1. Introduction

This document describes how to integrate Microsoft SQL Server with the nShield Database Security Option Pack (nDSOP V2.1) using an Entrust nShield hardware security module (HSM) as a Root of Trust.

1.1. Product configurations

Entrust tested the integration with the following versions:

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base OS</td>
<td>Windows Server 2019 Datacenter</td>
</tr>
<tr>
<td>SQL Server</td>
<td>Microsoft 2016 Enterprise</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Microsoft 2019 Enterprise</td>
</tr>
<tr>
<td>Management Studio</td>
<td>v18.8</td>
</tr>
</tbody>
</table>

1.2. Supported nShield hardware and software versions

Entrust tested the integration with the following nShield HSM hardware and software versions, and SQLEKM provider:

<table>
<thead>
<tr>
<th>Product</th>
<th>Security World</th>
<th>Firmware</th>
<th>Netimage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect XC</td>
<td>12.60.11</td>
<td>12.50.11 (FIPS Certified)</td>
<td>12.60.10</td>
</tr>
<tr>
<td>Connect XC</td>
<td>12.80.4</td>
<td>12.50.11 (FIPS Certified)</td>
<td>N/A</td>
</tr>
<tr>
<td>Connect XC</td>
<td>12.80.4</td>
<td>12.72.1 (FIPS Certified)</td>
<td>N/A</td>
</tr>
<tr>
<td>nShield 5c</td>
<td>13.2.2</td>
<td>13.2.2 (FIPS Pending)</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Supported nShield SQLEKM provider:

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>nDSOP</td>
<td>2.1</td>
</tr>
</tbody>
</table>

1.3. Supported nShield functionality
## Functionality vs. Support

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIPS 140-2 Level 3</td>
<td>Yes</td>
</tr>
<tr>
<td>Key Management</td>
<td>Yes</td>
</tr>
<tr>
<td>Key Generation</td>
<td>Yes</td>
</tr>
<tr>
<td>Key Recovery</td>
<td>Yes</td>
</tr>
<tr>
<td>1 of N Card Set</td>
<td>Yes</td>
</tr>
<tr>
<td>Softcards</td>
<td>Yes</td>
</tr>
<tr>
<td>Module Only Key</td>
<td>No</td>
</tr>
<tr>
<td>Fail Over</td>
<td>Yes</td>
</tr>
<tr>
<td>Load Balancing</td>
<td>Yes</td>
</tr>
<tr>
<td>nSaaS</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 1.4. Requirements

Be familiar with:

- The Microsoft SQL Server features and documentation.
- The Microsoft SQL Server Management Studio features and documentation.
- The T-SQL language. The minimum requirement for T-SQL is a basic understanding of SQL tasks such as creating a database or tables.
- Database security concepts and practices.
- The documentation for the HSM.

### 1.5. Terms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLEKM</td>
<td>SQL Server Extensible Key Management</td>
</tr>
<tr>
<td>TDEKEK</td>
<td>TDE Key Encryption Key</td>
</tr>
<tr>
<td>TDEDEK</td>
<td>TDE Database Encryption Key</td>
</tr>
</tbody>
</table>
2. Setup

Prerequisites:

- A Windows Server with Microsoft SQL server.
- SQL Server Management Studio installed.
- The database TestDatabase has been created and is available for the integration.

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

To install the Security World software and create a Security World:

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:
   - 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 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  S30E-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  S30E-02E0-D947
  mode  operational
...
```

6. Create a Security World if one does not already exist or copy an existing one. Follow your organization’s security policy for this. Create extra ACS cards as spares in case of a card failure or lost. These cannot be duplicated after the Security World is...
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 the nShield nDSOP

To install the nShield nDSOP:

1. Mount the `nDSOP_Windows-x.x.x.iso` file.
2. Double-click the `setup` file and follow the instructions.

2.3. Create the Operator Card Set (OCS) or Softcard

The OCS or Softcard and associated passphrase will be used to authorize access to specific keys protected by the SQLEKM provider. Typically, an organization’s security policies dictate the use of one or the other.

2.4. Create the OCS

A SQL Server credential (as used for EKM) maps one protecting token to one stored passphrase. It can store information for only one token at a time. An OCS does have a quorum of one.

Recovering from a power failure requires the OCS to be inserted in the HSM or the TVD.

1. Ensure the cardlist file located in the path `C:\ProgramData\nCipher\Key Management Data\config\` contains the serial number of the card(s) to be presented or the wildcard value.
2. Open a command window as administrator.
3. Execute the following command. Enter a passphrase or password at the prompt. Follow your organization’s security policy for the OCS values. After an OCS card set has been created, the cards cannot be duplicated. Notice that `slot 2`, remote via a Trusted Verification Device (TVD), is used to present the card.
Add the `-p` (persistent) option to the command above if you want:

- to be able to encrypt/decrypt the database after the OCS card has been removed from the HSM front panel slot or from the TVD.
- the ability to persist after a reboot.

The authentication provided by the OCS as shown in the command line above is non-persistent and only available while the OCS card is inserted in the HSM front panel slot or the TVD. If the TVD loses connection to the Remote Administration client the database will be inaccessible.

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
6c9c2b7c32fb884a531e08f7a37edc3924fb5e76  1/1  none-NL nDSOPocs
```

