Microsoft SQL Server 2019 Always Encrypted

nShield® HSM Integration Guide

01 Apr 2022

Contents

1. Introduction

Always Encrypted is a feature in Windows SQL Server 2019 designed to protect sensitive data both at rest and in flight between an on-premises 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 on–premises 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 on-premises client application server.

1.1. Product configurations

Entrust successfully tested nShield HSM integration with Windows SQL Server 2019 and the Always Encrypted feature in the following configurations:

1.1.1. Remote server

Product	Version
SQL Server	Microsoft SQL Server 2019
Base OS	Windows Server 2019 Datacenter

1.1.2. On-premises client

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

1.2. Supported nShield features

Entrust successfully tested nShield HSM integration with the following features:

Feature	Support
Module Only Key	Yes
OCS cards	Yes

1.3. Supported nShield hardware and software versions

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

1.3.1. Connect XC

Security World Software	Firmware	Image	OCS	Module
12.80.4	FIPS 12.50.11	12.60.10	✓	✓
12.80.4	CC 12.50.7	12.50.7	✓	✓

1.3.2. Connect +

Security World Software	Firmware	Image	OCS	Module
12.80.4	FIPS 12.50.8	12.60.10	✓	✓
12.40 Compatibility Package	CC 2.55.4	12.45.1	✓	✓

1.4. 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.

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

Process	Duty Role
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
`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

1.5. Using multiple on-premises client servers

Each client server wanting access to the contents of data encrypted with a given CEK must have access to an HSM in the same Security World and have 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.

2. Install and configure

This installation must be performed on the on-premises client computer.

The nShield Security World software will be installed on this computer. This computer will also be made a client of the HSM.

2.1. 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. Add the Security World utilities path C:\Program Files\nCipher\nfast\bin to the Windows system path.
3. Open the firewall port 9004 for the HSM connections.
4. 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:
   - https://nshieldsupport.entrust.com/hc/en-us/articles/360021378272-How-To-Locally-Set-up-a-new-or-replacement-nShield-Connect
   - https://nshieldsupport.entrust.com/hc/en-us/articles/360014011798-How-To-Remotely-Setup-a-new-or-replacement-nShield-Connect
   - https://nshieldsupport.entrust.com/hc/en-us/articles/360013253417-How-To-Remotely-Setup-a-new-or-replacement-nShield-Connect-XC-Serial-Console-Model
5. Open a command window and run the following to confirm that the HSM is `operational`.

```
C:\Users\dbuser>enquiry
Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        530E-02E0-D947 7724-8509-81E3 09AF-0BE9-53AA 9E10-03E0-D947
 mode                 operational
...
Module #1:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        530E-02E0-D947
 mode                 operational
 ...
```

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

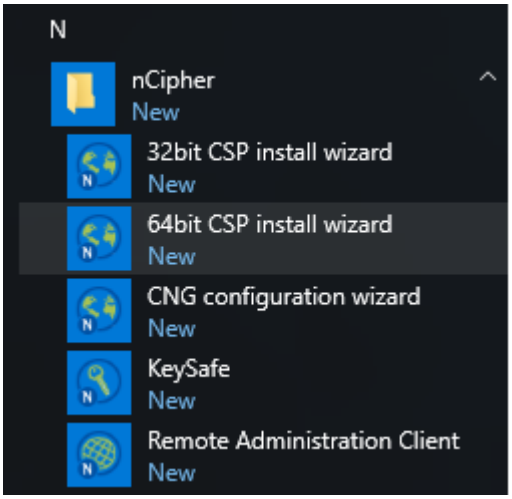
```
C:\Users\dbuser>nfkminfo
World
 generation  2
 state       0x37270008 Initialised Usable ...
 ...
Module #1
 generation 2
 state      0x2 Usable
 ...
```

2.2. Install and register the CNG provider

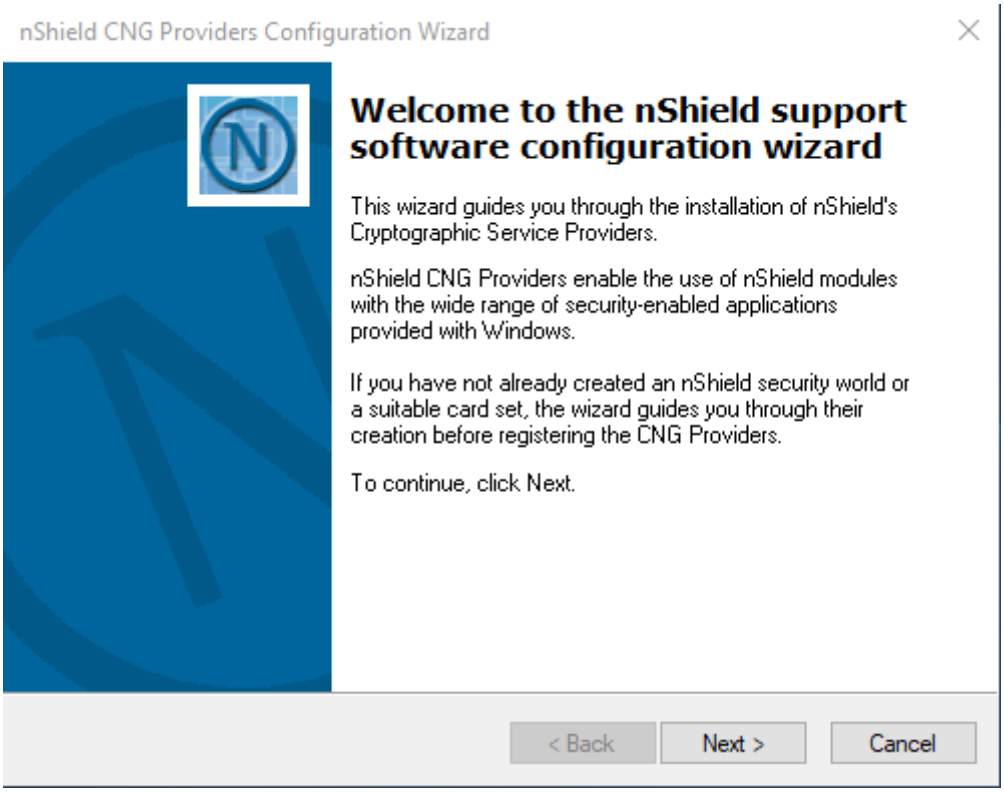
1. Open a command window as administrator and type the following to put the HSM in `pre-initialization` mode. This operation takes about a minute to complete.

```
C:\Windows\system32>enquiry -m 1
Module #1:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        530E-02E0-D947
 mode                 operational
 ...

C:\Windows\system32>nopclearfail -I -m 1
Module 1, command ClearUnitEx: OK

C:\Windows\system32>enquiry -m 1
Module #1:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        530E-02E0-D947
 mode                 pre-initialization
 ...
```

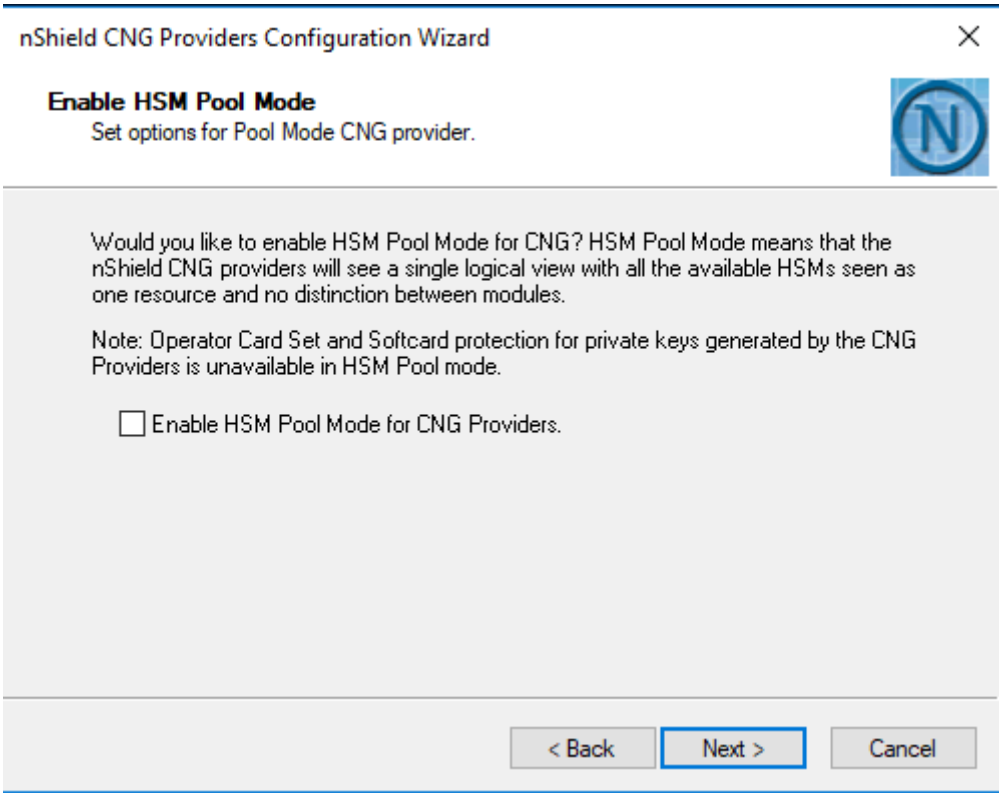
2. Select the **Start** button to access all applications. Look for the recently installed nShield utilities.
3. Double-click the CNG configuration wizard and run it as Administrator.



4. Select **Next** on the **CNG Install** welcome screen.



5. Select **Next** on the **Enable HSM Pool Mode** screen. Leave the **Enable HSM Pool Mode for CNG Providers** check box un-checked.



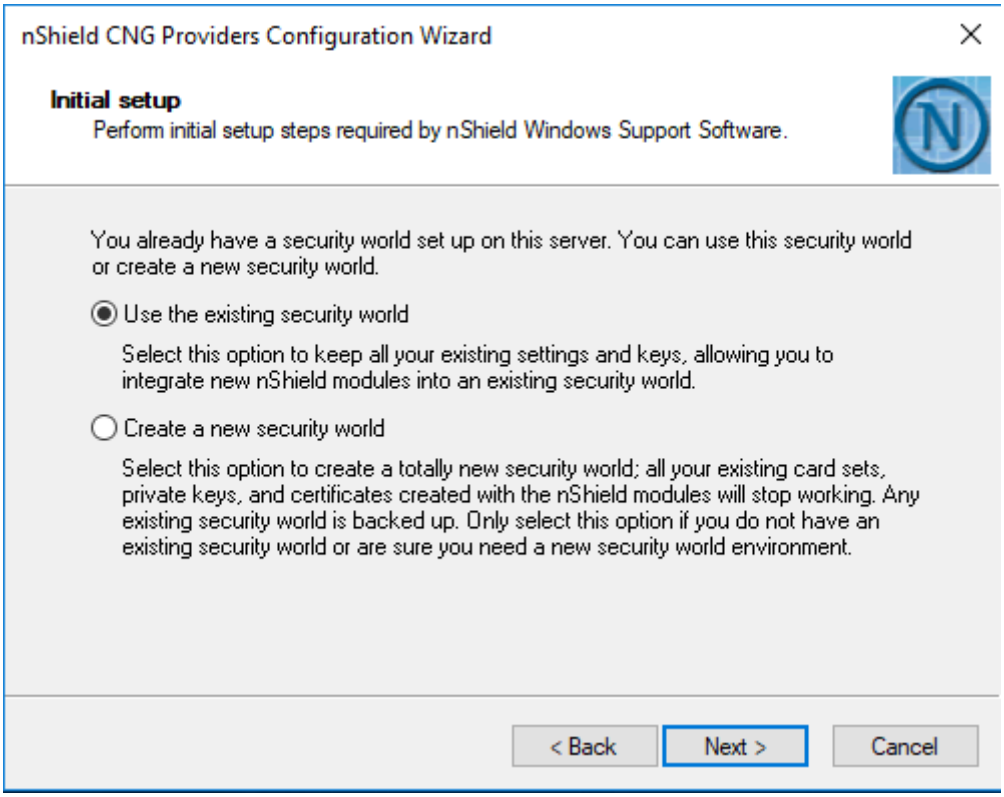
6. At the **Security World** screen, select:
   - **Use the existing security world** if you already have a Security World that you intend to use for Always Encrypted. The corresponding `world` and `module_xxxx-`

`xxxx-xxxx` files most be present in the `%NFAST_KMDATA%\local` folder. Be prepared to present the quorum of Administrator cards.

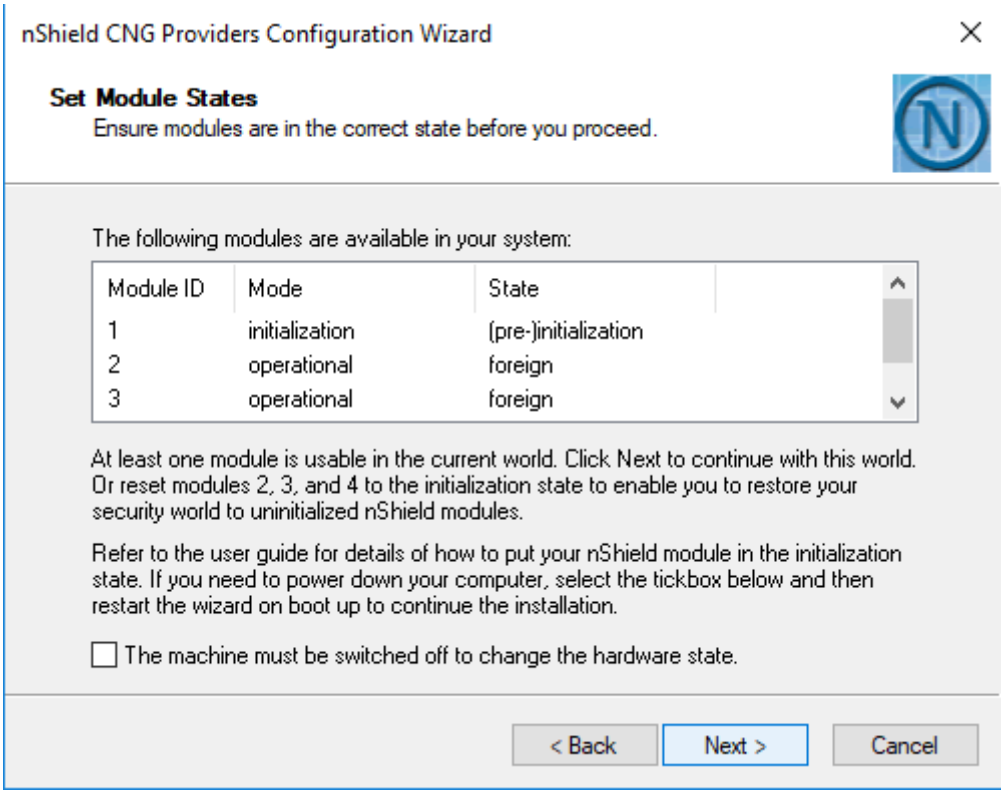
- **Create a new Security World** if you do not currently have a Security World or would like to create a new Security World.

For the purposes of this integration guide we have chosen to use an existing Security World. For instructions on how to create and configure a new Security World, see the *Installation Guide* and *User Guide* for your HSM.

Select **Next**.

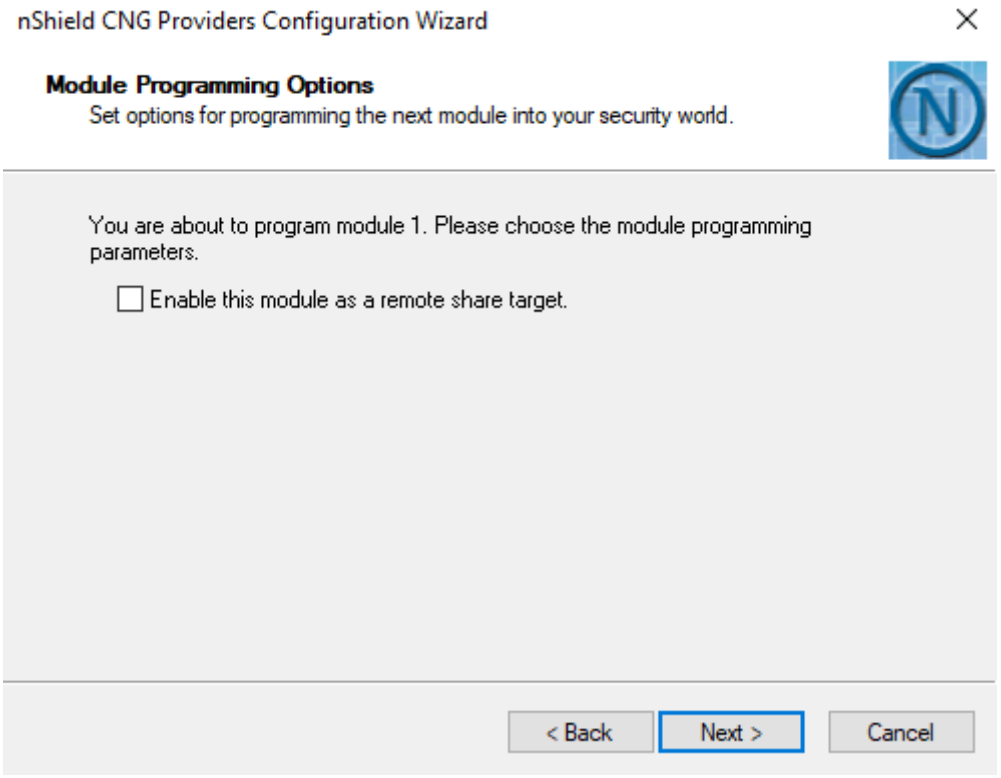


7. The **Set Module States** pop-up shows the available HSM(s). Select the desired HSM. The state of the selected HMS should be `(pre-)initialisation`. Select **Next**.



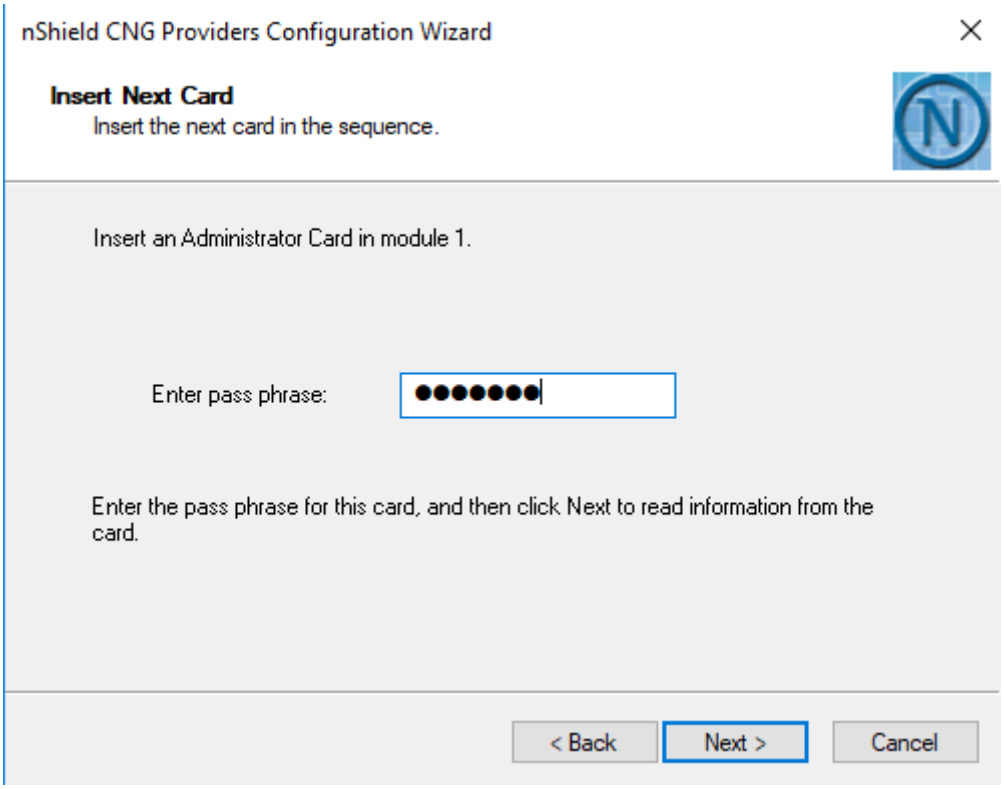
8. At the **Module Programming Options** screen, clear **Enable this module as a remote target** and select **Next**. It will take about a minute before the screen changes.

Please be aware that this is not to be confused with the nShield Remote Administration utility.



9. Insert the first Administrator Card in the HSM, enter the passphrase and select **Next**. Repeat this step for the other Administrator Cards as required.

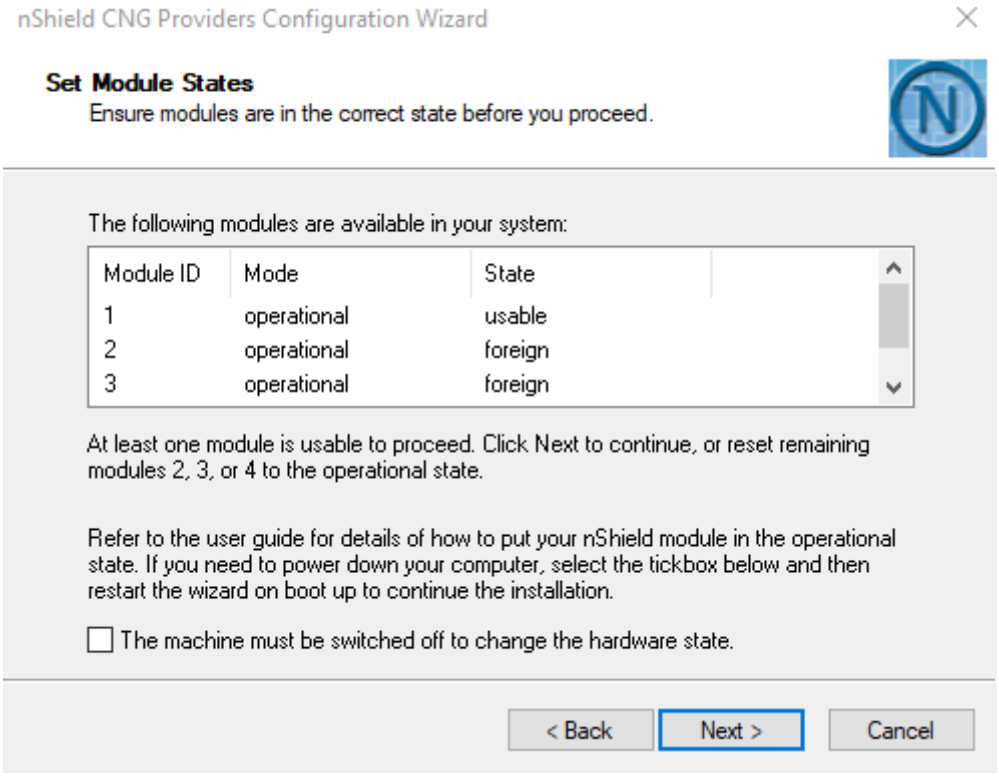
   Loading or creating the Security World takes about a minute.



10. Return the HSM to `Operational` mode. This operation takes about a minute to complete.

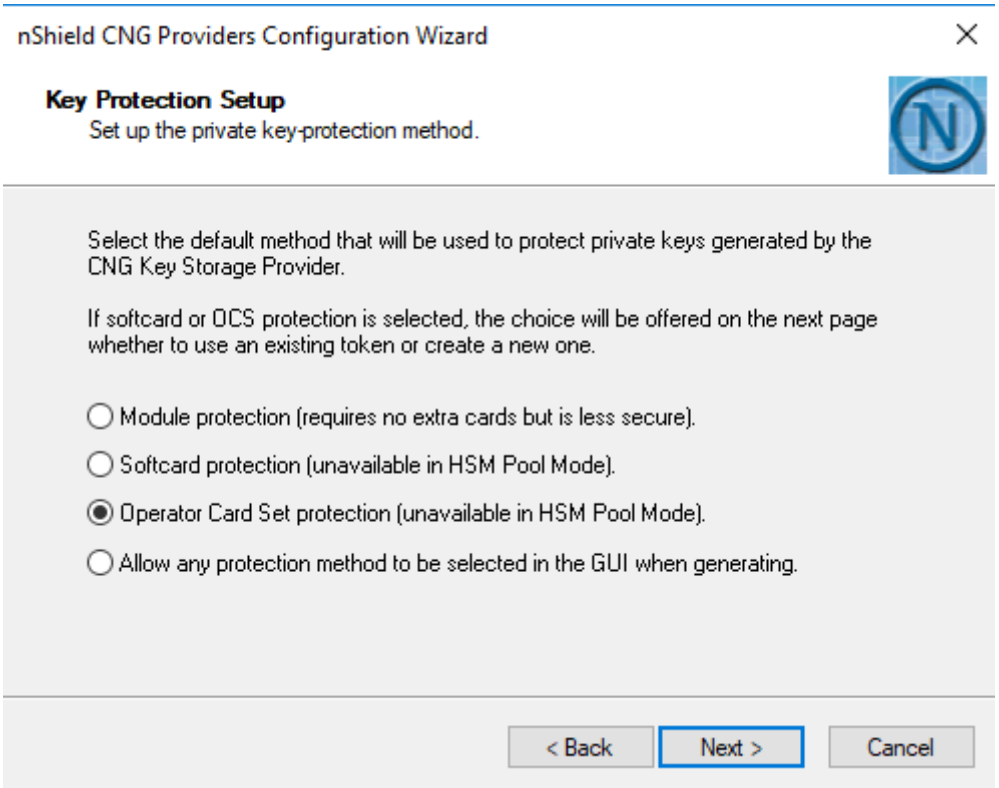
```
C:\Windows\system32>enquiry -m 1
Module #1:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        530E-02E0-D947
 mode                 initialization
 ...

C:\Windows\system32>nopclearfail -O -m 1
Module 1, command ClearUnitEx: OK

C:\Windows\system32>enquiry -m 1
Module #1:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        530E-02E0-D947
 mode                 operational
 ...
```

   The module state will change to `Usable`.



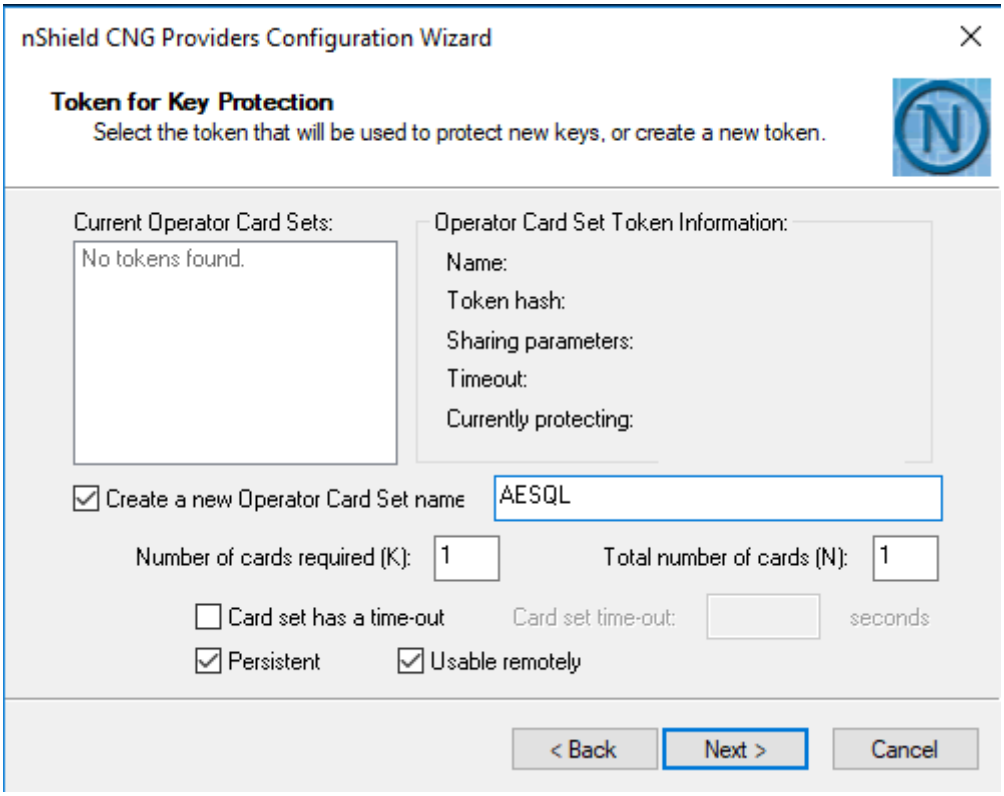
Select **Next**.

