Field Level Encryption Guide in Microsoft SQL Server

This guide is to help developers quickly understand the steps needed to implement field level encryption in SQL Server. Field level encryption is enabled via the *Always Encrypted* feature of SQL Server.

Always Encrypted will encrypt sensitive columns such as medical or credit card information.

Identifying the Crown Jewel

Customer ID	Customer Name	Customer Credit Card Number (Crown Jewel)
123	Jim Halpert	4111111111111111
456	Kevin Malone	3566002020360505
789	Pamela Beasly	371449635398431

The first step is to identity the sensitive data (referred to as Crown Jewels). Some examples are credit card info, medical records etc.

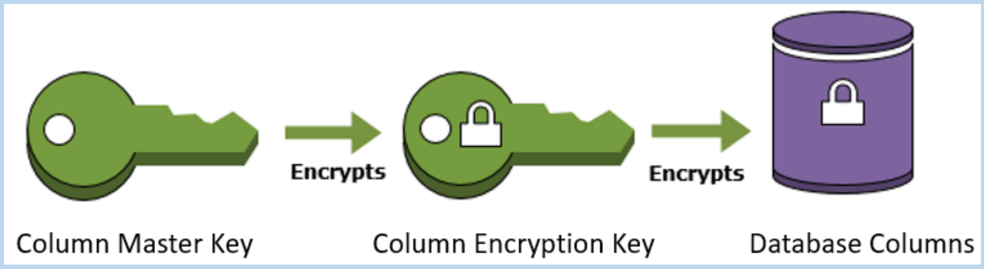
Customer ID	Customer Name	Customer Credit Card Number (Crown Jewel)
123	Jim Halpert	MTIzNC01NjctODkwMQ==
456	Kevin Malone	OTk5OS04ODg4LTc3Nz1==
789	Pamela Beasly	OTg3LTY1NC0zMj4QpR1E=

After enabling Field Level Encryption, only the sensitive column will be encrypted. To access the plaintext value, access to the appropriate keys is needed.

Creating Keys

There are 2 keys needed:

1. Column Master Key (CMK) - Encrypts the Column Encryption Keys and is stored externally to the SQL Server.
2. Column Encryption Key (CEK) - Encrypts the actual data columns and is encrypted by the CMK.



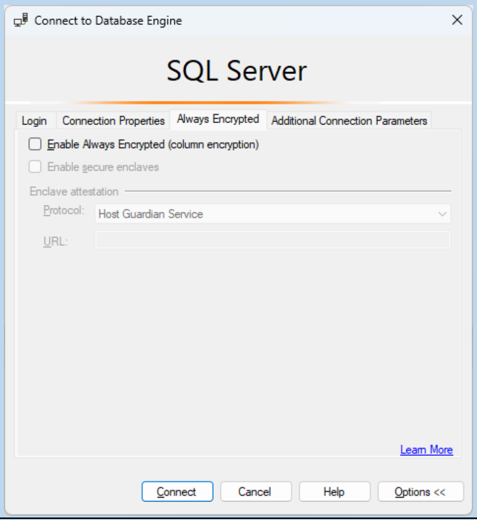
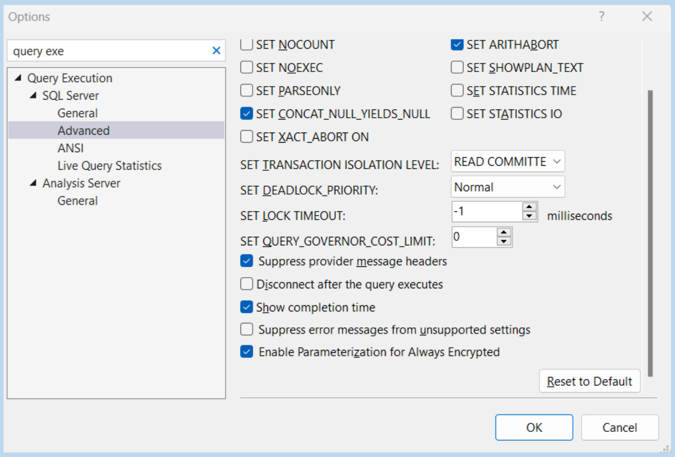
Requesting Keys

1. Identify column that requires encryption.
2. Choose between *Deterministic* or *Randomized* Encryption type.
   a. *Deterministic* encryption supports queries, such as point lookup searches that involve equality comparisons on encrypted columns.
   b. *Randomized* encryption doesn't support any computations on encrypted columns.
3. Raise a ServiceNow request to DBA for enabling encryption on this column.
4. DBA will generate the keys and pass keys to Application Developers
5. Test connectivity and decryption of encrypted column.

