Microsoft SQL Server Always Encrypted

nShield® HSM Integration Guide

2024-10-21

Member of Microsoft Intelligent Security Association

Microsoft Security

Table of Contents

1. Introduction . . . 1
    1.1. Product configurations . . . 1
    1.2. Supported nShield hardware and software versions . . . 1
    1.3. Role separation . . . 2
    1.4. Multiple Windows user accounts on a single client server . . . 3
    1.5. Multiple client servers . . . 3
    1.6. Always Encrypted and TDE . . . 3
2. Configure computers and accounts . . . 4
    2.1. Join the domain . . . 4
    2.2. Create domain accounts . . . 4
3. Install and configure client . . . 5
    3.1. Select the protection method . . . 5
    3.2. Install the Security World software and create a Security World . . . 5
    3.3. Create the OCS or Softcard . . . 8
    3.4. Install and register the CNG provider . . . 9
    3.5. Install and configure SqlServer PowerShell module . . . 12
    3.6. Install the SQL Server Management Studio . . . 13
    3.7. Allow Active Directory user to remote login . . . 13
4. Install and configure SQL server . . . 14
    4.1. Install the SQL database engine . . . 14
    4.2. Create the SQL logins . . . 15
5. Generate the encryption keys . . . 16
    5.1. Generate the Always Encrypted Column Master Key (CMK) . . . 16
    5.2. Generate My Column Master Key (MyCMK) and My Column Encryption Key (MyCEK) with SSMS . . . 19
    5.3. Generate MyCMK and MyCEK with PowerShell . . . 25
6. Encrypt or decrypt a column with SSMS . . . 27
    6.1. Encrypt a column . . . 27
    6.2. View an encrypted column . . . 30
    6.3. Remove column encryption . . . 32
7. Encrypt or decrypt a column with PowerShell . . . 35
    7.1. Encrypt a column . . . 35
    7.2. Remove column encryption . . . 36
8. Test access to Always Encrypted keys by another user . . . 37
9. Supported PowerShell SqlServer cmdlets . . . 38
10. Additional resources and related products . . . 40
    10.1. nShield Connect . . . 40

10.2. nShield as a Service 40
10.3. Entrust products 40
10.4. nShield product documentation 40

Chapter 1. Introduction

Always Encrypted is a feature in Windows SQL Server designed to protect sensitive data both at rest and in flight between a client application server and Azure or SQL Server database(s).

Data protected by Always Encrypted remains in an encrypted state until it has reached the client application server. This effectively mitigates man-in-the-middle attacks and provides assurances against unauthorized activity from rogue DBAs or admins with access to Azure or SQL server databases.

The nShield HSM secures the key used to protect the Column Master Key, stored in an encrypted state on the client application server.

1.1. Product configurations

Entrust successfully tested nShield HSM integration with Windows SQL Server and the Always Encrypted feature in the following configurations:

1.1.1. Remote server

Product	Version
SQL Server	Microsoft SQL Server 2022
Base OS	Windows Server 2022 Datacenter

1.1.2. Client

Product	Version
SQL Server GUI	Microsoft SQL Server Management Studio V18.8
Base OS	Windows 10 Enterprise

1.2. Supported nShield hardware and software versions

Entrust successfully tested with the following nShield hardware and software versions:

Product	Security World Software	Firmware	Netimage	OCS	Softcard	Module
Connect XC	13.4.5	12.50.11 (FIPS 140-2 certified) & 12.72.1 (FIPS 140-2 certified)	12.80.5 & 13.4.5	✓	✓	✓
nShield 5c	13.4.5	13.2.2	13.2.2	✓	✓	✓
nSaaS	12.80.4	12.72.1 (FIPS 140-2 certified)	12.80.5	✓	✓	✓

1.3. Role separation

The generation of keys and the application of these keys for encryption or decryption are separate processes. The processes can be assigned to users with various access permissions, or Duty Roles. The table below shows the processes and duty roles with reference to the Security Administrator and the database Administrator.

Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

Process	Duty Role
Generating the Column Master Key (CMK) and Column Encryption Key (CEK)	Security Administrator
Applying the CMK and CEK in the database	Database Administrator

Four database permissions are required for Always Encrypted.

Operation	Description
`ALTER ANY COLUMN MASTER KEY`	Required to generate and delete a column master key

Operation	Description
`ALTER ANY COLUMN ENCRYPTION KEY`	Required to generate and delete a column encryption key
`VIEW ANY COLUMN MASTER KEY`	Required to access and read the metadata of the column master keys to manage keys or query encrypted columns
`VIEW ANY COLUMN ENCRYPTION KEY`	Required to access and read the metadata of the column encryption key to manage keys or query encrypted columns

1.4. Multiple Windows user accounts on a single client server

To enable multiple Windows user accounts on a single oclient server, ask Entrust Support for a Hotfix patch to allow multiple users to use the same always encrypted key.

1.5. Multiple client servers

Each client server wanting access to the content of the encrypted data with a given CEK must have:

- An HSM in the same Security World.
- A Hotfix patch to allow multiple users to use the same always encrypted key. Ask Entrust Support for this.
- A copy of the CMK key token stored on its local drive.

1.6. Always Encrypted and TDE

The same Security World can be used for Always Encrypted and TDE.

Chapter 2. Configure computers and accounts

Installation steps:

1. Join the domain.
2. Create domain accounts.

2.1. Join the domain

Windows authentication is used in this integration for added security. The Entrust nShield HSM solution for Microsoft SQL Always Encrypted enables keys that are associated with one user to be used by other users, providing secure access to a common database.

Both the client computer and the remote server computer must join the same Windows domain.

2.2. Create domain accounts

Create the following three Windows domain accounts:

- <domain>\<SQL Administrator>
- <domain>\dbuser
- <domain>\dbuser2

Chapter 3. Install and configure client

This installation must be performed on the client using the <domain_name>\Administrator account.

Installation steps:

1. Select the protection method
2. Install the Security World software and create a Security World
3. Create the OCS or Softcard
4. Install and register the CNG provider
5. Install and configure SqlServer PowerShell module
6. Install the SQL Server Management Studio
7. Allow Active Directory user to remote login

3.1. Select the protection method

OCS or Module protection can be used to authorize access to the keys protected by the HSM. Follow your organization's security policy to select which one.

3.2. Install the Security World software and create a Security World

1. Install the Security World software. For instructions, see the *Installation Guide* and the *User Guide* for the HSM.
2. Install Hotfix TAC-996 if multiple Windows user accounts need access to the same data. Contact nShield support to download the Hotfix. To perform the installation:
   a. Open a command window as Administrator and uninstall the CNG:

```
C:\Users\Administrator.EXAMPLE>cnginstall32 --uninstall
ncksps w.dll removed.

ncpp.dll removed.

C:\Users\Administrator.EXAMPLE>cnginstall --uninstall
ncksps w.dll removed.

ncpp.dll removed.
```

   b. Reboot the server.
   c. Copy files as per the installation instructions in the Hotfix package:

```
C:\Users\Administrator.EXAMPLE>copy C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-
TAC996\hotfix-Z155163-TAC996\nfast\c\caping\vs2017-32\lib\* "C:\Program
Files\nCipher\nfast\c\caping\vs2017-32\lib\."
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-32\lib\nckspsw.dll
Overwrite C:\Program Files\nCipher\nfast\c\caping\vs2017-32\lib\.\nckspsw.dll? (Yes/No/All): All
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-32\lib\nckspsw.lib
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-32\lib\nckspsw.map
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-32\lib\nckspsw.pdb
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-32\lib\ncpp.dll
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-32\lib\ncpp.lib
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-32\lib\ncpp.map
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-32\lib\ncpp.pdb
        8 file(s) copied.

C:\Users\Administrator.EXAMPLE>copy C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-
TAC996\hotfix-Z155163-TAC996\nfast\c\caping\vs2017-64\lib\* "C:\Program
Files\nCipher\nfast\c\caping\vs2017-64\lib\."
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-64\lib\nckspsw.dll
Overwrite C:\Program Files\nCipher\nfast\c\caping\vs2017-64\lib\.\nckspsw.dll? (Yes/No/All): All
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-64\lib\nckspsw.lib
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-64\lib\nckspsw.map
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-64\lib\nckspsw.pdb
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-64\lib\ncpp.dll
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-64\lib\ncpp.lib
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-64\lib\ncpp.map
C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-TAC996\hotfix-Z155163-
TAC996\nfast\c\caping\vs2017-64\lib\ncpp.pdb
        8 file(s) copied.

C:\Users\Administrator.EXAMPLE>copy C:\Users\Administrator.EXAMPLE\Downloads\hotfix-Z155163-
TAC996\hotfix-Z155163-TAC996\nfast\lib\versions\caping-atv.txt "C:\Program
Files\nCipher\nfast\lib\versions\."
Overwrite C:\Program Files\nCipher\nfast\lib\versions\.\caping-atv.txt? (Yes/No/All): All
        1 file(s) copied.
```

d. Open a command window as Administrator and install the CNG:

```
C:\Users\Administrator.EXAMPLE>cnginstall32 --install
nckspsw.dll installed.

ncpp.dll installed.

C:\Users\Administrator.EXAMPLE>cnginstall --install
nckspsw.dll installed.

ncpp.dll installed.
```

e. Reboot the server.

3. Add the Security World utilities path `C:\Program Files\nCipher\nfast\bin` to the Windows system path.
4. Open the firewall port 9004 for the HSM connections.
5. Install the nShield Connect HSM locally, remotely, or remotely via the serial console. See the following nShield Support articles and the *Installation Guide* for the HSM:

   - How to locally set up a new or replacement nShield Connect
   - How to remotely set up a new or replacement nShield Connect
   - How to remotely set up a new or replacement nShield Connect XC Serial Console model

   Access to the Entrust nShield Support Portal is available to customers under maintenance. To request an account, contact nshield.support@entrust.com.

6. Open a command window and run the following to confirm that the HSM is `operational`:

```
C:\Users\Administrator.EXAMPLE>enquiry
Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        5F08-02E0-D947 6A74-1261-7843
 mode                 operational
 version              12.80.4
...
Module #1:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        5F08-02E0-D947
 mode                 operational
 version              12.72.1
 ...
```

7. Create your Security World if one does not already exist, or copy an existing one. Follow your organization's security policy for this.
8. Confirm that the Security World is `usable`:

```
C:\Users\Administrator.EXAMPLE>nfkminfo
World
 generation  2
 state       0x3737000c Initialised Usable ...
 ...
Module #1
 generation 2
 state      0x2 Usable
 ...
```

3.3. Create the OCS or Softcard

If using OCS protection, create the OCS now. Follow your organization's security policy for the value N of K/N. As required, create extra OCS cards, one for each person with access privilege, plus spares.

Administrator Card Set (ACS) authorization is required to create an OCS in FIPS 140 level 3.

After an OCS card set has been created, the cards cannot be duplicated.