11. Select **Operator Card Set** in the **Key Protection Setup**. Select **Next**.



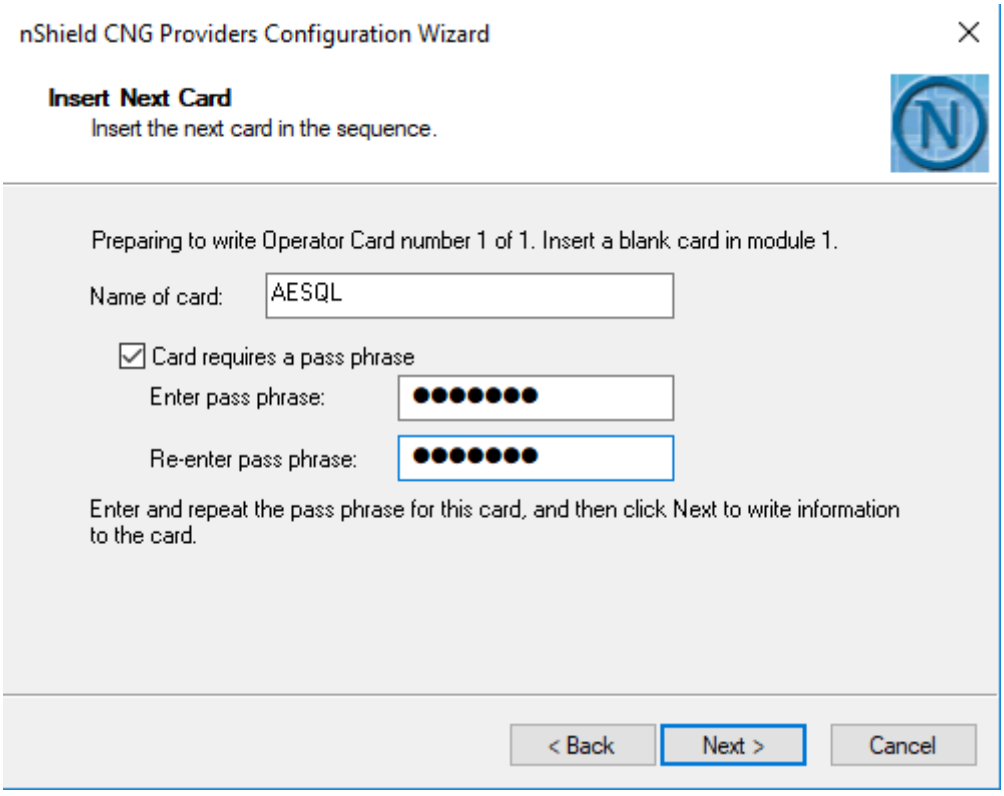
Select **Next**.

12. Enter the OCS name, K of N values, and check **Persistent** and **Usable remotely** as show.



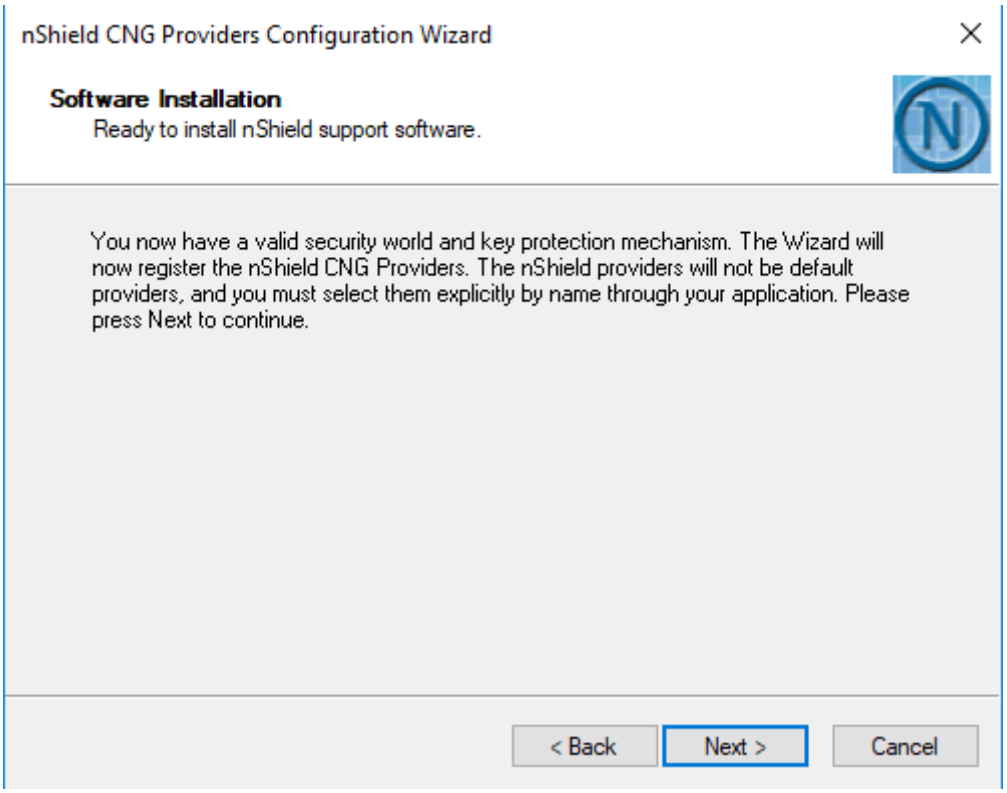
Select **Next**.

13. Insert a blank Operator Card in the HSM. On the **Insert Next Card** screen enter a name to for the OCS card and passphrase.



Select **Next**.

14. Select **Next** and **Finish**. The nShield CNG providers will now be installed and the key Storage Provider will be registered.

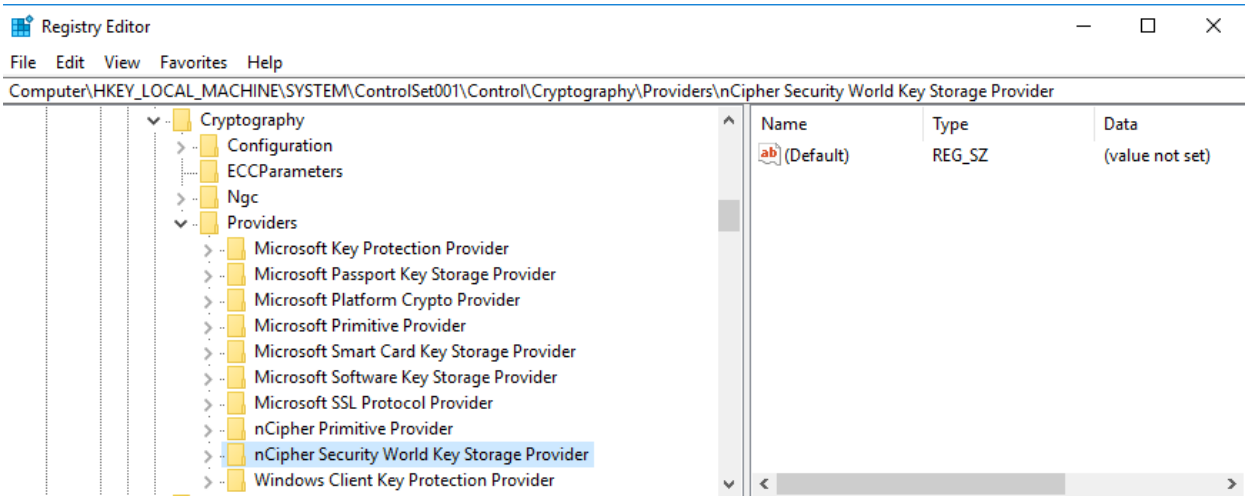


15. Open a command window as administrator and type the following to confirm the KSP has been successfully registered. Look for **nCipher Security World Key Storage Provider**.

```
C:\Windows\system32>cnglist.exe --list-providers
Microsoft Key Protection Provider
Microsoft Passport Key Storage Provider
Microsoft Platform Crypto Provider
Microsoft Primitive Provider
Microsoft Smart Card Key Storage Provider
Microsoft Software Key Storage Provider
Microsoft SSL Protocol Provider
Windows Client Key Protection Provider
nCipher Primitive Provider
nCipher Security World Key Storage Provider
```

16. Check the registry in `CNGRegistry`:

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Cryptography\Providers\nCipherSecurityWorldKeyStorageProvider
```



2.3. Install and configure SqlServer PowerShell module

1. Open a PowerShell session as Administrator and run:

```
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

Install-PackageProvider Nuget –force –verbose
```

2. Update PowerShellGet:

```
Install-Module –Name PowerShellGet –force –verbose
```

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

```
Install-Module -Name SqlServer -force -verbose -AllowClobber
```

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

4. Once installed (if you are using PowerShell ISE refresh the Commands pane if you are using PowerShell open a new session), confirm the install by running:

```
Get-Module -list -Name SqlServer
```

5. You should see something similar to the output below:

```
Directory: C:\Program Files\WindowsPowerShell\Modules

ModuleType Version    Name          ExportedCommands
---------- -------    ----          ----------------
Manifest   21.0.17152 SqlServer     {Add-SqlColumnEncryptionKeyValue, Complete-SqlColumnMasterKeyRotatio···
```

3. Generate the encryption keys

3.1. Generate the Always Encrypted Column Master Key (CMK) protected by the nShield HMS

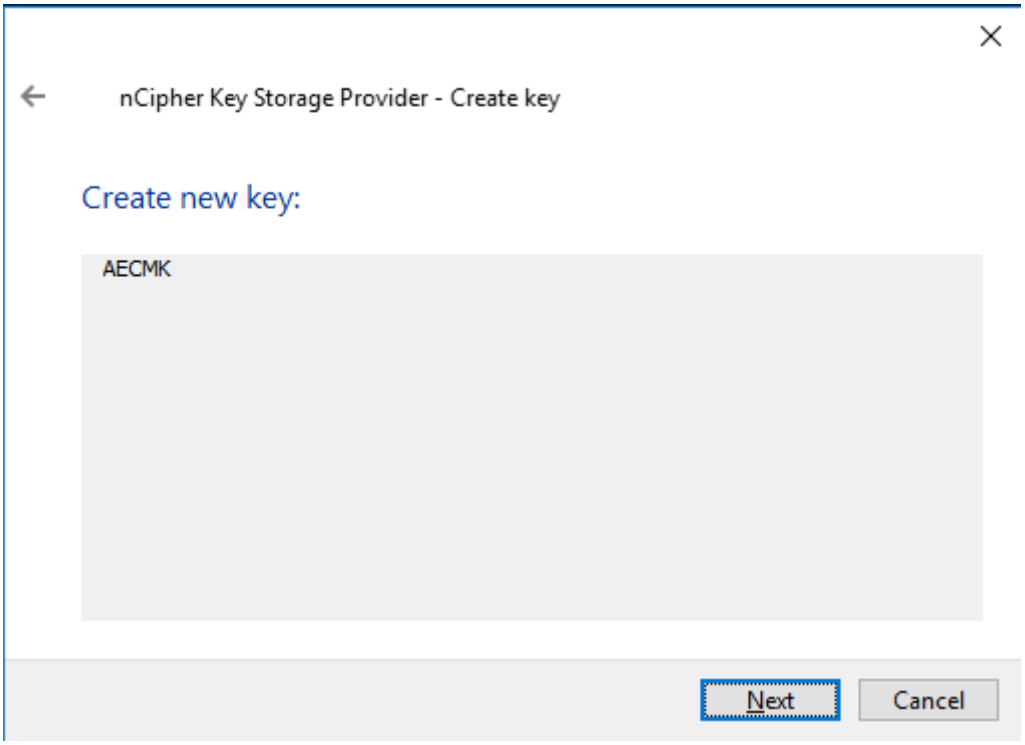
1. Launch PowerShell on the on-premises client computer as Administrator, and run the `Generate_CMK.ps1` script.

```
$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)
```

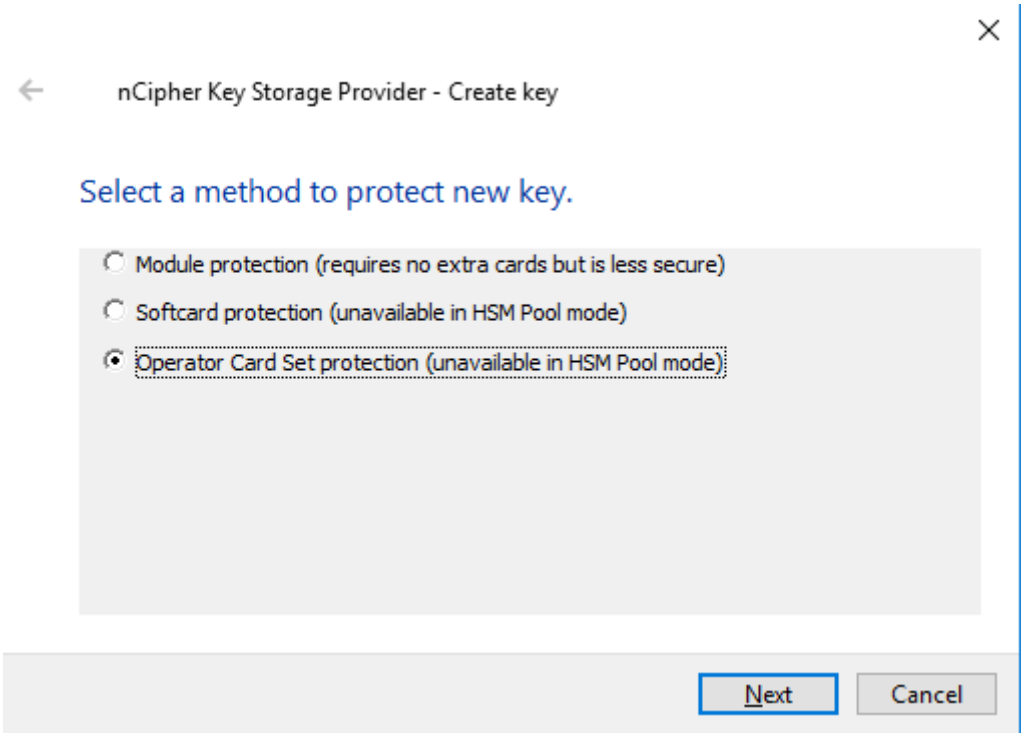
   The command line is

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

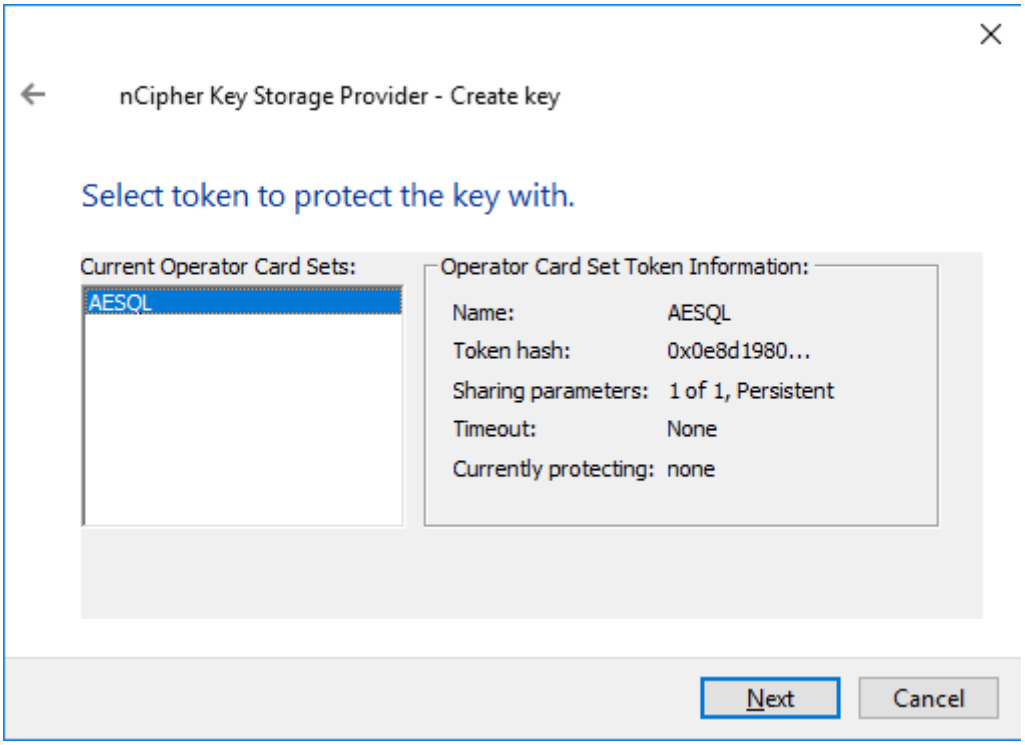
   a. The following pop-window should appear. Select **Next**.



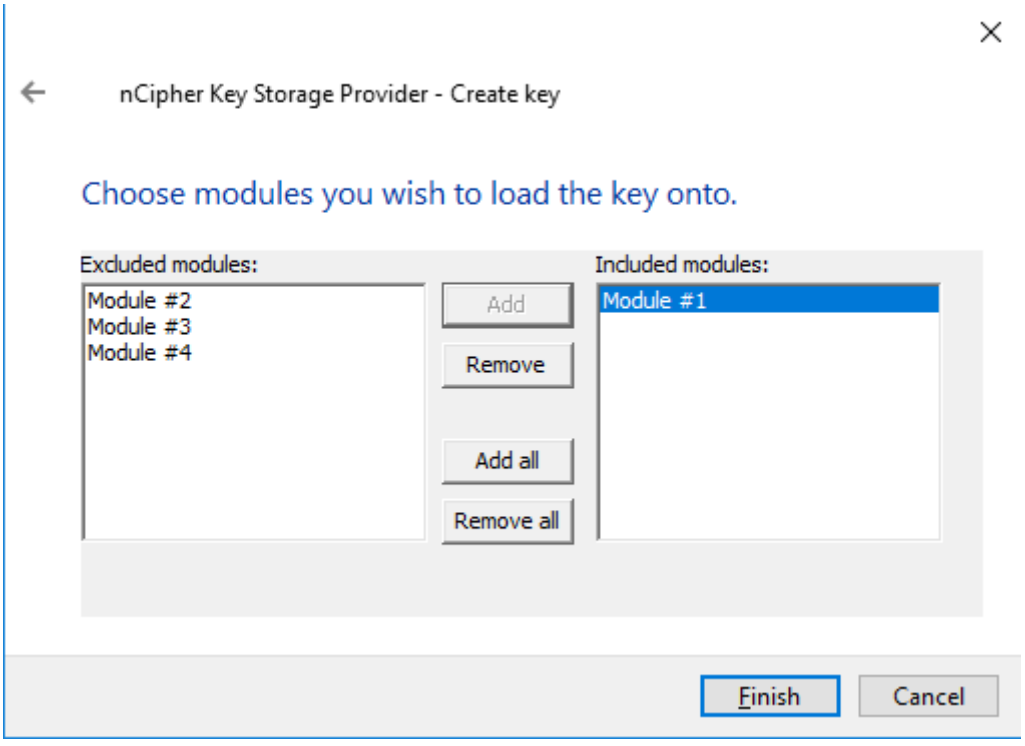
b. Select the **Operator Card Set Protection**. Insert the OCS card in the HSM and select **Next**.



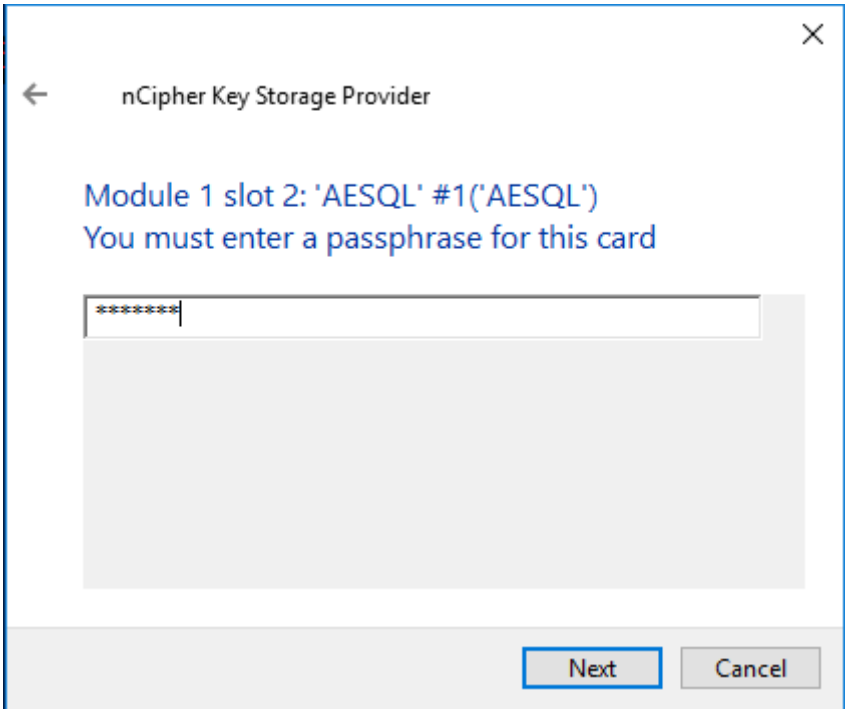
c. Select **Next**.



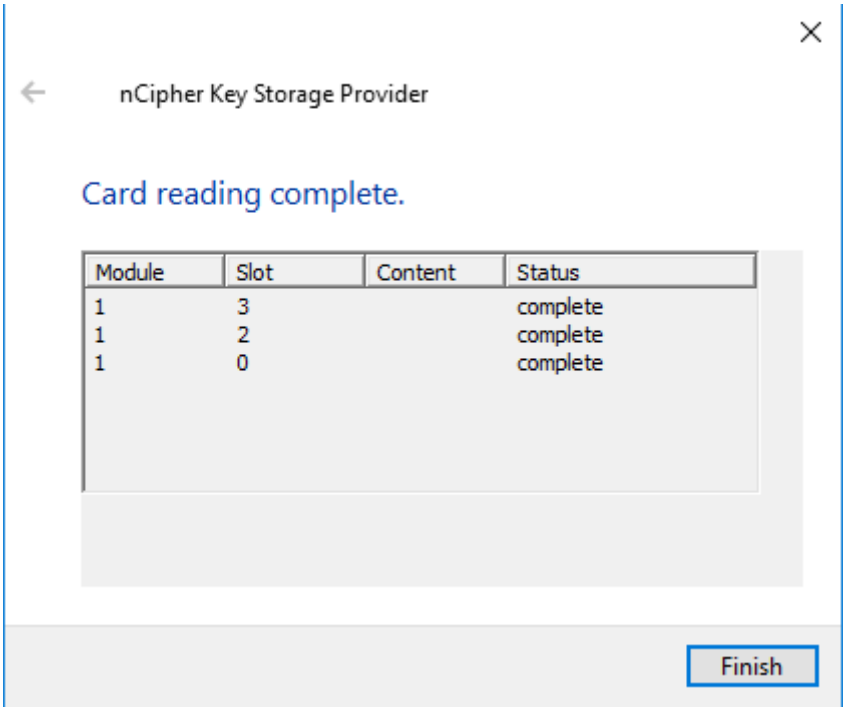
d. Select the HSM and select **Finish**.



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



f. 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 On-Premise Client server, where it is stored as an Application Key Token in the `%NFAST_KMDATA%\local folder` (`:\ProgramData\nCipher\Key Management Data\local`).

2. Verify the new key as follows on a command window.

