients in the DB
/// </summary>
/// <returns>A datareader with the query resultset</returns>
private static SqlDataReader FindAndPrintPatientInformationAll()
{
    SqlCommand cmd = _sqlconn.CreateCommand();

    // Normal select statement.
    //
    cmd.CommandText = @"SELECT [SSN], [FirstName], [LastName], [BirthDate] FROM
[dbo].[Patients] ORDER BY [PatientId]";
    SqlDataReader reader = cmd.ExecuteReader();
    return reader;
}

/// <summary>
/// Implementation for querying a single patient, based on SSN
/// </summary>
/// <param name="ssn">Patient's SSN</param>
/// <returns>A datareader with the query resultset</returns>
private static SqlDataReader FindAndPrintPatientInformationSpecific(string ssn)
{
    SqlCommand cmd = _sqlconn.CreateCommand();

    // Use parameterized SQL to query the data.
    //
    cmd.CommandText = @"SELECT [SSN], [FirstName], [LastName], [BirthDate] FROM
[dbo].[Patients] WHERE [SSN] = @SSN";

    SqlParameter paramSSN = cmd.CreateParameter();
    paramSSN.ParameterName = @"@SSN";
    paramSSN.DbType = DbType.AnsiStringFixedLength;
    paramSSN.Direction = ParameterDirection.Input;

```

```

        paramSSN.Value = ssn;
        paramSSN.Size = 11;
        cmd.Parameters.Add(paramSSN);

        SqlDataReader reader = cmd.ExecuteReader();
        return reader;
    }

    /// <summary>
    /// Format and print the patient data in the console.
    /// </summary>
    /// <param name="reader">the rowset with the patient's data</param>
    private static void PrintPatientInformation(SqlDataReader reader)
    {
        string breaker = new string('-', (19 * 4) + 9);
        Console.WriteLine();
        Console.WriteLine(breaker);
        Console.WriteLine(breaker);
        Console.WriteLine(@"| {0,15} | {1,15} | {2,15} | {3,25} |", reader.GetName(0),
reader.GetName(1), reader.GetName(2), reader.GetName(3));
        if (reader.HasRows)
        {
            while (reader.Read())
            {
                Console.WriteLine(breaker);
                Console.WriteLine(@"| {0,15} | {1,15} | {2,15} | {3,25} |", reader[0],
reader[1], reader[2], ((DateTime)reader[3]).ToLongDateString());
            }
            Console.WriteLine(breaker);
            Console.WriteLine(breaker);
            Console.WriteLine();
            Console.WriteLine();
        }

        /// <summary>
        /// Print usage help on console
        /// </summary>
        static void PrintUsage()
        {
            Console.WriteLine(@"Usage: AlwaysEncryptedDemo <server_name> <database_name>");
            Console.WriteLine();
        }

        static void Main(string[] args)
        {
            if (args.Length != 2)
            {
                PrintUsage();
                return;
            }

            SqlConnectionStringBuilder strblldr = new SqlConnectionStringBuilder();

            strblldr.DataSource = args[0];
            strblldr.InitialCatalog = args[1];
            strblldr.IntegratedSecurity = true;

            // Enable Always Encrypted in the connection we will use for this demo
            //
            strblldr.ColumnEncryptionSetting = SqlConnectionColumnEncryptionSetting.Enabled;

```

```
_sqlconn = new SqlConnection(strbldr.ConnectionString);

_sqlconn.Open();

try
{
    // Add a few rows to the table.
    // Please notice that as far as the app is concerned, all data is in plaintext
    //
    AddNewPatient("123-45-6789", "John", "Doe", new DateTime(1971, 5, 21));
    AddNewPatient("111-22-3333", "Joanne", "Doe", new DateTime(1974, 12, 1));
    AddNewPatient("562-00-6354", "Michael", "Park", new DateTime(1928, 11, 18));

