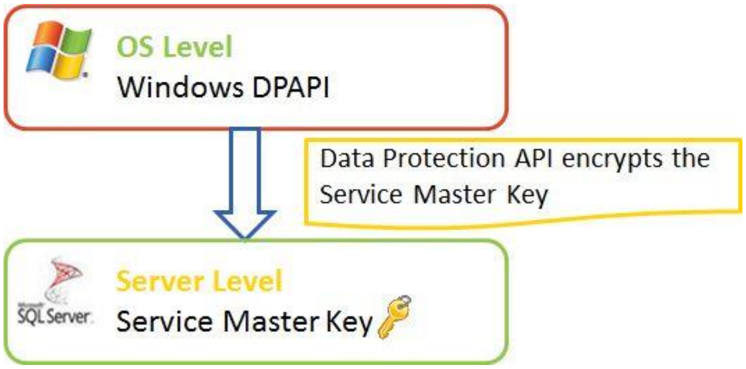
Encrypt and Decrypt in SQL Server

https://www.c-sharpcorner.com/UploadFile/db2972/encrypt-and-decrypt-in-sql-server-part-1/

Introduction

In this article, we will discuss Service Master Key, backup, restore and alter Service Master Key.



Service Master Key

The Service Master Key is created at the time of installation of SQL Server. There is only one Service Master Key per SQL instance. The Windows data protection API uses the SQL Server service account credentials to encrypt the Service Master Key. And the Service Master Key secures all other keys on the server.

Backup Service Master Key

Since there can be only one Service Master Key per instance, it is advisable to take a backup of this key.

```
BACKUP SERVICE MASTER KEY TO FILE= 'D:\SQLServer2008R2.SMK'
ENCRYPTION BY PASSWORD = '@k$h@yPatel'
GO
```

If you execute the SQL statement above, the following error might be thrown:

Cannot write into file "D:\SQLServer2008R2.SMK". Verify that you have write permissions, that the file path is valid, and that the file does not already exist.

It is a SQL Server service account permission issue. Rather than granting permission to the account, we can take a backup from the default path. So for that remove the full path and just pass in the file name.

```
BACKUP SERVICE MASTER KEY TO FILE= 'SQLServer2008R2.SMK'
ENCRYPTION BY PASSWORD = '@k$h@yPatel'
GO
```

Now you can find the "SQLServer2008R2.SMK" file on the following path:

C:\Program Files\Microsoft SQL Server\MSSQL10_50.SQLXPR2008\MSSQL\DATA

Note: Take a backup of the Service Master Key and store it in a secure location immediately after installing SQL Server.

Restore Service Master Key

We can restore this file whenever it is required. The syntax is as follows:

```
RESTORE SERVICE MASTER KEY FROM FILE= 'SQLServer2008R2.SMK'
DECRYPTION BY PASSWORD = '@k$h@yPatel'
GO
```

Note: Password should be the password previously used to encrypt the backup.

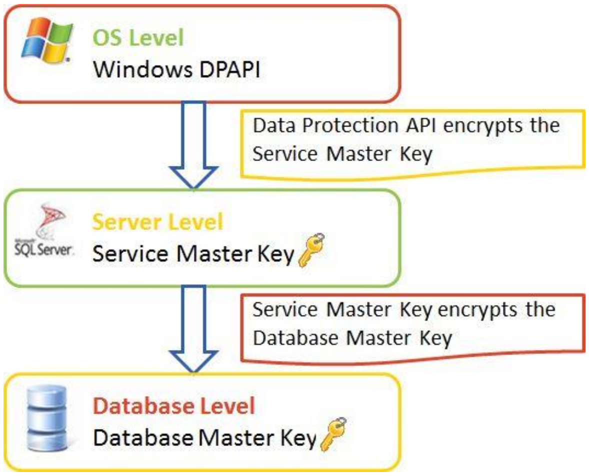
Alter Service Master Key

```
ALTER SERVICE MASTER KEY REGENERATE;
```

Note: It is advisable to regenerate the SMK whenever we make any changes in the service account.

https://www.c-sharpcorner.com/UploadFile/db2972/encrypt-and-decrypt-in-sql-server-part-2/

In this article, we will discuss Database Master Key.