```
C:\Users\dbuser>nfkminfo -k

Key list - 1 keys
 AppName caping              Ident s-1-5-21-2556418611-2173580918-1658130183-1001--
7b7eb65c095c556e5da059480e6ca2ed512dacc1
```

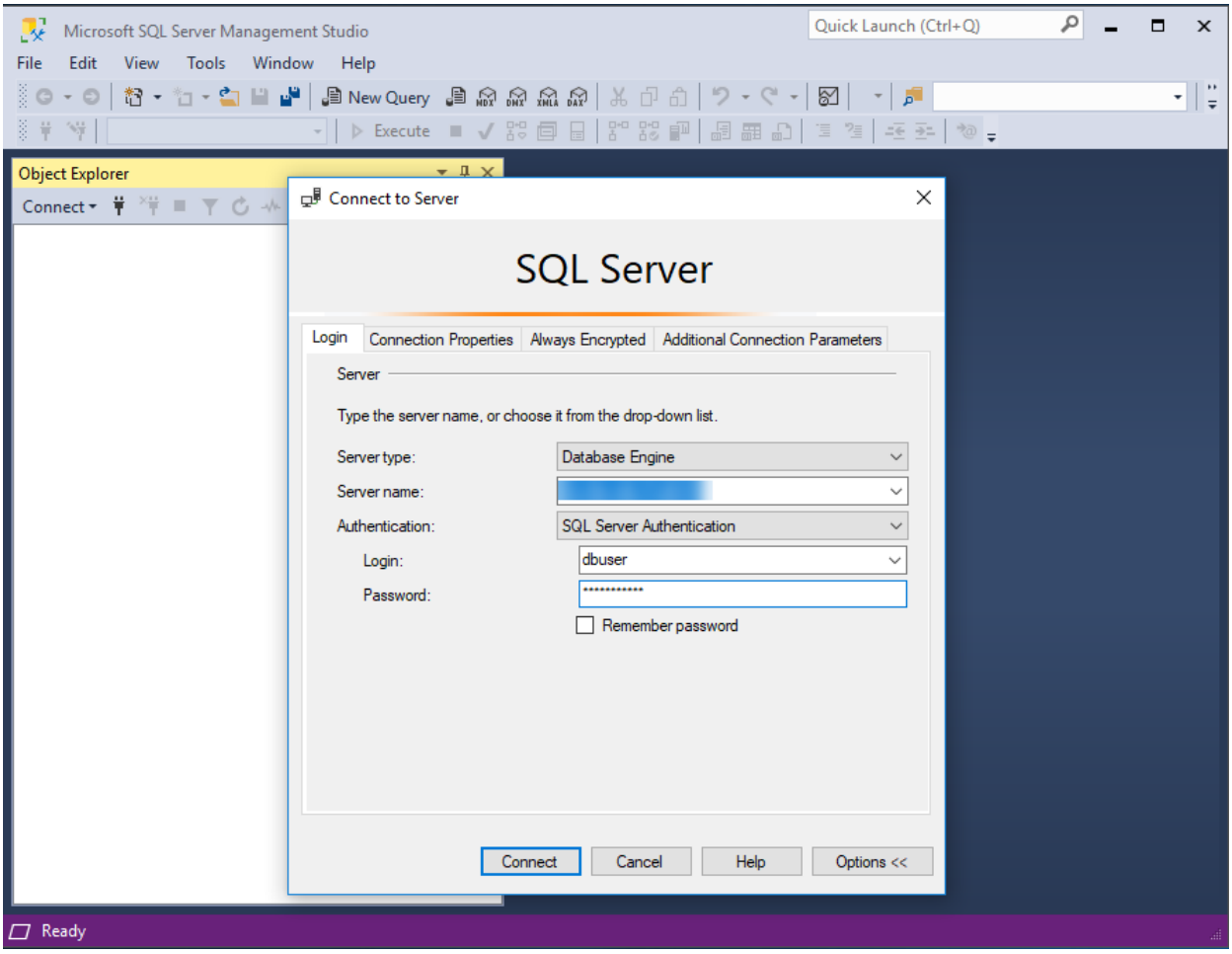
3. Display the information about the key by copy-pasting the key name above as follows.

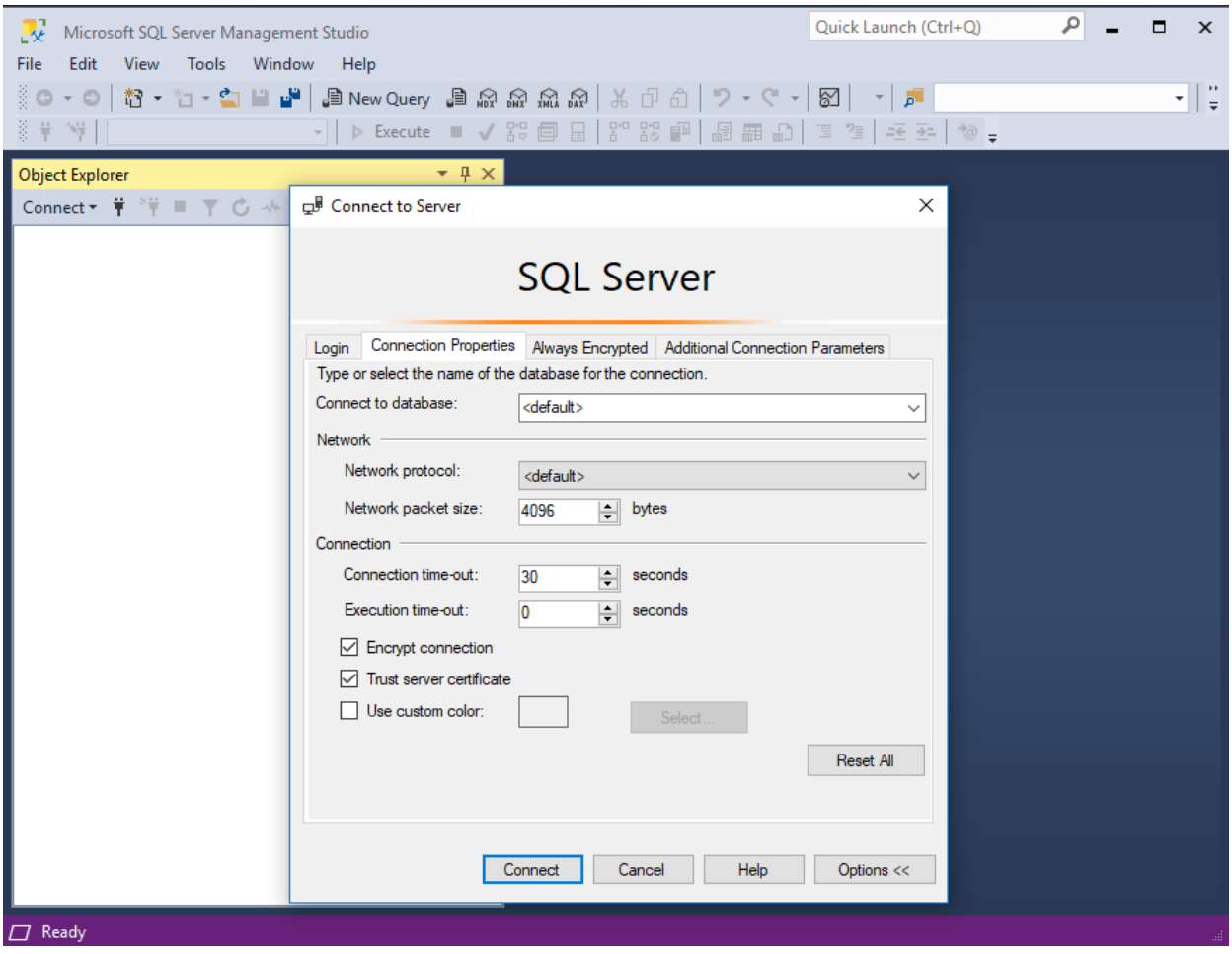
```
C:\Users\dbuser>nfkminfo -k caping s-1-5-21-2556418611-2173580918-1658130183-1001--
7b7eb65c095c556e5da059480e6ca2ed512dacc1
Key AppName caping Ident s-1-5-21-2556418611-2173580918-1658130183-1001--7b7eb65c095c556e5da059480e6ca2ed512dacc1
 BlobKA length         1128
 BlobPubKA length      484
 BlobRecoveryKA length 1496
 name                  "AECMK"
 hash                  76071834044810539e7354f468cc2cae61a448da
 recovery              Enabled
 protection            CardSet
 other flags           PublicKey !SEEAppKey !NVMemBlob +0x0
 cardset               0e8d19801b25d774c3b2bab5a643ec7c20a5255d
 gentime               2021-03-30 19:18:14
 SEE integrity key     NONE

BlobKA
 format                6 Token
 other flags           0x0
 hkm                   2a2e6b22ad6a72673473511d91304efd2f76e197
 hkt                   0e8d19801b25d774c3b2bab5a643ec7c20a5255d
 hkr                   none

BlobRecoveryKA
 format                9 UserKey
 other flags           0x0
 hkm                   none
 hkt                   none
 hkr                   fc4cbd1a6e88c08dd35912d0aecabf47ff1e0c2a

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

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

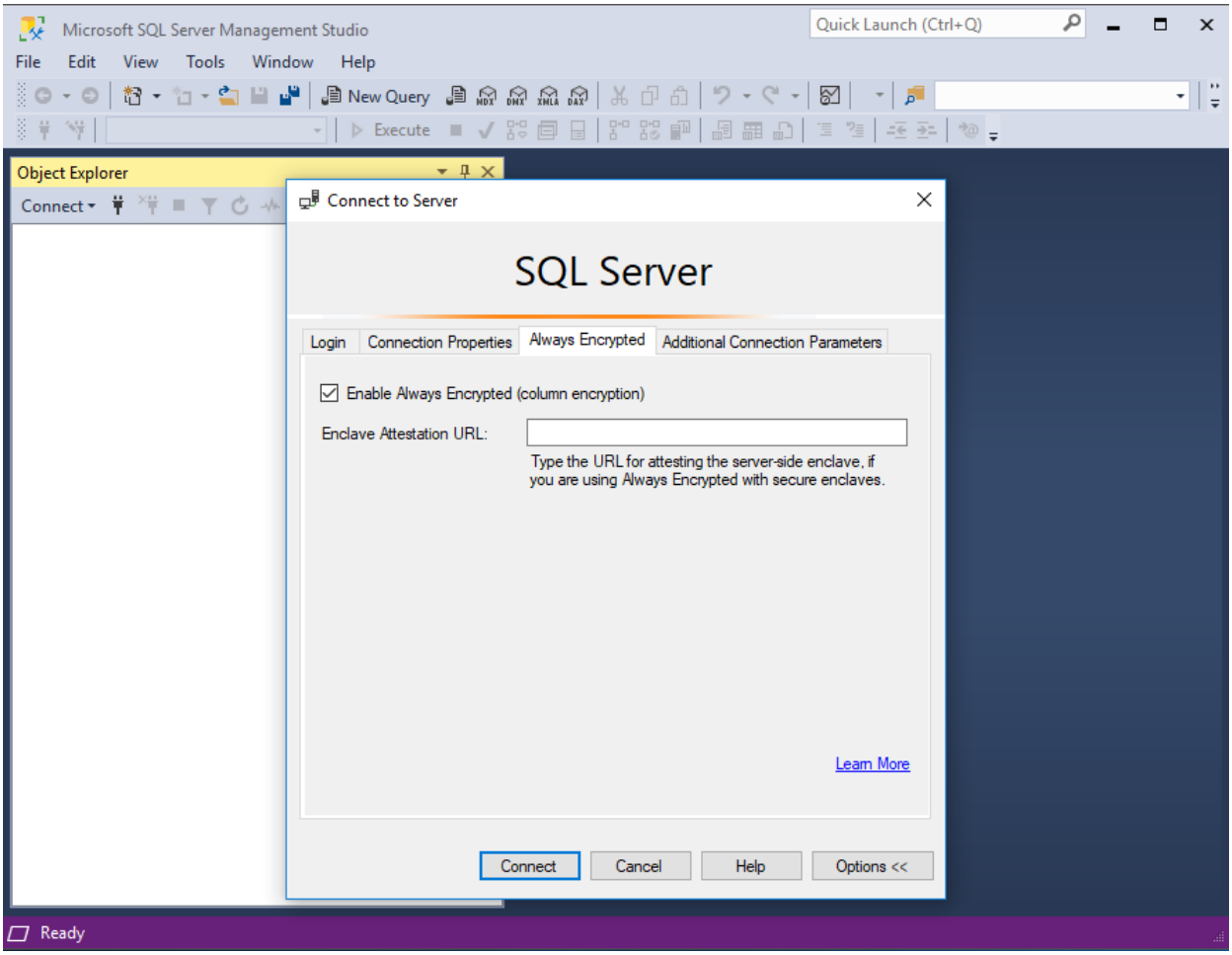
3.2. Generate My Column Master Key (MyCMK) and My Column Encryption Key (MyCEK) with SSMS

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

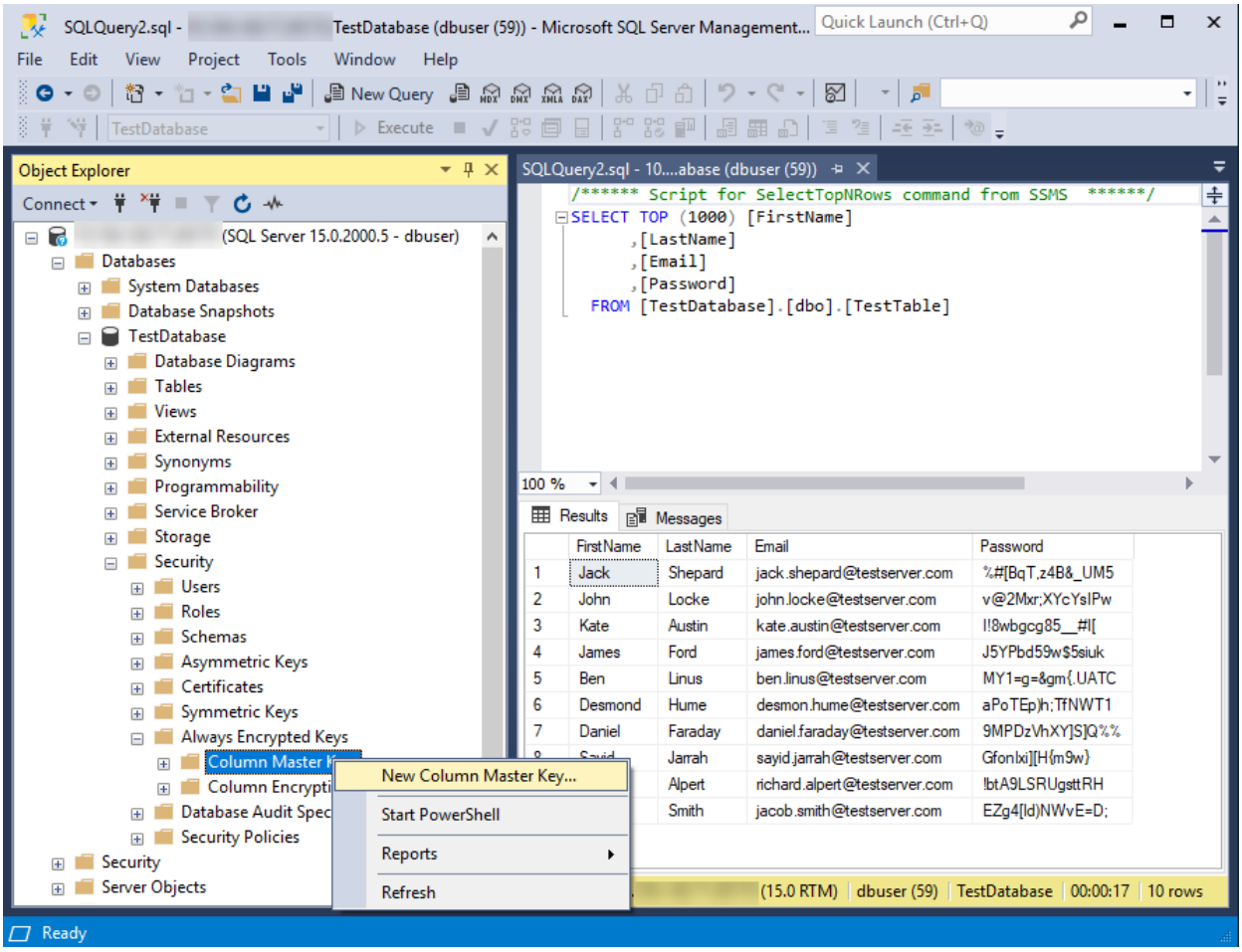
1. Launch **Microsoft SQL Server Management Studio** on the on-premises client computer.
2. As the `dbuser` user, connect to the database on the SQL server on the hosting site.


Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Tools Window Help
New Query
Execute
Object Explorer
Connect
Connect to Server
SQL Server
Login
Connection Properties
Always Encrypted
Additional Connection Parameters
Server
Type the server name, or choose it from the drop-down list.
Server type:
Database Engine
Server name:
Authentication:
SQL Server Authentication
Login:
dbuser
Password:
Remember password
Connect
Cancel
Help
Options <<
Ready


Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Tools Window Help
New Query
Execute
Object Explorer
Connect
Connect to Server
SQL Server
Login
Connection Properties
Always Encrypted
Additional Connection Parameters
Type or select the name of the database for the connection.
Connect to database:
<default>
Network
Network protocol:
<default>
Network packet size:
4096
bytes
Connection
Connection time-out:
30
seconds
Execution time-out:
0
seconds
Encrypt connection
Trust server certificate
Use custom color:
Select...
Reset All
Connect
Cancel
Help
Options <<
Ready



3. Using **Object Explorer**, select the **Security** directory under the desired Database. Select **Always Encrypted Keys** to expand it, then select **New Column Master Key**.



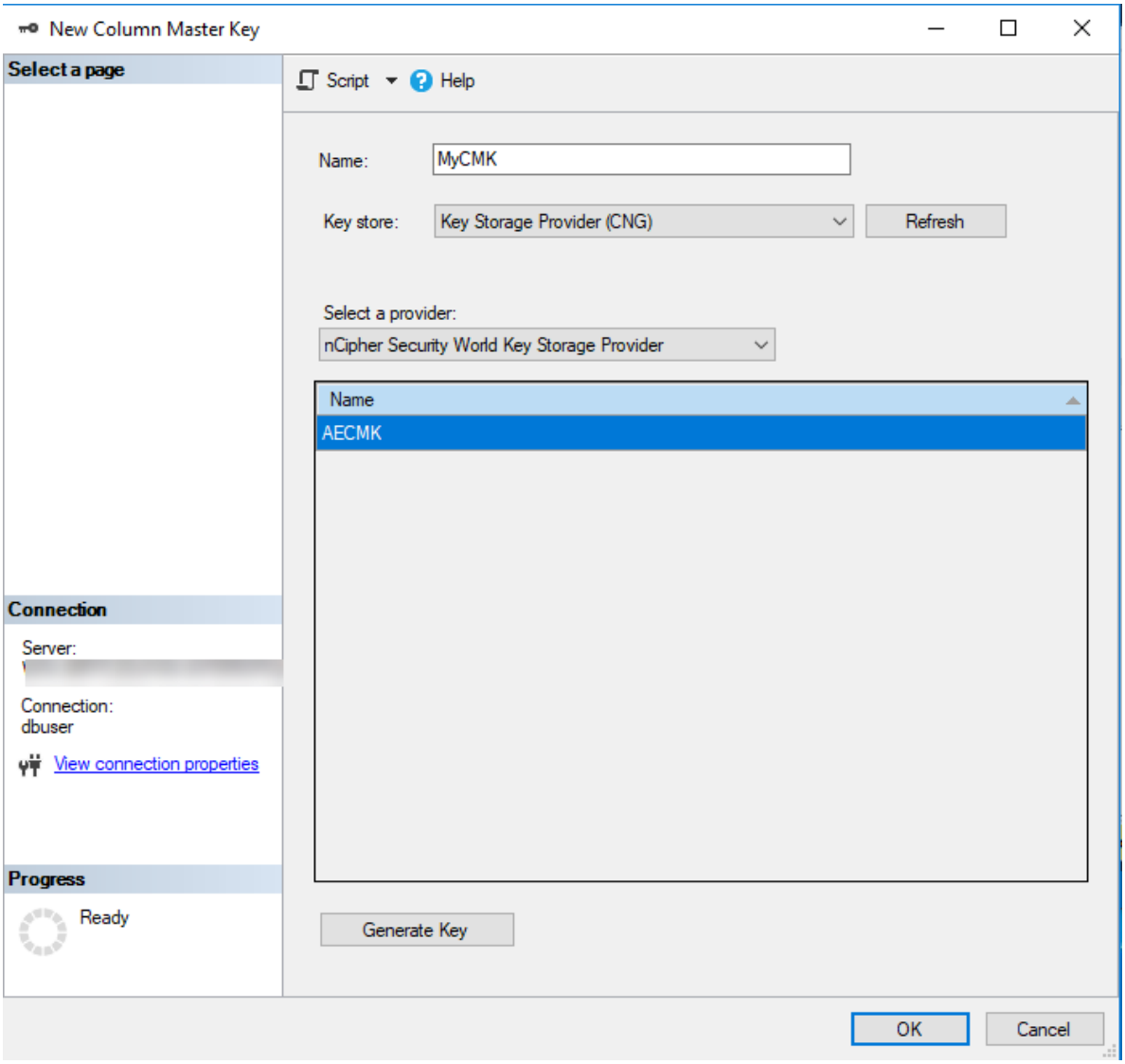
4. Enter the following information on the **Column Master Keys** pop-up window, then select **Next**

   a. Enter a name, for example `MyCMK`.

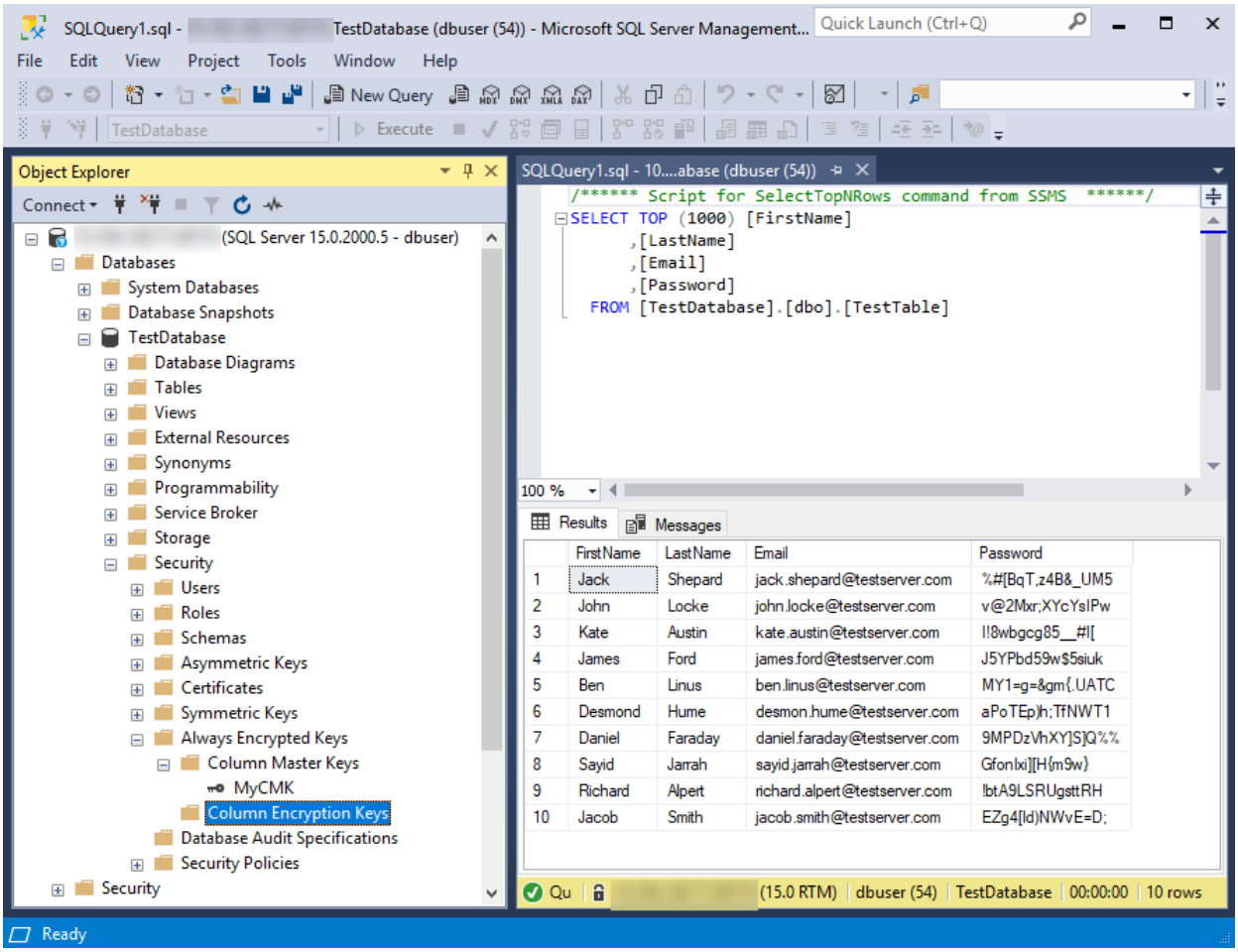
   b. Select **Key Storage Provider (CNG)** from the **Key store** drop-down list. This will then present the option to **Select a provider**.

   c. Select **nCipher Security World Key Storage Provider** from the drop-down list.

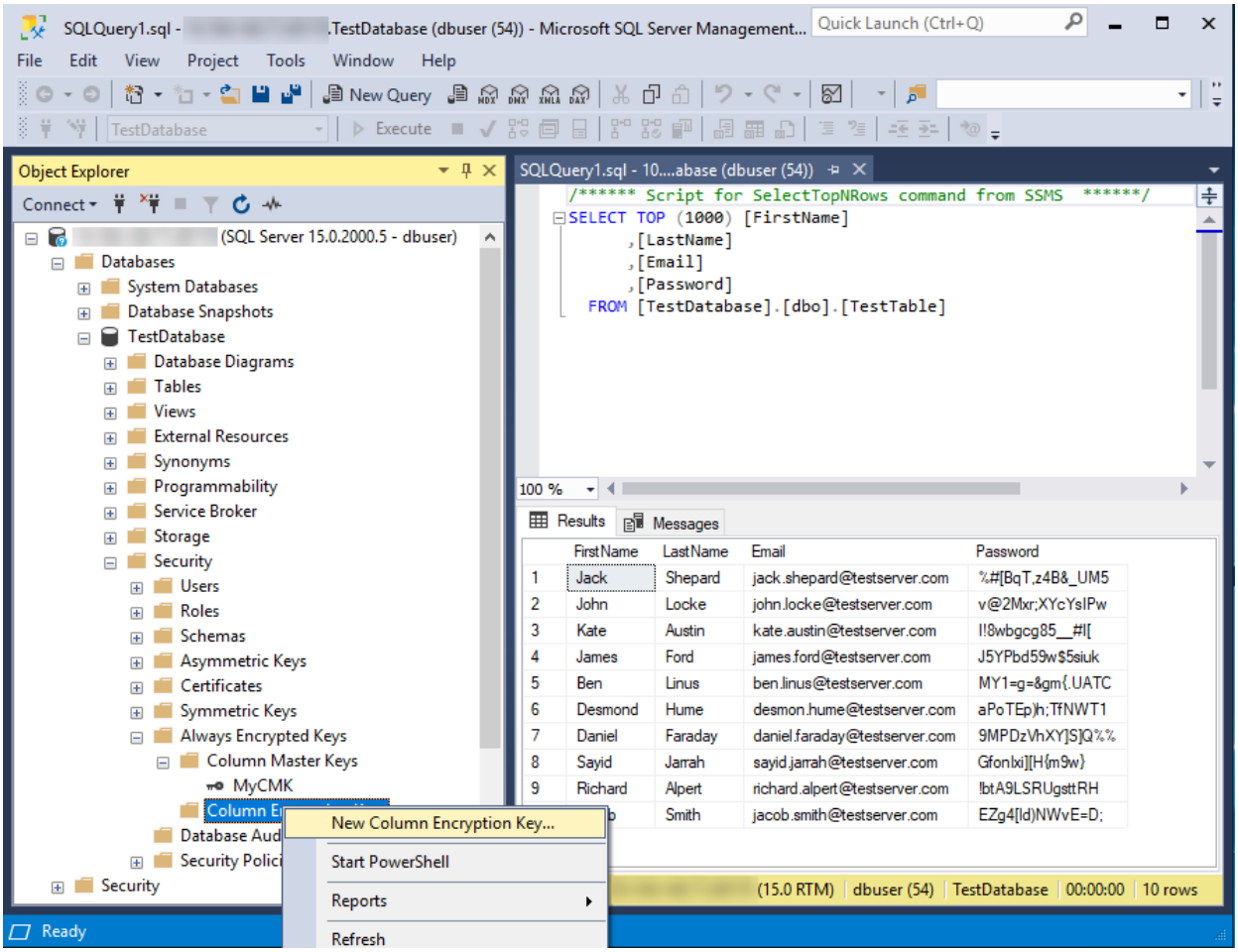
   d. The `AECMK` key created in an earlier step appears in **Name**. Select **OK** to create a new key using the nShield HSM and CNG KSP.