2.5. Create the Softcard

A SQL Server credential (as used for EKM) maps one protecting token to one stored passphrase. Softcards are singular and do not have a quorum, so the SQL Server credential matches them quite well.

Unlike OCS protection, which requires a smart card and a passcode, a softcard does not require additional input for recovery after a power failure.

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. Execute the following command. Enter a passphrase at the prompt.
3. Verify the Softcard created:

```
>nkminfo -s
SoftCard summary - 1 softcards:
Operator logical token hash name
629a3a902eeec139fb5136afec7b0f0801f45b60d nDSOPsoftcard
```

The `rocs` utility 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 nDSOPocs                 1 (1)  1 of 1
   2 nDSOPsoftcard            0 (0)  (softcard)
rocs> quit
```

2.6. Enable EKM and register the SQLEKM provider

To enable EKM and register the SQLEKM provider:

1. Launch the SQL Server Management Studio GUI.
2. Enable EKM by executing the following query:

```
sp_configure 'show advanced', 1
GO
RECONFIGURE
GO
sp_configure 'EKM provider enabled', 1
GO
RECONFIGURE
GO
```
3. Register the SQLEKM provider with the SQL Server by executing the following query:

```sql
CREATE CRYPTOGRAPHIC PROVIDER nDSOP
FROM FILE = 'C:\Program Files\nCipher\fast\bin\ncsqlekm.dll'
```

4. Check the SQLEKM provider is listed in the SQL Server Management Studio GUI. Go to Security > Cryptographic Providers. nDSOP should be visible. Right-click it to verify that it is enabled.
2.7. Verify the SQLEKM provider configuration

To verify the SQLEKM provider configuration:

1. Run the following query:

   ```sql
   SELECT * FROM sys.cryptographic_providers;
   ```

   Verify the following:
   - The version matched that of the nDSOP installation iso.
   - Path to dll is correct.
   - is_enabled column set to 1.

2. Run the following query:

   ```sql
   SELECT * FROM sys.dm_cryptographic_provider_properties;
   ```
Verify the following:

<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>friendly_name</td>
<td>nCipher SQLEKM Provider</td>
</tr>
<tr>
<td>authentication_type</td>
<td>BASIC</td>
</tr>
<tr>
<td>symmetric_key_support</td>
<td>1</td>
</tr>
<tr>
<td>asymmetric_key_support</td>
<td>1</td>
</tr>
</tbody>
</table>

3. Verify the supported cryptographic algorithms can be queried by running the following query:

```sql
DECLARE @ProviderId int;
SET @ProviderId = (SELECT TOP(1) provider_id FROM sys.dm_cryptographic_provider_properties
WHERE friendly_name LIKE 'nCipher SQLEKM Provider');
SELECT * FROM sys.dm_cryptographic_provider_algorithms(@ProviderId);
GO```
Notice each key type has its set of valid algorithms.