OS Level
Windows DPAPI
Data Protection API encrypts the Service Master Key
SQL Server
Server Level
Service Master Key
Service Master Key encrypts the Database Master Key
Database Level
Database Master Key

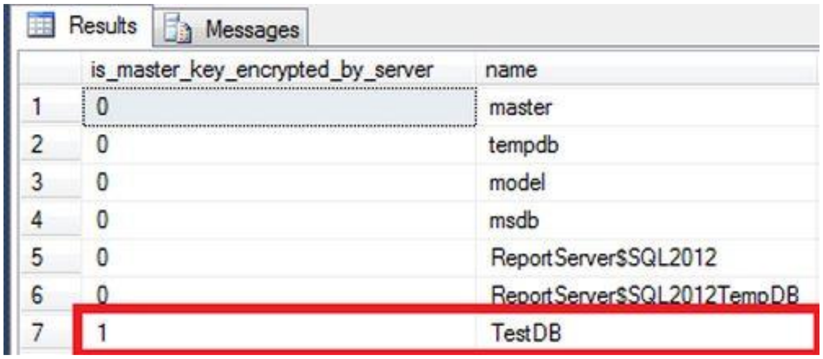
In order to create a Database Master Key, first create the database "TestDB" in SQL Server and execute the following commands.

```
USE TestDB
GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD='@k$h@yPatel'
```

- The database master key is a symmetric key used to protect the private keys of certificates and asymmetric keys that are present in the database.
- To enable the automatic decryption of the master key, a copy of the key is encrypted by using the service master key and stored in the database and in the master.
- Now let's see how to check whether it is encrypted by the service master key or not.
- For that execute the following statement.

```
SELECT is_master_key_encrypted_by_server,* FROM sys.databases
```



	is_master_key_encrypted_by_server	name
1	0	master
2	0	tempdb
3	0	model
4	0	msdb
5	0	ReportServer$SQL2012
6	0	ReportServer$SQL2012TempDB
7	1	TestDB

The Is_master_key_encrypted_by_server column value of the TestDB database specifies that the master key is encrypted by the service master key.

We can change this setting by altering the master key.

Alter Database Master Key

```
ALTER MASTER KEY DROP ENCRYPTION BY SERVICE MASTER KEY
```

We can reset the above setting by executing the following command. Since we drop encryption by the service master key, we must explicitly open the database master key with a password.

```
OPEN MASTER KEY DECRYPTION BY PASSWORD = '@k$h@yPatel'
ALTER MASTER KEY ADD ENCRYPTION BY SERVICE MASTER KEY
```

Back up Database Master Key

```
USE TestDB
GO
BACKUP MASTER KEY TO FILE = 'D:\TestDB.DMK'
ENCRYPTION BY PASSWORD='@k$h@yPatel'
```

- The database master key is used to encrypt other keys and certificates.
- If this key is deleted or corrupted then it is very difficult to decrypt those keys and the data that are encrypted using those keys may be lost, so it is advisable to take a backup of the database master key.
- We can restore the database master key by executing the following statement.
- If there is no master key available in the database then the following statement creates a new master key, but the only difference is that it is not encrypted automatically with the service master key.