Querying Encrypted Field

To query the encrypted data, we have to ensure the client driver has access to the keys.

1. If using SSMS as a client, enable the Always Encrypted options as below.

Enable Always Encrypted in DB connection	Enable Always Encrypted in SSMS options
	

Connect to Database Engine

SQL Server

Login | Connection Properties | Always Encrypted | Additional Connection Parameters

☐ Enable Always Encrypted (column encryption)
☐ Enable secure enclaves

Enclave attestation
Protocol: Host Guardian Service
URL:

Learn More

Connect | Cancel | Help | Options <<

Options

query exe

- Query Execution
  - SQL Server
    - General
    - Advanced
    - ANSI
    - Live Query Statistics
  - Analysis Server
    - General

☐ SET NOCOUNT
☑ SET ARITHABORT
☐ SET NOEXEC
☐ SET SHOWPLAN_TEXT
☐ SET PARSEONLY
☐ SET STATISTICS TIME
☑ SET CONCAT_NULL_YIELDS_NULL
☐ SET STATISTICS IO
☐ SET XACT_ABORT ON
SET TRANSACTION ISOLATION LEVEL: READ COMMITTE
SET DEADLOCK_PRIORITY: Normal
SET LOCK TIMEOUT: -1 milliseconds
SET QUERY_GOVERNOR_COST_LIMIT: 0
☑ Suppress provider message headers
☐ Disconnect after the query executes
☑ Show completion time
☐ Suppress error messages from unsupported settings
☑ Enable Parameterization for Always Encrypted

Reset to Default

OK | Cancel

2. If using other clients (eg code), ensure the client has access to the keys.
   a. The access to the keys can be configured in the connection string to the SQL Server. Below is an example with the ODBC driver, where the keys are stored in a Windows Certificate Store:

```
“DRIVER=ODBC Driver 18 for SQL
Server;SERVER=localhost,1433;Encrypt=yes;Trusted_Connection=Yes;DATABASE=master;
ColumnEncryption=Enabled;KeyStoreAuthentication=WindowsCertificateStore;”
```

   b. For more details (such as using other DB drivers and programming languages), please refer to the Developer Guide.

Useful References

Developer Guide for SQL Server's Always Encrypted (Microsoft), code examples, database drivers - https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-client-development?view=sql-server-ver16

SQL Server Always Encrypted Docs (Microsoft) - https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine?view=sql-server-ver16

Microsoft documentation on creating a column master key (scroll to bottom) - https://learn.microsoft.com/en-us/sql/t-sql/statements/create-column-master-key-transact-sql?view=sql-server-ver16#:~:text=creating%20a%20column

Microsoft documentation on creating a column encryption key (scroll to bottom) - https://learn.microsoft.com/en-us/sql/t-sql/statements/create-column-encryption-key-transact-sql?view=sql-server-ver16#:~:text=Examples

Setting up Always Encrypted on AWS RDS (AWS) - https://aws.amazon.com/blogs/database/set-up-always-encrypted-with-amazon-rds-for-sql-server/

Setting up Always Encrypted with Secure Enclaves on AWS EC2 (AWS) - https://aws.amazon.com/blogs/modernizing-with-aws/sql-server-always-encrypted-with-secure-enclaves/

Using HSM with Always Encrypted (Microsoft) - https://techcommunity.microsoft.com/t5/sql-server-blog/using-hardware-security-modules-with-always-encrypted/ba-p/384575

Key Stores in Always Encrypted (Microsoft) - https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/create-and-store-column-master-keys-always-encrypted?view=sql-server-ver16

Key Rotation in Always Encrypted (Microsoft) - https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/rotate-always-encrypted-keys-using-ssms?view=sql-server-ver16

Proof-of-Concept (POC)

In this POC, a sample table was created with column encryption enabled for the “*CustomerCreditCard*” column.