1. If using remote administration, ensure the `C:\ProgramData\nCipher\Key Management Data\config\cardlist` file contains the serial number of the card(s) to be presented.
2. Open a command window as Administrator.
3. Run the following command. Follow your organization's security policy for the values K/N. The OCS cards cannot be duplicated after created. Enter a passphrase or password at the prompt. Notice that `slot 2`, remote via a Trusted Verification Device (TVD), is used to present the card. In this example, K=1 and N=1.

```
>createocs -m1 -s2 -N testOCS -Q 1/1

FIPS 140-2 level 3 auth obtained.

Creating Cardset:
 Module 1: 0 cards of 1 written
 Module 1 slot 0: Admin Card #1
 Module 1 slot 2: empty
 Module 1 slot 3: empty
 Module 1 slot 2: blank card
 Module 1 slot 2:- passphrase specified - writing card
Card writing complete.

cardset created; hkltu = a165a26f929841fe9ff2acdf4bb6141c1f1a2eed
```

   Add the `-p` (persistent) option to the command above to retain authentication after the OCS card has been removed from the HSM front panel slot, or from the TVD. If using OCS card protection and the non-persistent card configuration, OCS cards need to be inserted in the nShield front panel or always present in the TVD. The authentication provided by the OCS as shown in the command line above is non-persistent and only available for K=1 and while the OCS card is present in the HSM front panel slot or TVD.

4. Verify the OCS created:

```
nfkminfo -c
Cardset list - 1 cardsets:  (P)ersistent/(N)ot, (R)emoteable/(L)ocal-only
```

```
 Operator logical token hash              k/n timeout  name
 a165a26f929841fe9ff2acdf4bb6141c1f1a2eed  1/1  none-NL testOCS
```

The `rocs` utility also shows the OCS created:

```
>rocs
'rocs' key recovery tool
Useful commands: 'help', 'help intro', 'quit'.
rocs> list cardset
No. Name                    Keys (recov) Sharing
  1 testOCS                 0 (0)        1 of 1
rocs> quit
```

If using Softcard protection, create the Softcard now.

1. Ensure the `C:\Program Files\nCipher\nfast\cknfastrc` file exists with the following content. Otherwise create it.

```
> type "C:\Program Files\nCipher\nfast\cknfastrc"
CKNFAST_LOADSHARING=1
```

2. Run the following command and enter a passphrase/password at the prompt:

```
>ppmk -n testSC

Enter new pass phrase:
Enter new pass phrase again:
New softcard created: HKLTU d9414ed688c6405aab675471d3722f8c70f5d864
```

3. Verify the Softcard was created:

```
>nfkminfo -s
SoftCard summary - 1 softcards:
 Operator logical token hash              name
 d9414ed688c6405aab675471d3722f8c70f5d864  testSC
```

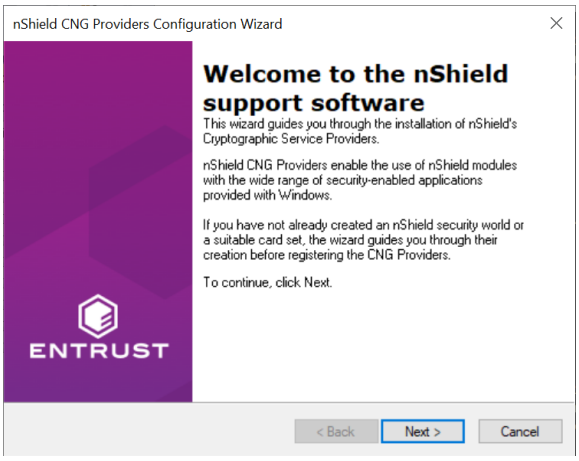
The `rocs` utility also shows the OCS and Softcard created.

```
>rocs
'rocs' key recovery tool
Useful commands: 'help', 'help intro', 'quit'.
rocs> list cardset
No. Name                    Keys (recov) Sharing
  1 testOCS                 0 (0)        1 of 1
  2 testSC                  0 (0)        (softcard)
rocs>quit
```

3.4. Install and register the CNG provider

To install and register the CNG provider:

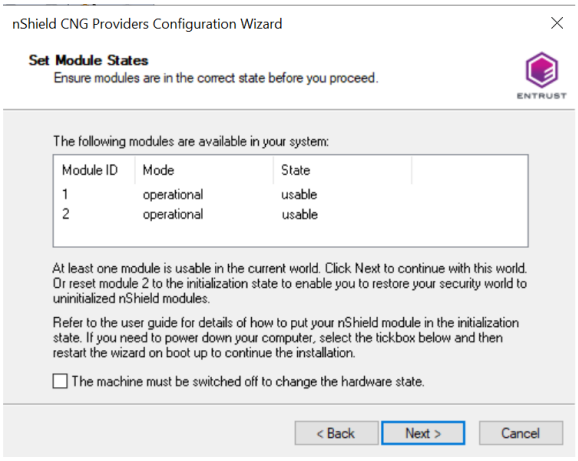
1. Select **Start** > **Entrust** > **CNG configuration wizard**.
2. Select **Next** on the **Welcome** window.



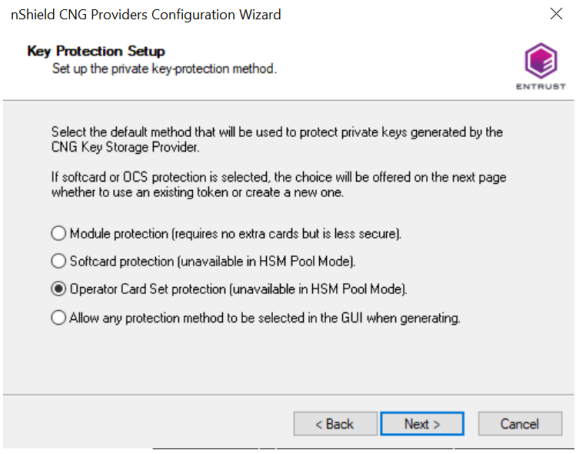
3. Select **Next** on the **Enable HSM Pool Mode** window, leaving **Enable HSM Mode for CNG Providers** un-checked.

If you intend to use multiple HSMs in a failover and load-sharing capacity, select **Enable HSM Pool Mode for CNG Providers**. If you do, you can only use module protected keys. Module protection does not provide conventional 1 or 2 factor authentication. Instead, the keys are encrypted and stored as an application key token, also referred to as a Binary Large Object (blob), in the `kmdata/local` directory.

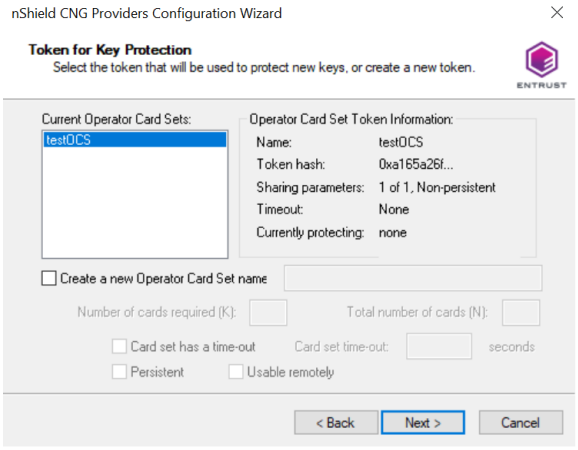
4. Select **Use existing security world** on the **Initial setup** window. Then select **Next**.
5. Select the HSM (Module) if more than one is available on the **Set Module States** window. Then select **Next**.



6. In **Key Protection Setup**, select **Operator Card Set protection**. Then select **Next**.



7. Choose from the **Current Operator Card Sets** or **Current Softcards** list. These were created above. Then select **Next** and **Finish**.

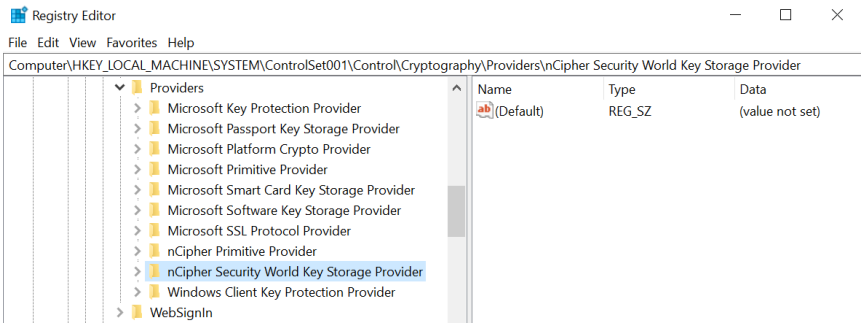


8. Verify the provider with the following commands:

```
>certutil -csplist | findstr nCipher
Provider Name: nCipher DSS Signature Cryptographic Provider
Provider Name: nCipher Enhanced Cryptographic Provider
Provider Name: nCipher Enhanced DSS and Diffie-Hellman Cryptographic Provider
Provider Name: nCipher Enhanced DSS and Diffie-Hellman SChannel Cryptographic Provider
Provider Name: nCipher Enhanced RSA and AES Cryptographic Provider
Provider Name: nCipher Enhanced SChannel Cryptographic Provider
Provider Name: nCipher Signature Cryptographic Provider
Provider Name: nCipher Security World Key Storage Provider

>cnglist.exe --list-providers | findstr nCipher
nCipher Primitive Provider
nCipher Security World Key Storage Provider
```

9. Check the registry in `CNGRegistry`:

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Cryptography\Providers\nCipherSecurityWorldKeyStorageProvid
er
```



3.5. Install and configure SqlServer PowerShell module

1. Open a PowerShell session as Administrator and run:

```
PS C:\Users\Administrator.EXAMPLE> [Net.ServicePointManager]::SecurityProtocol =
[Net.SecurityProtocolType]::Tls12
PS C:\Users\Administrator.EXAMPLE> Install-PackageProvider Nuget -force -verbose
VERBOSE: Acquiring providers for assembly: C:\Program
Files\WindowsPowerShell\Modules\PackageManagement\1.4.7\fullclr\Microsoft.PackageManagement.CoreProviders.d
ll
...
VERBOSE: Imported provider 'C:\Program
Files\PackageManagement\ProviderAssemblies\nuget\2.8.5.208\Microsoft.PackageManagement.NuGetProvider.dll' .
```

2. Update PowerShellGet:

```
PS C:\Users\Administrator.EXAMPLE> Install-Module -Name PowerShellGet -force -verbose
VERBOSE: Using the provider 'PowerShellGet' for searching packages.
...
VERBOSE: Module 'PowerShellGet' was installed successfully to path 'C:\Program
Files\WindowsPowerShell\Modules\PowerShellGet\2.2.5'.
```

3. Download and install the SqlServer module to configure Always Encrypted using Power Shell:

```
PS C:\Users\Administrator.EXAMPLE> Install-Module -Name SqlServer -force -verbose -AllowClobber
VERBOSE: Using the provider 'PowerShellGet' for searching packages.
...
VERBOSE: Module 'SqlServer' was installed successfully to path 'C:\Program
Files\WindowsPowerShell\Modules\SqlServer\21.1.18256'.
```

The `-AllowClobber` parameter allows you to import the specified commands if it exists in the current session.

4. Once installed, confirm the install by running the command below.

If you are using PowerShell ISE, refresh the Commands pane. If you are using PowerShell, open a new session.