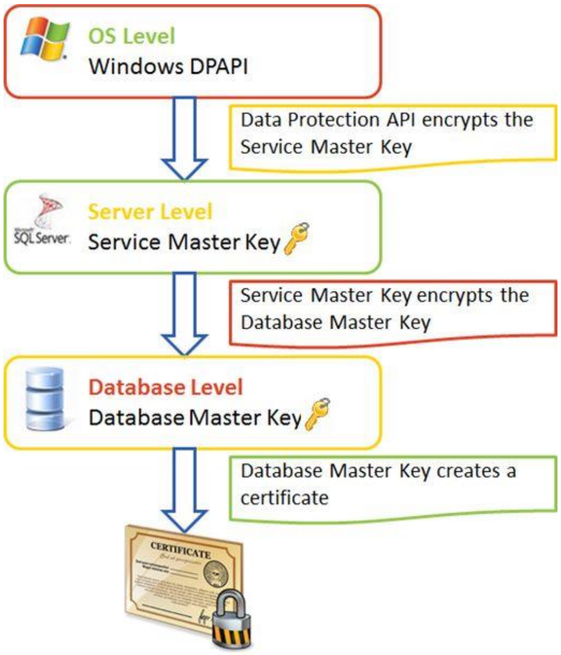
Restore Database Master Key

```
RESTORE MASTER KEY FROM FILE = 'D:\TestDB.DMK'
DECRYPTION BY PASSWORD = '@k$h@yPatel'
ENCRYPTION BY PASSWORD='@k$h@yPatelC#'
```

"Encryption by password" specifies the password used to encrypt the database master key after it has been loaded into the database.

https://www.c-sharpcorner.com/UploadFile/db2972/encrypt-and-decrypt-in-sql-server-part-3/

In this article, we will generate a certificate and use this certificate to encrypt and decrypt the string.


OS Level
Windows DPAPI
Data Protection API encrypts the Service Master Key
SQL Server
Server Level
Service Master Key
Service Master Key encrypts the Database Master Key
Database Level
Database Master Key
Database Master Key creates a certificate
CERTIFICATE

Create Certificate

```
CREATE CERTIFICATE TESTCERT
ENCRYPTION BY PASSWORD ='@k$h@yPatel'
WITH SUBJECT ='TEST CERTIFICATE',
START_DATE='01/10/2013',
EXPIRY_DATE='01/10/2014'
```

If start_date is not provided then the current date will be startdate and if expiry_date is not provided then after one year, startdate will be considered.

Backup Certificate

```
BACKUP CERTIFICATE TESTCERT
TO FILE = 'd:\TestCert.CER'
WITH PRIVATE KEY
(
    FILE='d:\TestCert.PVK',
    ENCRYPTION BY PASSWORD='@k$h@yPatel',
    DECRYPTION BY PASSWORD='@k$h@yPatel'
)
GO
```

Restore Certificate

```
DROP CERTIFICATE TESTCERT
CREATE CERTIFICATE TESTCERT
FROM FILE='D:\TestCert.CER'
```

Encrypt & Decrypt

```sql
DECLARE @Text VARCHAR(50)
DECLARE @EncryptedText VARBINARY(128)
DECLARE @DecryptedText VARCHAR(MAX)
SET @Text = 'I am Akshay Patel'
SET @EncryptedText=ENCRYPTBYCERT(CERT_ID('TESTCERT'),@Text)
SET @DecryptedText=DECRYPTBYCERT(CERT_ID('TESTCERT'),@EncryptedText,N'@k$h@yPatel')
SELECT @Text AS 'TextToEncrypt',@EncryptedText AS 'EncryptedText',@DecryptedText as 'DecryptedText'
```

https://www.c-sharpcorner.com/UploadFile/db2972/encrypt-and-decrypt-in-sql-server-part-4/

Introduction

In this article, we will encrypt plain text and decrypt encrypted text using an asymmetric key.

An asymmetric key is a combination of public key and private key. A public key is used to encrypt the data and a private key to decrypt the data.