<table>
<thead>
<tr>
<th>Key Type</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symmetric</td>
<td>AES_128, AES_192, ASE_256</td>
</tr>
<tr>
<td>Asymmetric</td>
<td>RSA_2048, RSA_3072, RSA_4096</td>
</tr>
</tbody>
</table>

2.8. Create the user SQL Server credential

To create the user SQL Server credential:

1. Verify the OCS or Softcard created above:

```
> nfkminfo -c
Cardset list - 1 cardsets: (P)ersistent/(N)ot, (R)emoteable/(L)ocal-only
Operator logical token hash k/n timeout name
6c9c2b7c32fb884a531e08f7a37e3924fb5e76 1/1 none-NL nDSOPocs
```

```
> nfkminfo -s
SoftCard summary - 1 softcards:
Operator logical token hash name
629a3a9828e13f9f5136afec7b8f088f45b68d nDSOPsoftcard
```

3. Right-click Credentials, then select New Credential.
4. Enter the credential name, and the OCS card name and passphrase. Check Use Encryption Provider and select nDSOP. Select OK.
5. Verify the new credential in **Security > Credentials**. May need to right-click and select **Refresh**.

6. Navigate to **Security > Logins**. Right-click the login used to access the TestDatabase and select **Properties**.

7. Check **Map to Credentials** in the dialog. Select the server credential created above in the drop-down to the right. Then select **Add**, and select **OK**.
3. Configure TDE

The TDE Database Encryption Key (TDEDEK) is a symmetric key that is used to perform the actual encryption of the database and are unique to a given database. It is created by SQL Server and cannot be exported from the database, meaning it cannot be created or directly protected by the SQLEKM provider (nShield HSM).

The TDEDEK is protected within the database by encrypting it with a wrapping key. The wrapping key is called the TDE Key Encryption Key (TDEKEK). The TDEKEK is an asymmetric key protected by the SQLEKM provider in the nShield HSM. It is possible to have a single TDEKEK for multiple databases, or different TDEKEKs for different databases.

The TDEKEK must be created under the tdeLogin/tdeCredential. However, the current user does not have to use the tdeCredential, so long as the user credential is using the same OCS or Softcard as the tdeCredential.

3.1. Create a TDEKEK

To create a TDEKEK in the master database:

1. Insert the OCS in the HSM slot or TVD. If using Softcard protection, no action is needed.
2. Run the following query:

   ```sql
   USE master;
   CREATE ASYMMETRIC KEY "<name_of_key_in_database>"
   FROM PROVIDER "<SQLEKM_provider>"
   WITH
   PROVIDER_KEY_NAME = '<name_of_key_in_SQLEKM_provider>',
   CREATION_DISPOSITION = CREATE_NEW,
   ALGORITHM = <asymmetric_algorithm_desc>;
   GO
   ```

   Where:

   - **name_of_key_in_database**  The name given to the key in the database.
   - **name_of_key_in_SQLEKM_provider**  The name given to the key in the SQLEKM provider.
   - **asymmetric_algorithm_desc**  A valid asymmetric key algorithm descriptor.

   See Verify the SQLEKM provider configuration.
Example:

```
USE master;
CREATE ASYMMETRIC KEY "AsymTestWrappingKeyDatabase"
FROM PROVIDER "nDSOP"
WITH
  PROVIDER_KEY_NAME = 'AsymTestWrappingKeySQLEKM',
  CREATION_DISPOSITION = CREATE_NEW,
  ALGORITHM = RSA_2048;
GO
```

Notice the newly created key highlighted in the object explorer.

3. The key generated can also be verified using a CLI command:

```
> nfkminfo -l

Keys protected by cardsets:
key_simple_sqlekm-6c9c2b7c32fb884a531e88f7a37edc3924fb5e76-67703bab0c40915c62f6afe64c49c741c41e24b
'AsymTestWrappingKeySQLEKM'
```

The `rocs` utility shows the names and protection methods of the keys.

```
> rocs
  'rocs' key recovery tool
  Useful commands: 'help', 'help intro', 'quit'.
  rocs> list keys
<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>App</th>
<th>Protected by</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AsymmetricTestKeySQLEKM</td>
<td>simple</td>
<td>nDSOPocs</td>
</tr>
</tbody>
</table>
  rocs> quit
```

3.2. Create a TDE login and credential

A tdeLogin and tdeCredential allows an ordinary database user, who is fully authorized to
use the database, but has no SQLEKM credentials of their own, to perform query operations using a TDE encrypted database. Without the tdeLogin and tdeCredential, then every user would need their own credentials. It is beyond the scope of this document to provide an example of how to use these credentials, only on how to create them.

3.3. Create a TDE credential

To create a TDE credential:

1. In SQL Server Management Studio, navigate to Security > Credentials.
2. Right-click Credentials, then select New Credential.
3. Enter the Credential name.
4. Enter the Identity as the name of the OCS or Softcard created above.
5. Enter the Password as the passphrase on the OCS or Softcard.
6. Select Use Encryption Provider. Select the provider and select OK.