```
PS C:\Users\Administrator.EXAMPLE> Get-Module -list -Name SqlServer

    Directory: C:\Program Files\WindowsPowerShell\Modules

ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Script     21.1.18256 SqlServer                           {Add-RoleMember, Add-SqlAvailabilityDatabase,
Add-SqlAvail...
```

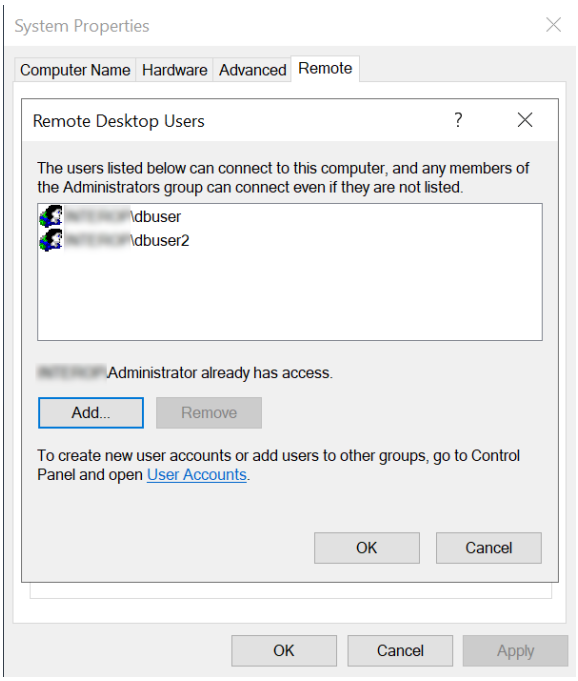
3.6. Install the SQL Server Management Studio

Install the SQL Server Management Studio.

3.7. Allow Active Directory user to remote login

To allow an Active Directory user to remote login:

1. Select **Control Panel** > **System** > **Advance system settings**.
2. Select the **Remote** tab in the **System Properties** dialog. Then select **Select Users...**.
3. Add the following users:
    - <domain>\dbuser
    - <domain>\dbuser2.



Chapter 4. Install and configure SQL server

This installation must be performed on the remote server.

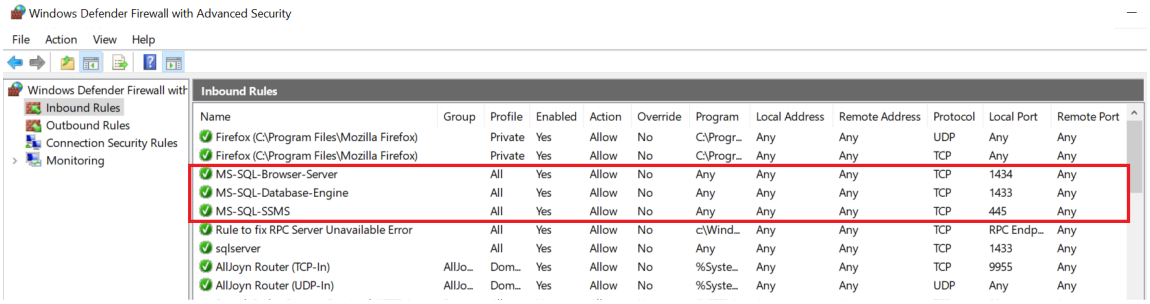
Installation steps:

1. Install the SQL database engine.
2. Create the SQL logins.

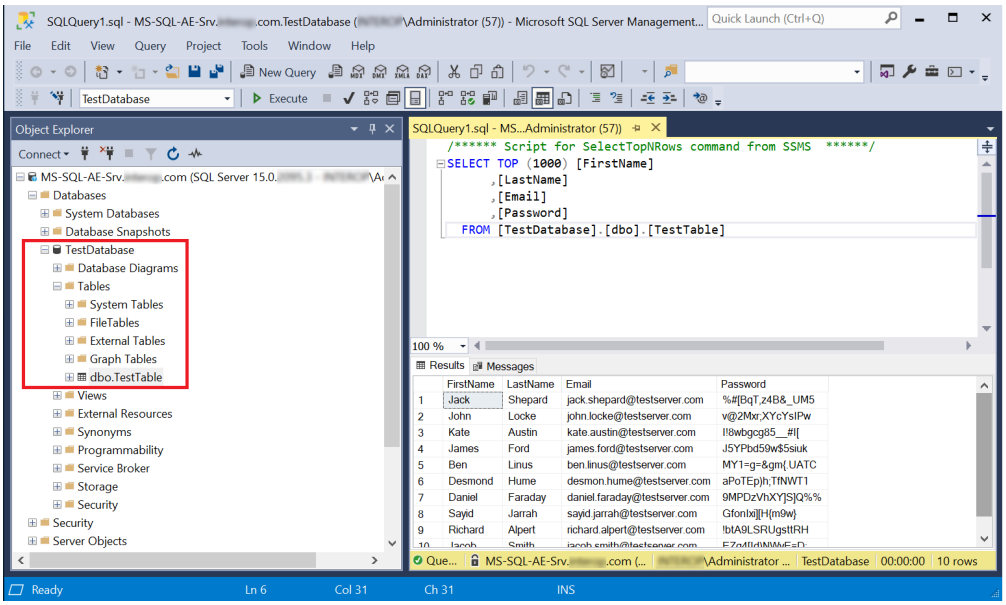
4.1. Install the SQL database engine

This installation must be performed on the remote server using the <domain_name>\Administrator account.

1. Install the SQL engine.
2. Open the firewall ports 1433, 1434, and 445 for access by the SQL database engine, SQL browser, and Active Directory for domain account authorization.

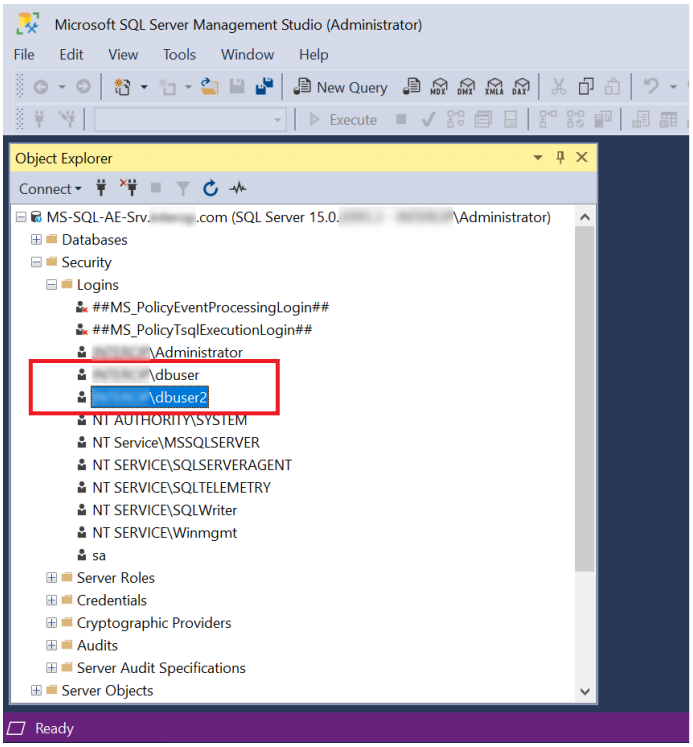


3. Create a test database, if a suitable is not available, for the purpose of this integration.

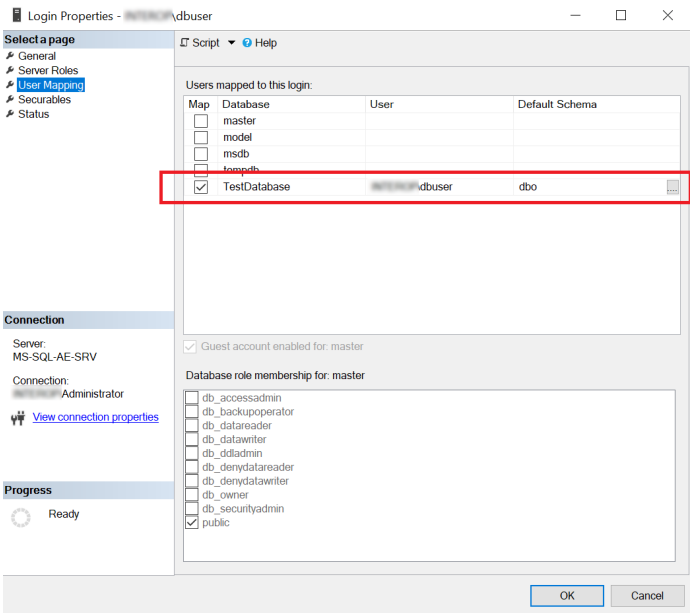


4.2. Create the SQL logins

1. Create two SQL logins with the domain accounts <domain>\dbuser and <domain>\dbuser2 with **Default Database** equal to "TestDatabase".



2. Set the **User Mapping** as database owners, **db_owner**, of TestDatabase.



Chapter 5. Generate the encryption keys

To generate encryption keys:

- Generate the Always Encrypted Column Master Key (CMK).
- Generate My Column Master Key (MyCMK) and My Column Encryption Key (MyCEK) with SSMS.
- Generate MyCMK and MyCEK with PowerShell.

5.1. Generate the Always Encrypted Column Master Key (CMK)

The CMK is protected by the nShield HMS.

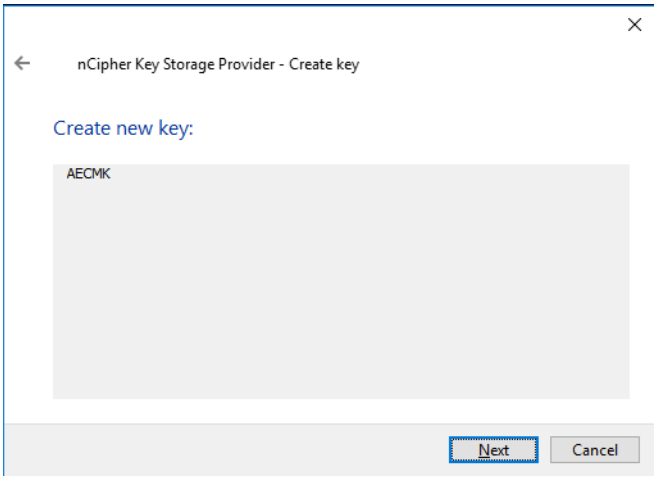
1. Log in to the client using the <domain>\dbuser, or a suitable security administrator account.
2. Launch PowerShell and run the `Generate_AECMK.ps1` script (shown below).