Create Asymmetric Key

```
CREATE ASYMMETRIC KEY AsymKey
WITH ALGORITHM = RSA_1024
go
```

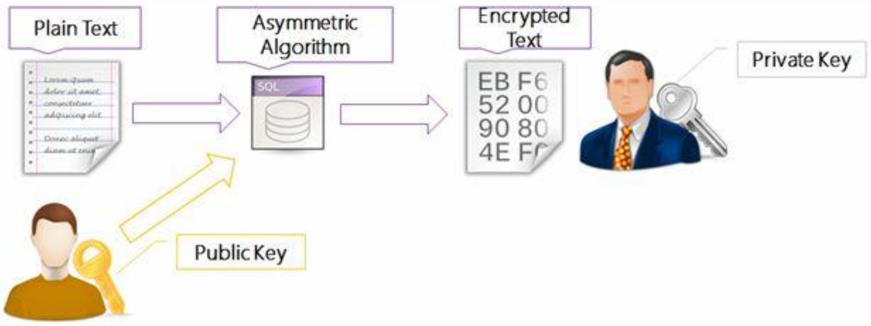
It is mandatory to create a or open a database key to execute the statement above successfully otherwise you will get the following error message:

Msg 15581, Level 16, State 6, Line 1

Please create a master key in the database or open the master key in the session before performing this operation.

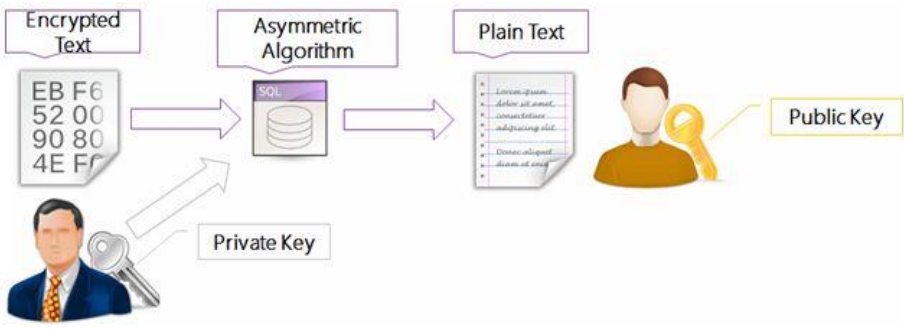
Encrypt

```
DECLARE @EncryptedText VARBINARY(128)
SET @EncryptedText=ENCRYPTBYASYMKEY(ASYMKEY_ID(N'AsymKey'),@Text)
```

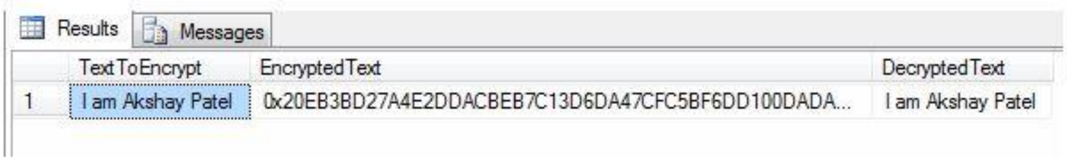


Decrypt

```
DECLARE @DecryptedText VARCHAR(MAX)
SET @DecryptedText=DECRYPTBYASYMKEY (ASYMKEY_ID(N'AsymKey'),@TextEnrypt)
```



```
SELECT @Text AS 'TextToEncrypt',@TextEnrypt AS 'EncryptedText',@TextDecrypt AS 'DecryptedText'
GO
```



	TextToEncrypt	EncryptedText	DecryptedText
1	I am Akshay Patel	0x20EB3BD27A4E2DDACBEB7C13D6DA47CFC5BF6DD100DADA...	I am Akshay Patel

Drop Asymmetric Key

```
DROP ASYMMETRIC KEY AsymKey
```

https://www.c-sharpcorner.com/UploadFile/db2972/encrypt-and-decrypt-in-sql-server-part-5/

In this article, we will create a symmetric key and encrypt and decrypt a string using this key.