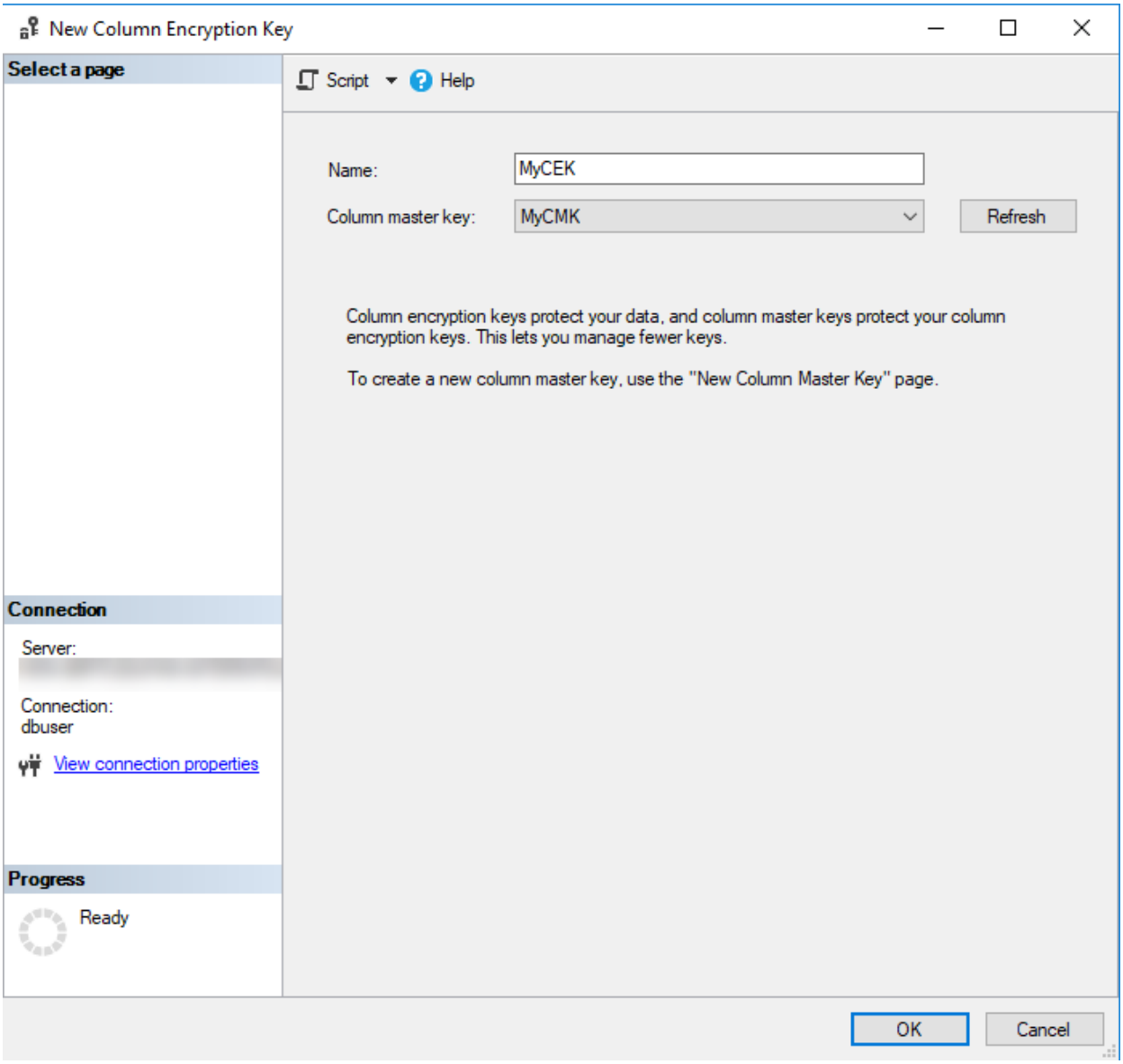
5. Notice the newly created **MyCMK** in the database **Security\Always Encrypted Keys\Column Master Keys**.



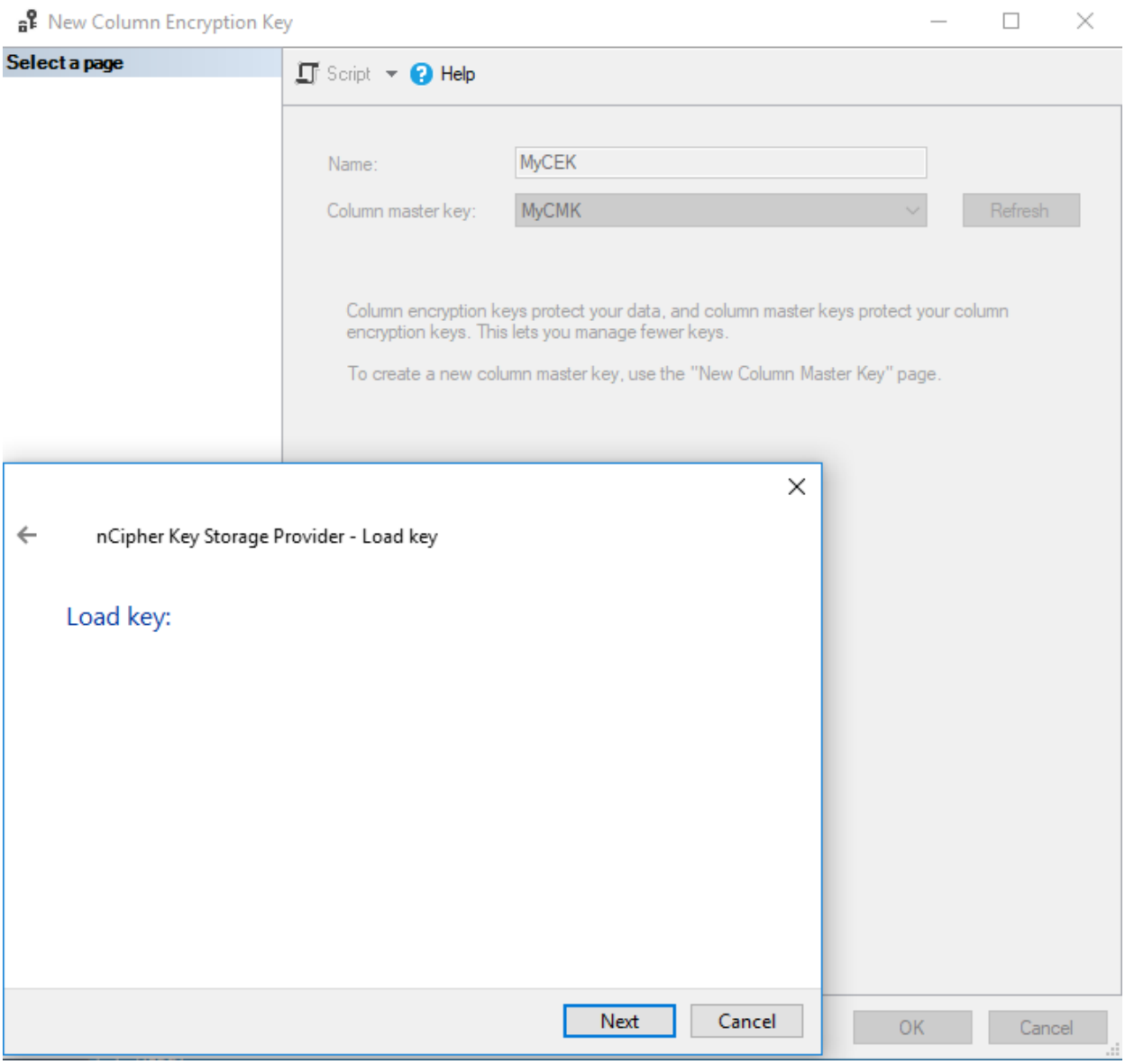
6. Using **Object Explorer**, select the **Security** directory under the desired Database. Select **Always Encrypted Keys** to expand it, then select **New Column MEncryption Key**.



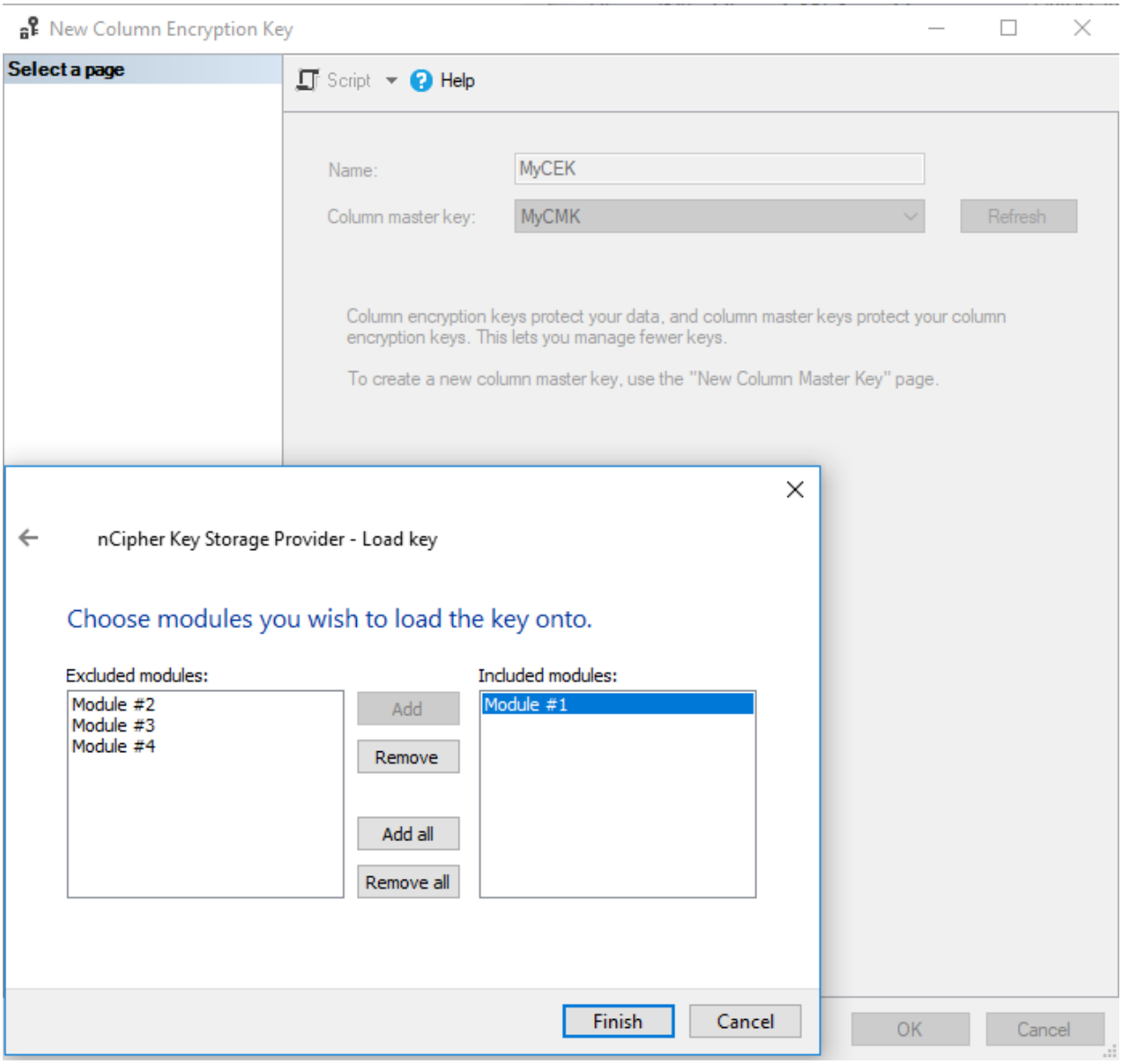
7. Enter **Name** and select **OK**.



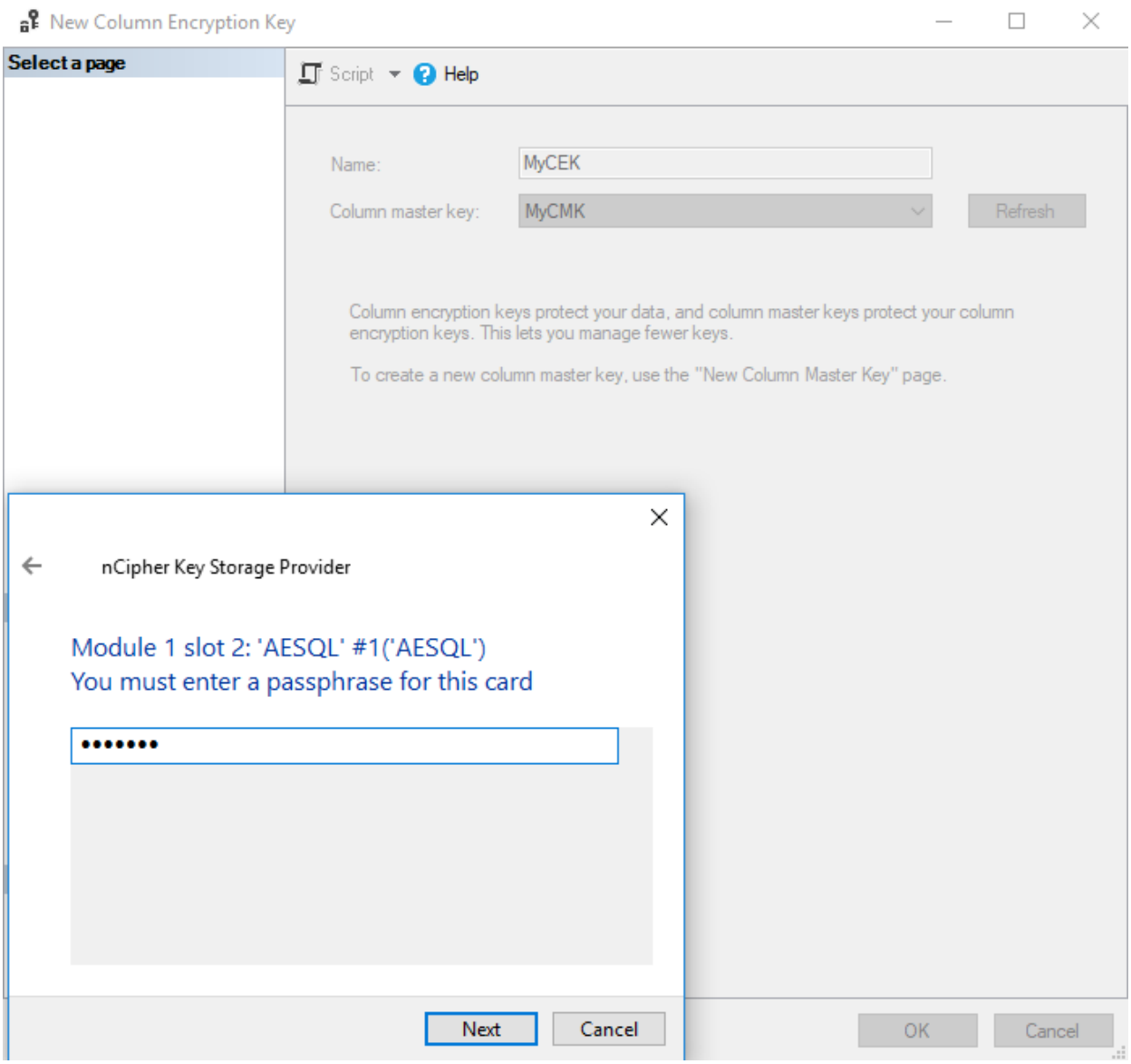
8. Present the OCS and select **Next**.



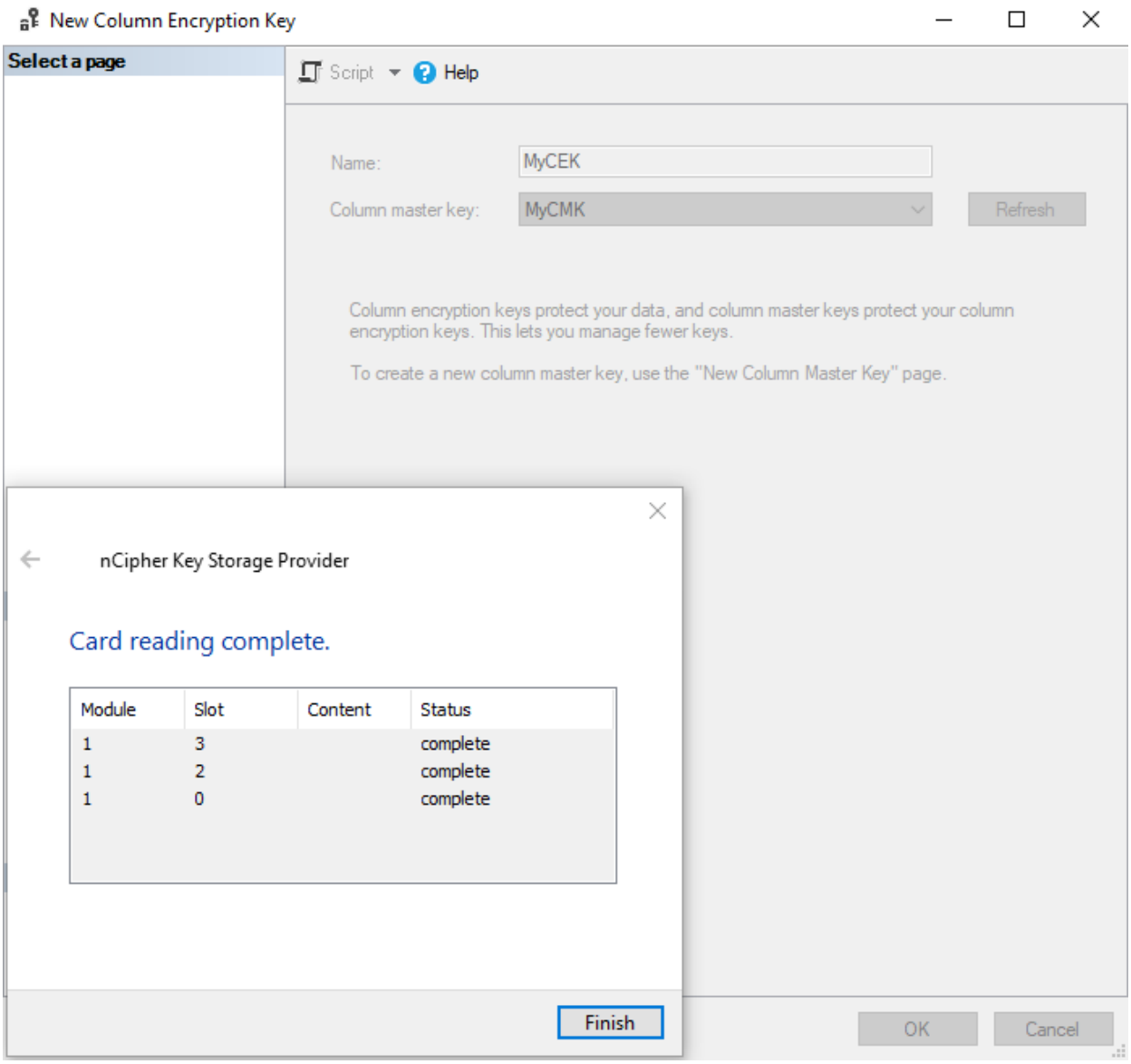
9. Select the HSM and select **Finish**.



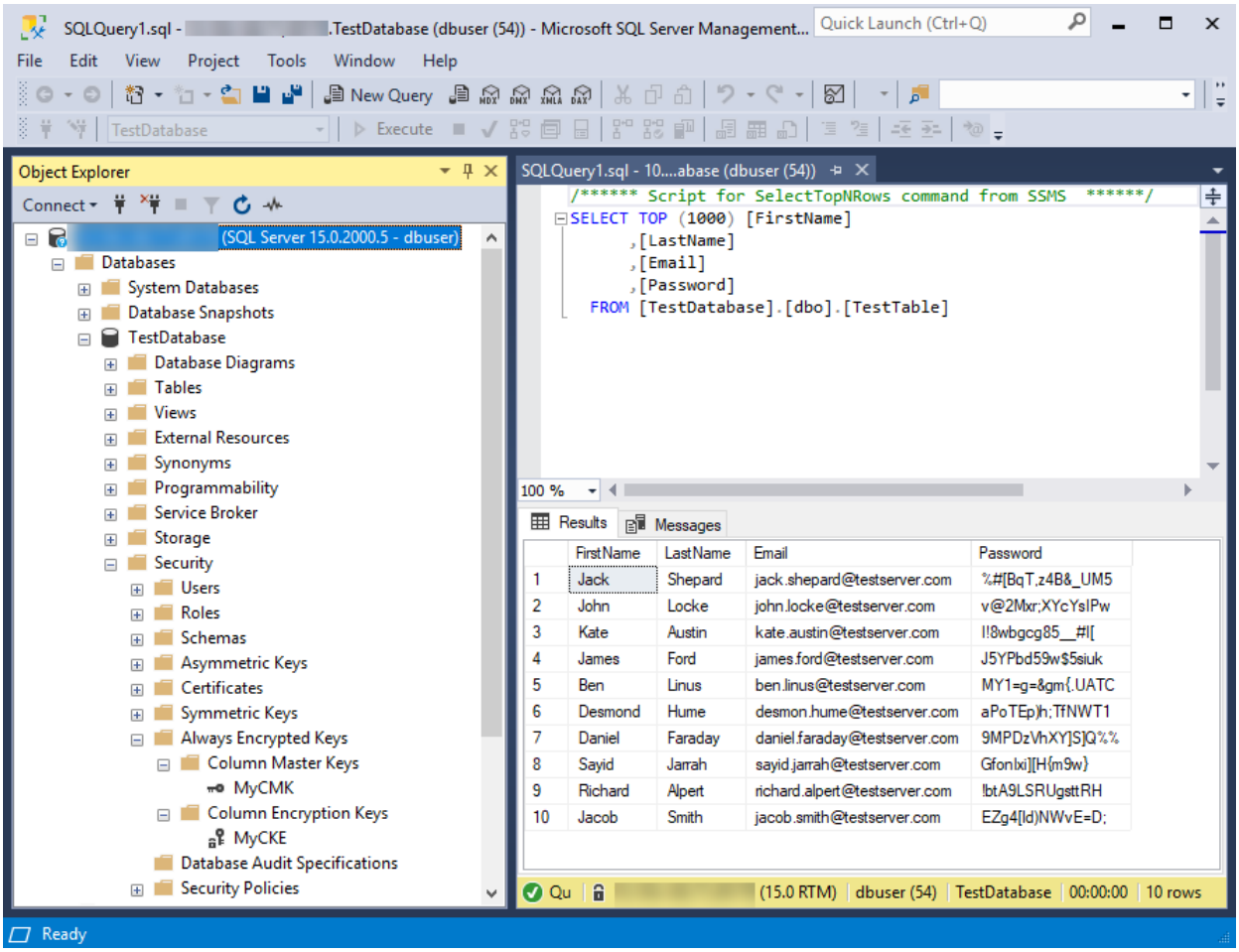
10. Enter the passphrase and select **Next**.



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



12. Notice the newly created **MyCEK** in the database **Security\Always Encrypted Keys\Column Encryption Keys**.



3.3. Generate My Column Master Key (MyCMK) and My Column Encryption Key (MyCEK) with PowerShell

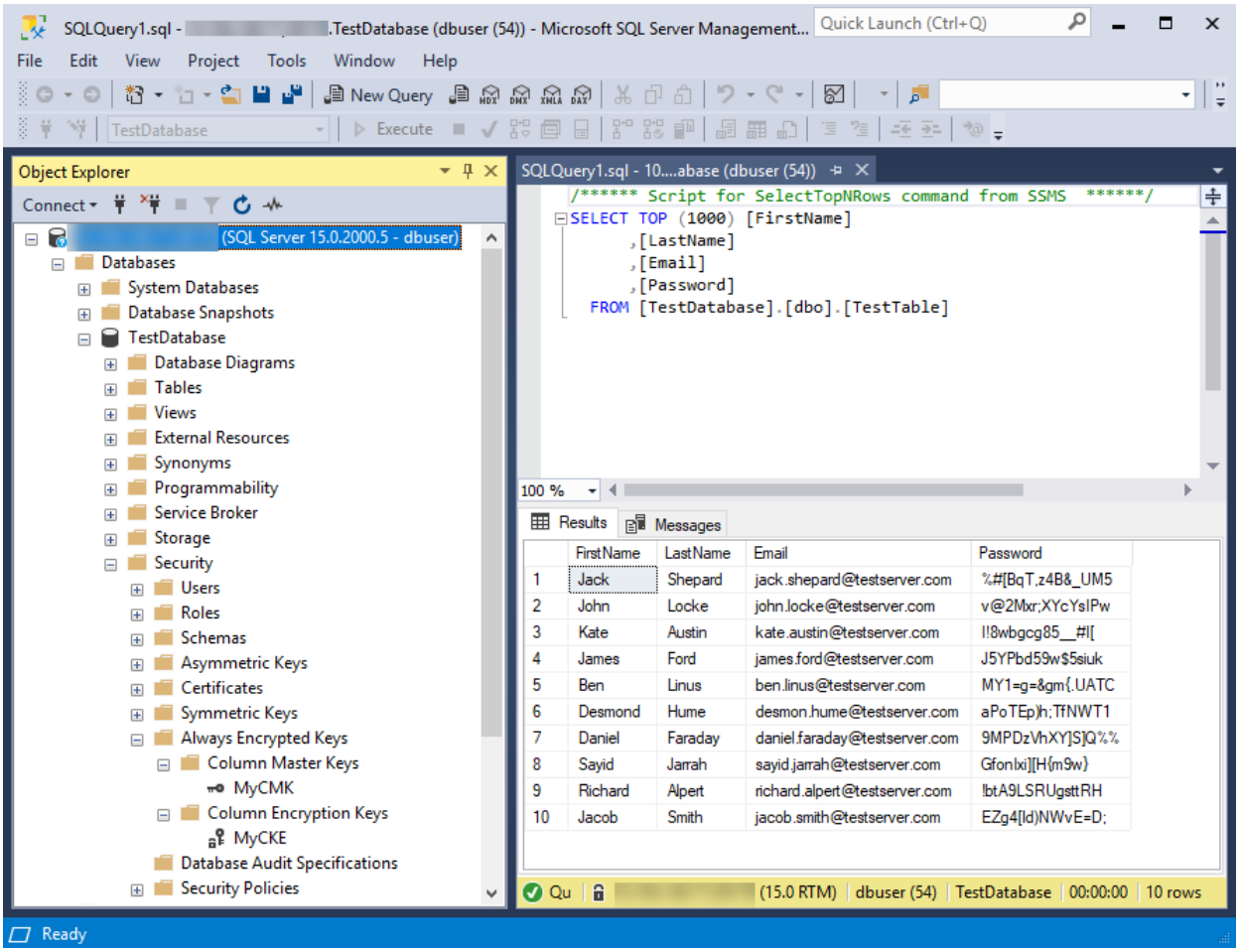
1. Delete MyCEK and MyCMK created above by right-clicking each key and selecting **Delete**.
2. Launch PowerShell on the on-premises client computer and run the `Generate_MyCMK_and_MyCEK.ps1` script.