7. Notice the credential created.
3.4. Create the TDE login

To create the TDE login:

1. In SQL Server Management Studio, navigate to Security > Logins.
2. Right-click Logins, then select New Login.
3. Enter the Login name.
4. Select Mapped to asymmetric key. Then select the asymmetric key created earlier.
5. Select Map to Credential. Then select the TDE credential created earlier. Then select Add.
6. Select OK.
7. Notice the login created.

3.5. Create the TDEDEK and switch on encryption

To create the TDEDEK and switch on encryption:

1. In SQL Server Management Studio, navigate to Databases > TestDatabase.
2. Right-click TestDatabase, then select Tasks > Manage Database Encryption.
3. Set Encryption Algorithm to AES 256 or your choice.
4. Select Use server asymmetric key. Then select the asymmetric key created earlier.
5. Select Set Database Encryption On. Then select OK. Restart the Microsoft SQL Server Management Studio and repeat these steps if it fails.

6. Run the following query to verify the encryption state:
/* Script for SelectTopNRows command from SSMS */
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.certificates AS c ON e.encryptor_thumbprint = c.thumbprint

The following table shows the value returned for encryption state and the meaning.

<table>
<thead>
<tr>
<th>Encryption state</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Encryption disabled (or no encryption key)</td>
</tr>
<tr>
<td>1</td>
<td>Unencrypted or Decrypted</td>
</tr>
<tr>
<td>2</td>
<td>Unencrypted or Decrypted</td>
</tr>
<tr>
<td>3</td>
<td>Encrypted</td>
</tr>
<tr>
<td>4</td>
<td>Key change in progress</td>
</tr>
<tr>
<td>5</td>
<td>Decryption in progress</td>
</tr>
<tr>
<td>6</td>
<td>Protection change in progress (The certificate or asymmetric key that is encrypting the database encryption key is being changed)</td>
</tr>
</tbody>
</table>

7. Turn off encryption by clearing **Set Database Encryption On** in the steps above.
3.6. Key rotation - Replace the TDEKEK

This is the wrapping key called TDE Key Encryption Key, an asymmetric key protected by the SQLEKM provider in the nShield HSM.

1. Follow the procedure in Create a TDEKEK to create a new asymmetric TDEKEK.

Example:

```sql
USE master;
CREATE ASYMMETRIC KEY "AsymTestWrappingKeyDatabase2"
FROM PROVIDER "nDSOP"
WITH
PROVIDER_KEY_NAME = 'AsymTestWrappingKeySQLEKM2',
CREATION_DISPOSITION = CREATE_NEW,
ALGORITHM = RSA_2048;
GO
```

2. In SQL Server Management Studio, navigate to Databases > TestDatabase.
3. Right-click TestDatabase, then select Tasks > Manage Database Encryption.
4. Select Re-Encrypt Database Encryption Key and Use server asymmetric.
5. Select the newly created asymmetric key AsymTestWrappingKeyDatabase2.
6. Deselect Regenerate Database Encryption Key.
7. Select Set Database Encryption On.
8. Select OK.

3.7. Key rotation - Replace the TDEDEK

This is the key called TDE Database Encryption Key, a symmetric used to perform the actual encryption of the database. It is created by SQL Server and cannot be exported
from the database. It is protected within the database by encrypting it with a wrapping key TDEKEK.

1. In SQL Server Management Studio, navigate to Databases > TestDatabase.
2. Right-click TestDatabase, then select Tasks > Manage Database Encryption.
3. Deselect Re-Encrypt Database Encryption Key.
4. Select Regenerate Database Encryption Key.
5. Select AES 256.
7. Select OK.

8. Verify the encryption state as shown in Create the TDEDEK and switch on encryption.
4. Perform backup and recovery

A rigorous backup regimen is recommended to provide a means to recover both the database and associated keys used for encryption. Consult your corporate IT and security team for best practice and corporate policy requirements.

4.1. Back up the Security World

The Security World data is inherently encrypted and does not require any further encryption operation to protect it. It can only be used by someone who has access to a quorum of the correct ACS cards, or the OCS card, Softcard, their passphrases, an nShield HSM and nShield Security World Software. Therefore, backup simply consists of making a copy of the Security World files and saving the copy in a safe location, as necessary to restore the keys used by the database.