Create Symmetric Key

```
CREATE SYMMETRIC KEY TestSymKey
WITH ALGORITHM =AES_256
ENCRYPTION BY CERTIFICATE TestCert
GO
```

Open Symmetric Key

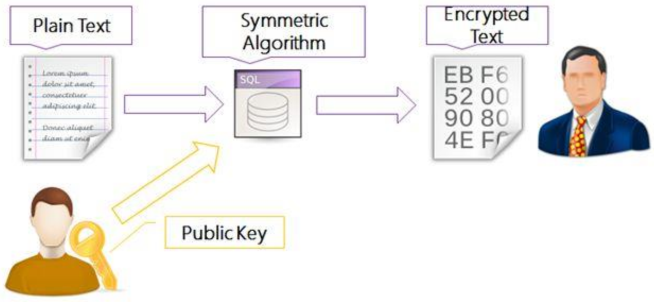
Once we create the symmetric key, we need to open it before use.

```
OPEN SYMMETRIC KEY TestSymKey
DECRYPTION BY CERTIFICATE TestCert
WITH PASSWORD ='@k$h@yPatel'
GO
```

Encrypt

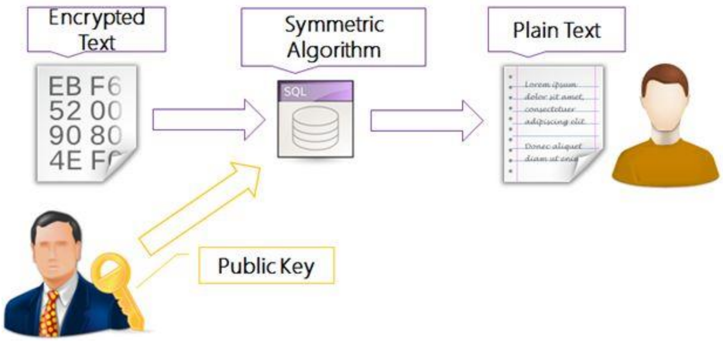
```
DECLARE @Text VARCHAR(MAX)
SET @Text = 'I am Akshay Patel'

DECLARE @EncryptedText VARBINARY(128)
SET @EncryptedText = (SELECT ENCRYPTBYKEY(KEY_GUID(N'TestSymKey'),@Text))
```

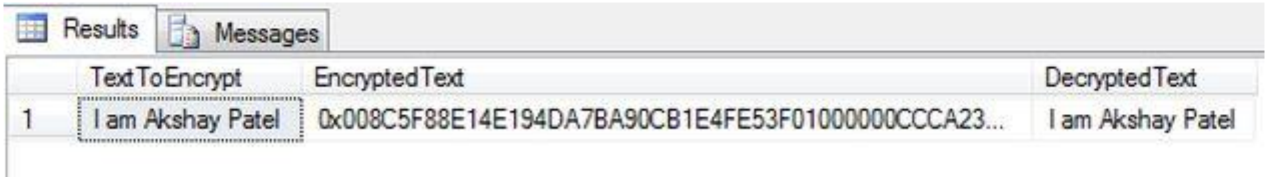

Plain Text
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec aliquet diam at enim
Symmetric Algorithm
SQL
Encrypted Text
EB F6 52 00 90 80 4E FC
Public Key

Decrypt

```
DECLARE @DecryptedText VARCHAR(MAX)
SET @DecryptedText = (SELECT CONVERT(VARCHAR(MAX),DECRYPTBYKEY(@EncryptedText)))
```


Encrypted Text
EB F6
52 00
90 80
4E FF
Symmetric Algorithm
SQL
Plain Text
Lorem ipsum dolor sit amet, consectetuer adipiscing elit
Donec aliquet diam ut eros
Public Key

```
SELECT @Text AS 'TextToEncrypt',@TextEnrypt AS 'EncryptedText',@TextDecrypt AS 'DecryptedText'
GO
```



	TextToEncrypt	EncryptedText	DecryptedText
1	I am Akshay Patel	0x008C5F88E14E194DA7BA90CB1E4FE53F01000000CCCA23...	I am Akshay Patel

Drop Asymmetric Key

```
DROP SYMMETRIC KEY TestSymKey
GO
```

In this five article series, we have seen Service Master Key, Database Master Key, and Encrypt & Decrypt using Certificate, Asymmetric Key and Symmetric Key.