```
$cngProviderName = "nCipher Security World Key Storage Provider"

$cngAlgorithmName = "RSA"

$cngKeySize = 2048

$cngKeyName = "AECMK"

$cngProvider = New-Object System.Security.Cryptography.CngProvider($cngProviderName)

$cngKeyParams = New-Object System.Security.Cryptography.CngKeyCreationParameters

$cngKeyParams.provider = $cngProvider

$cngKeyParams.KeyCreationOptions =
[System.Security.Cryptography.CngKeyCreationOptions]::OverwriteExistingKey

$keySizeProperty = New-Object System.Security.Cryptography.CngProperty("Length",
[System.BitConverter]::GetBytes($cngKeySize), [System.Security.Cryptography.CngPropertyOptions]::None);

$cngKeyParams.Parameters.Add($keySizeProperty)

$cngAlgorithm = New-Object System.Security.Cryptography.CngAlgorithm($cngAlgorithmName)

$cngKey = [System.Security.Cryptography.CngKey]::Create($cngAlgorithm, $cngKeyName, $cngKeyParams)
```

   a. Run the following command:

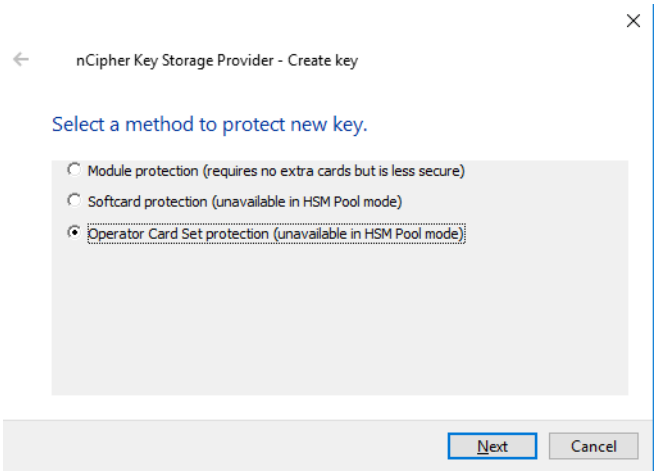
```
> PowerShell -ExecutionPolicy Bypass -File Generate_AECMK.ps1
```

   The following dialog appears.

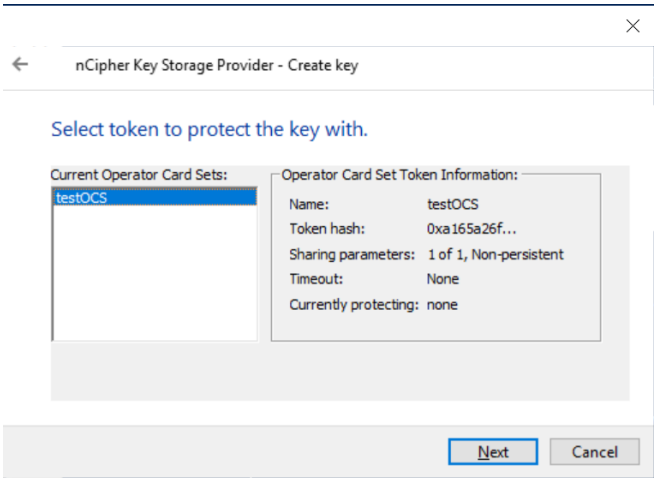


b. Select **Next**.

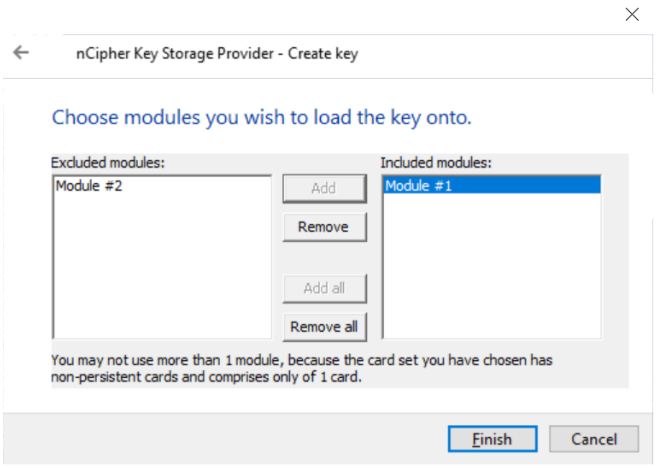
c. Select the **Operator Card Set Protection**. Insert the OCS card in the HSM and select **Next**.



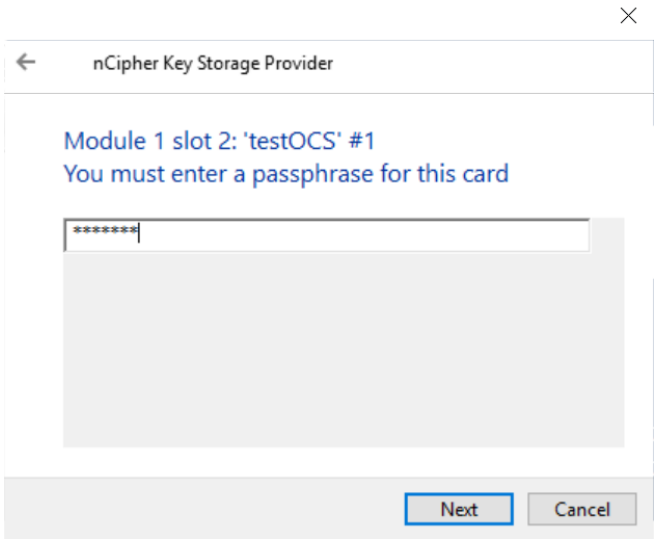
d. Select the OCS and then Select **Next**.



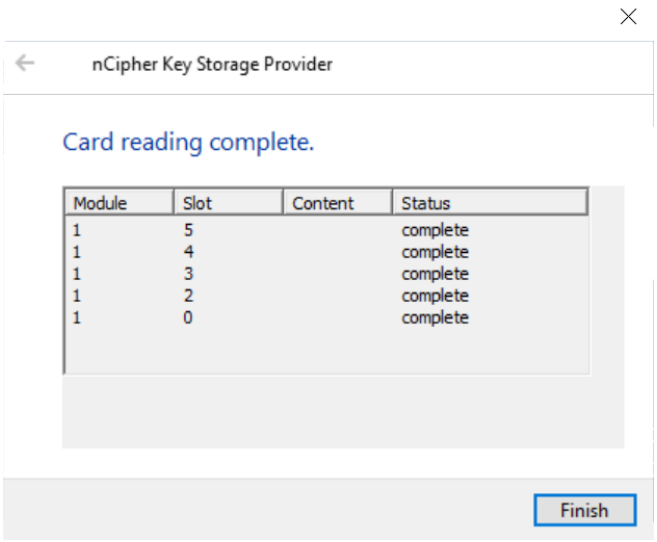
e. Select the HSM and select **Finish**.



f. Enter the OCS passphrase and select **Next**.



g. Select **Finish**.



A 2048-bit RSA key pair, called AECMK, has been generated. The key is encrypted in the HSM and then pushed to the requesting client server, where it is stored as an Application Key Token in the %NFAST_KMDATA%\local folder. That is, :\ProgramData\nCipher\Key Management Data\local.

3. Verify the new key:

```
C:\Users\Administrator.EXAMPLE>nfkminfo -k

Key list - 1 keys
 AppName caping              Ident user--e57798f862740453d02379579c1758ddfa2189db
```

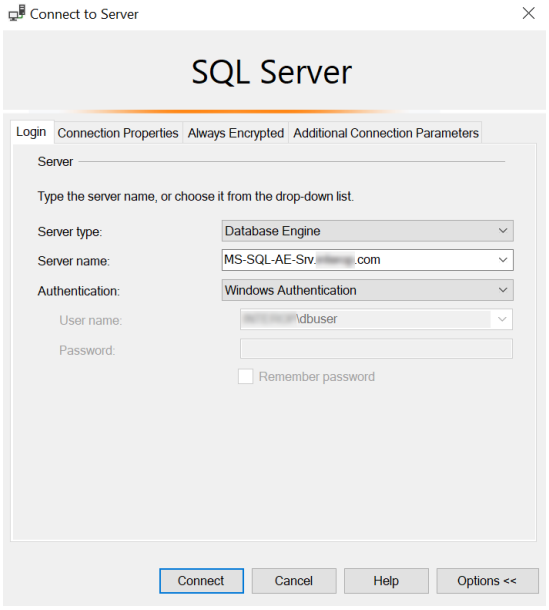
4. Display the information about the key by copy-pasting the key name above as follows:

```
C:\Users\Administrator.EXAMPLE>nfkminfo -k caping user--e57798f862740453d02379579c1758ddfa2189db
Key AppName caping Ident user--e57798f862740453d02379579c1758ddfa2189db
 BlobKA length         1128
 BlobPubKA length      484
 BlobRecoveryKA length 1496
 name                  "AECMK"
 hash                  d9253d650283dafd8d62659f9fb74102b9edcf8c
 recovery              Enabled
 protection            CardSet
 other flags           PublicKey !SEEAppKey !NVMemBlob +0x0
 cardset               a165a26f929841fe9ff2acdf4bb6141c1f1a2eed
 gentime               2022-12-30 19:46:54
 SEE integrity key     NONE

BlobKA
 format                6 Token
 other flags           0x0
 hkm                   28ee9f7cfceba95992f1f3f31b39c8dba7cfa960
 hkt                   a165a26f929841fe9ff2acdf4bb6141c1f1a2eed
 hkr                   none

BlobRecoveryKA
 format                9 UserKey
 other flags           0x0
 hkm                   none
 hkt                   none
 hkr                   55c38c84103d95278fd54b6b5b3e67d614db8538

BlobPubKA
 format                5 Module
 other flags           0x0
 hkm                   c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
 hkt                   none
 hkr                   none