1. Back up C:\ProgramData\nCipher\Key Management Data.
2. Securely store and keep a record of ACS and OCS cards associated with each Security World, preferable using the serial number on the cards.
3. The Softcard, used instead of OCS, resides in the Key Management Data folder. It is backed up at C:\ProgramData\nCipher\Key Management Data.
4. Keep a record of which database and which Security World backups correspond to each other.

4.2. Restore the Security World

Restoring a Security World simply means restoring a backup copy of the Security World folder C:\ProgramData\nCipher\Key Management Data.

The ACS is required if the Security World being restored is not already loaded onto the HSM. See the Installation Guide and the User Guide for the HSM. A short version is available at How to locally set up a new or replacement nShield Connect.

4.3. Back up the database

To back up the database:

1. Create the backup devices by running the following query:
Notice the devices created.

2. Create the backup by running the following query:

```
-- Encrypted Backup
USE master;
GO

-- Provide backup device and locations
EXEC sp_addumpdevice 'disk', 'EncryptedTestDatabaseBackup', 'C:\Program Files\Microsoft SQL Server\MSSQL15.INTEROPDATABASE\MSSQL\Backup\TestDatabaseEncrypted.bak';
GO

EXEC sp_addumpdevice 'disk', 'EncryptedTestDatabaseBackupLog', 'C:\Program Files\Microsoft SQL Server\MSSQL15.INTEROPDATABASE\MSSQL\Backup\TestDatabaseEncryptedLog.bak';
GO

Notice the backup files created.
```
If the database is encrypted, the backup will also be encrypted. If the database is not encrypted, then the backup will not be encrypted. If you want to create an encrypted backup from a non-encrypted database, you will have to create the non-encrypted backup file, and then encrypt the file using an independent encryption tool.

4.4. Restore the database

Restore a TDE encrypted database in a similar manner as an un-encrypted database. But for TDE encrypted database the Security World needs to be restored before restoring the encrypted database. The OCS, if used, needs to be inserted in the HSM before restoring the encrypted database. Otherwise, the restored database will appear as (Restore Pending).

1. Install the Security World software and the nShield nDSOP if rebuilding the server. Do not create a Security World.
2. Restore the Security World.
3. Insert the OCS in the HSM front panel slot, or the TVD if using OCS protection.
4. Enable EKM and register the SQLEKM provider if rebuilding the server.
5. Create the SQL Server credential if rebuilding the server. The OCS and Softcard are in the restored Security World.
6. Verify the SQLEKM provider configuration if rebuilding the server.
7. Import the database wrapping key (TDEKEK) into the master database by running the following query. This key should already exist in the restored Security World.

```sql
USE master;
GO
-- Import TDEKEK
CREATE ASYMMETRIC KEY "AsymTestWrappingKeyDatabase2"
FROM PROVIDER "nDSOP"
WITH
  PROVIDER_KEY_NAME = 'AsymTestWrappingKeySQLEKM2',
  CREATION_DISPOSITION = OPEN_EXISTING;
GO
```
8. Recreate the TDE logins and credentials by running the following query. Notice the name of the OCS (`nDSOPocs`), and Softcard (`nDSOPsoftcard`) created earlier.

- **OCS:**

  ```sql
  USE master;
  GO
  -- tdeLogin and tcdCredential
  CREATE LOGIN tdeLogin FROM ASYMMETRIC KEY AsymTestWrappingKeyDatabase2;
  CREATE CREDENTIAL tdeCredential WITH IDENTITY = 'nDSOPocs', SECRET = 'ncipher'
  FOR CRYPTOGRAPHIC PROVIDER nDSOP;
  ALTER LOGIN tdeLogin ADD CREDENTIAL tdeCredential;
  GO
  ```

- **Softcard:**

  ```sql
  ```
USE master;
GO
-- tdeLogin and tdeCredential
CREATE LOGIN tdeLogin FROM ASYMMETRIC KEY AsymTestWrappingKeyDatabase2;
CREATE CREDENTIAL tdeCredential WITH IDENTITY = 'nDSOPsoftcard', SECRET = 'ncipher'
FOR CRYPTOGRAPHIC PROVIDER nDSOP;
ALTER LOGIN tdeLogin ADD CREDENTIAL tdeCredential;
GO

9. Restore the database by running the following query:

USE master
ALTER DATABASE TestDatabase SET SINGLE_USER WITH ROLLBACK IMMEDIATE
RESTORE DATABASE [TestDatabase] FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL15.INTEROPDATABASE\MSSQL\Backup\TestDatabaseEncrypted.bak'
WITH REPLACE
GO

10. Return to multiple user mode by running the following script:

USE master;
ALTER DATABASE TestDatabase SET MULTI_USER;
GO
5. Column level encryption

Table Column data can be protected by an Entrust HSM protected key. These nDSOP EKM keys are generated within the SQL Server database and can encrypt/decrypt data in a column.