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

# Connect to database.
$ConnectionString = "Data Source=<DB_Server_IP>,49170;Initial Catalog=TestDatabase;User
ID=dbuser;Password=<dbuser_Password>;MultipleActiveResultSets=False;Connect
Timeout=30;Encrypt=True;TrustServerCertificate=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 "MyCKE" -InputObject $Database -ColumnMasterKey "MyCMK"
```

The command line is

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

Name
-----
MyCMK
MyCEK
```

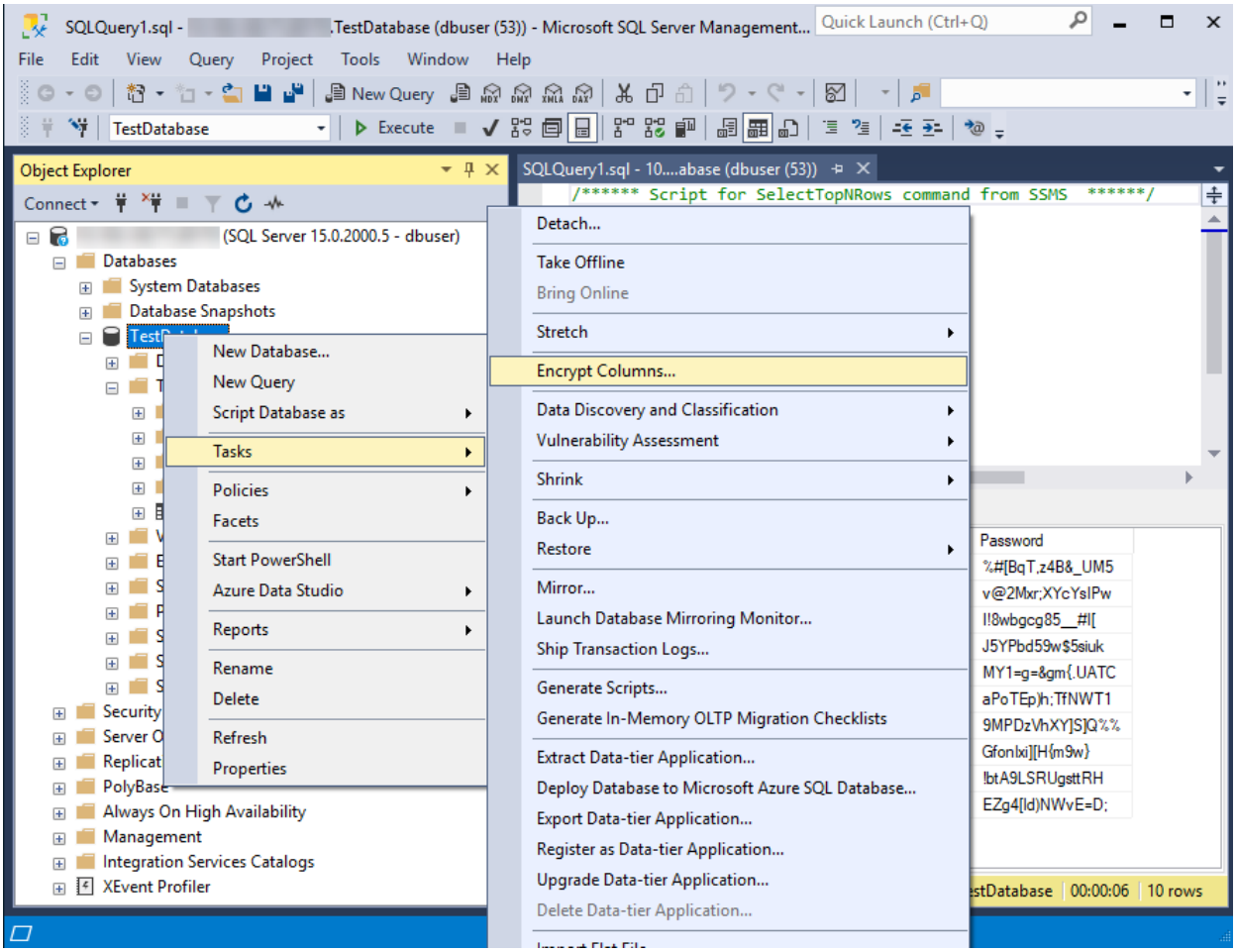
3. Present the OCS, select the HSM, and enter the passphrase.
4. Notice the newly created **MyCMK** in the database **Security\Always Encrypted Keys\Column Master Keys**.


SQLQuery1.sql - .TestDatabase (dbuser (54)) - Microsoft SQL Server Management...
Quick Launch (Ctrl+Q)
File Edit View Project Tools Window Help
New Query
TestDatabase
Execute
Object Explorer
Connect
(SQL Server 15.0.2000.5 - dbuser)
Databases
System Databases
Database Snapshots
TestDatabase
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Users
Roles
Schemas
Asymmetric Keys
Certificates
Symmetric Keys
Always Encrypted Keys
Column Master Keys
MyCMK
Column Encryption Keys
MyCKE
Database Audit Specifications
Security Policies
SQLQuery1.sql - 10....abase (dbuser (54))
/****** Script for SelectTopNRows command from SSMS ******/
SELECT TOP (1000) [FirstName]
,[LastName]
,[Email]
,[Password]
FROM [TestDatabase].[dbo].[TestTable]
100 %
Results
Messages
FirstName LastName Email Password
1 Jack Shepard jack.shepard@testserver.com %#[BqT,z4B&_UM5
2 John Locke john.locke@testserver.com v@2Mxr;XYcYsIPw
3 Kate Austin kate.austin@testserver.com I!8wbgcg85__#I[
4 James Ford james.ford@testserver.com J5YPbd59w$5siuk
5 Ben Linus ben.linus@testserver.com MY1=g=&gm{.UATC
6 Desmond Hume desmon.hume@testserver.com aPoTEp)h;TfNWT1
7 Daniel Faraday daniel.faraday@testserver.com 9MPDzVhXY]S]Q%%
8 Sayid Jarrah sayid.jarrah@testserver.com Gfonlxi][H{m9w}
9 Richard Alpert richard.alpert@testserver.com !btA9LSRUgsttRH
10 Jacob Smith jacob.smith@testserver.com EZg4[Id)NWvE=D;
Qu (15.0 RTM) dbuser (54) TestDatabase 00:00:00 10 rows
Ready

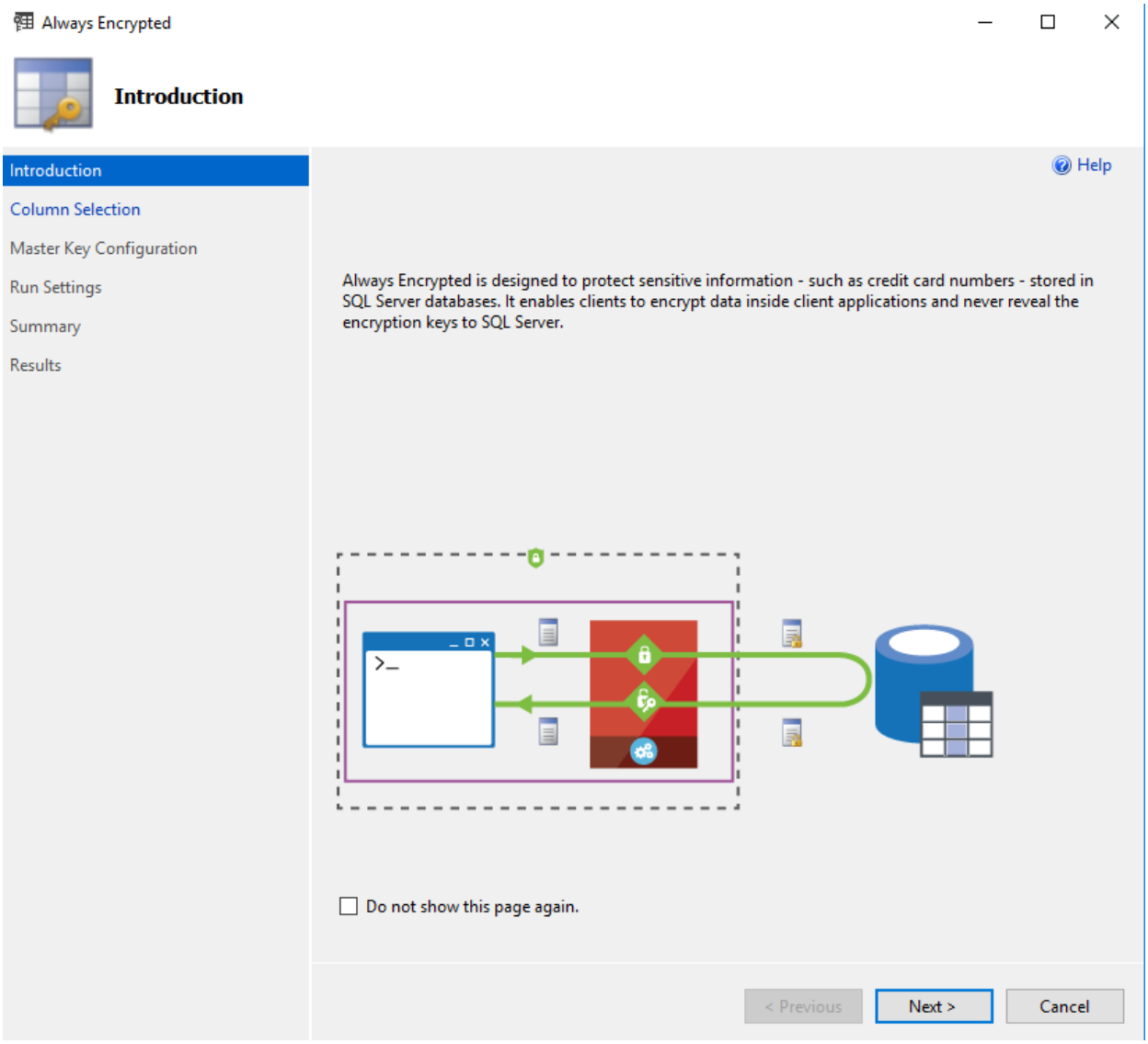
4. Encrypt or decrypt a column with SSMS

4.1. Encrypt a column

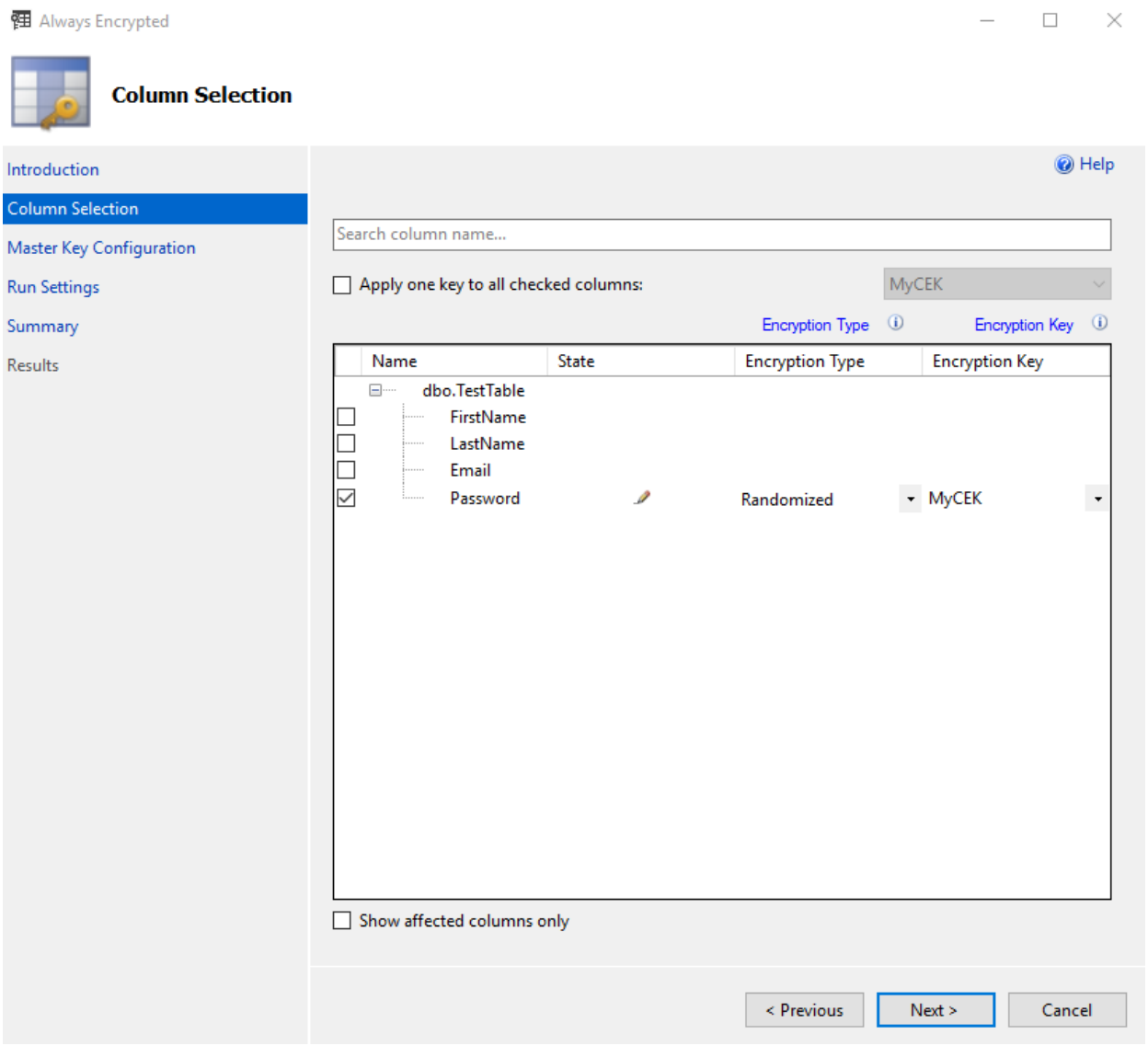
1. Launch **Microsoft SQL Server Management Studio** on the on-premises client. Connect with the dbuser account to the database on the SQL server.
2. Right-click the database, **TestDatabase**, and select **Tasks** > **Encrypt Columns**.



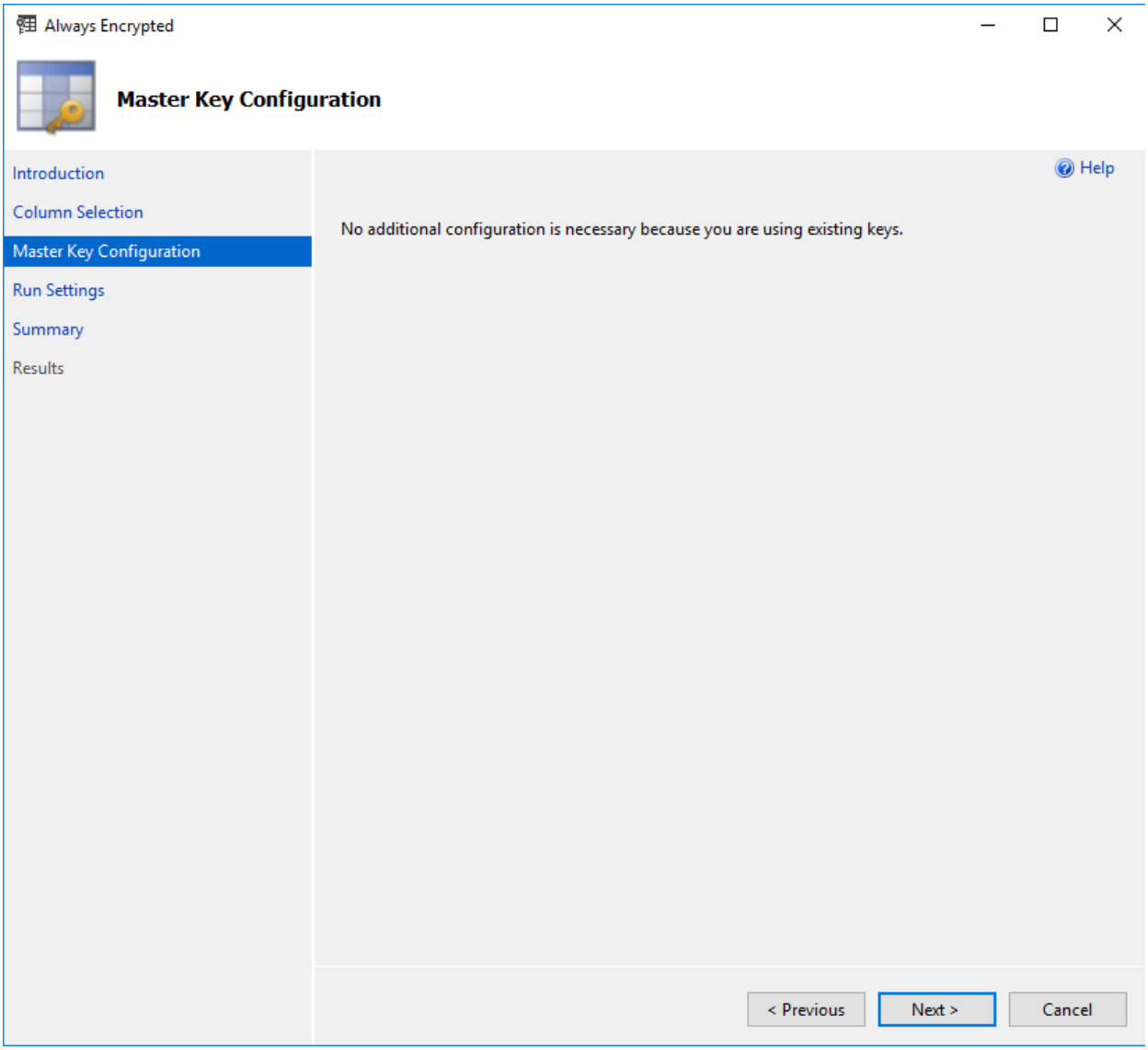
3. Select **Next** on the Introduction screen.



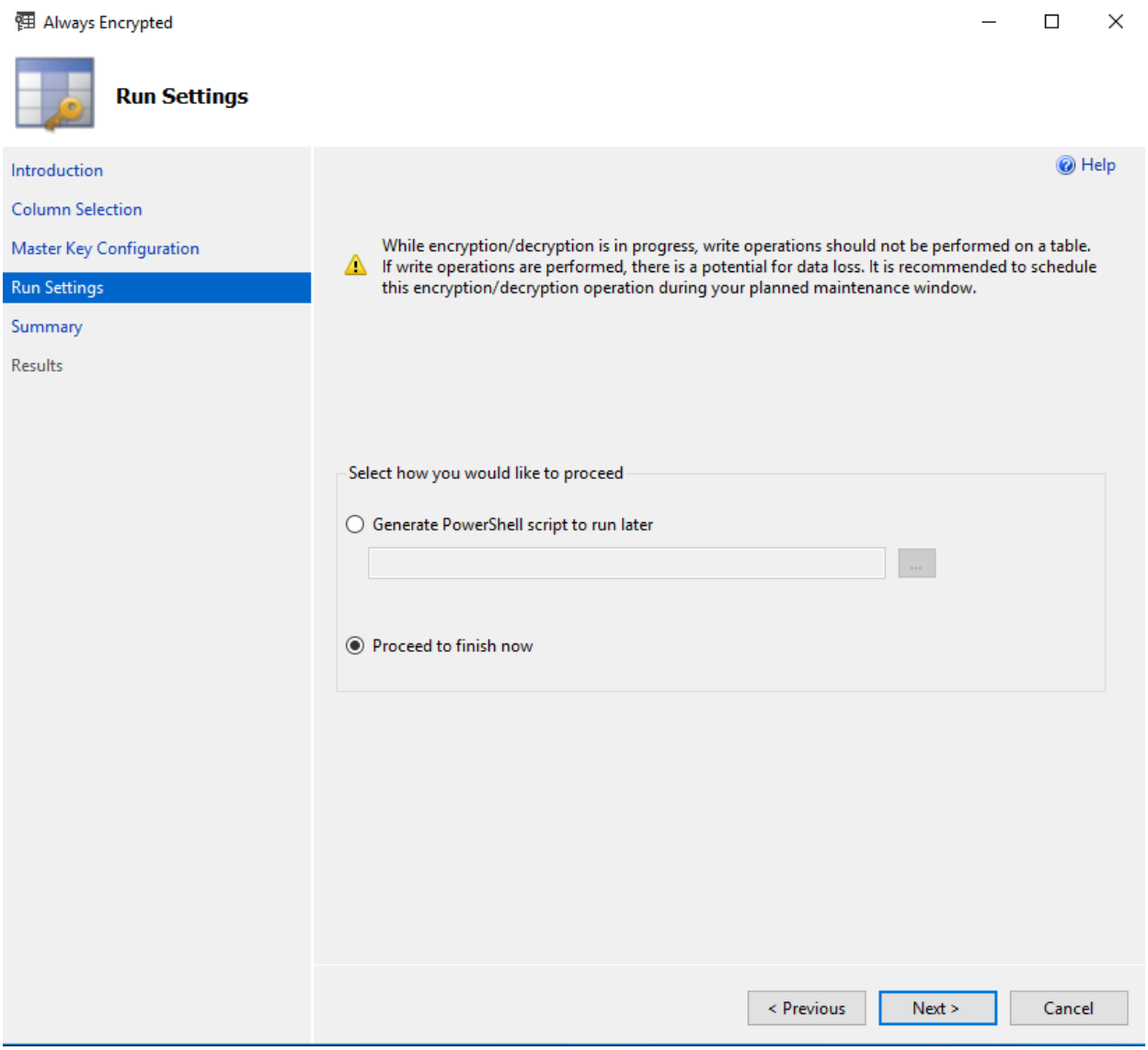
4. Select the column and encryption type on the **Column Selection** screen and select **Next**.



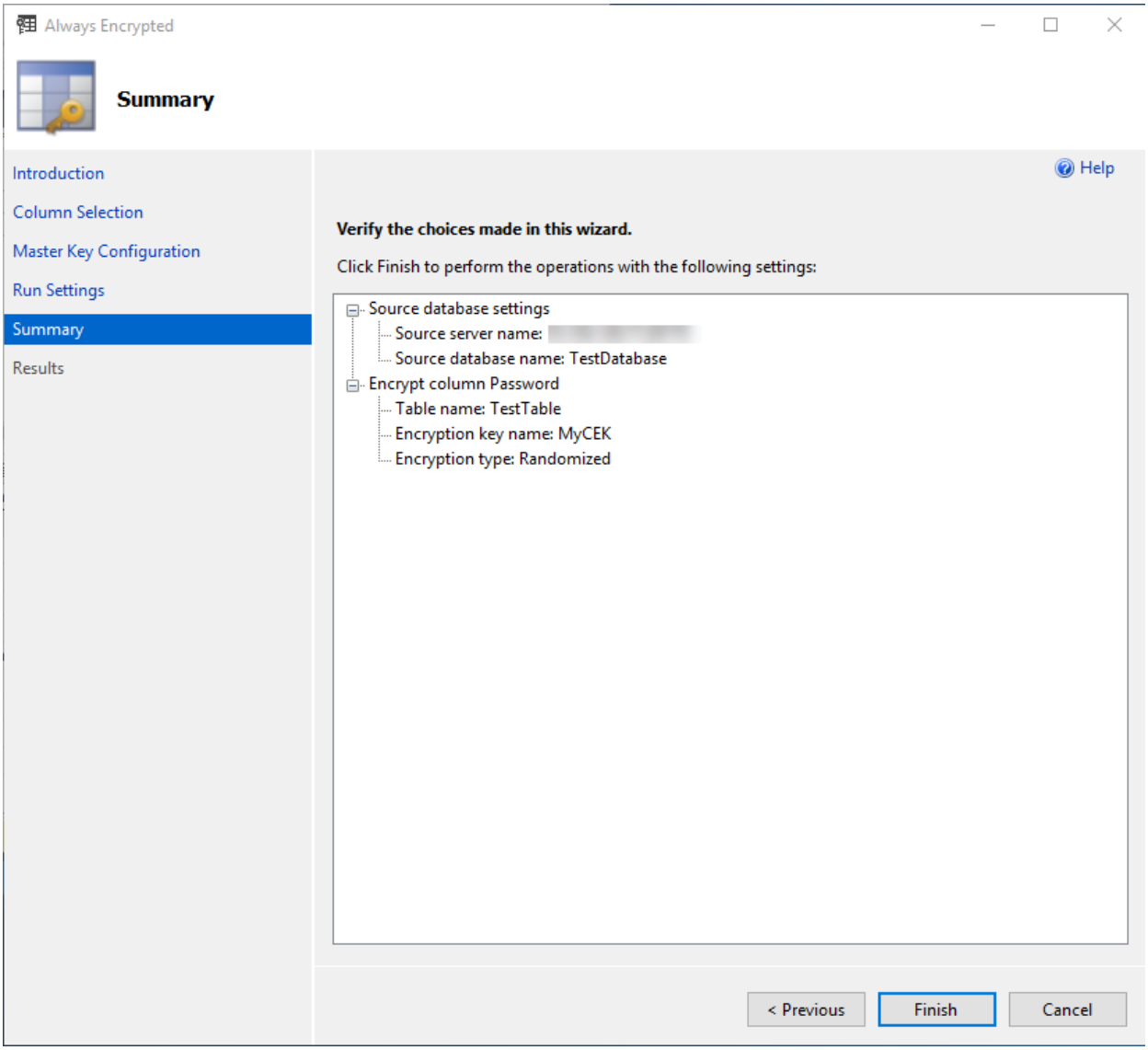
5. Select **MyCMK** on the **Master key Configuration** window. Select **Next**.



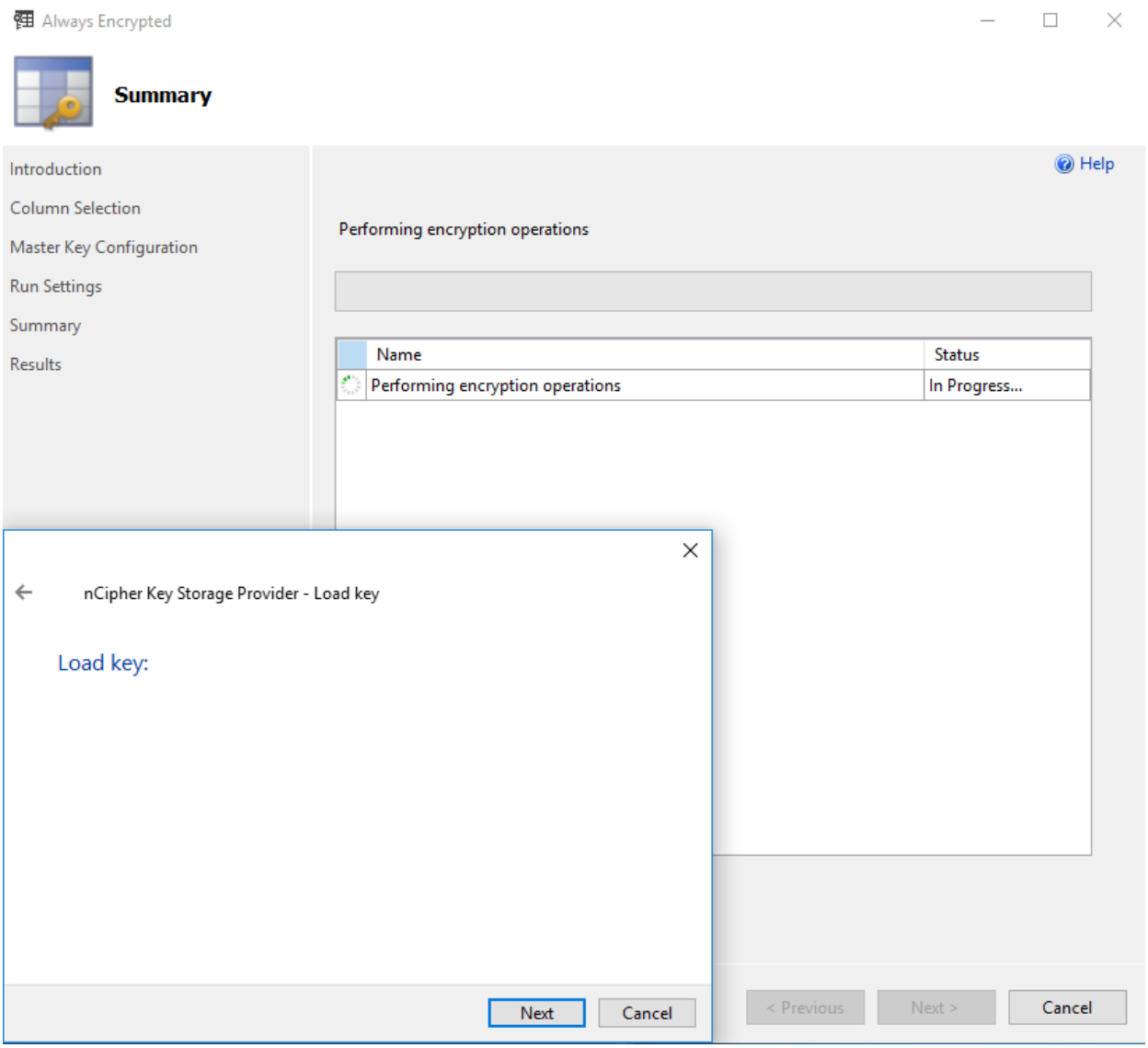
6. Select **Proceed to finish now** radio button and select **Next**.