Extra entry #1
 typecode              0x10000 65536
 length                60
Not a blob
```

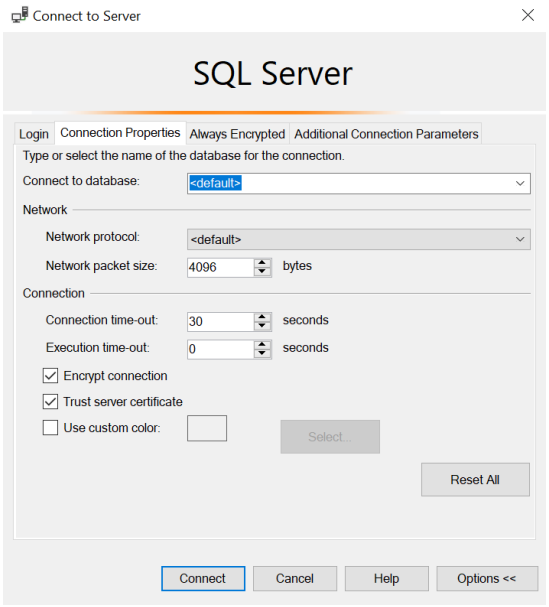
5.2. Generate My Column Master Key (MyCMK) and My Column Encryption Key (MyCEK) with SSMS

This key will encrypt all subsequent Column Encryption keys (CEKs) in your database.

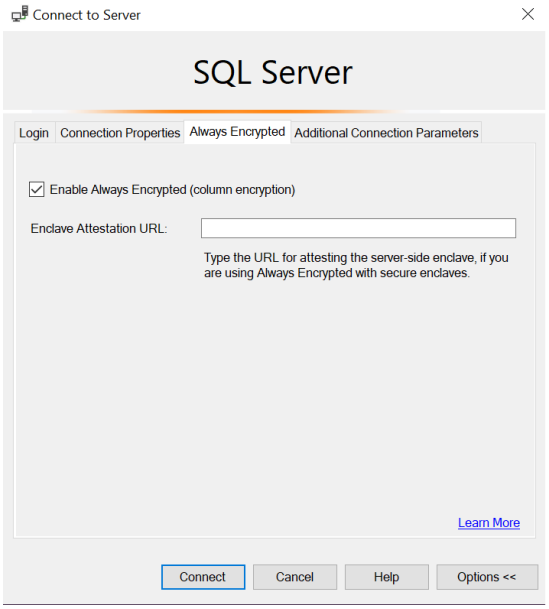
1. Log in to the client using the <domain>\dbuser account.
2. Launch **Microsoft SQL Server Management Studio**.
3. Connect to the database on the remote SQL server:
    a. Select the **Login** tab and set it as follows:



    b. Select the **Connection Properties** tab, as set as follows:

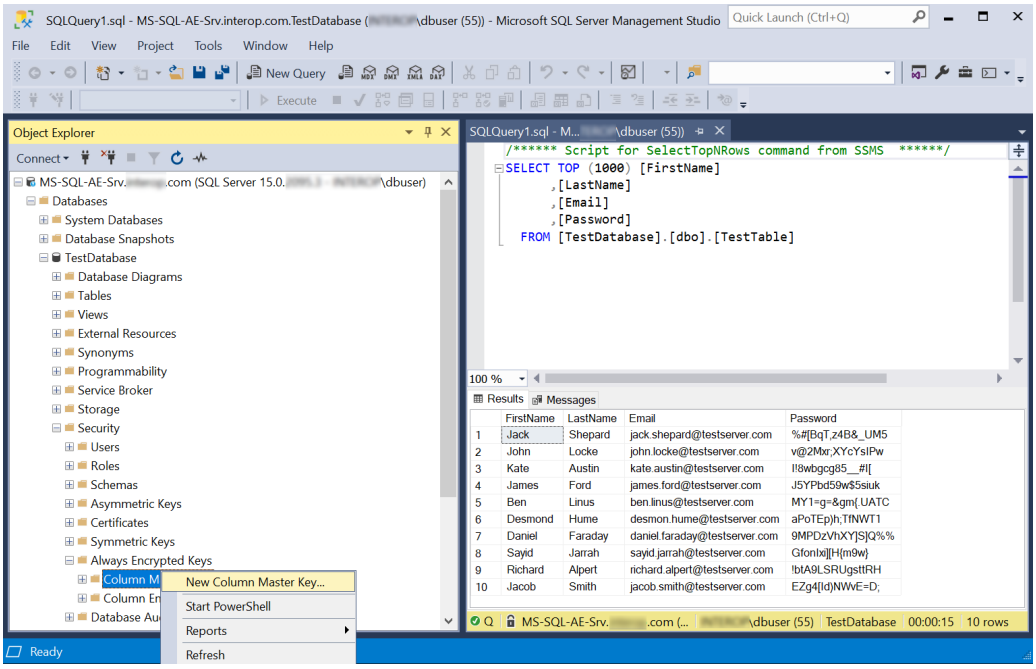


    c. Select the **Always Encrypted** tab and select **Enable Always Encrypted**:



d. Select **Connect**.

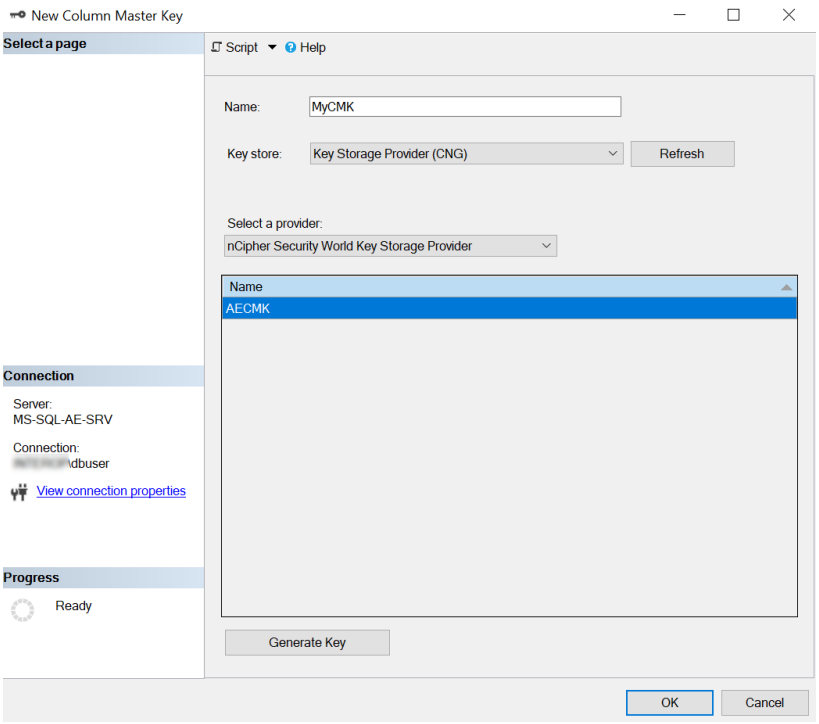
4. Using the **Object Explorer**, select the **Security** directory under the required database, then select **Always Encrypted Keys** > **Column Master Key** > **New Column Master Key**.



5. Enter the following information on the **Column Master Keys** dialog:
   a. Enter a **Name**, for example **MyCMK**.
   b. Select **Key Storage Provider (CNG)** from the **Key store** drop-down list and then **Select a provider**.
   c. Select **nCipher Security World Key Storage Provider** from the drop-down list.

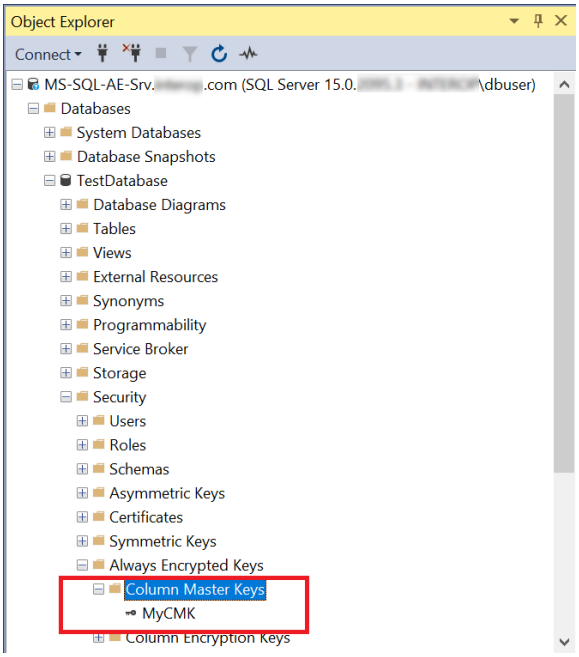
The **AECMK** key created in an earlier step appears in **Name**.

d. Select **OK** to create a new key using the nShield HSM and CNG KSP.



6. Select **Next**.

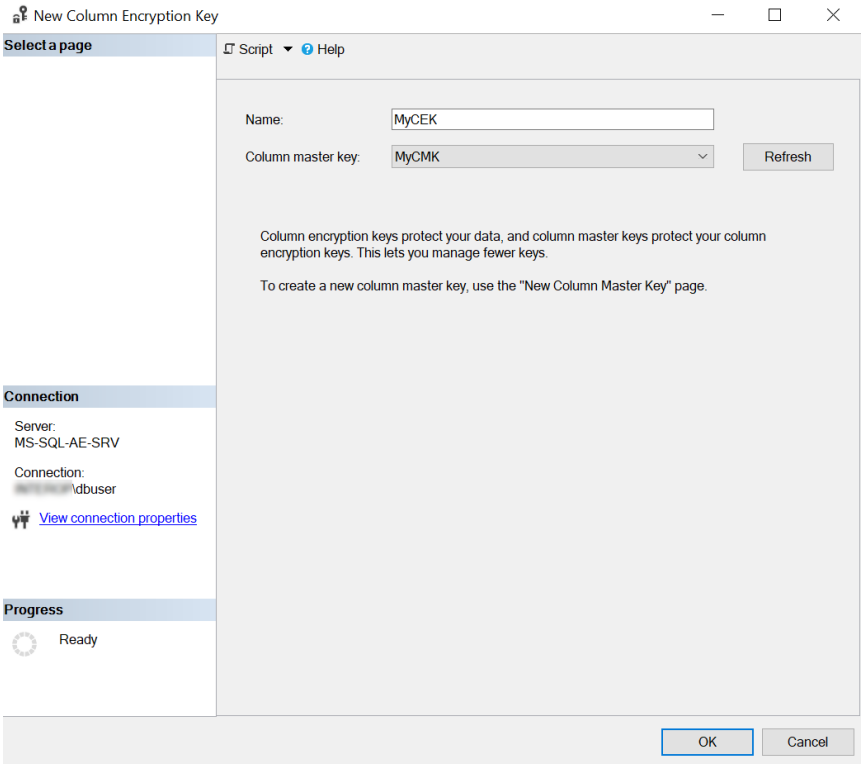
The newly-created **MyCMK** is created in the database under **Security** > **Always Encrypted Keys** > **Column Master Keys**.



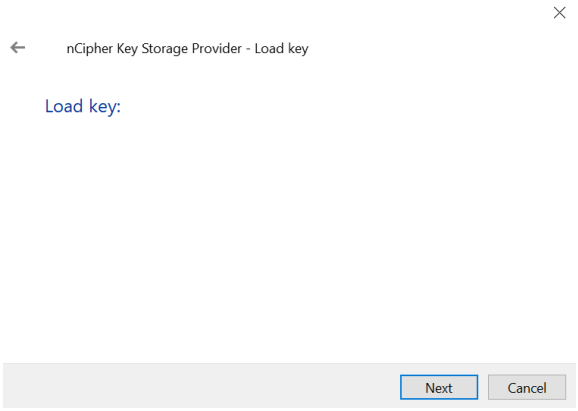
7. Using **Object Explorer**, select the **Security** directory under the required database.

Select **Always Encrypted Keys** to expand it, then select **New Column Encryption Key**.

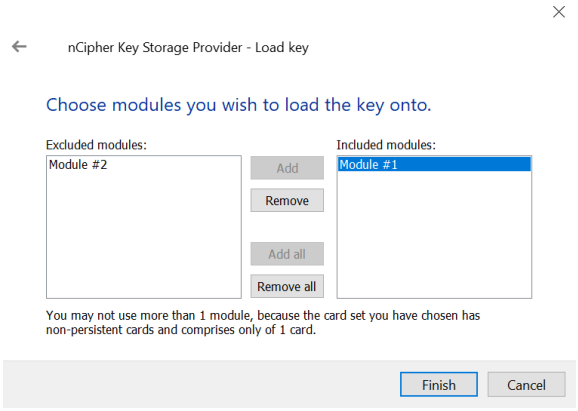
8. Enter **Name**, select the CMK, then select **OK**.



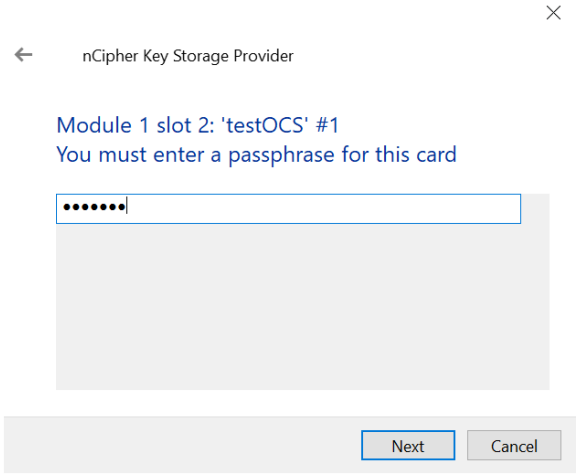
9. Present the OCS and then select **Next**.



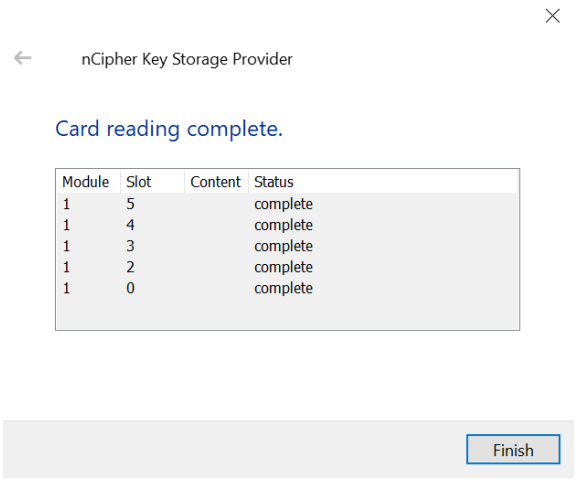
10. Select the HSM and then select **Finish**.



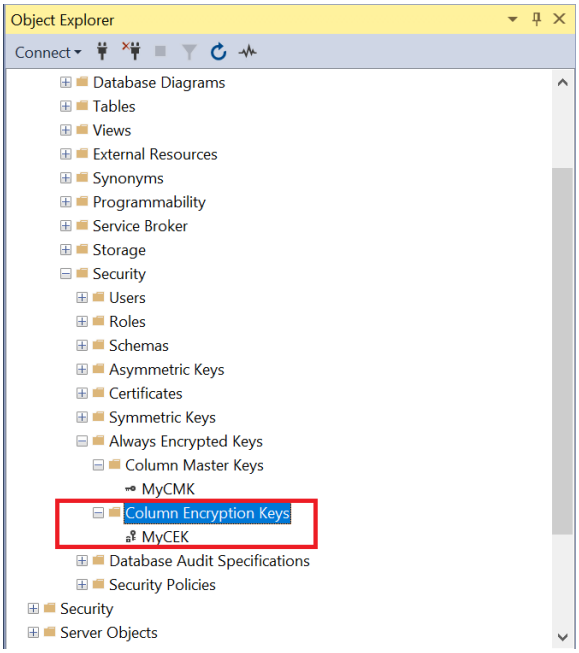
11. Enter the passphrase and then select **Next**.



12. Select **Finish** after the OCS card reading completes.



The newly-created **MyCEK** is in the database under **Security** > **Always Encrypted Keys** > **Column Encryption Keys**.



5.3. Generate MyCMK and MyCEK with PowerShell

To generate MyCMK and MyCEK with PowerShell:

1. Delete MyCEK and MyCMK in that order created above by right-clicking each key and selecting **Delete**.
2. Launch PowerShell and run the `Generate_MyCMK_and_MyCEK.ps1` script (below).