    // Print a few individual entries as well as the whole table
    // Once again, the app handles the data as plaintext
    //
    FindAndPrintPatientInformation("123-45-6789");
    FindAndPrintPatientInformation("111-22-3333");
    FindAndPrintPatientInformation();
}
finally
{
    _sqlconn.Close();
}
}
}
```

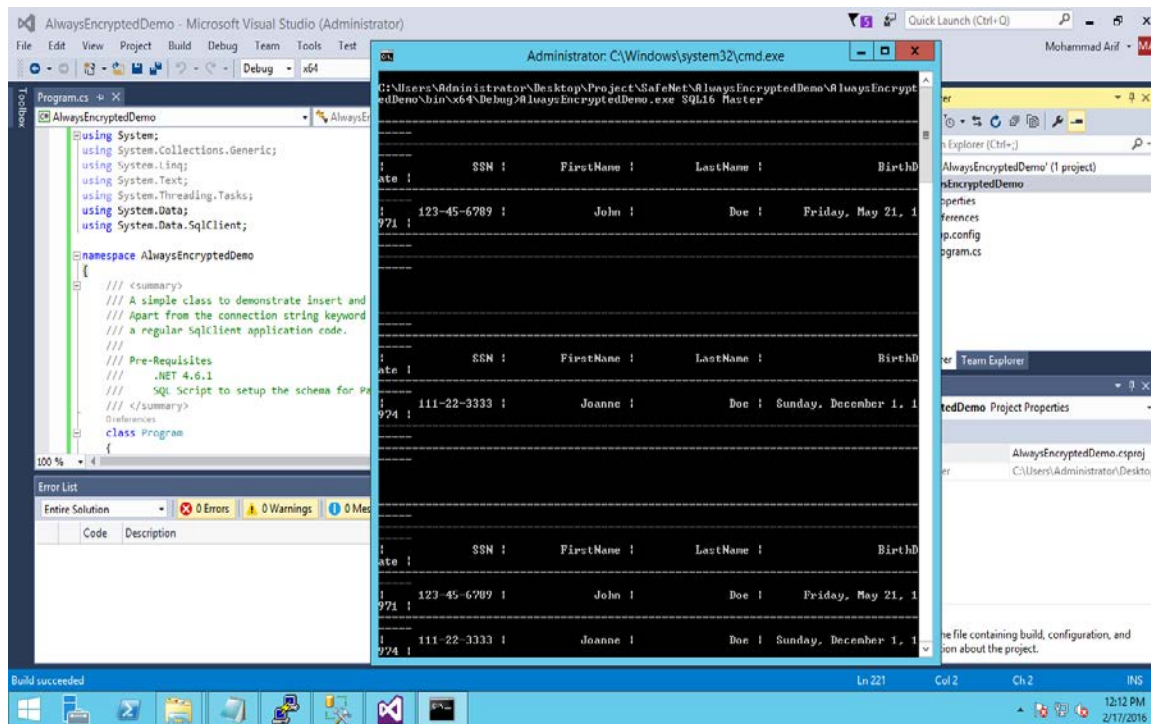
8. When the project build successfully open the application in command prompt, run the following command:

AlwaysEncryptedDemo.exe <Server Name> <Database Name>

For Example: AlwaysEncryptedDemo.exe SQL16 Master



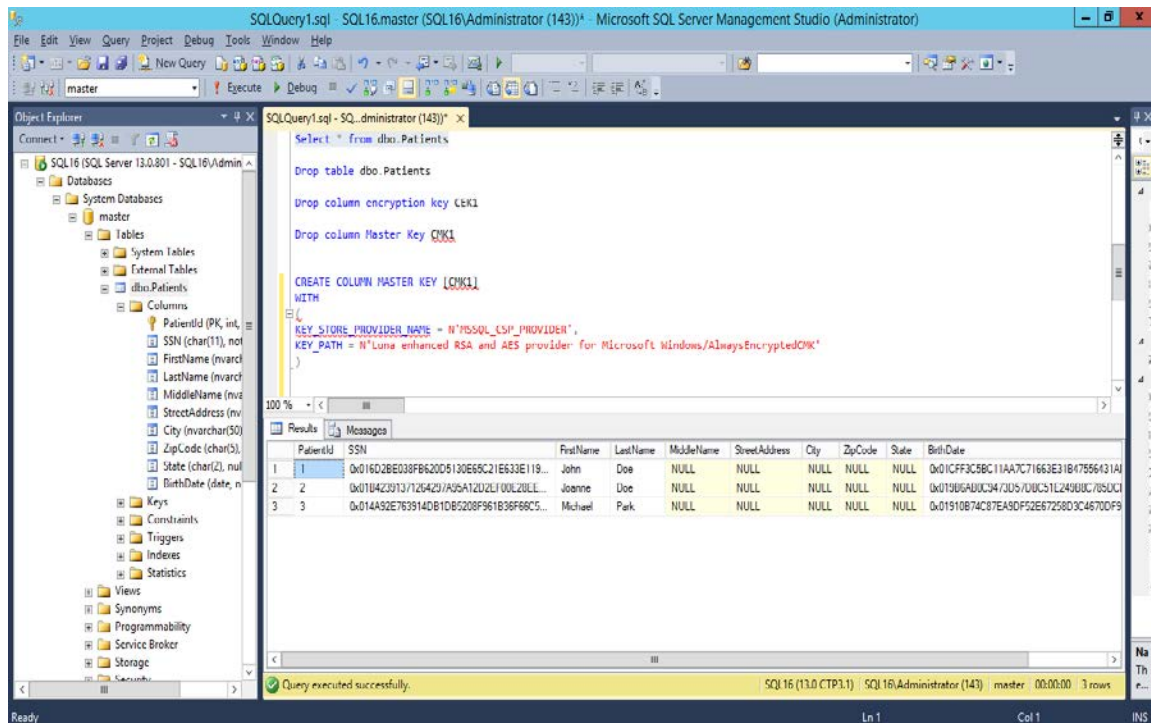
NOTE: Where “AlwaysEncryptedDemo.exe”, “SQL16” and “Master” are the name of the build application, Database Server, and Database respectively.



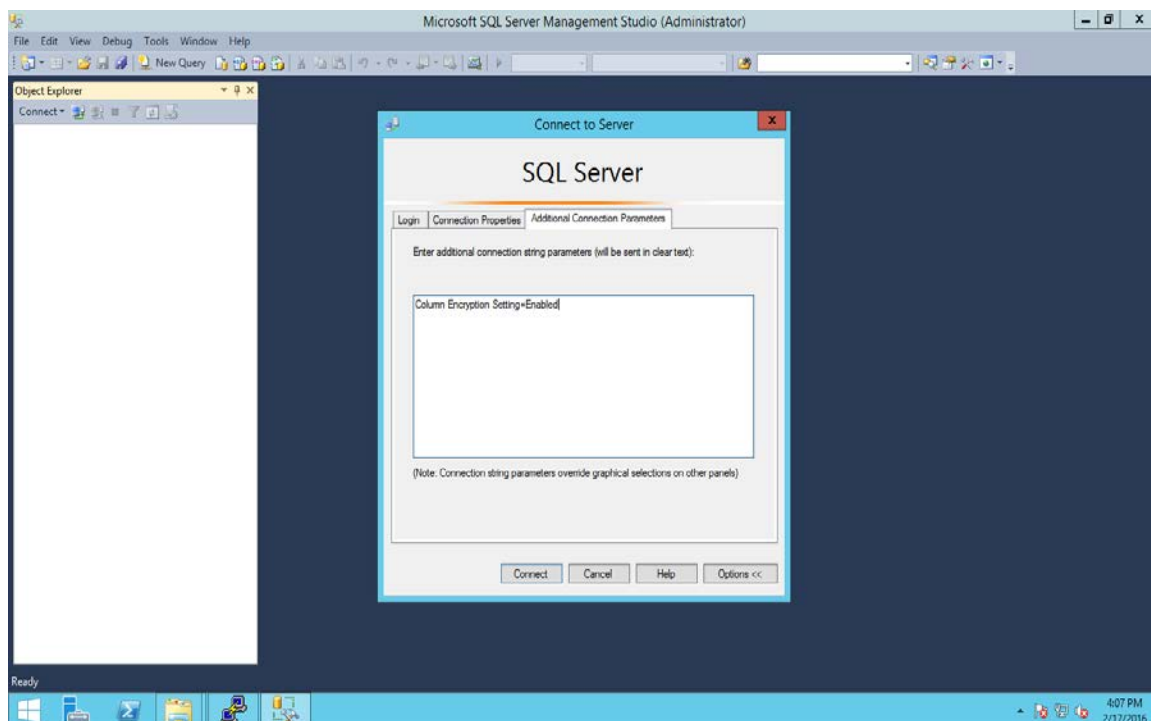
After execution as shown in the above screenshot, it inserts data in the dbo.Patients table in the encrypted form and select the records from table and display in the clear text.

9. Open the **SQL Server Management Studio** and run the select query on the dbo.Patients table to ensure that records are inserted in the encrypted form and even DBA could not able to see it.

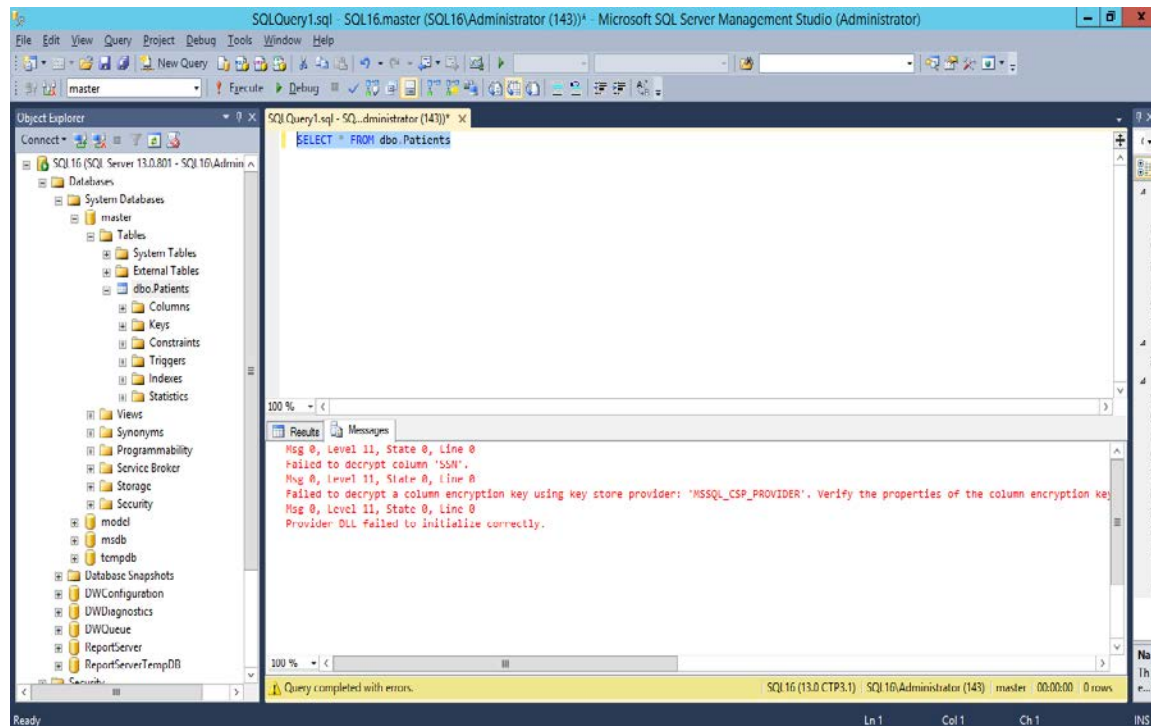
Select * from dbo.Patients;



Another way to ensure that DBA could not able to see the encrypted data, use “Column Encryption Setting=Enabled” in **Additional Connection Parameters** while connecting the database.