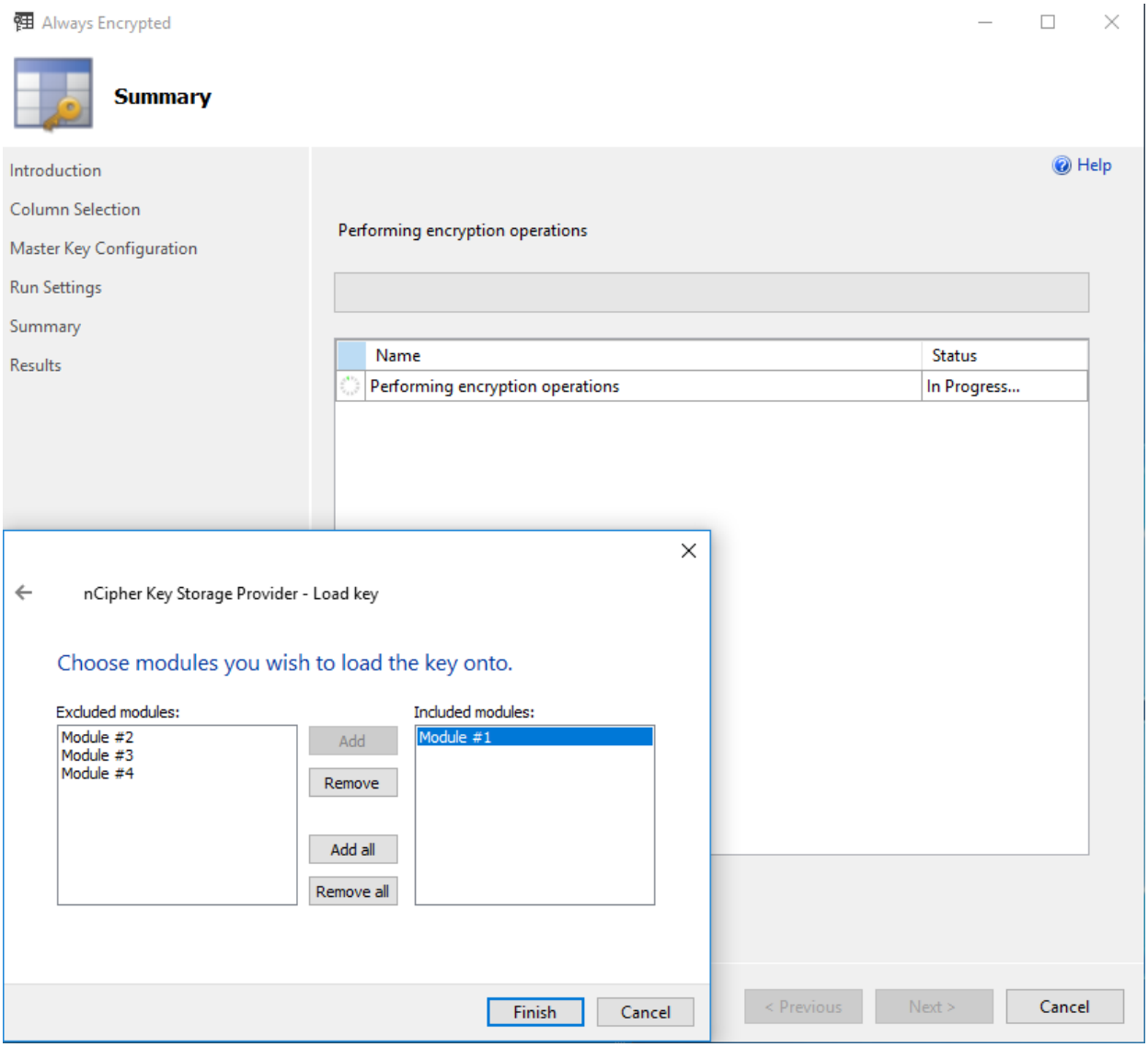
7. Verify the configuration choices on the **Summary** screen. Select **Next**.



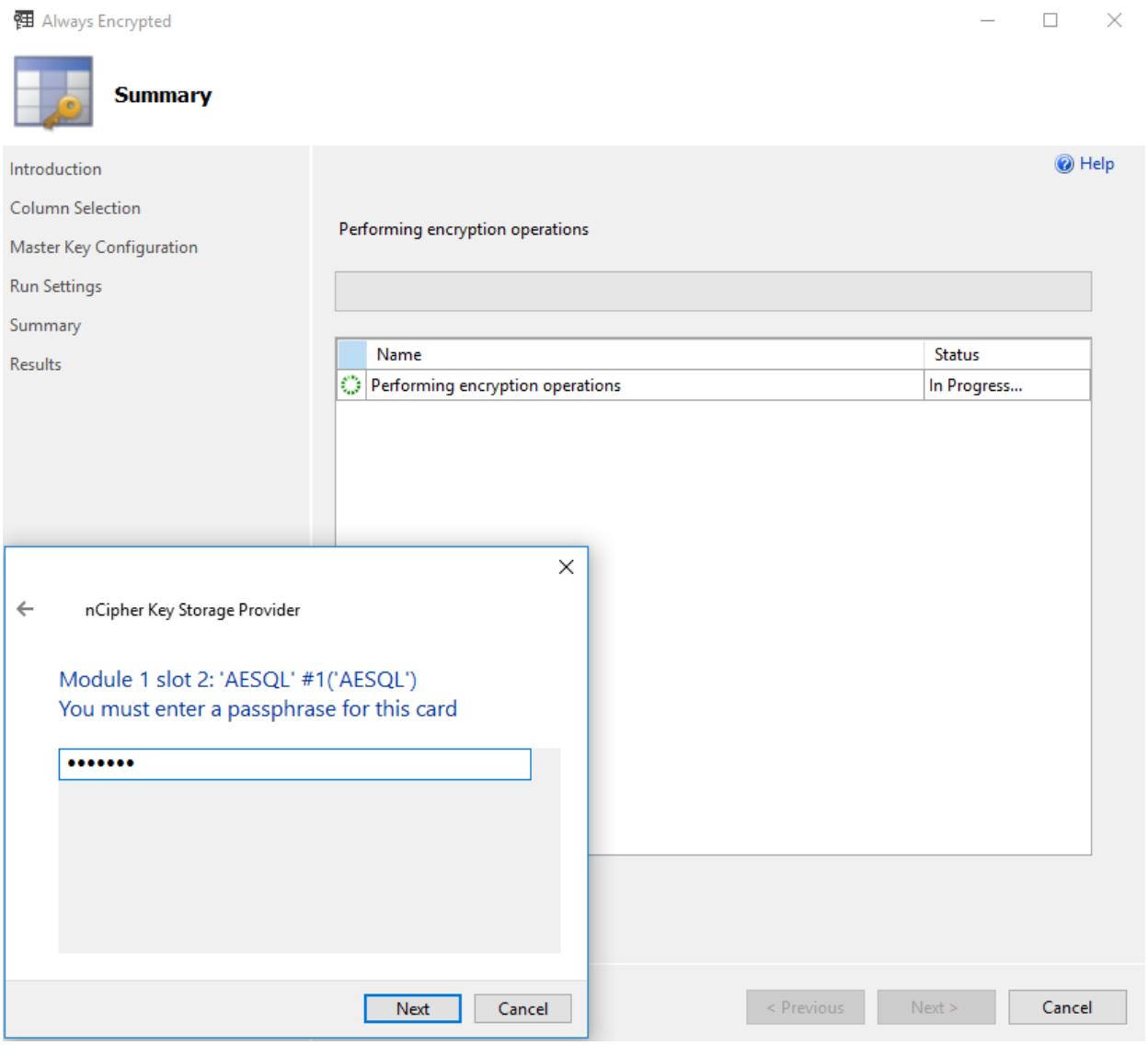
8. Present the OCS that is protecting the CMK and select **Finish**.



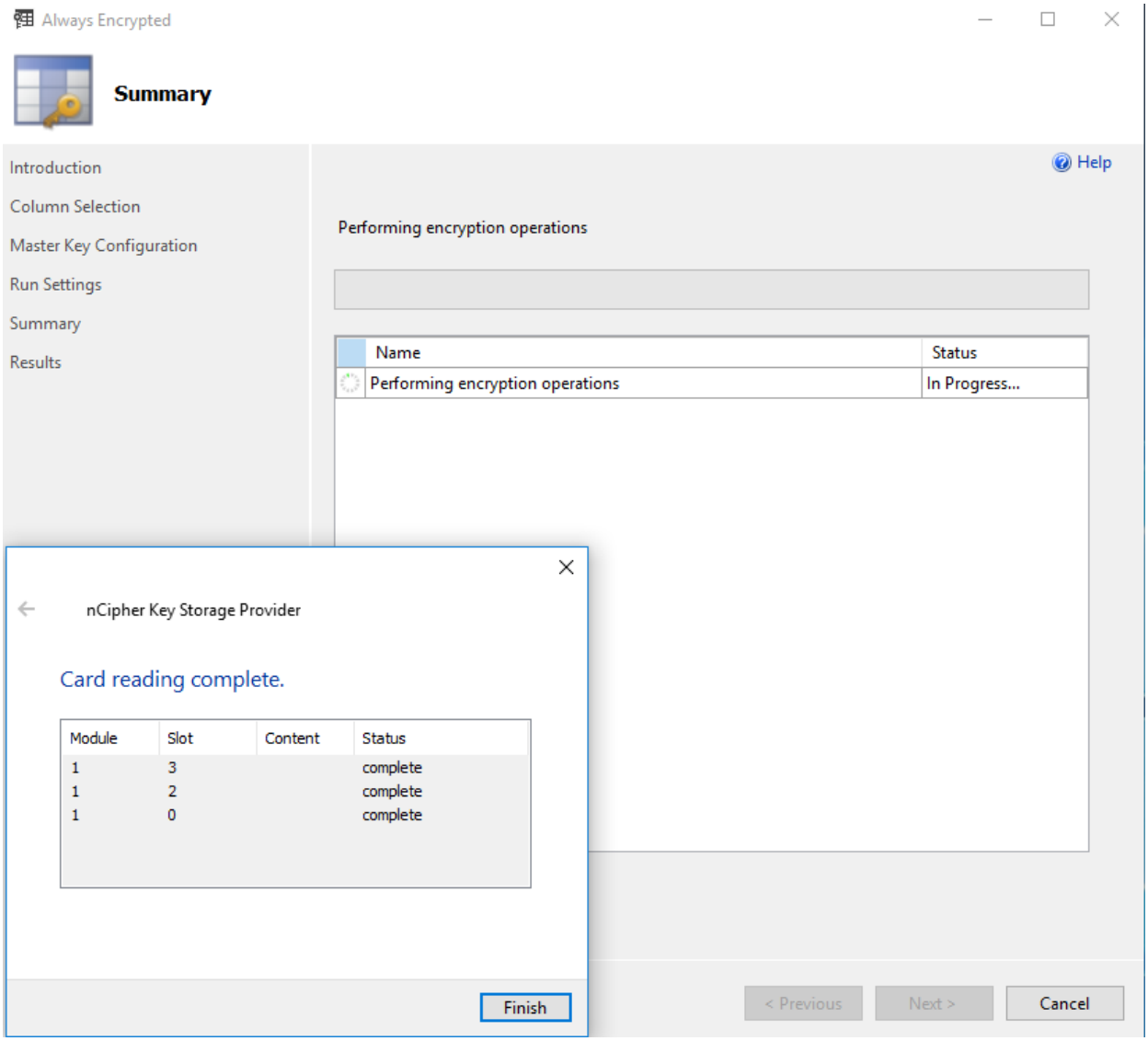
9. Select the HSM and select **Next**.



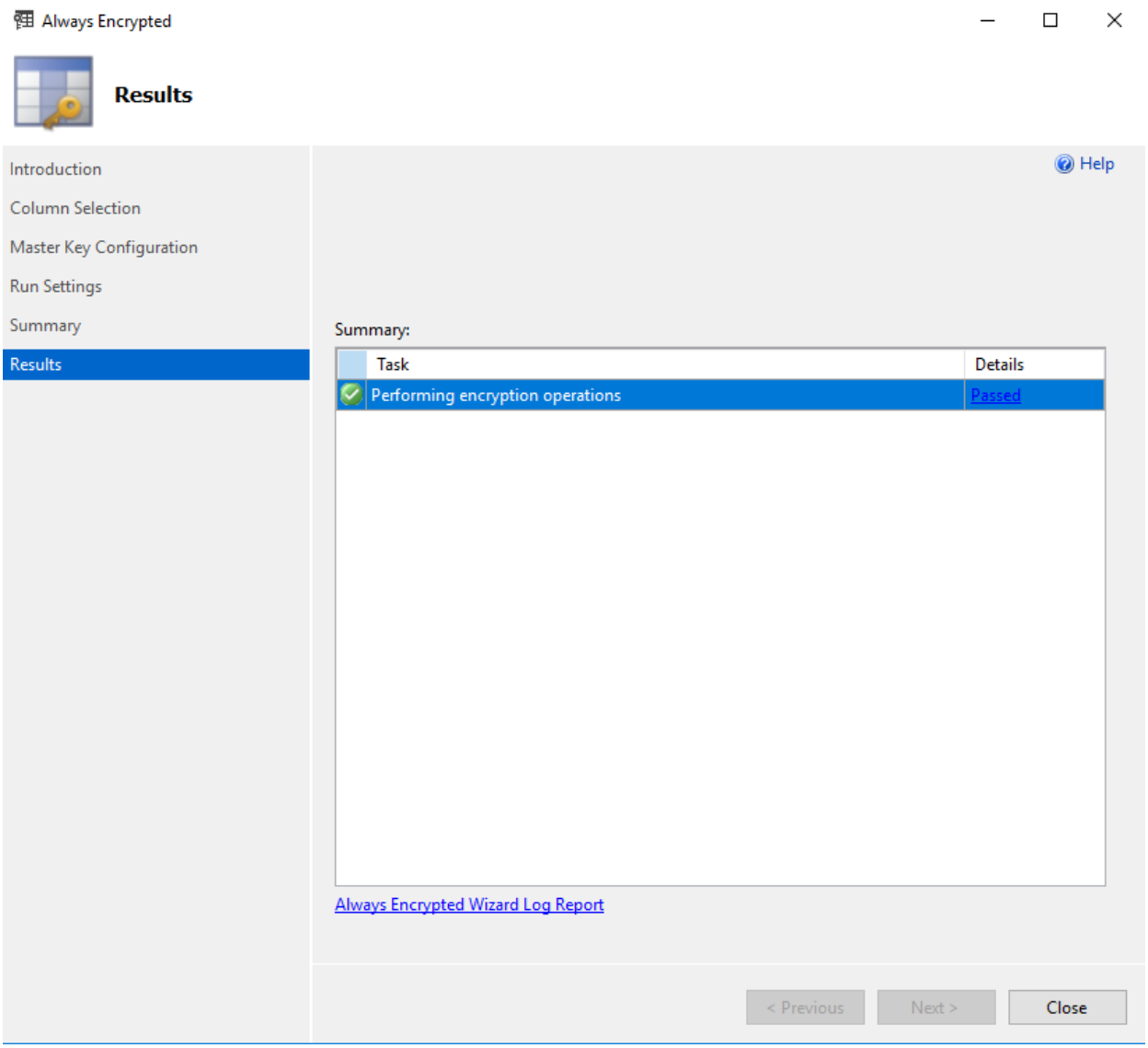
10. Enter the passphrase, and select **Next**.



11. Select **Finish**.



12. Select **Close**.

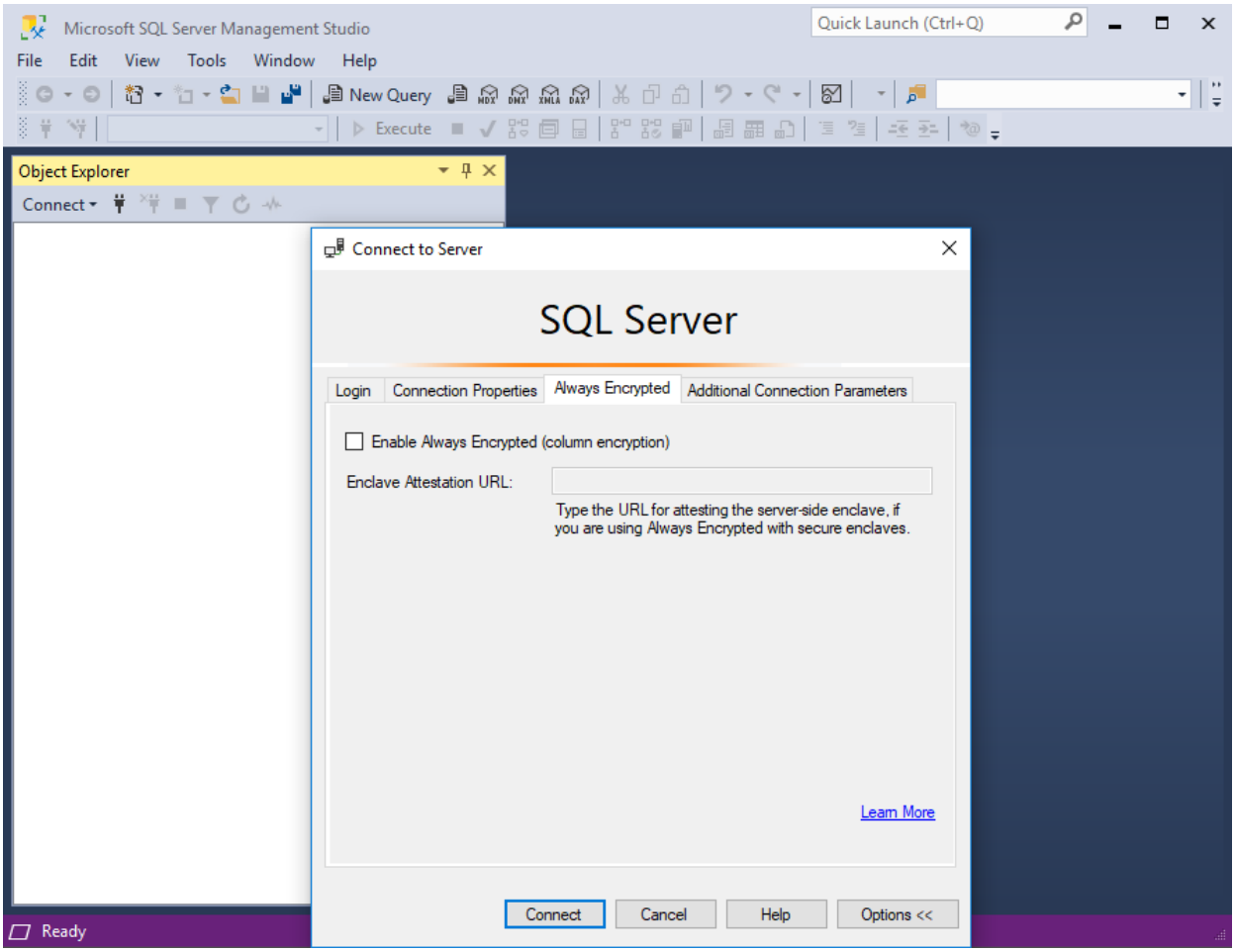


The column has been encrypted in the SQL server, but it shows as clear text on the **Microsoft SQL Server Management Studio** GUI on the on-premises client. This is because **Always Encrypted** is performing the decryption at the on-premises client site.

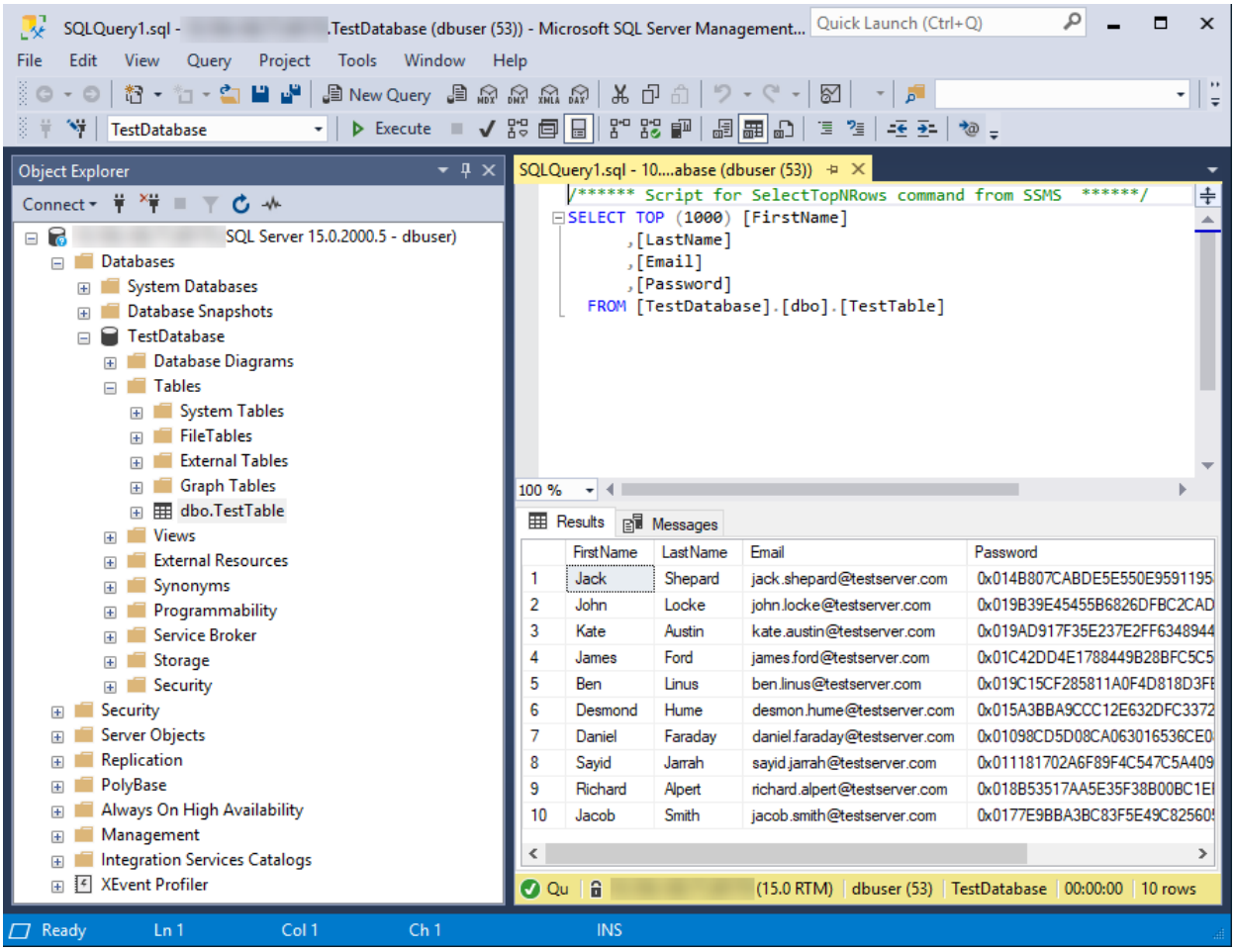
4.2. View an encrypted column

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

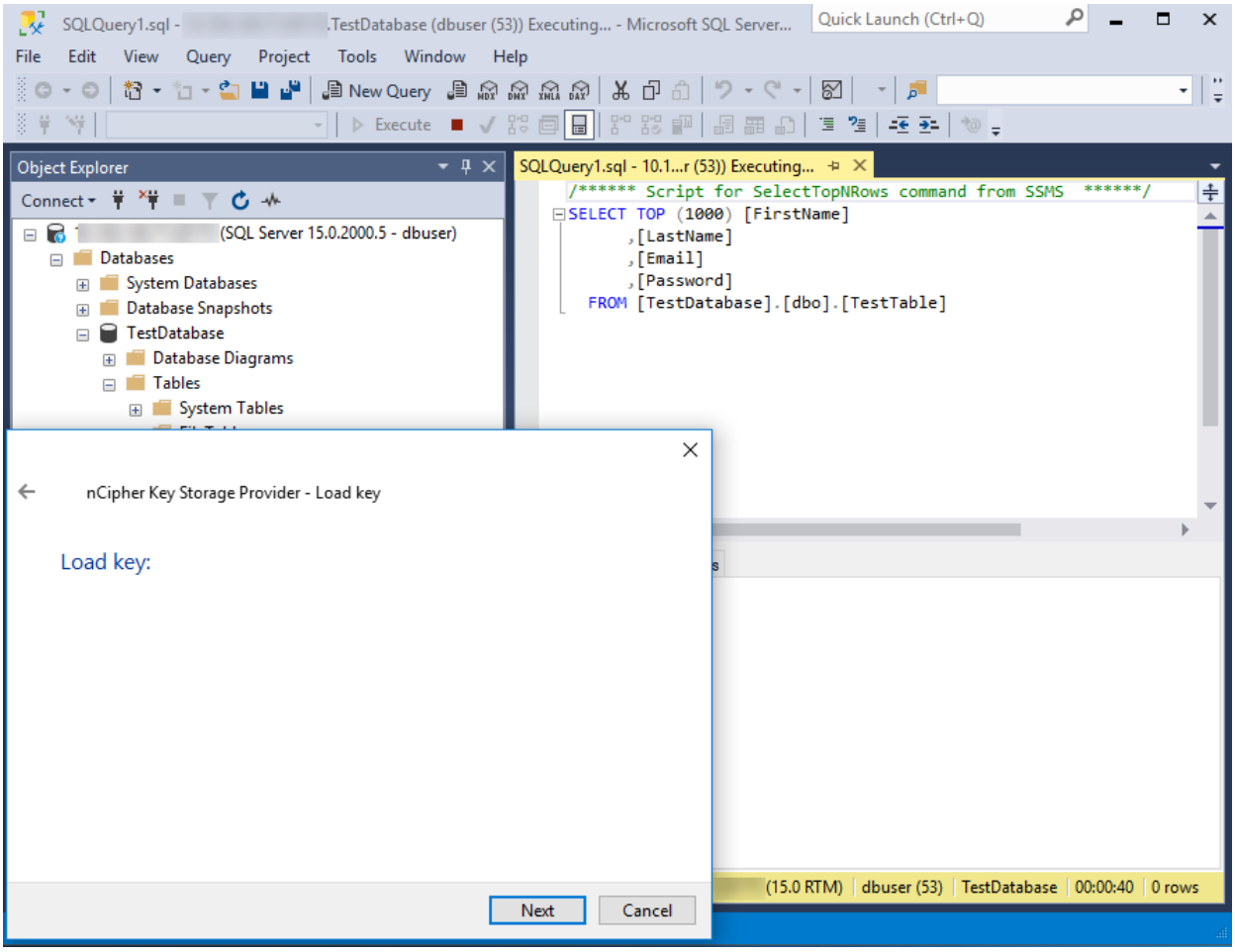
1. Connect to the SQL server from the on-premises client, but with the **Enable Always Encrypted** unchecked.