```
# Import the SqlServer module.
Import-Module SqlServer

# Connect to database.
$ConnectionString = "Data Source=MS-SQL-AE-Srv.interop.com,1433;Initial
Catalog=TestDatabase;Trusted_Connection=True;MultipleActiveResultSets=False;Encrypt=True;TrustServerCertifi
cate=True;Packet Size=4096;Application Name=`"Microsoft SQL Server Management Studio`""
$Database = Get-SqlDatabase -ConnectionString $ConnectionString

# Create a SqlColumnMasterKeySettings object for your column master key.
$cmkSettings = New-SqlCngColumnMasterKeySettings -CngProviderName "nCipher Security World Key Storage
Provider" -KeyName "AECMK"

# Create column master key metadata in the database.
New-SqlColumnMasterKey -Name "MyCMK" -InputObject $Database -ColumnMasterKeySettings $cmkSettings

# Generate a column encryption key, encrypt it with the column master key and create column encryption key
metadata in the database.
New-SqlColumnEncryptionKey -Name "MyCEK" -InputObject $Database -ColumnMasterKey "MyCMK"
```

The command line is:

```
> PowerShell -ExecutionPolicy Bypass -File Generate_MyCMK_and_MyCEK.ps1

Name
-----
```

```
MyCMK
MyCEK
```

3. Present the OCS, select the HSM, and enter the passphrase.
4. Check the newly-created **MyCMK** and **MyCEK** are present.

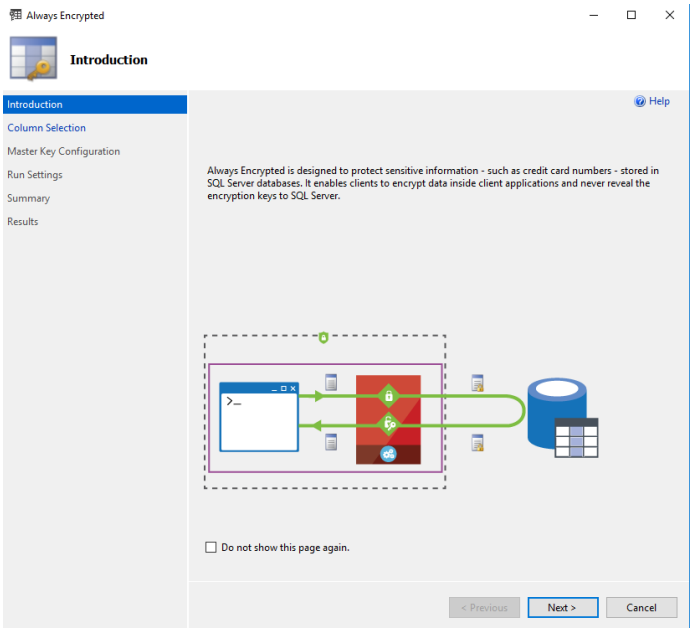
Chapter 6. Encrypt or decrypt a column with SSMS

To encrypt or decrypt a column with SSMS:

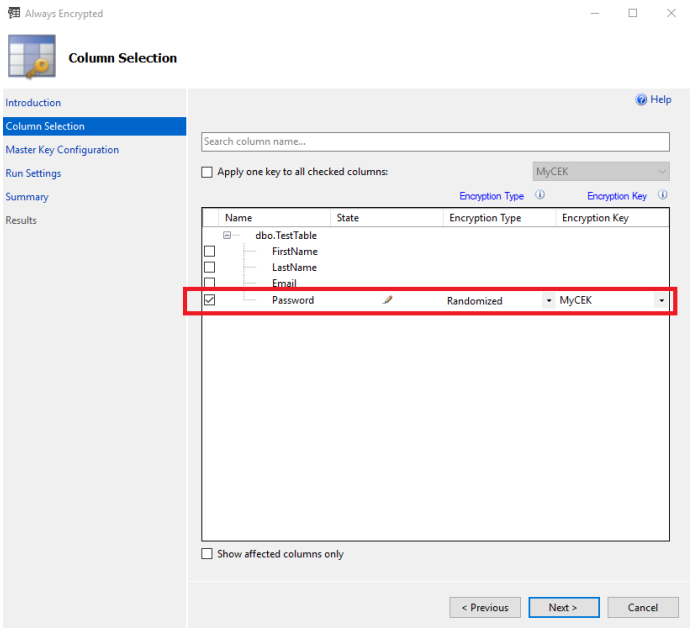
- Encrypt a column
- View an encrypted column
- Remove column encryption

6.1. Encrypt a column

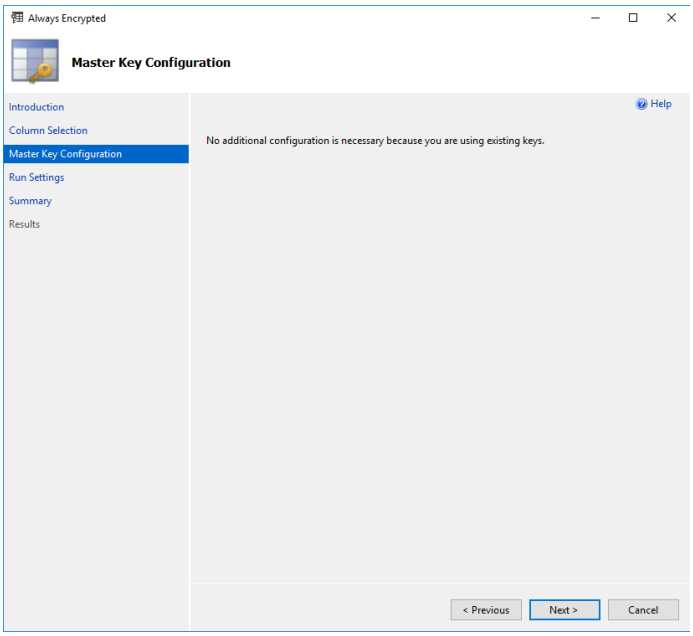
1. Log in to the client with the <domain>\dbuser account.
2. Launch **Microsoft SQL Server Management Studio**.
3. Connect to the database on the remote SQL server, enabling **Always Encrypted**, see [encrypt-decrypt-column-with-ssms:::generate-mycmk-mycek-ssms].
4. In the **Object Explorer**, right-click the **TestDatabase** database and select **Tasks** > **Encrypt Columns...**.
5. On the **Introduction** screen, select **Next**.



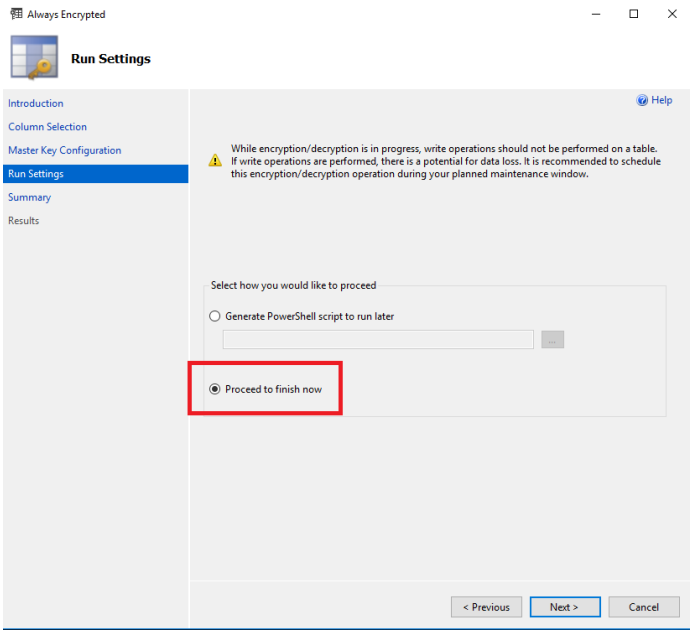
6. On the **Column Selection** screen, select the column **Name**, **Encryption Type**, and **Encryption Key**. Then select **Next**.



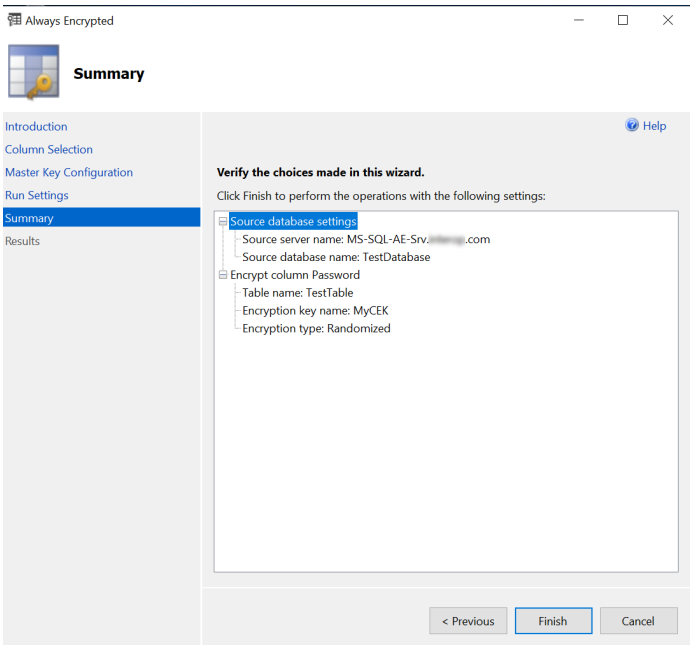
7. On the **Master Key Configuration** screen, select **Next**.



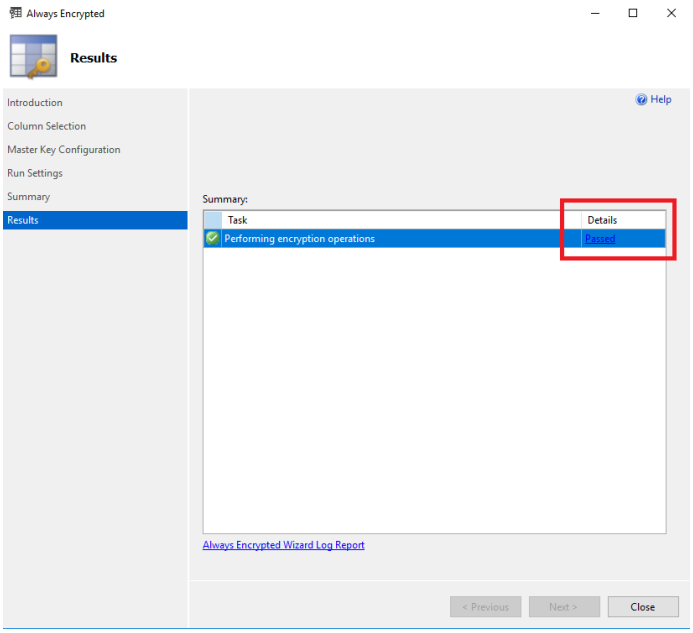
8. On the **Run Settings** screen, select **Proceed to finish now**. Then select **Next**.



9. On the **Summary** screen, verify the configuration choices. Then select **Finish**.



10. Present the OCS, select the HSM, and enter the passphrase.