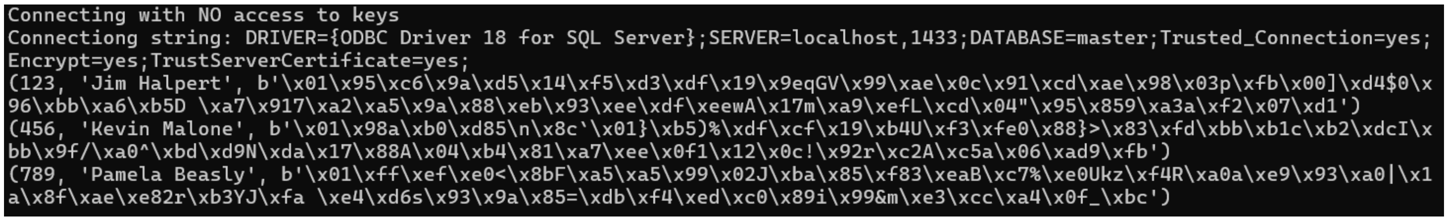
Table Schema

```
CREATE TABLE CustomerInfo (

    CustomerID INT NOT NULL PRIMARY KEY,

    CustomerName NVARCHAR(50) NOT NULL,

    CustomerCreditCard INT ENCRYPTED WITH (

        ENCRYPTION_TYPE = Deterministic,

        ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',

        COLUMN_ENCRYPTION_KEY = CEK_1

    ) NOT NULL

)
```

Table Contents

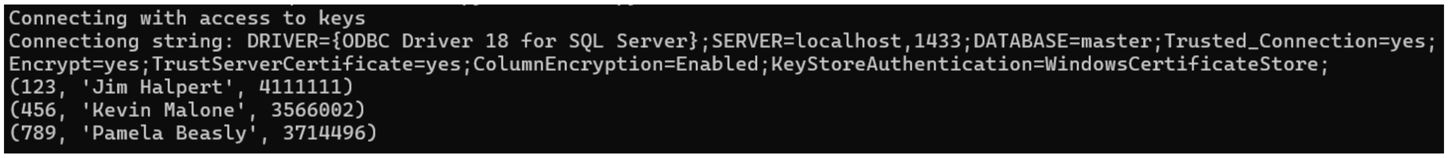
CustomerID	CustomerName	CustomerCreditCard
123	Jim Halpert	4111111
456	Kevin Malone	3566002
789	Pamela Beasly	3714496

When a connection was made, without specifying the keystore authentication options, the data for “*CustomerCreditCard*” is encrypted:



```
Connecting with NO access to keys
Connectiong string: DRIVER={ODBC Driver 18 for SQL Server};SERVER=localhost,1433;DATABASE=master;Trusted_Connection=yes;
Encrypt=yes;TrustServerCertificate=yes;
(123, 'Jim Halpert', b'\x01\x95\xc6\x9a\xd5\x14\xf5\xd3\xdf\x19\x9eqGV\x99\xae\x0c\x91\xcd\xae\x98\x03p\xfb\x00]\xd4$0\x
96\xbb\xa6\xb5D \xa7\x917\xa2\xa5\x9a\x88\xeb\x93\xee\xdf\xeewA\x17m\xa9\xefL\xcd\x04"\x95\x859\xa3a\xf2\x07\xd1')
(456, 'Kevin Malone', b'\x01\x98a\xb0\xd85\n\x8c`\x01}\xb5)%\xdf\xcf\x19\xb4U\xf3\xfe0\x88}>\x83\xfd\xbb\xb1c\xb2\xdcI\x
bb\x9f/\xa0^\xbd\xd9N\xda\x17\x88A\x04\xb4\x81\xa7\xee\x0f1\x12\x0c!\x92r\xc2A\xc5a\x06\xad9\xfb')
(789, 'Pamela Beasly', b'\x01\xff\xef\xe0<\x8bF\xa5\xa5\x99\x02J\xba\x85\xf83\xeaB\xc7%\xe0Ukz\xf4R\xa0a\xe9\x93\xa0|\x1
a\x8f\xae\xe82r\xb3YJ\xfa \xe4\xd6s\x93\x9a\x85=\xdb\xf4\xed\xc0\x89i\x99&m\xe3\xcc\xa4\x0f_\xbc')
```

