Symmetric Key Encryption

A symmetric key can be created a number of ways – here I’m outlining how to make certificate based symmetric keys and password-based symmetric keys.

*General Outline/Overview*

-- SETUP - USE RELEVANT DB

USE cooldatabase;

-- 1. Create master key

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'super\_pass';

-- 2. Verify mkey exists

SELECT \* FROM sys.symmetric\_keys

-- 3. Create Certificate

CREATE CERTIFICATE ATestCertificate WITH SUBJECT = 'A Test Certificate';

-- 4. Verify certificate exists

SELECT \* FROM sys.certificates

-- 5. Create Symmetric Key [Certificate-based]

CREATE SYMMETRIC KEY supersymmetrickey

WITH ALGORITHM = AES\_256

ENCRYPTION BY CERTIFICATE ATestCertificate;

GO

-- 5. Create Symmetric Key [Password-based]

CREATE SYMMETRIC KEY supersymmetrickeywpass

WITH ALGORITHM = AES\_256

ENCRYPTION BY PASSWORD='mycoolpassword';

-- 6. Verify new Symmetric Key exists

SELECT \* FROM sys.symmetric\_keys

-- 7. Save something into DB by encrypting it [Using certificate-based symmetric key]

OPEN SYMMETRIC KEY supersymmetrickey

DECRYPTION BY CERTIFICATE ATestCertificate

INSERT key\_testing

VALUES(

EncryptByKey(key\_GUID('supersymmetrickey'), 'pleaseencryptme' )

)

CLOSE SYMMETRIC KEY supersymmetrickey

-- 7. Save something into DB by encrypting it [Using password-based symmetric key]

OPEN SYMMETRIC KEY supersymmetrickeywpass

DECRYPTION BY PASSWORD='mycoolpassword'

INSERT key\_testing

VALUES(

EncryptByKey(key\_GUID('supersymmetrickeywpass'), 'pleaseencryptmemoar' )

)

CLOSE SYMMETRIC KEY supersymmetrickeywpass

-- 8. Decrypt an encrypted string [Using certificate-based symmetric key]

(using varchar max but if you have an initial string length pre-defined on the plaintext to be encrypted that you can use the minimum - for example char(9))

OPEN SYMMETRIC KEY supersymmetrickey

DECRYPTION BY CERTIFICATE ATestCertificate

SELECT CONVERT(VARCHAR(MAX),DECRYPTBYKEY(encpass)) FROM key\_testing WHERE id='3'

CLOSE SYMMETRIC KEY supersymmetrickey

-- 8. Decrypt an encrypted string [Using password-based symmetric key]

OPEN SYMMETRIC KEY supersymmetrickeywpass

DECRYPTION BY PASSWORD='mycoolpassword'

SELECT CONVERT(VARCHAR(MAX),DECRYPTBYKEY(encpass))

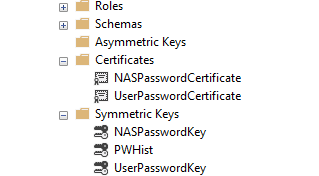
FROM key\_testing WHERE id='5'

CLOSE SYMMETRIC KEY supersymmetrickeywpass

*Create Symmetric Key*

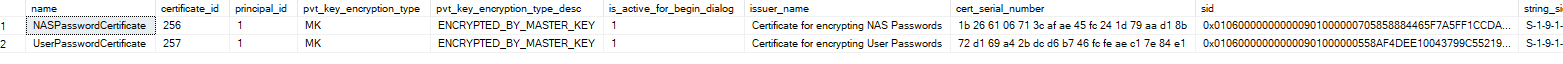
1. **View any certificates/checking if exists**

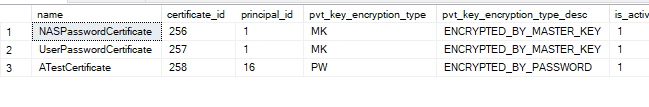
Keys will be saved, not under databases schema, but under Security > Certificates



You can also view using  sys.certificates

SELECT \* FROM sys.certificates





1. **Create Certificate**

<https://docs.microsoft.com/en-us/sql/t-sql/statements/create-certificate-transact-sql?view=sql-server-ver15>

<https://sqlity.net/en/2375/create-certificate/>

A certificate can only be protected by either a password or the database master key. If a password is specified during creation, that same password is required every time the private key needs to be accessed.