11. Check that **Passed** appears in the **Details** column of the **Results** screen.



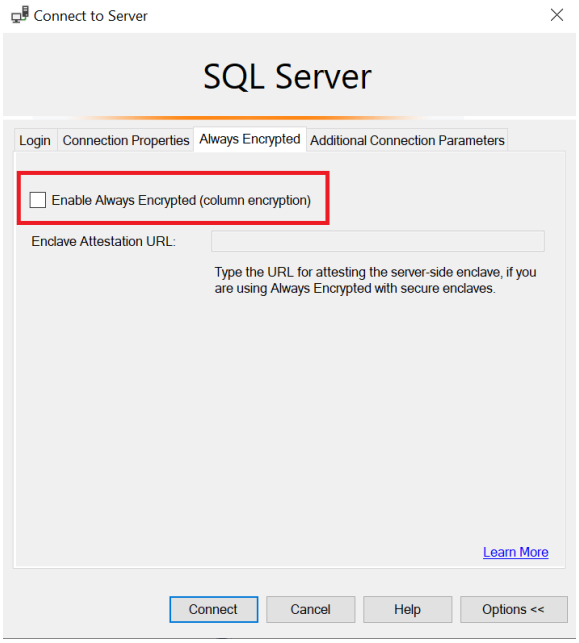
The column is encrypted in the SQL server, but it shows as clear text on the **Microsoft SQL Server Management Studio** GUI on the client. This is because **Always Encrypted** is performing the decryption at the client site.

12. Select **Close**.

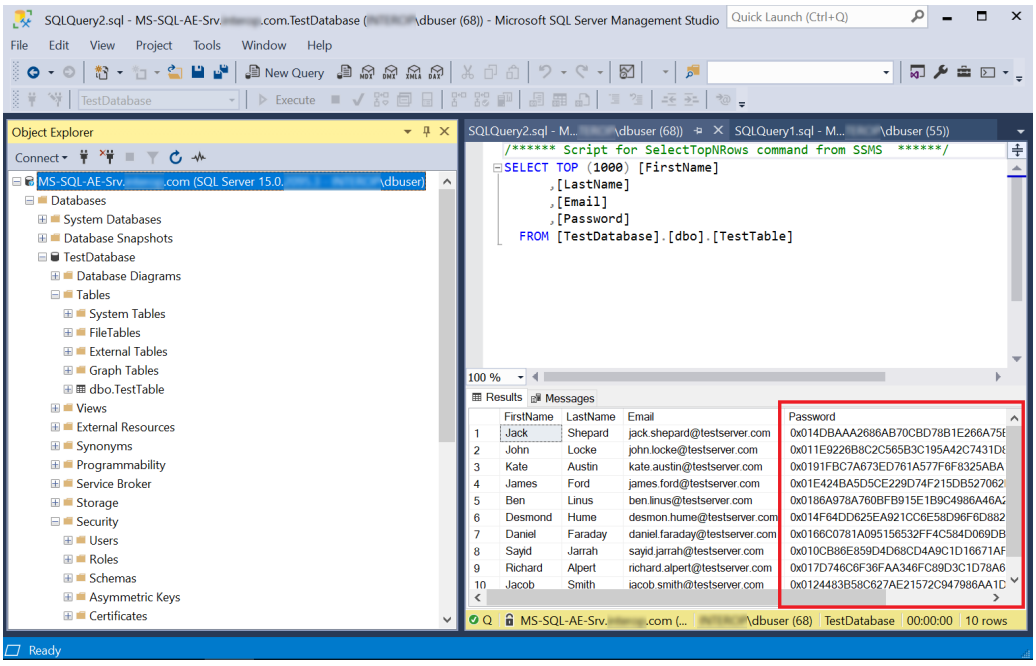
6.2. View an encrypted column

Reconnect to the SQL server with **Enable Always Encrypted** disabled to view the encrypted data stored in the SQL server.

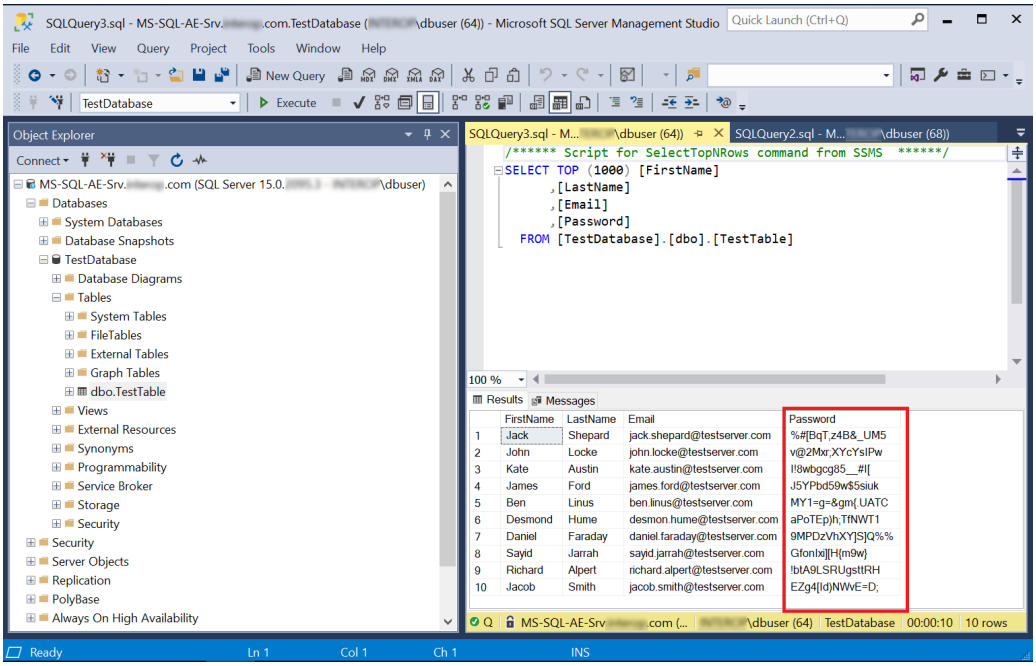
1. Connect to the SQL server but with the **Enable Always Encrypted** unchecked.



2. Right-click **dbo.Table** and select **Select Top 1000 Rows**. The column that was chosen for encryption now appears as ciphertext, that is, as an encrypted value.

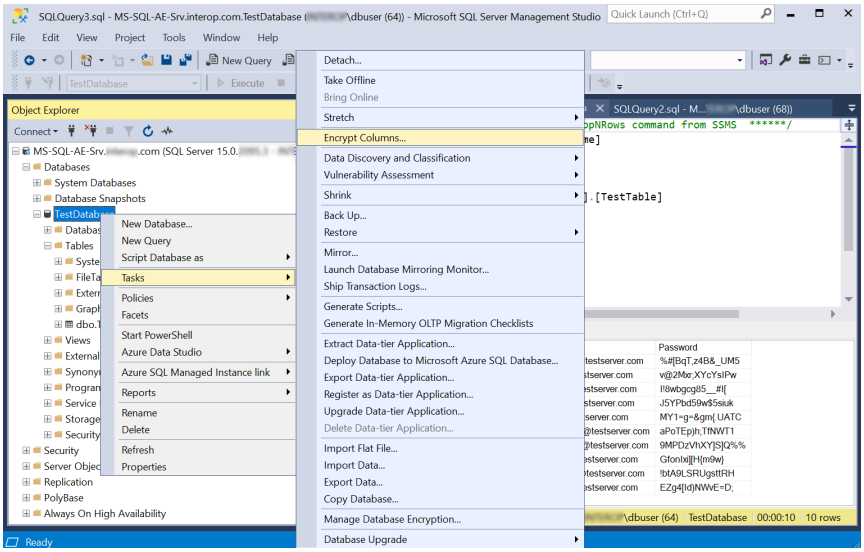


3. Reconnect to the SQL server, but with the **Enable Always Encrypted** checked.
4. Present the OCS, select the HSM, and enter the passphrase.
5. Right-click **dbo.Table** and select **Select Top 1000 Rows**. The column that was chosen for encryption is now being decrypted by **Always Encrypted** with the key protected by the nShield HSM.

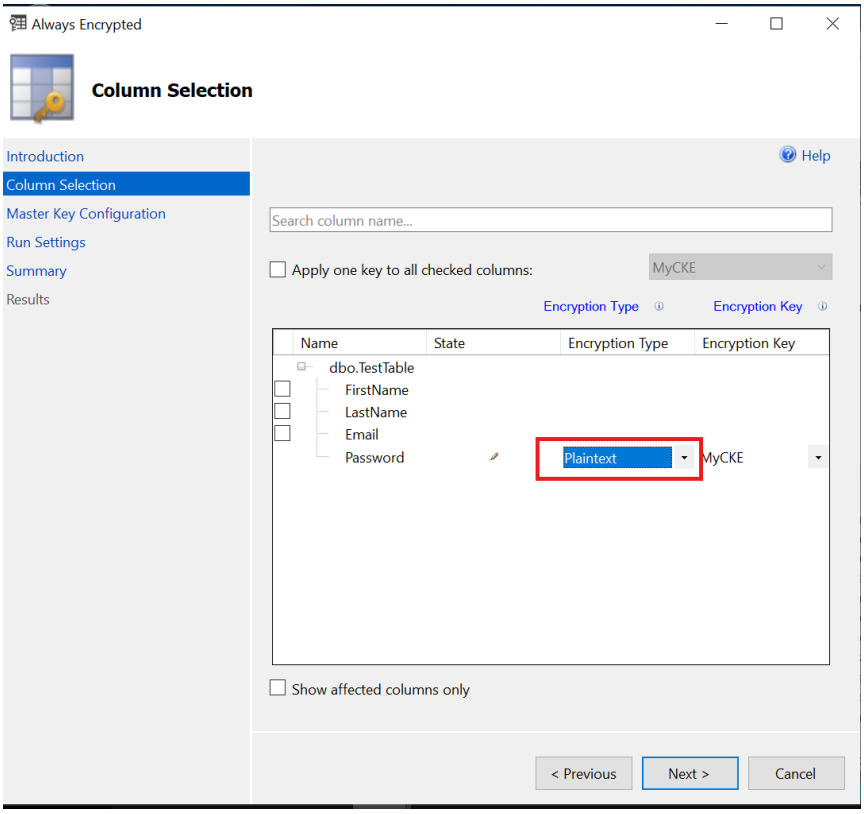


6.3. Remove column encryption

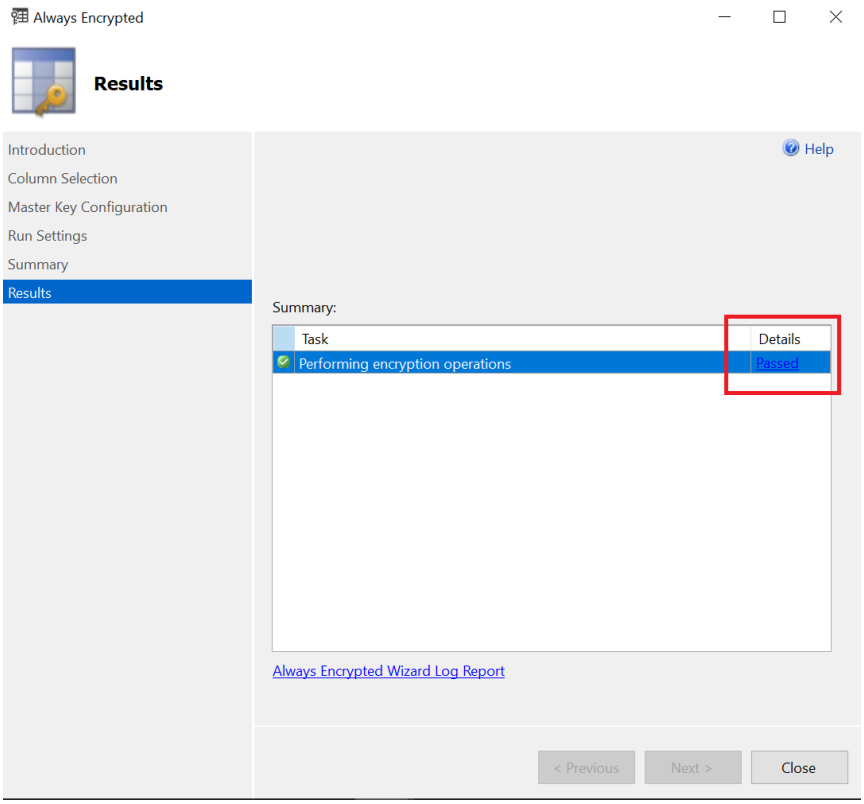
1. In the **Object Explorer**, right-click the **TestDatabase** database, and select **Tasks** > **Encrypt Columns...**.



2. On the **Introduction** screen, select **Next**.
3. On the **Column Selection** screen, for **Encryption Type** select **Plaintext**. Then select **Next**.



4. On the **Master Key Configuration** screen, select **Next**.
5. On the **Run Settings** screen, select **Proceed to finish now**. Then select **Next**.
6. On the **Summary** screen, verify the configuration choices. Then select **Finish**.
7. Present the OCS, select the HSM, and enter the passphrase.
8. Check that **Passed** appears in the **Details** column of the **Results** screen.



The column has been decrypted in the SQL server. To view the plain text data stored SQL server, reconnect to the server with Always Encrypted disabled, see [encrypt-decrypt-column-with-ssms:::view-encrypted-column].

9. Select **Close**.

Chapter 7. Encrypt or decrypt a column with PowerShell

To encrypt or decrypt a column with PowerShell:

- Encrypt a column
- Remove column encryption

7.1. Encrypt a column

To encrypt a column:

1. Log in to the client using the <domain>\dbuser account.
2. Launch PowerShell on the client computer and run the `Encrypt_Column_Named_Password.ps1` script (below).