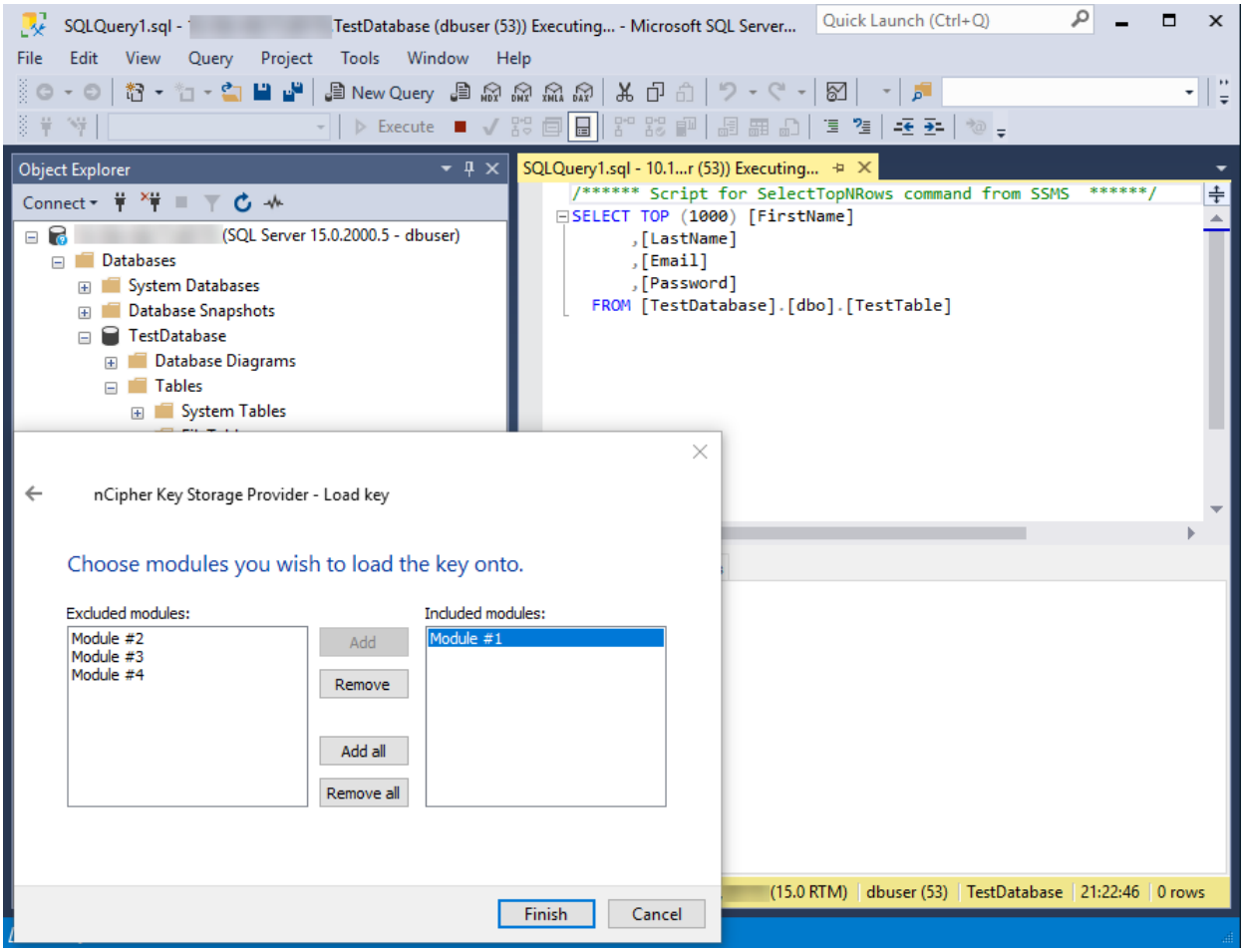
2. Right-click **dbo.Table**, and select **Select Top 1000 Rows**. The column that was chosen for encryption now appears as ciphertext, that is, as an encrypted value.



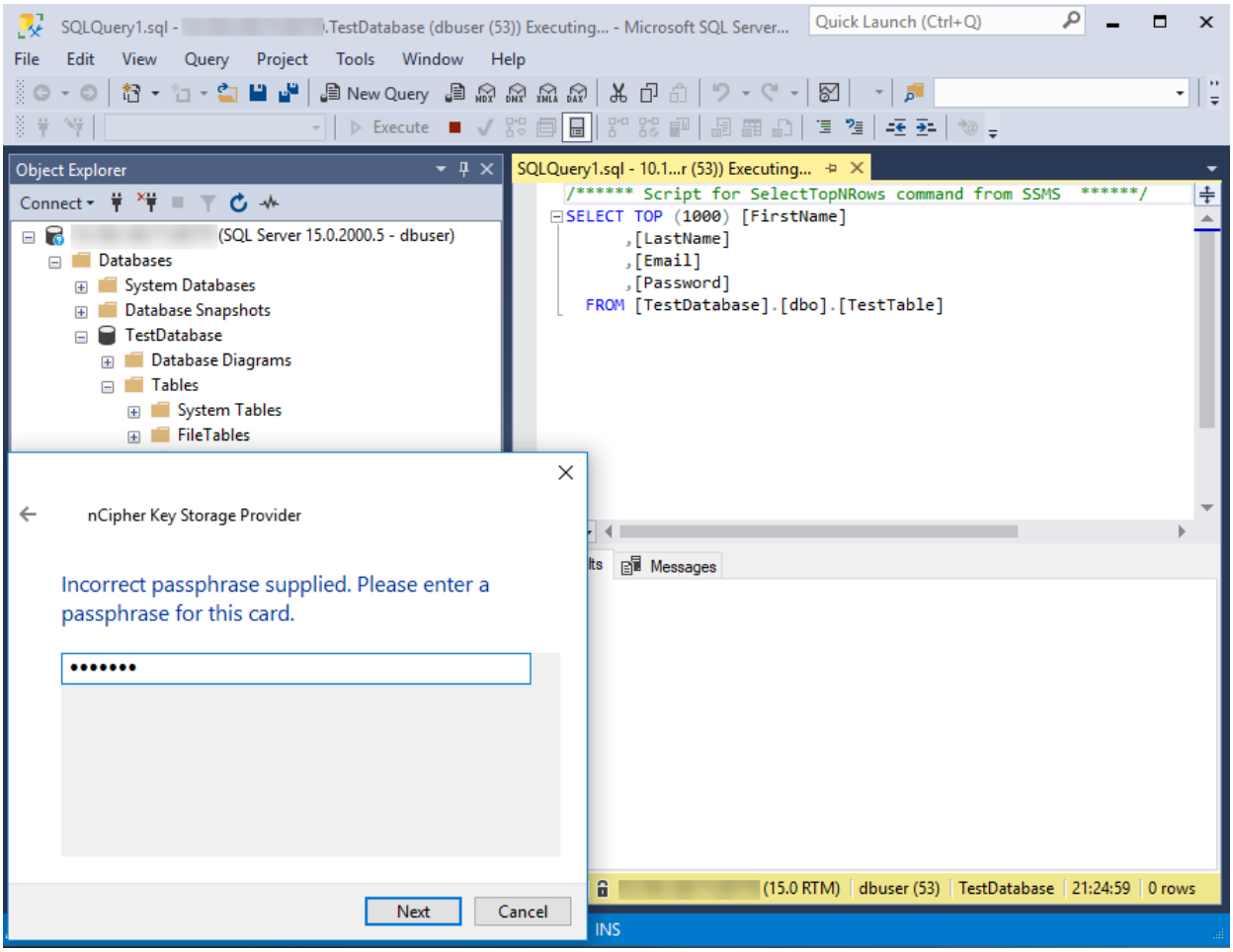
3. Reconnect to the SQL server from the on-premises client, but with the **Enable Always Encrypted** checked. Be prepared to provide the OCS. Select **Next**.



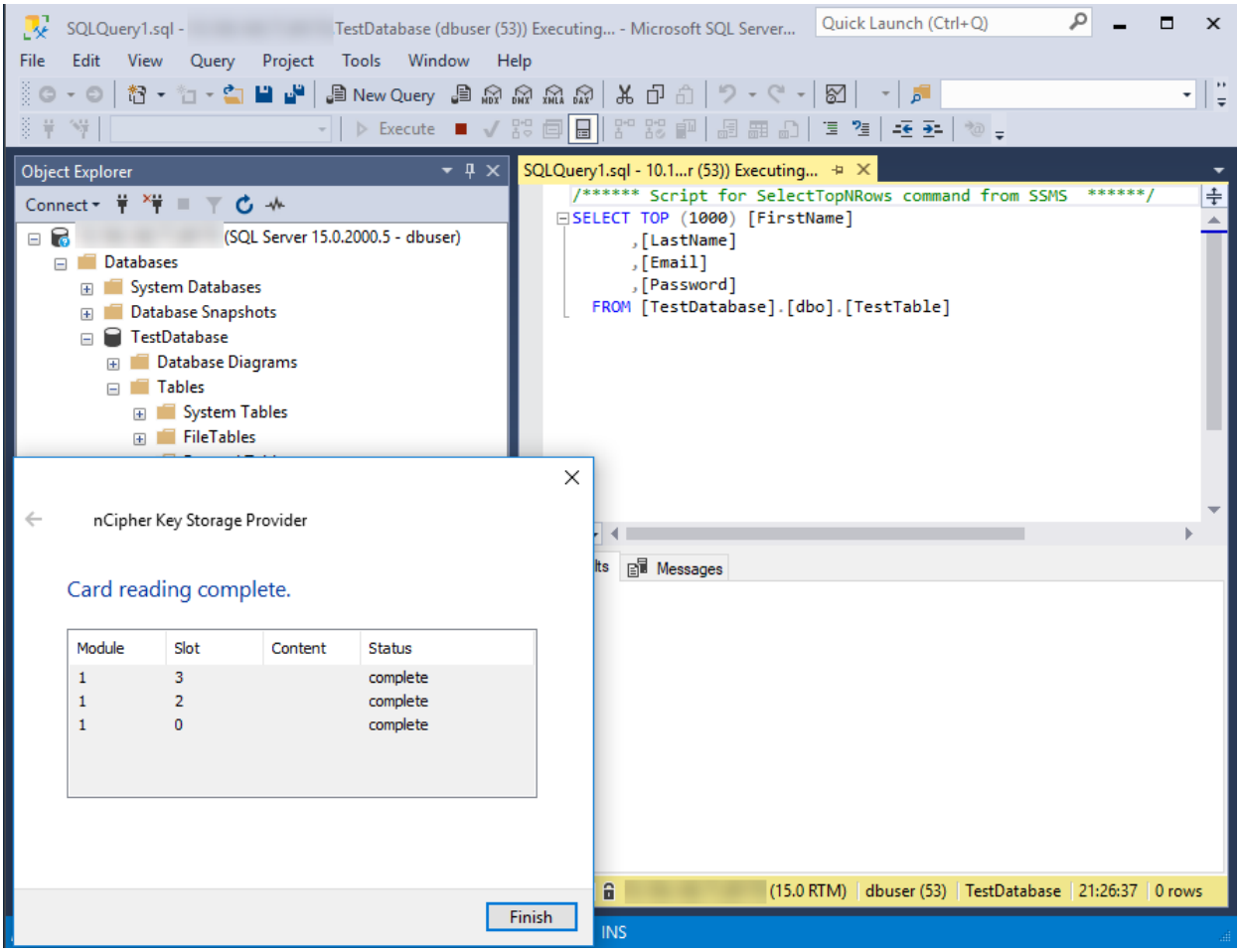
4. Select the HSM. Select **Finish**.



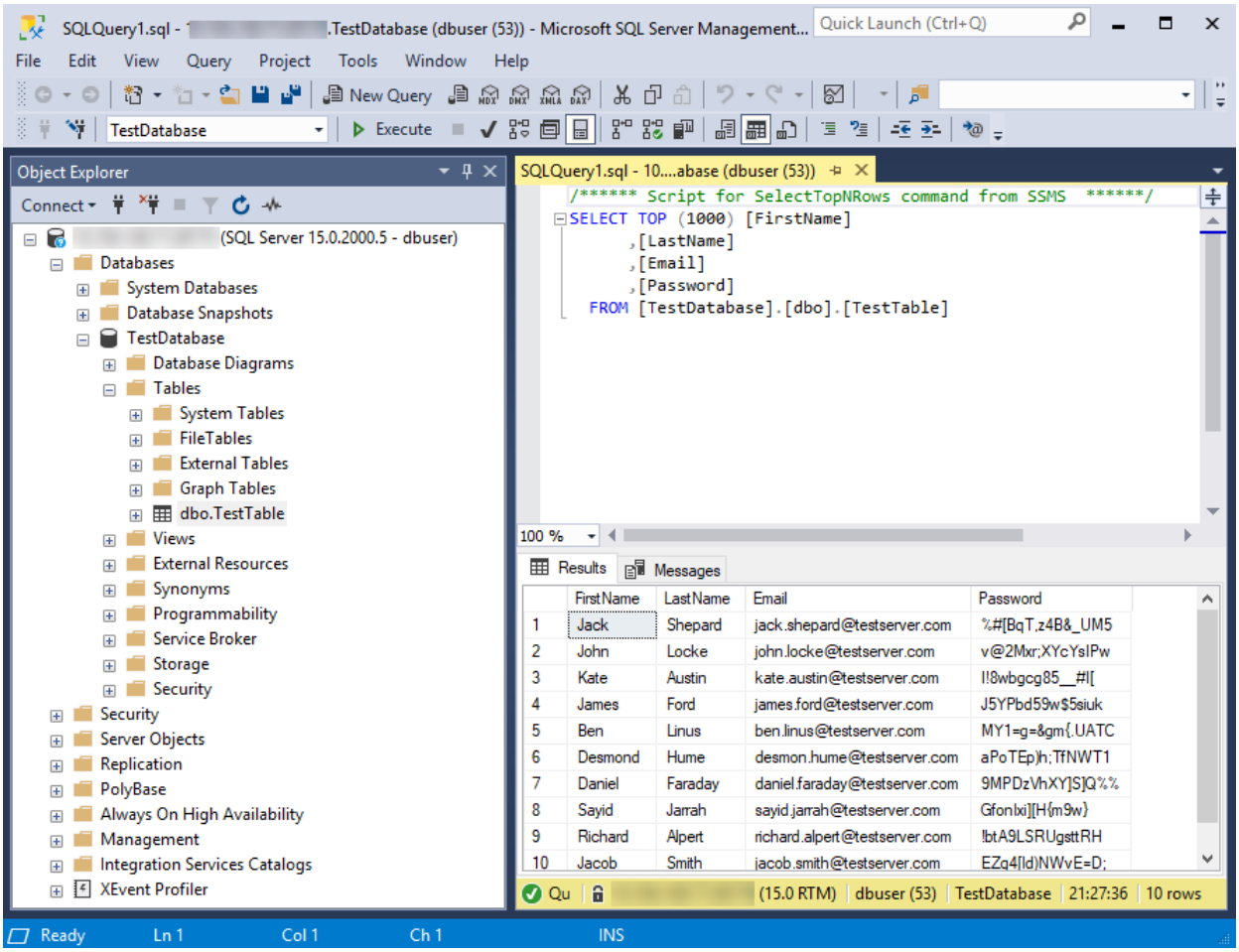
5. Enter the passphrase. Select **Next**.



6. Select **Finish**.

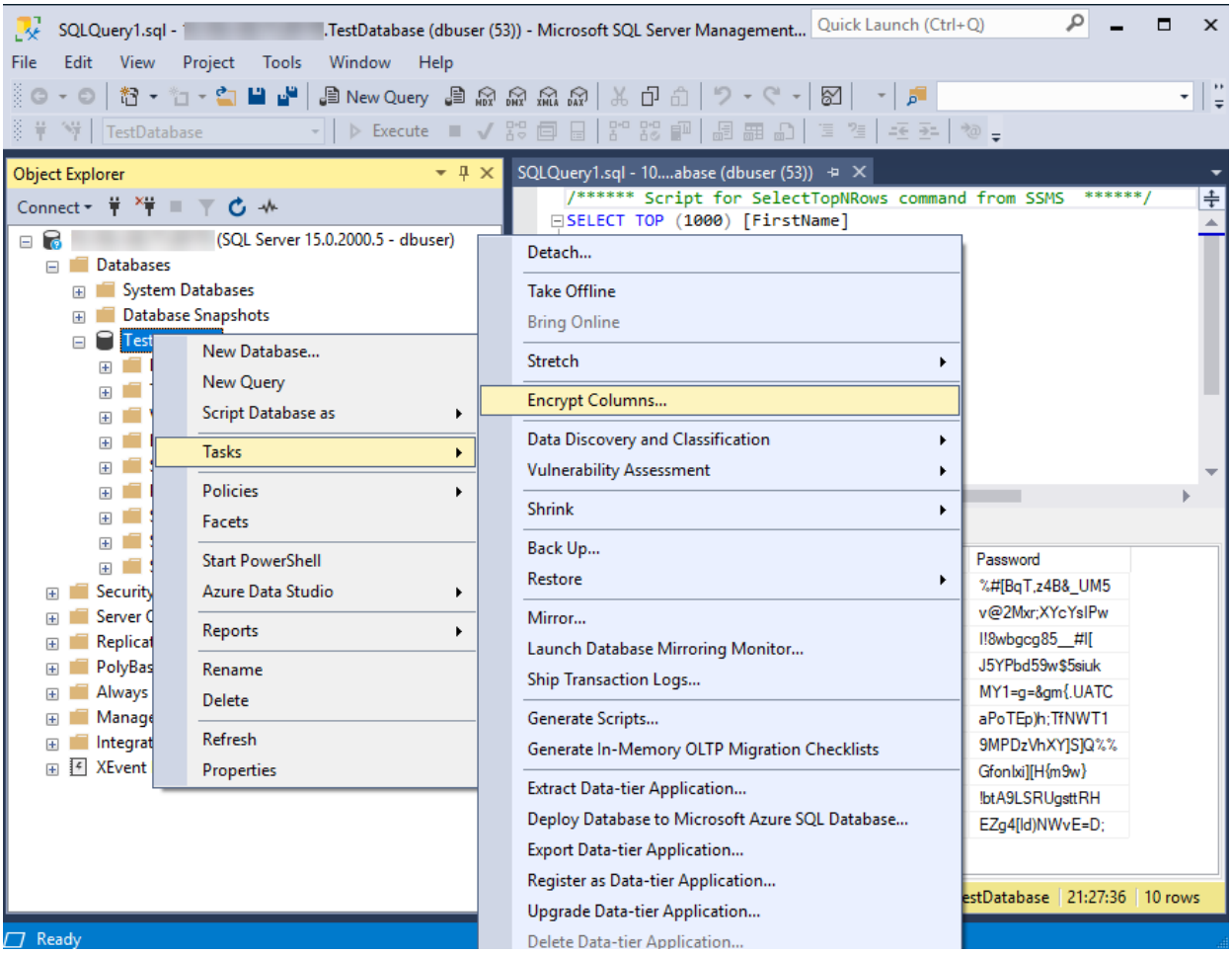


7. Right-click **dbo.Table**, and select **Select Top 1000 Rows**. The column that was chosen for encryption is now being decrypted by **Always Encrypted** with the key protected by the nCipher HSM.
8. Select **Finish**.

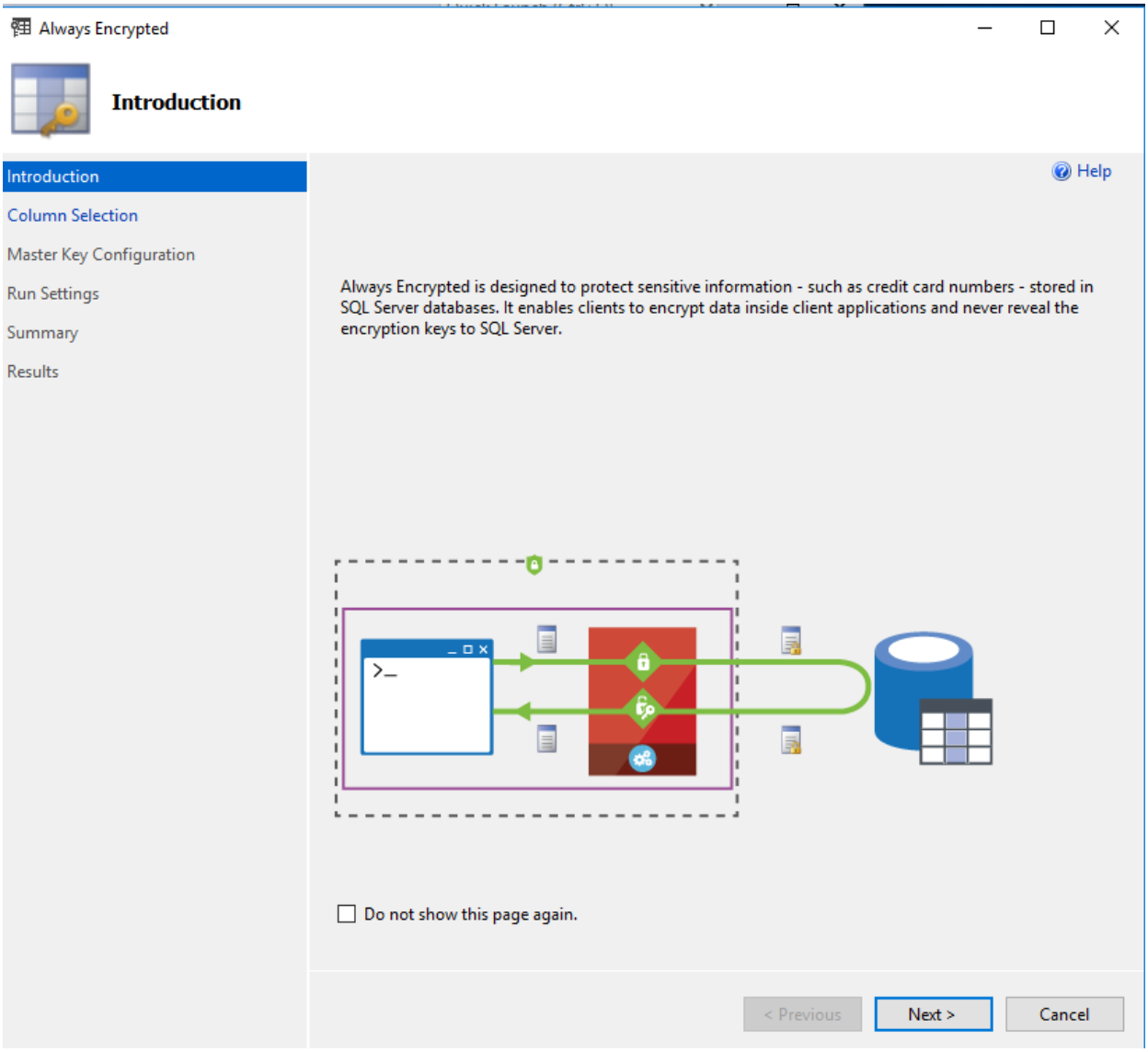


4.3. Remove column encryption

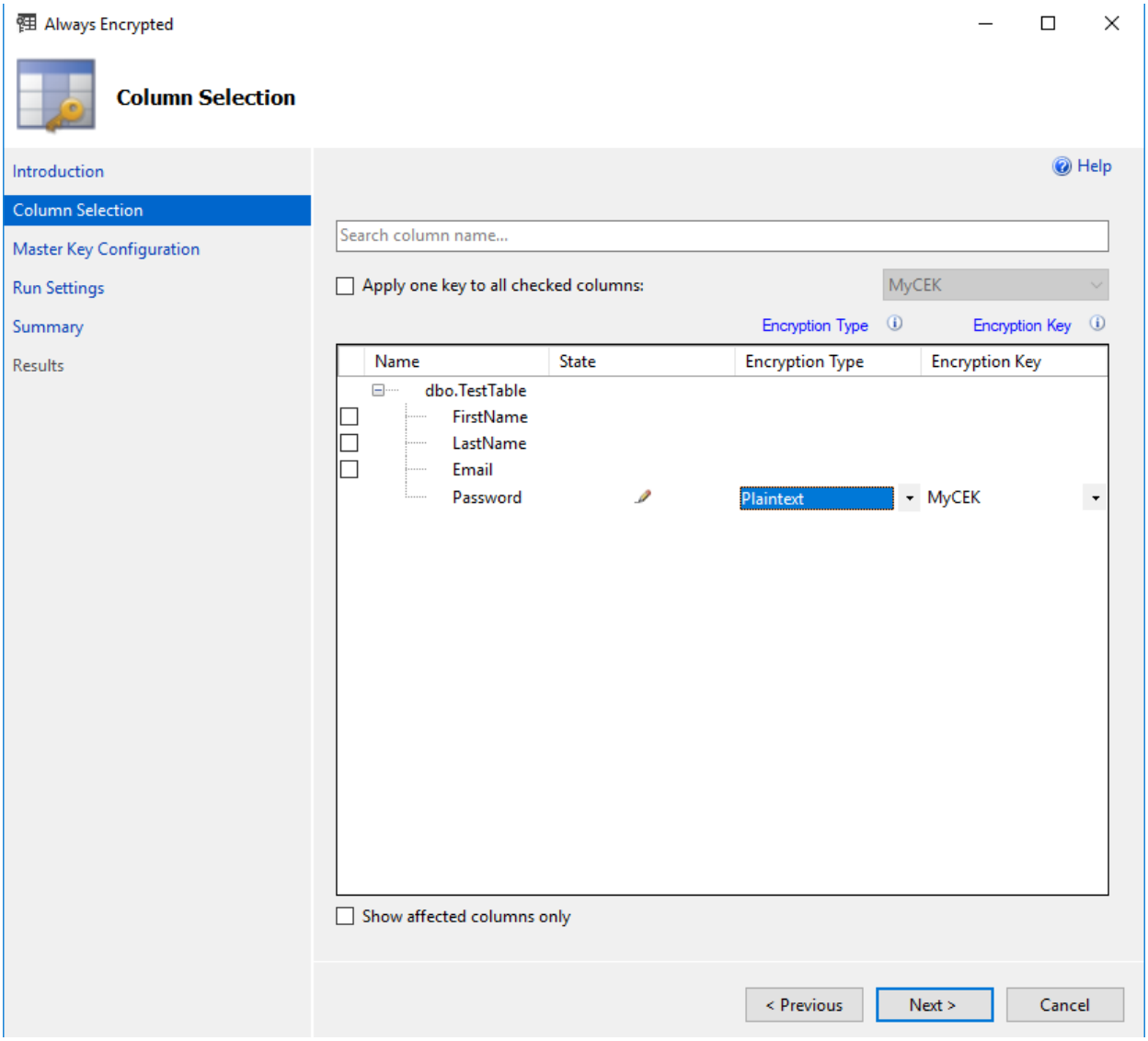
1. Right-click the required database and in the **Tasks** menu and select **Encrypt Columns**.