When the key store authentication options are specified and the client has access to the keys, the “*CustomerCreditCard*” data is visible:



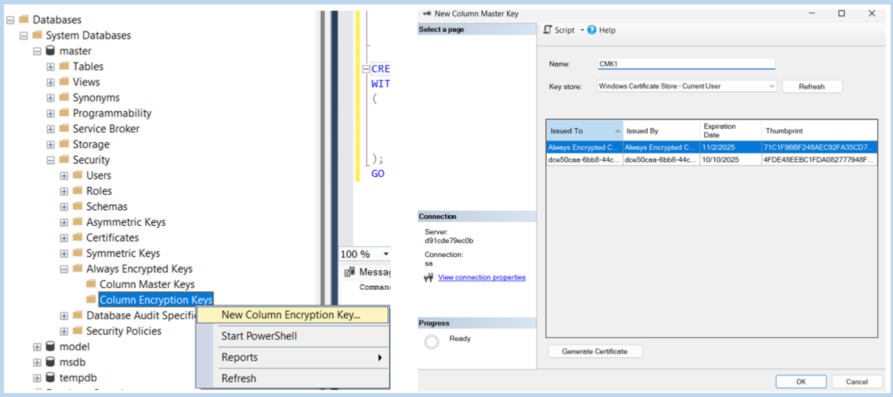
```
Connecting with access to keys
Connectiong string: DRIVER={ODBC Driver 18 for SQL Server};SERVER=localhost,1433;DATABASE=master;Trusted_Connection=yes;
Encrypt=yes;TrustServerCertificate=yes;ColumnEncryption=Enabled;KeyStoreAuthentication=WindowsCertificateStore;
(123, 'Jim Halpert', 4111111)
(456, 'Kevin Malone', 3566002)
(789, 'Pamela Beasly', 3714496)
```

Generating Column Master Key (CMK) – For DBA only

1. There are multiple options available for the key store. In this guide, we will use the Windows Certificate Store.
2. Run the command to create the CMK. In the below example, a CMK is created at the Certificate Store.

```
CREATE COLUMN MASTER KEY MyCMK

WITH (

     KEY_STORE_PROVIDER_NAME = N'MSSQL_CERTIFICATE_STORE',

     KEY_PATH = 'Current User/Personal/f2260f28d909d21c642a3d8e0b45a830e79a1420'

   );
```

3. Alternatively, use SSMS to generate the CMK via the GUI



Name: CMK1

Key store: Windows Certificate Store - Current User

Issued To	Issued By	Expiration Date	Thumbprint
Always Encrypted C...	Always Encrypted C...	11/2/2025	71C1F9BBF248AEC92FA35CD7...
dce50caa-6bb8-44c...	dce50caa-6bb8-44c...	10/10/2025	4FDE48EEBC1FDA082777948F...