```
# Import the SqlServer module.
Import-Module SqlServer

# Set up connection and database SMO objects
$sqlConnectionString = "Data Source=MS-SQL-AE-Srv.interop.com; Initial Catalog=TestDatabase; Integrated
Security=True; MultipleActiveResultSets=False; Connect Timeout=30; Encrypt=True;
TrustServerCertificate=True; Packet Size=4096; Application Name=`"Microsoft SQL Server Management Studio`""
$smoDatabase = Get-SqlDatabase -ConnectionString $sqlConnectionString

# If your encryption changes involve keys in Azure Key Vault, uncomment one of the lines below in order to
authenticate:
#   * Prompt for a username and password:
#Add-SqlAzureAuthenticationContext -Interactive

#   * Enter a Client ID, Secret, and Tenant ID:
#Add-SqlAzureAuthenticationContext -ClientID '<Client ID>' -Secret '<Secret>' -Tenant '<Tenant ID>'

# Change encryption schema
$encryptionChanges = @()

# Add changes for table [dbo].[TestTable]
$encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName dbo.TestTable.Password -EncryptionType
Randomized -EncryptionKey "MyCEK"
Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase
```

   The command line is:

```
> PowerShell -ExecutionPolicy Bypass -File Encrypt_Column_Named_Password.ps1
```

3. Present the OCS, select the HSM, and enter the passphrase.
4. Launch **Microsoft SQL Server Management Studio**. Do as indicated in encrypt-decrypt-column-with-powershell:::encrypt-decrypt-column-with-ssms.pdf to verify the column has been encrypted.

7.2. Remove column encryption

To remove column encryption:

1. Launch PowerShell on the client computer and run the `Decrypt_Column_Named_Password.ps1` script (below).

```
# Import the SqlServer module.
Import-Module SqlServer

# Set up connection and database SMO objects
$sqlConnectionString = "Data Source=MS-SQL-AE-Srv.interop.com; Initial Catalog=TestDatabase; Integrated
Security=True; MultipleActiveResultSets=False; Connect Timeout=30; Encrypt=True;
TrustServerCertificate=True; Packet Size=4096; Application Name=`"Microsoft SQL Server Management Studio`""
$smoDatabase = Get-SqlDatabase -ConnectionString $sqlConnectionString

# If your encryption changes involve keys in Azure Key Vault, uncomment one of the lines below in order to
authenticate:
#   * Prompt for a username and password:
#Add-SqlAzureAuthenticationContext -Interactive

#   * Enter a Client ID, Secret, and Tenant ID:
#Add-SqlAzureAuthenticationContext -ClientID '<Client ID>' -Secret '<Secret>' -Tenant '<Tenant ID>'

# Change encryption schema
$encryptionChanges = @()

# Add changes for table [dbo].[TestTable]
$encryptionChanges += New-SqlColumnEncryptionSettings -ColumnName dbo.TestTable.Password -EncryptionType
Plaintext
Set-SqlColumnEncryption -ColumnEncryptionSettings $encryptionChanges -InputObject $smoDatabase
```

   The command line is:

```
> PowerShell -ExecutionPolicy Bypass -File Decrypt_Column_Named_Password.ps1
```

2. Present the OCS, select the HSM, and enter the passphrase.
3. Launch **Microsoft SQL Server Management Studio**. Do as indicated in encrypt-decrypt-column-with-powershell:::encrypt-decrypt-column-with-ssms.pdf to verify the column has been encrypted.

Chapter 8. Test access to Always Encrypted keys by another user

To test access to Always Encrypted keys by another user:

1. Log in to the client using the <domain>\dbuser2 account.
2. Launch **Microsoft SQL Server Management Studio**.
3. Connect to the database on the remote SQL server, enabling **Always Encrypted**.
4. Present the OCS, select the HSM, and enter the passphrase.
5. Perform operations on the TestDatabase, which is possible since <domain>\dbuser2 has access to the same MyCMK and MyCEK keys created by <domain>\dbuser.

Chapter 9. Supported PowerShell SqlServer cmdlets

PowerShell cmdlet	Description
`Add-SqlColumnEncryptionKeyValue`	Adds a new encrypted value for an existing column encryption key object in the database.
`Complete-SqlColumnMasterKeyRotation`	Completes the rotation of a column master key.
`Get-SqlColumnEncryptionKey`	Returns all column encryption key objects defined in the database, or returns one column encryption key object with the specified name.
`Get-SqlColumnMasterKey`	Returns the column master key objects defined in the database, or returns one column master key object with the specified name.
`Invoke-SqlColumnMasterKeyRotation`	Initiates the rotation of a column master key.
`New-SqlAzureKeyVaultColumnMasterKeySettings`	Creates a `SqlColumnMasterKeySettings` object describing an asymmetric key stored in Azure Key Vault.
`New-SqlCngColumnMasterKeySettings`	Creates a `SqlColumnMasterKeySettings` object describing an asymmetric key stored in a key store supporting the Cryptography Next Generation (CNG) API.
`New-SqlColumnEncryptionKey`	Creates a new column encryption key object in the database.
`New-SqlColumnEncryptionKeyEncryptedValue`	Produces an encrypted value of a column encryption key.
`New-SqlColumnEncryptionSettings`	Creates a new `SqlColumnEncryptionSettings` object that encapsulates information about a single column's encryption, including CEK and encryption type.

PowerShell cmdlet	Description
`New-SqlColumnMasterKey`	Creates a new column master key object in the database.
`New-SqlCspColumnMasterKeySettings`	Creates a `SqlColumnMasterKeySettings` object describing an asymmetric key stored in a key store with a Cryptography Service Provider (CSP) supporting Cryptography API (CAPI).
`Remove-SqlColumnEncryptionKey`	Removes the column encryption key object from the database.
`Remove-SqlColumnEncryptionKeyValue`	Removes an encrypted value from an existing column encryption key object in the database.
`Remove-SqlColumnMasterKey`	Removes the column master key object from the database.
`Set-SqlColumnEncryption`	Encrypts, decrypts or re-encrypts specified columns in the database.

The full list of cmdlets and additions to the `SqlServer` module can be found in the Microsoft Online Documentation.

Chapter 10. Additional resources and related products

10.1. nShield Connect

10.2. nShield as a Service

10.3. Entrust products

10.4. nShield product documentation