5.1. Generate a key

1. Insert the OCS in the HSM slot or TVD. If using Softcard protection, no action is needed.

2. To create an asymmetric key, run the following query:

   ```sql
   USE master;
   CREATE ASYMMETRIC KEY "<name_of_key_in_database>"
   FROM PROVIDER "<SQLEKM_provider>"
   WITH
   PROVIDER_KEY_NAME = '<name_of_key_in_SQLEKM_provider>',
   CREATION_DISPOSITION = CREATE_NEW,
   ALGORITHM = <asymmetric_algorithm_desc>;
   GO
   ```

   Where:

   - `name_of_key_in_database` The name given to the key in the database. For example, `AsymKey`.
   - `SQLEKM_provider` The SQLEKM provider. For example, `nDSOP`.
   - `name_of_key_in_SQLEKM_provider` The name given to the key in the SQLEKM provider. For example, `AsymKey`.
   - `asymmetric_algorithm_desc` A valid asymmetric key algorithm descriptor. For example, `RSA_2048`. See Verify the SQLEKM provider configuration.

3. To create a symmetric key, run the following query:

   ```sql
   USE master;
   CREATE SYMMETRIC KEY "SymKey"
   FROM PROVIDER "nDSOP"
   WITH
   PROVIDER_KEY_NAME = 'SymKey',
   CREATION_DISPOSITION = CREATE_NEW,
   ALGORITHM = AES_256;
   GO
   ```
4. To import an existing nDSOP EKM protected key into the SQL Server, run the following query:

```sql
CREATE SYMMETRIC KEY "ExistingSymKey"
FROM PROVIDER "nDSOP"
WITH
PROVIDER_KEY_NAME = 'ExistingSymKey',
CREATION_DISPOSITION=OPEN_EXISTING
GO
```

5. Verify the key:

```bash
>nkminfo -l

Keys protected by cardsets:
key_simple_sqlekm-f82cca5299d7f8df581154de03b736998fd59f89-5d272ab1d5d69ab1300209ea5d0d1e50f59fa4 'SymKey'
```

5.2. Column encryption

1. Create a table:

```sql
USE master;
CREATE TABLE testdb
(
employee_name varchar(255) NOT NULL,
employee_secret varbinary(MAX) NOT NULL
)
GO
```

2. Insert a row into the table. The data within the second column is encrypted using the previously created key. For example:

```sql
INSERT INTO testdb
VALUES ('Employee Name', ENCRYPTBYKEY(KEY_GUID('SymKey'), 'Employee Password'))
```

3. Run the following to decrypt the values of the column:

```sql
SELECT CONVERT(varchar, DECRYPTBYKEY(employee_secret)) employee_secret FROM testdb
```

Asymmetric encryption and decryption examples:

```sql
INSERT INTO testdb
VALUES ('Employee Name', ENCRYPTBYASYMKEY(ASYMKEY_ID('AsymKey'), 'Secret Value'))
```

```sql
SELECT CONVERT(varchar, DECRYPTBYASYMKEY(ASYMKEY_ID('AsymKey'), employee_secret)) employee_secret FROM testdb
```
6. Upgrade nDSOP

This section will perform the migration of the Entrust Database Security Option Pack (nDSOP).

<table>
<thead>
<tr>
<th>From Version</th>
<th>To Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.0</td>
<td>v2.1</td>
</tr>
</tbody>
</table>

6.1. Product configurations

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base OS</td>
<td>Windows Server 2016 Datacenter</td>
</tr>
<tr>
<td>SQL Server</td>
<td>Microsoft 2016 Enterprise with Service Pack 2</td>
</tr>
<tr>
<td>Microsoft SQL Server Management Studio</td>
<td>v18.8</td>
</tr>
</tbody>
</table>

6.2. Supported nShield hardware and software versions

<table>
<thead>
<tr>
<th>Product</th>
<th>Security World</th>
<th>Firmware</th>
<th>Netimage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect XC</td>
<td>12.60.11 with v2 Compatibility Package</td>
<td>12.50.11 (FIPS Certified)</td>
<td>12.60.10</td>
</tr>
</tbody>
</table>

6.3. Procedure

The following procedure will be performed on a Windows Server 2016 with Microsoft SQL Server 2106, and nDSOP v1.0. A database called TestDatabase has been created and encrypted and will be used in this procedure.