After connecting the database, run the select query again. It displays the failed to decrypt the column encryption key error that signifies that DBA do not have access the Column Master Key generated on the HSM and only client application can see the data in clear form.



Always Encrypted feature using SafeNet CSP and .NET Framework 4.6.1 works successfully on SQL Server. As you have seen, Always Encrypted transparently encrypts/decrypts sensitive data in the application as long as the application has access to the CMK generated on HSM. Users and applications without access to the CMK, including SQL Server itself, will not be able to decrypt the sensitive data.

Remarks

By following the above steps, a column master key and a column encryption key is generated. Both keys are defined in the database, and an application is implemented using both keys and accessing encrypted database columns. In reality, for a production application, these steps will likely be the responsibility of individuals assuming different roles in an organization. Role separation is important to maximize the benefits of Always Encrypted.

Here is how the key management steps can be allocated to various roles:

Step	Responsible Role
Generating column master key in an HSM.	Security Administrator
Generating/encrypting column encryption key.	Security Administrator or Application Operations
Defining column/master encryption key in the database.	DBA

After generating a column master key, the Security Administrator would also generate and encrypt a column encryption key. Alternatively, the Security Administrator can grant permission to access and use the column master key to the Application Operations team, who would generate and encrypt the column encryption key. The Security Administrator or Application Operations team would then pass the column master key location information (key path) and the encrypted value of the column encryption key to the DBA, who would issue T-SQL statements to create key metadata in the database.

5

Troubleshooting Tips

Problem – 1

Failed to verify Authenticode signature on DLL 'C:\Program Files\LunaPCI\EKM\LunaEKM.dll'.

Solution

This error could appear in SQL logs if the certificate in the signature of dll cannot be verified because there are no corresponding certificates for this issuer and therefore it is not trusted.

Go to <http://www.verisign.com/support/roots.html> and download the all root certificates. Install the certificate and install/import it to Trusted Root Certification Authorities store.

Problem – 2

"The decryption key is incorrect" error when you open a symmetric key that is encrypted by an asymmetric key in SQL Server 2008, SQL Server 2012 or SQL Server 2008 R2 on a computer that is running Windows 8 or Windows Server 2012.

Solution