*Using Masterkey*

Masterkeys are database-specific and there can only be one per database

**Create masterkey**

<https://sqlity.net/en/2373/create-database-master-key/>

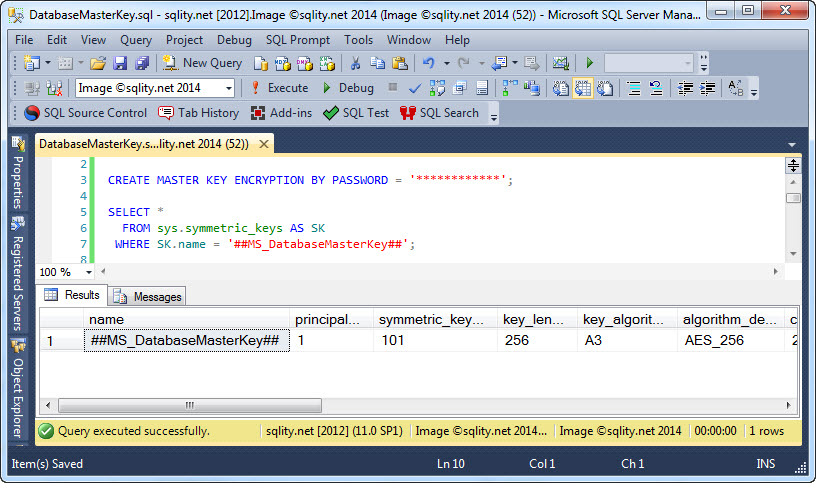
CREATE MASTER KEY ENCRYPTION BY PASSWORD = '\*\*\*\*\*\*\*\*\*\*\*\*';

View all keys…

 SELECT \* FROM sys.symmetric\_keys

There will only be one master key (per db) and it will have name “##MS\_DatabaseMasterKey##”

SELECT \*FROM sys.symmetric\_keys AS SK  
WHERE SK.name = '##MS\_DatabaseMasterKey##';



**Create Cert**

The CREATE CERTIFICATE statement in its simplest form has the following format:

CREATE CERTIFICATE ATestCertificate WITH SUBJECT = 'A Test Certificate';

If that doesn’t work, make sure the master key is opened…

USE<database\_name>;

OPEN MASTER KEY DECRYPTION BY PASSWORD = ‘\*\*\*\*\*’;

GO

CLOSE MASTER KEY;

GO

*Using Password*

[sql]

CREATE CERTIFICATE ATestCertificate  
ENCRYPTION BY PASSWORD = '\*\*\*\*\*\*\*\*\*\*'  
WITH SUBJECT = 'A Test Certificate';  
[/sql]

*Note: Above we are creating a self-signed certificate (not using external authority) using a password we provide (instead of the default – which is a database masterkey).*

ATestCertificate is the name for the certificate. . Subject is a field defined by the X.509 standard and certificates created by this command follow that standard. In SQL Server, the subject can be up to 128 characters long. This is a SQL Server restriction, the X.509 standard itself allows for longer subjects.

The private key for SQL Server generated certificates is always a 1024-bit key for the RSA encryption algorithm. SQL Server supports imported certificate key lengths from 384 to 4096 bits but can only generate this one key length for certificates.

1. **Create Symmetric Key and handle encryption/decryption**

<https://docs.microsoft.com/en-us/sql/t-sql/statements/create-symmetric-key-transact-sql?view=sql-server-ver15>

*[Method 1] Create Symmetric Key (encrypted by certificate)*

***Abstract***

CREATE SYMMETRIC KEY <cool\_key\_name>

WITH ALGORITHM = AES\_256

ENCRYPTION BY CERTIFICATE <certificate\_name>;

GO

The following example creates a symmetric key called **“supersymmetrickey”** by using the AES 256 algorithm, and then encrypts the new key with certificate named **“ATestCertificate”** (which was created using a masterkey in this example).