Connection

Server: d91cde79ec0b

Connection: sa

View connection properties

Progress

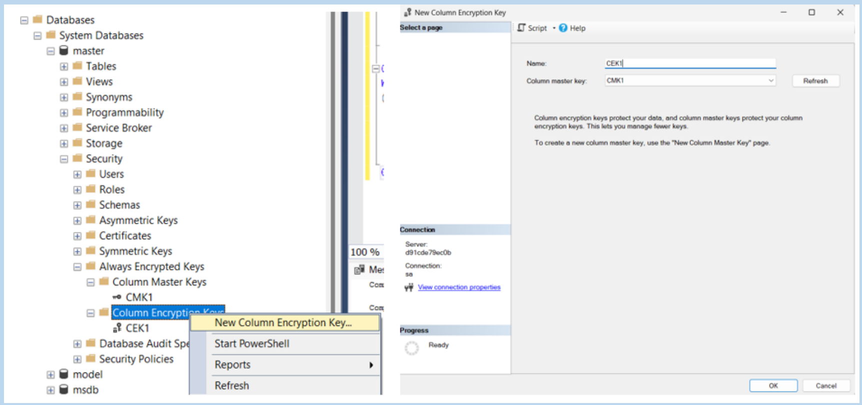
Ready

Generating Column Encryption Key (CEK) – For DBA only

1. Run the command to create the CMK. In the below example, a CMK is created at the Certificate Store.

```
CREATE COLUMN ENCRYPTION KEY cek1

WITH (

     COLUMN_MASTER_KEY = CMK1

     ALGORITHM = ‘RSA_OAEP’,

     ENCRYPTED_VALUE={INSERT_YOUR_ENCRYPTED_VALUE}

   );
```

2. Alternatively, use SSMS to generate the CEK via the GUI



Name: CEK1

Column master key: CMK1

Column encryption keys protect your data, and column master keys protect your column encryption keys. This lets you manage fewer keys.

To create a new column master key, use the "New Column Master Key" page.

Connection

Server: d91cde79ec0b

Connection: sa

View connection properties

Progress

Ready

Enabling Encryption – For DBA Only

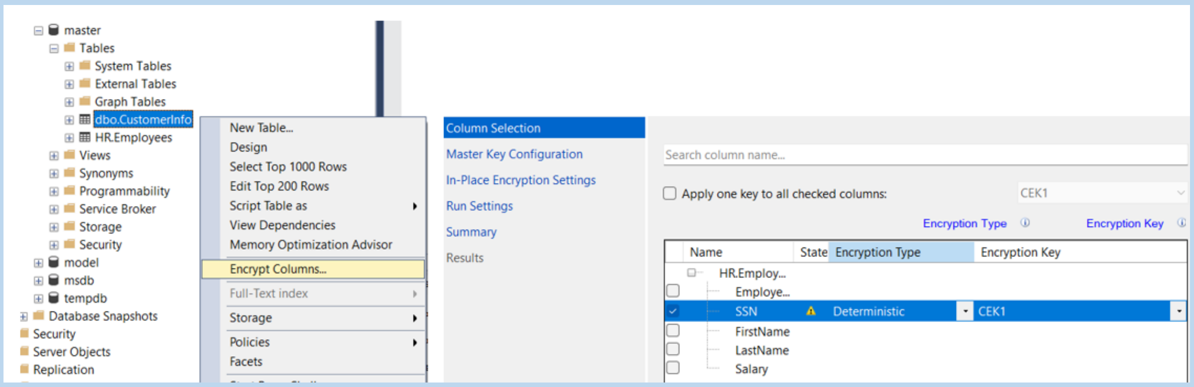
1. Choose between *Deterministic* or *Randomized* Encryption type.
    a. *Deterministic* encryption supports queries, such as point lookup searches that involve equality comparisons on encrypted columns.
    b. *Randomized* encryption doesn't support any computations on encrypted columns.

2. Create the table in SQL Server and specify the column that requires encryption.

    In the example below, the column ‘CustomerCreditCard’ will be encrypted using the CEK.

```
CREATE TABLE CustomerInfo (

    CustomerID INT NOT NULL PRIMARY KEY,

    CustomerName NVARCHAR(50) NOT NULL,

    CustomerCreditCard VARBINARY(256) ENCRYPTED WITH (

        ENCRYPTION_TYPE = Deterministic,

        ALGORITHM = 'AEAD_AES_256_CBC_HMAC_SHA_256',

        COLUMN_ENCRYPTION_KEY = CEK1

    ) NOT NULL
```

3. Alternatively, for an existing column, select it via SSMS and select the encryption using the CEK.



Apply one key to all checked columns: CEK1

Name	State	Encryption Type	Encryption Key
HR.Employ...			
Employe...			
✓ SSN	⚠	Deterministic	CEK1
FirstName			
LastName			
Salary			