Download the cumulative update package and apply for SQL Server provided by Microsoft to resolve the issue:

- For SQL Server 2008 SP3 on Windows Server 2012 platform:
<http://support.microsoft.com/kb/2863205>
- For SQL Server 2008 R2 SP2 on Windows Server 2012 platform:
<http://support.microsoft.com/kb/2871401>
- For SQL Server 2012 on Windows Server 2012 platform:
<http://support.microsoft.com/kb/2867319>

Problem – 3

CREATE CRYPTOGRAPHIC PROVIDER EKMPProvider FROM FILE = <Path to EKM DLL>' fails with below error on Windows 2012:

```
Error:
Msg 33029, Level 16, State 1, Line 3
Cannot initialize cryptographic provider. Provider error code: 1. (Failure - Consult EKM
Provider for details)
```

Solution

Reboot the OS server and try to create cryptographic provider. It resolves the above problem.

Problem – 4

When the key is generated on secondary node using CREATION_DISPOSITION=OPEN_EXISTING:

```
CREATE ASYMMETRIC KEY <Name of the SQL Key>  
FROM Provider LunaEKMProvider  
WITH PROVIDER_KEY_NAME = '<Name of the HSM Key>',  
CREATION_DISPOSITION=OPEN_EXISTING
```

Thumbprint of the key generated on secondary node of SQL servers (For backup/restore of an encrypted database) is different from original thumbprint of key generated on primary node which gives error while restoring the database on secondary nodes.

Solution

Use the EKM v1.2 and regenerate the key on secondary nodes using the same command. This issue is fixed in EKM v1.2.