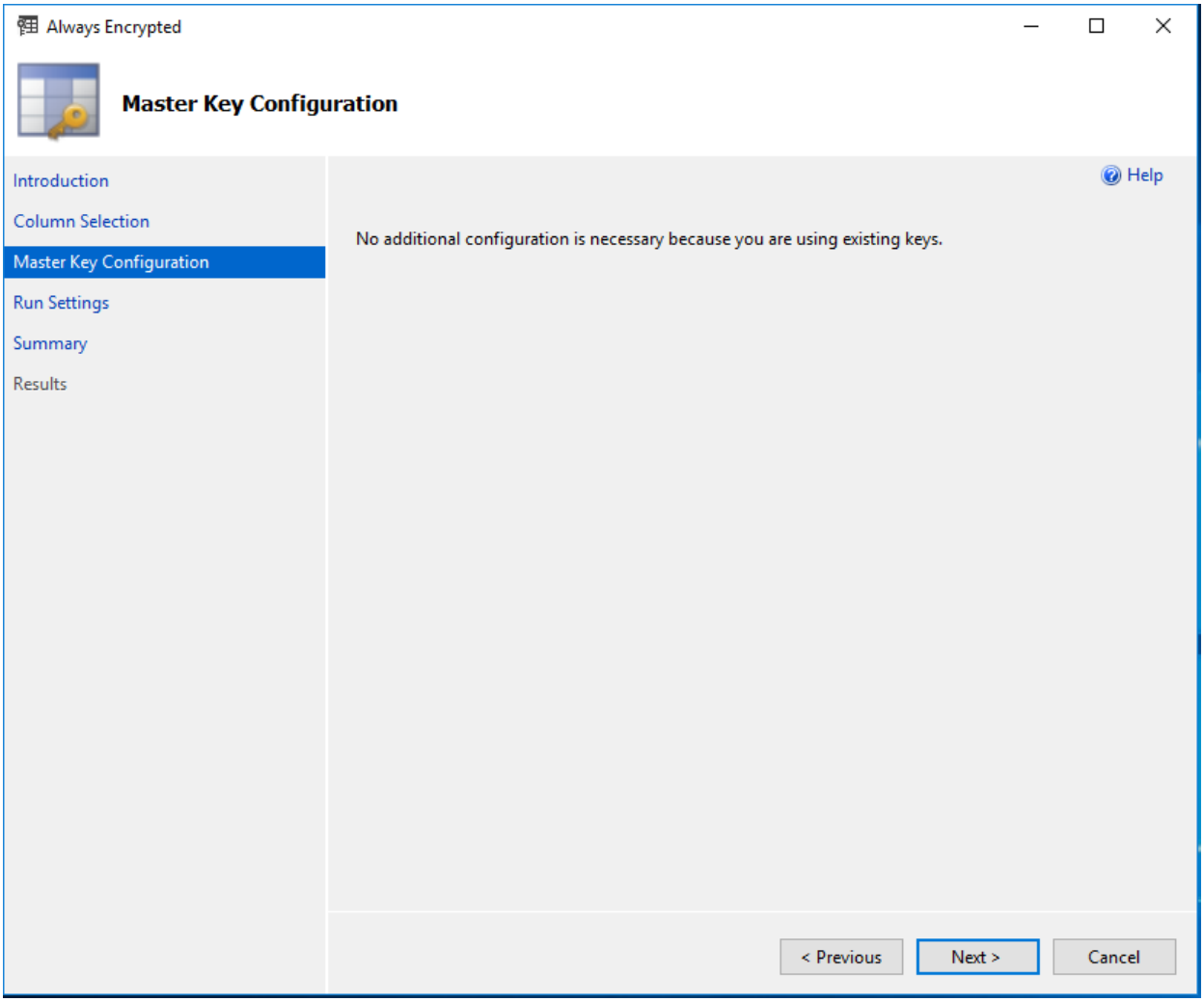
2. Select **Next** on the Introduction screen.



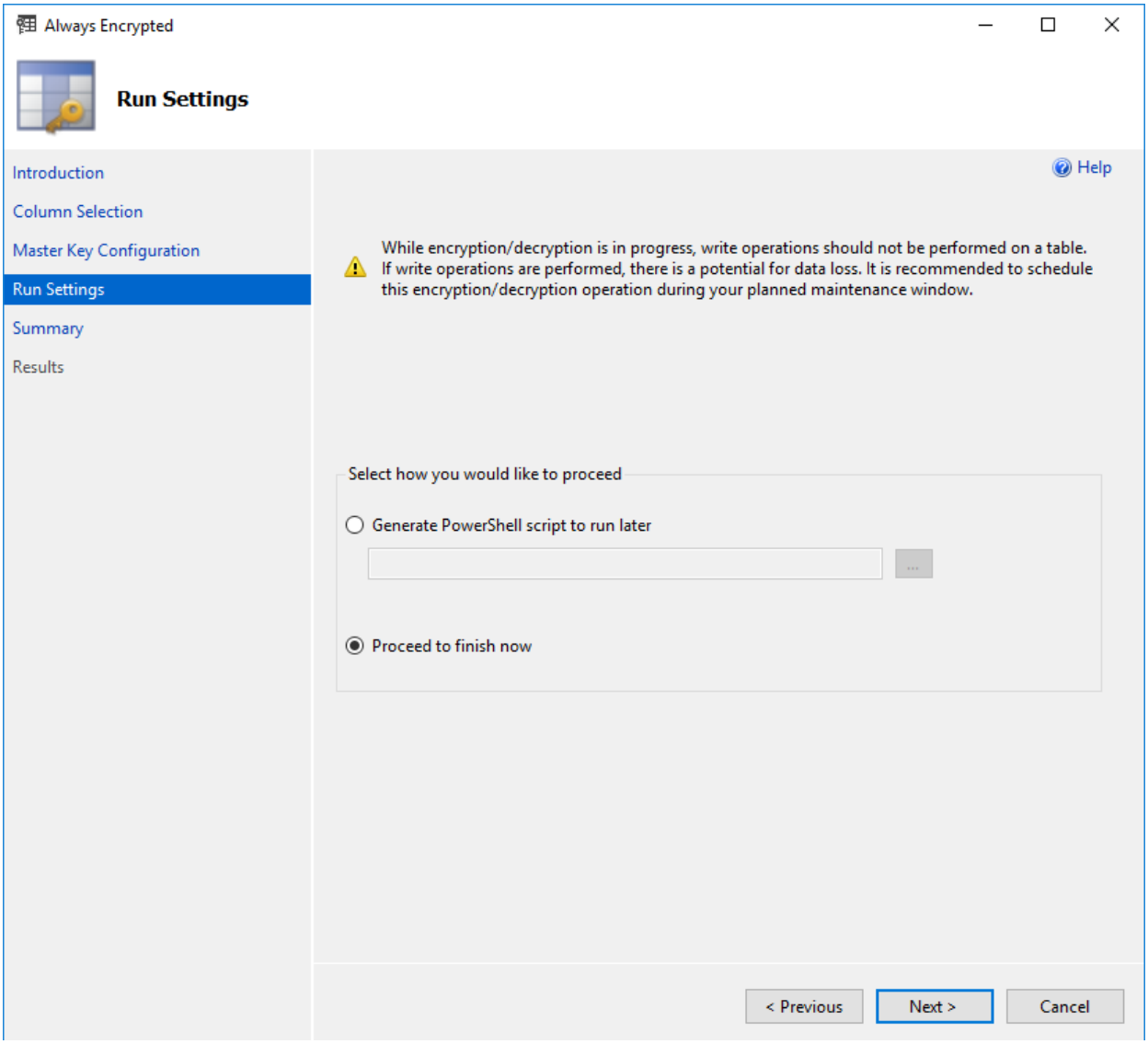
3. Select **Plaintext** from the drop down list in the **Encryption Type** and select **Next**.



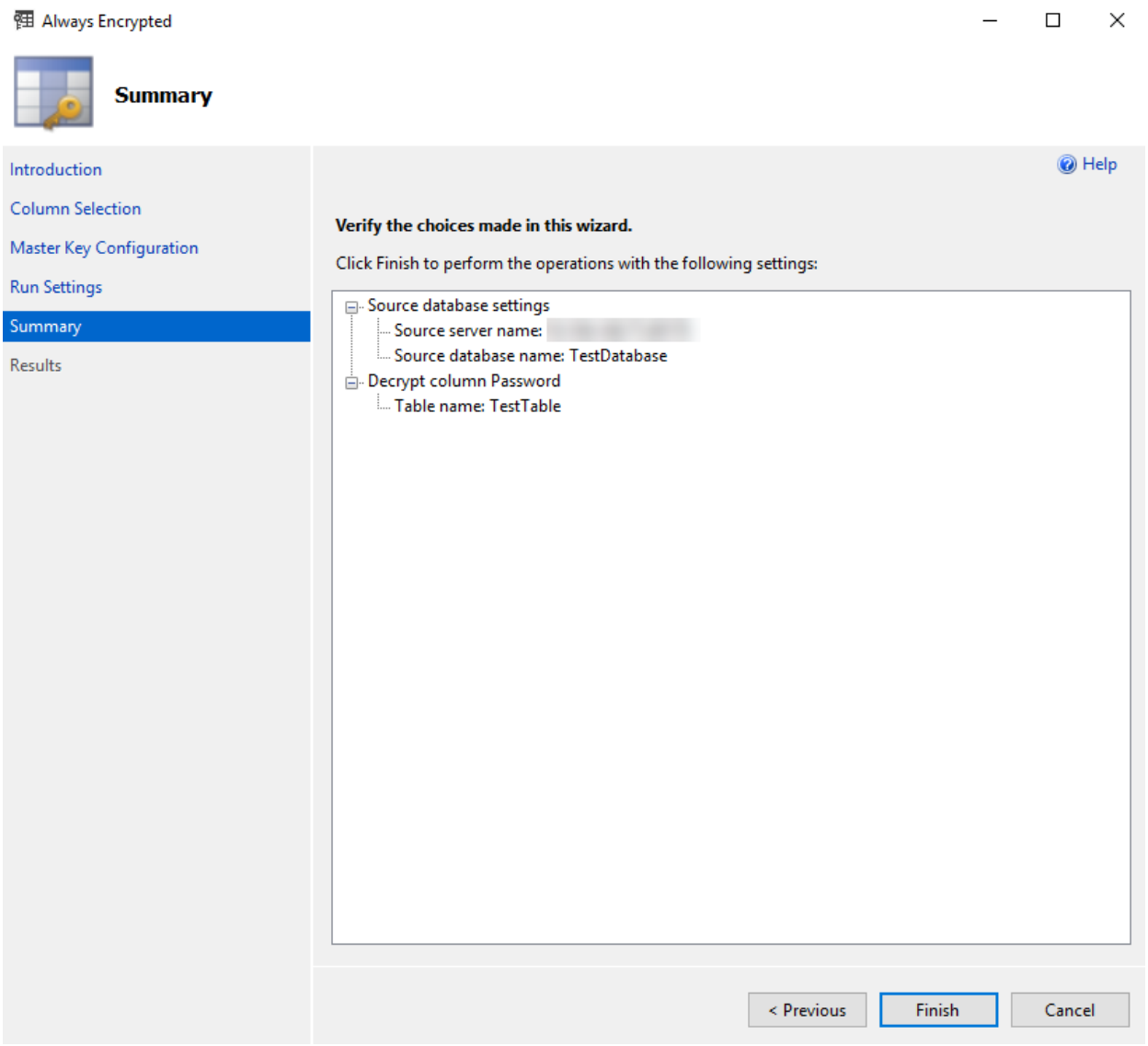
4. Select **Next** on the **Master Key Configuration** window.



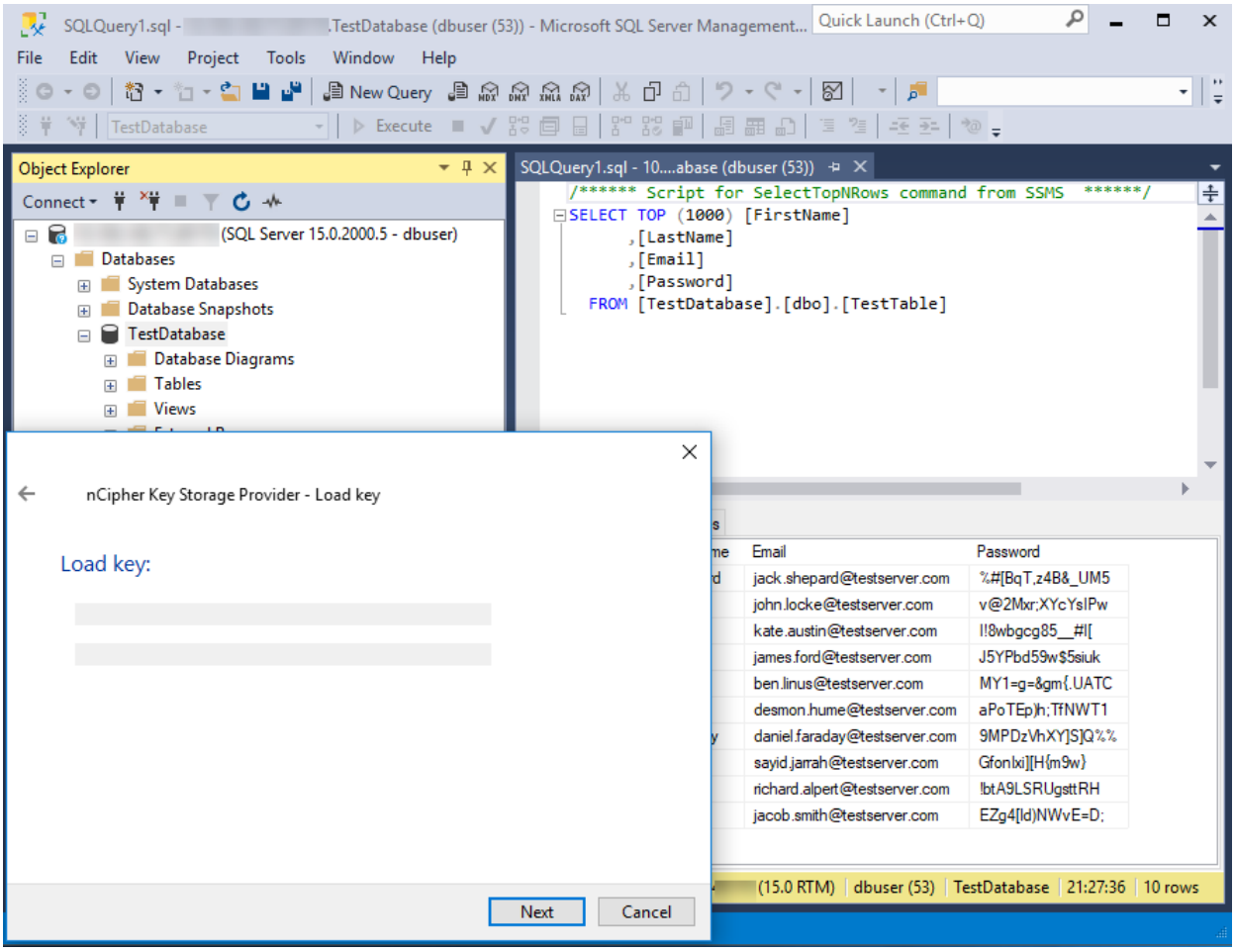
5. Select the **Proceed to finish now** radio button and select **Next**.



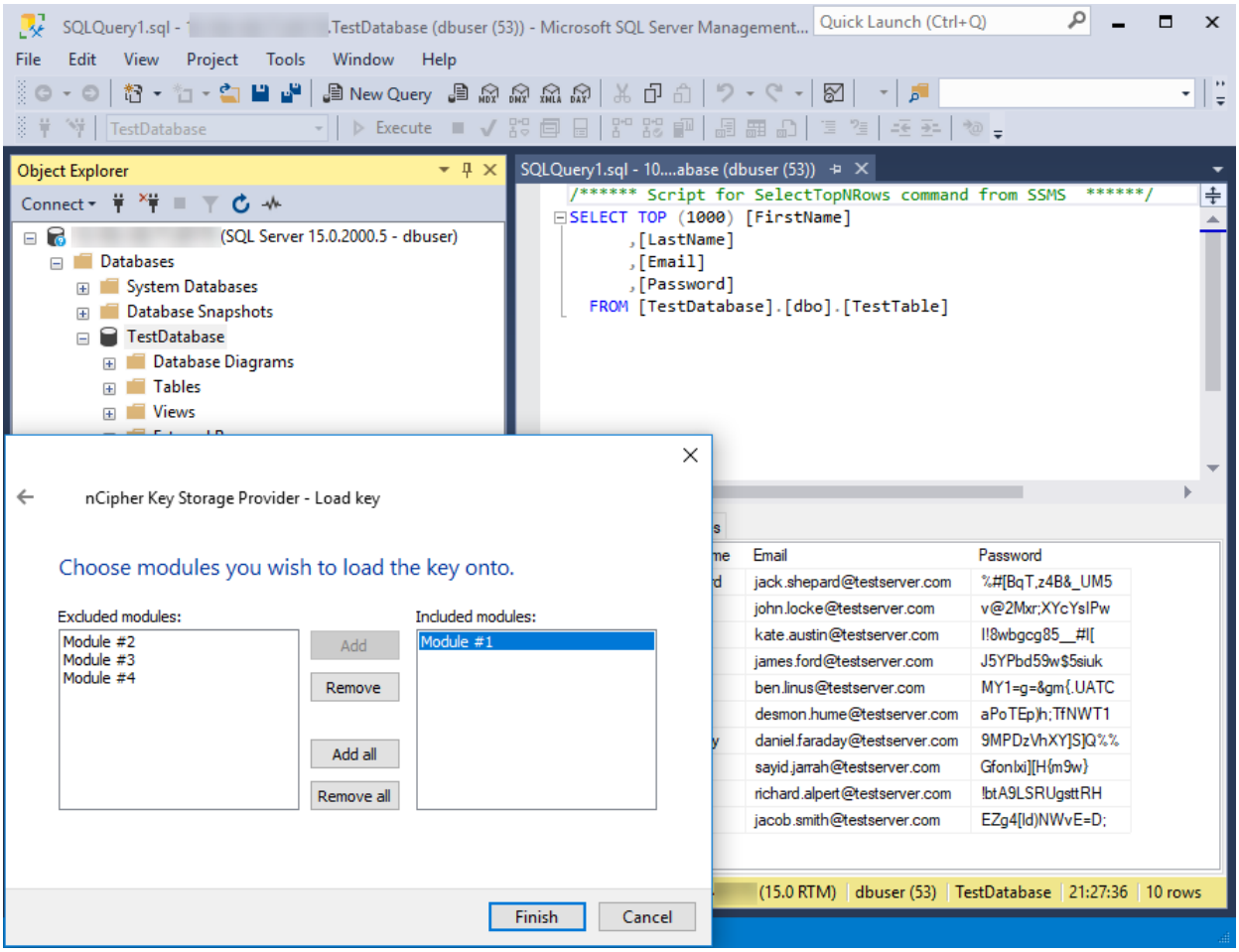
6. Select **Finish** on the **Summary** window.



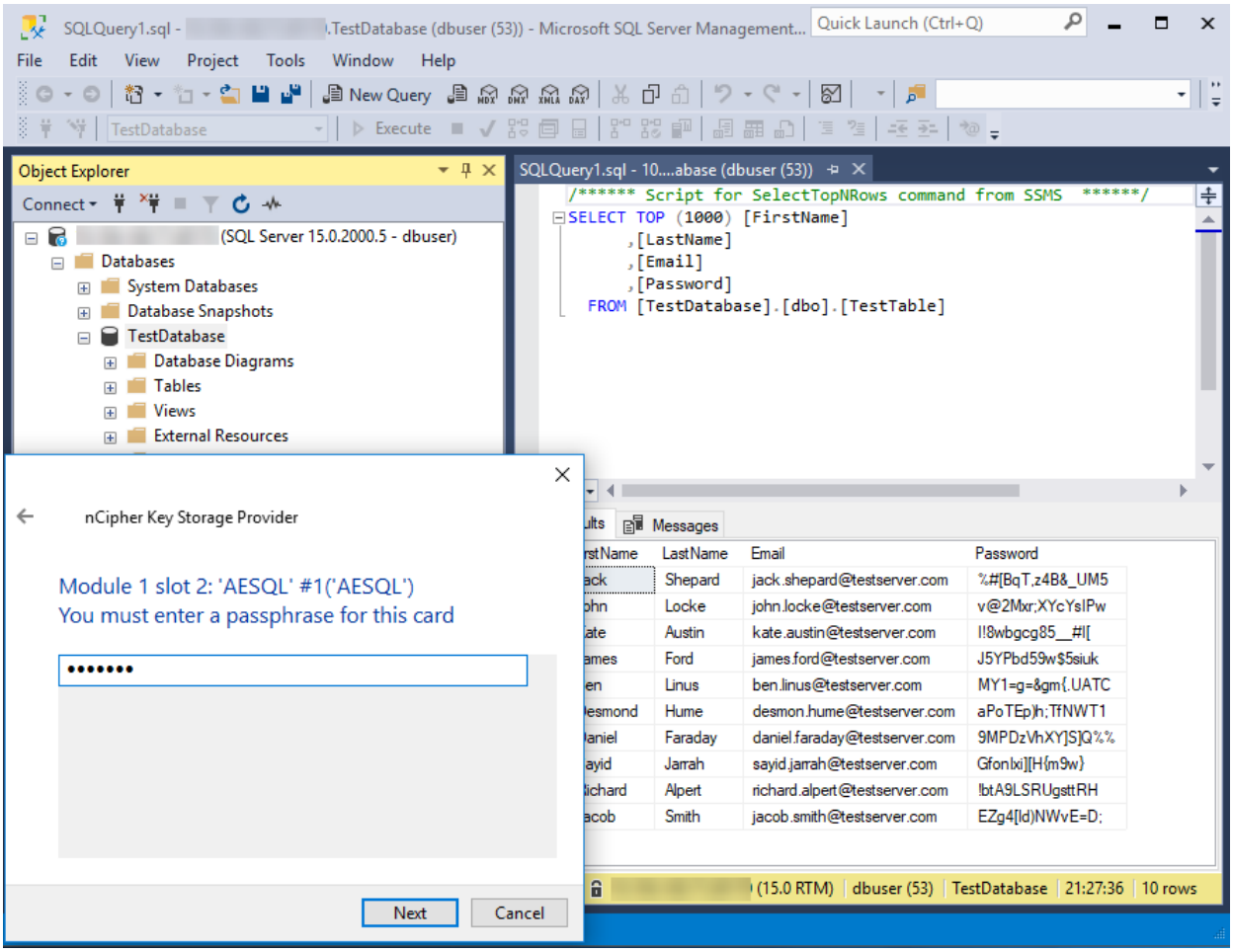
7. Present the OCS and select **Next**.



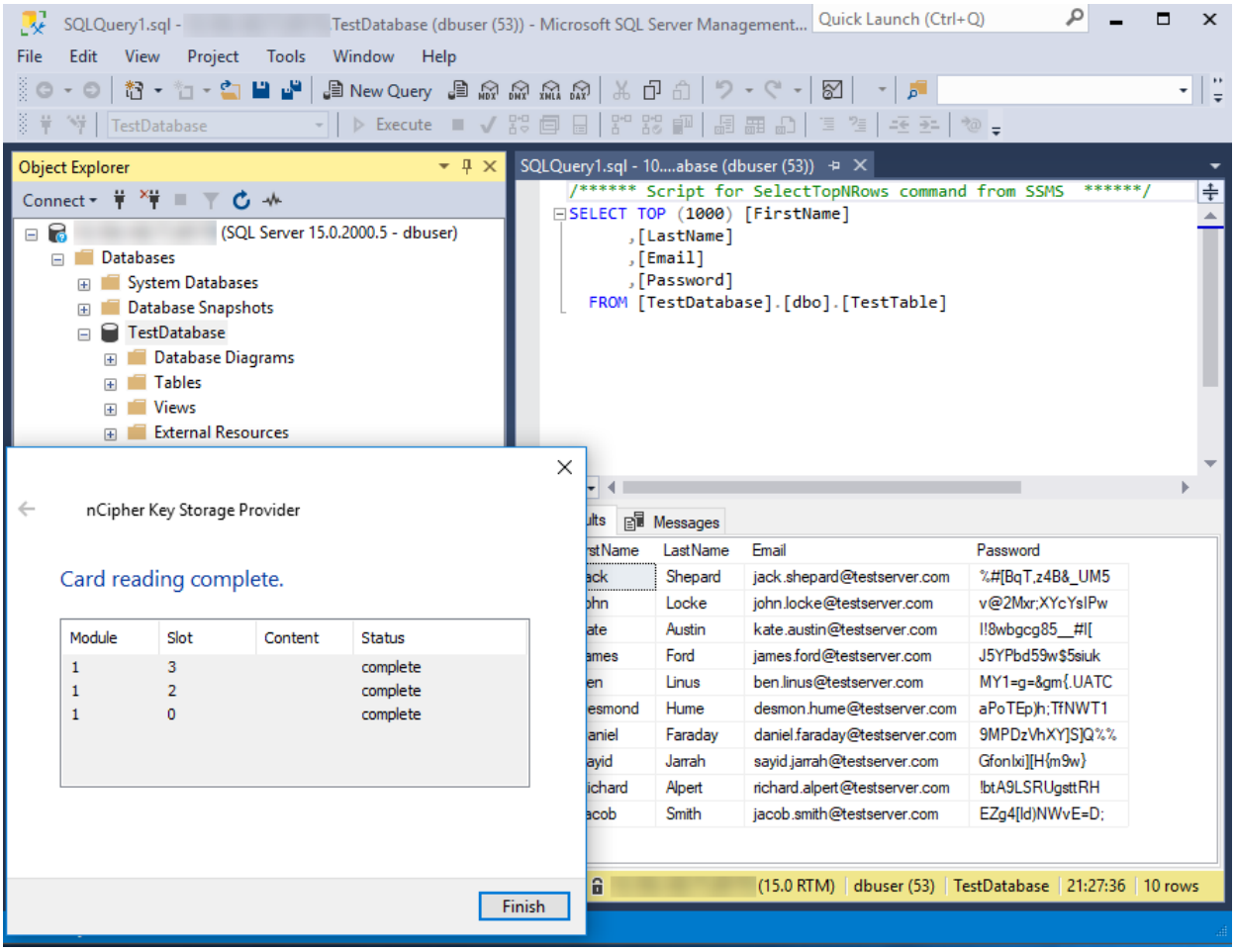
8. Select the HSM and select **Finish**.



9. Enter the passphrase and select **Next**.



10. Select **Finish** upon **Card reading complete**.



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 View an encrypted column.

5. Encrypt or decrypt a column with PowerShell

5.1. Encrypt a column

1. Launch PowerShell on the on-premises client computer and run the following script named Encrypt_Column_Named_Password.ps1.

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

# Connect to database.
$ConnectionString = "Data Source=<DB_Server_IP>,49170;Initial Catalog=TestDatabase;User
ID=dbuser;Password=<dbuser_Password>;MultipleActiveResultSets=False;Connect
Timeout=30;Encrypt=True;TrustServerCertificate=True;Packet Size=4096;Application Name=`"Microsoft SQL Server
Management Studio`""
$Database = Get-SqlDatabase -ConnectionString $ConnectionString

# 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 $Database
```

   The command line is

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

2. Present the OCS, select the HSM, and enter the passphrase.

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

5.2. View an encrypted column

Reconnect to the SQL server with **Always Encrypted** disabled to view the encrypted data stored in the SQL server. See View an encrypted column.

5.3. Remove column encryption

1. Launch PowerShell on the on-premises client computer and run the following script named Decrypt_Column_Named_Password.ps1.

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

# Connect to database.
$ConnectionString = "Data Source=<DB_Server_IP>,49170;Initial Catalog=TestDatabase;User
ID=dbuser;Password=<dbuser_Password>;MultipleActiveResultSets=False;Connect
Timeout=30;Encrypt=True;TrustServerCertificate=True;Packet Size=4096;Application Name=`"Microsoft SQL Server
Management Studio`""
$Database = Get-SqlDatabase -ConnectionString $ConnectionString

# Change encryption schema
$encryptionChanges = @()

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

The command line is

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

2. Present the OCS, select the HSM, and enter the passphrase.

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 View an encrypted column.

6. 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 at https://docs.microsoft.com/en-us/powershell/module/sqlserver/?view=sqlserver-ps.