2. Backup the database.
3. Run the following query to verify the encryption state.

5. Restart the SQL Server from the Windows MSSMS or services.
6. Wait for 60 seconds after the restart. Then check the database status. Notice **Recovery Pending** next to **TestDatabase**.

7. Un-install nDSOP v1.01 EKM provider using the Windows **Control Panel > Programs > Programs and Features**.

8. Install nDSOP v2.1 EKM provider by mounting the **.iso** file and double-clicking **setup**.

9. Insert the OCS in the HSM slot or TVD. No action is needed if you are using Softcard protection.
10. Retarget the keys by running the `sqlekm_retarget_keys` command:

```
C:\Users\Administrator>nfkinfo -k
Key list - 2 keys
AppName pkcs11  Ident uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-56ac051f249f91e641b065dc12fedca9f9e2419
AppName pkcs11  Ident uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-c88b06f082b2ca29f2a89b9c9352da9f91f8e2afed
C:\Users\Administrator>sqlekm_retarget_keys --all
Found 2 keys to retarget
Retargetted: key_pkcs11_uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-c88b06f082b2ca29f2a89b9c9352da9f91f8e2afed
Retargetted: key_pkcs11_uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-56ac051f249f91e641b065dc12fedca9f9e2419
C:\Users\Administrator>nfkinfo -k
Key list - 4 keys
AppName pkcs11  Ident uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-56ac051f249f91e641b065dc12fedca9f9e2419
AppName pkcs11  Ident uc79dfaf7c3311d22d240a7257e5e760ede89fbc70-c88b06f082b2ca29f2a89b9c9352da9f91f8e2afed
AppName simple  Ident sqlekm-79dfaf7c3311d22d240a7257e5e760ede89fbc70-b1844c56b4e6db1166dcdb64f5d59e4e408c
AppName simple  Ident sqlekm-79dfaf7c3311d22d240a7257e5e760ede89fbc70-fa938ba3e111df225b0e02d37c1233da89b8e2afed
```

11. Open the `C:\ProgramData\nCipher\Key Management Data\local` folder. Move all `pkcs11` keys to another folder. Leave the `simple` keys in the current folder.

```
C:\ProgramData\Cipher\Key Management Data>mkdir local_pcks11_keys
C:\ProgramData\Cipher\Key Management Data>move local\key_pkcs11* local_pcks11_keys\.
C:\ProgramData\Cipher\Key Management Data>nfkinfo -k
Key list - 2 keys
AppName simple  Ident sqlekm-79dfaf7c3311d22d240a7257e5e760ede89fbc70-b1844c56b4e6db1166dcdb64f5d59e4e408c
AppName simple  Ident sqlekm-79dfaf7c3311d22d240a7257e5e760ede89fbc70-fa938ba3e111df225b0e02d37c1233da89b8e2afed
```

12. Set the new provider by running the following script:

```
--ChangeToNewProvider.sql
ALTER CRYPTOGRAPHIC PROVIDER nDSOP
FROM FILE = 'C:\Program Files\nCipher\fast\bin\ncsqlekm.dll';
GO
```

13. Enable the EKM provider. Select **Security > Cryptographic Providers**. Right-click the provider and select **Enable**.
14. Verify the new EKM provider version by running the following script. Notice the `provider_version`.

```sql
SELECT * FROM sys.dm_cryptographic_provider_properties;
```

15. Restart the SQL Server from the Windows MSSMS or services. Wait for 60 seconds after the restart.

16. Check and refresh database status. Notice the **Recovery Pending** message next to the TestDatabase goes away.

17. Verify the encryption state by running the following script. Notice the `encryption_state_desc` shown as **Encrypted**.

```sql
/******* Script for SelectTopNRows command from SSMS *******/
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.certificates AS c ON e.encryptor_thumbprint = c.thumbprint
```