***Concrete Example***

CREATE SYMMETRIC KEY supersymmetrickey

WITH ALGORITHM = AES\_256

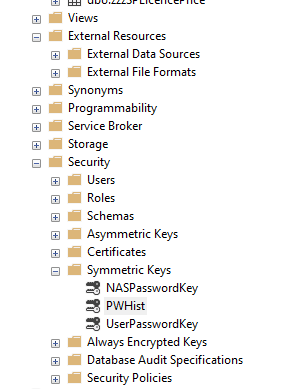
ENCRYPTION BY CERTIFICATE ATestCertificate;

GO

Verify key exists

SELECT \* FROM sys.symmetric\_keys

Keys will be saved, not under databases schema, but under Security > Symmetric schema



**Encrypt data (by symmetric key generated from certificate)**

***Abstract***

OPEN SYMMETRIC KEY <symmetric key name>

DECRYPTION BY CERTIFICATE <cert that was used to create sym key>

INSERT INTO <some table>

VALUES(

EncryptByKey(key\_GUID(‘<symmetric key name >'), 'pleaseencryptme' )

)

CLOSE SYMMETRIC KEY < symmetric key name >

***Concrete example***

OPEN SYMMETRIC KEY supersymmetrickey

DECRYPTION BY CERTIFICATE ATestCertificate

INSERT INTO key\_testing

VALUES(

EncryptByKey(key\_GUID('supersymmetrickey'), 'pleaseencryptme' )

)

CLOSE SYMMETRIC KEY supersymmetrickey

**Decrypt data (by symmetric key generated from certificate)**

**Abstract**

OPEN SYMMETRIC KEY <symmetric\_key\_name>

DECRYPTION BY PASSWORD='<password>'

SELECT CONVERT(VARCHAR(MAX),DECRYPTBYKEY(encpass))

FROM key\_testing WHERE id='5'

CLOSE SYMMETRIC KEY <symmetric\_key\_name>

**Concrete example**

OPEN SYMMETRIC KEY supersymmetrickeywpass

DECRYPTION BY PASSWORD='mycoolpassword'

SELECT CONVERT(VARCHAR(MAX),DECRYPTBYKEY(encpass))

FROM key\_testing WHERE id='5'

CLOSE SYMMETRIC KEY supersymmetrickeywpass

*[Method] Create Symmetric key (Encrypted by password)*

***Abstract***

CREATE SYMMETRIC KEY <cool\_key\_name>

WITH ALGORITHM = AES\_256

ENCRYPTION BY PASSWORD='<password>';

***Concrete Example***

The following example creates a symmetric key called **“supersymmetrickey”** by using the AES 256 algorithm, and then encrypts the new key with certificate named **“ATestCertificate”** (which was created using a masterkey in this example).

CREATE SYMMETRIC KEY supersymmetrickeywpass

WITH ALGORITHM = AES\_256

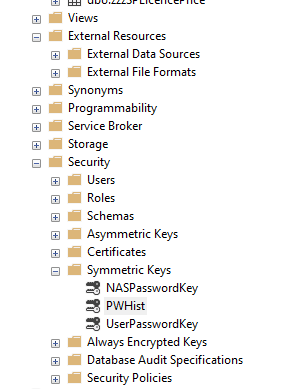
ENCRYPTION BY PASSWORD='mycoolpassword';

GO

Verify key exists

SELECT \* FROM sys.symmetric\_keys

Keys will be saved, not under databases schema, but under Security > Symmetric schema



**Encrypt data (using symmetric key generated by password)**

***Abstract***

OPEN SYMMETRIC KEY <symmetric\_key\_name>

DECRYPTION BY PASSWORD='mycoolpassword'

INSERT INTO key\_testing

VALUES(

EncryptByKey(key\_GUID(<symmetric\_key\_name>), 'pleaseencryptmemoar' )

)

CLOSE SYMMETRIC KEY supersymmetrickeywpass

***Concrete Example***

OPEN SYMMETRIC KEY supersymmetrickeywpass

DECRYPTION BY PASSWORD='mycoolpassword'

INSERT INTO key\_testing

VALUES(

EncryptByKey(key\_GUID('supersymmetrickeywpass'), 'pleaseencryptmemoar' )

)

CLOSE SYMMETRIC KEY supersymmetrickeywpass

**Decrypt data (using symmetric key generated by password)**

-- 8. Decrypt an encrypted string (using varchar max but if you have an initial string length pre-defined on the plaintext to be encrypted that you can use the minimum - for example char(9))

OPEN SYMMETRIC KEY PWHist

DECRYPTION BY PASSWORD='r4pkkdSCGdWdEV4F3SHFrBTygZr6DJKX8Zre4RYaUyAuYqPuTFayquVVRDcT7PVB'

SELECT CONVERT(VARCHAR(MAX),DECRYPTBYKEY(encpass))

FROM key\_testing WHERE id='1'

CLOSE SYMMETRIC KEY PWHist