SQL 2019 TDE Encryption - Database Restore Procedure Guide

Submitted by:
Oxcyon, Inc.
17520 Engle Lake Dr.
Middleburg Hts., OH 44130
P: 440-239-8611

SQL 209 TDE Encryption and Database Restore Procedure

This topic describes how to protect a database by using transparent data encryption (TDE), and then move the database to another instance of SQL Server by using SQL Server Management Studio or Transact-SQL. TDE performs real-time I/O encryption and decryption of the data and log files. The encryption uses a database encryption key (DEK), which is stored in the database boot record for availability during recovery. The DEK is a symmetric key secured by using a certificate stored in the master database of the server or an asymmetric key protected by an EKM module.

Limitations and Restrictions

- When moving a TDE protected database, you must also move the certificate or asymmetric key that is used to open the DEK. The certificate or asymmetric key must be installed in the master database of the destination server, so that SQL Server can access the database files.
- You must retain copies of both the certificate file and the private key file in order to recover the certificate. The password for the private key does not have to be the same as the database master key password.
- SQL Server stores the files created here in D:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA by default. Your file names and locations might be different.

These steps will cover how to setup TDE Encryption using Windows Server.

Permissions

- Requires **CONTROL DATABASE** permission on the **master** database to create the database master key.
- Requires **CREATE CERTIFICATE** permission on the **master** database to create the certificate that protects the DEK.
- Requires **CONTROL DATABASE** permission on the encrypted database and **VIEW DEFINITION** permission on the certificate or asymmetric key that is used to encrypt the database encryption key.

To create a database protected by transparent data encryption

The following procedures show you how to create a database protected by TDE using SQL Server Management Studio and by using Transact-SQL.

Please find the steps below to create the private keys needed:

Creating a Master Key:

We need to create a master key in the master database. creating a master key is performed at the master database level. Execute the following T-SQL script which creates a master key in the master database. Replace it with strong password of yours. This database master key is encrypted by service master key at instance level which is created at the time of SQL Server instance setup.

```
USE MASTER
GO
CREATE MASTER KEY ENCRYPTION
BY PASSWORD = 'UseAStrongPasswordHere!f$7';
CREATE CERTIFICATE MyTDECert
WITH SUBJECT = 'Certificate used for TDE in the TestTDE database';
        GO
```

This certificate is critical to you being able to access data encrypted by TDE, so you should make sure you back it up:

Creating a certificate in the master database

The second step in enabling Transparent Data Encryption (TDE) is creating a certificate in the master database. Once we create a master key, we must create a certificate which is protected by the database master key created in the above step. Execute the following T-SQL script to create a certificate in the master database.

MyTDECert is the name of the certificate. You can input the name and the subject of your choice.

```
CREATE CERTIFICATE MyTDECert
WITH SUBJECT = 'Certificate used for TDE in the TestTDE Database';
go
```

Creating database encryption key (DEK)

Once the certificate is created in the master database, we must create database encryption key (DEK) which is encrypted by the certificate created in the above step. Creation of database encryption key is performed at the user database. Execute the following T-SQL script which creates database encryption key (DEK) in the database called **MyProductsDB.** Replace the database name with yours.

```
USE MyProductsDB
GO
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE MyTDECert;
```

Enable encryption on the database

Once the database encryption key (DEK) is created, we must enable transparent data encryption (TDE) on the database. Execute the following T-SQL script by replacing the database name which you are going to encrypt.

```
ALTER DATABASE TestTDE
  SET ENCRYPTION ON;
  GO
```

And that's all there is to it in practice.

Restoring Transparent Data Encryption (TDE) enable database backup to a different server

When the database is enabled for transparent data encryption, the database backup files are also encrypted. If we try to restore a TDE enabled database backup on a different server it throws error "Cannot find server certificate with thumbprint".

We need the certificate which was used to encrypt the database to restore the backup on a different server. Following are the steps involved in restoring Transparent Data encryption (TDE) enabled database.

- Backup the certificate on the source server
- Copy the backup file and create a certificate from the file
- Copy the key file and from the file
- Restore the database backup

We will go through these steps one by one.

Backup the certificate on the source server

First, we must back up the certificate that was used to encrypt the database. Execute the following T-SQL script to create the certificate backup and the private key file in the mentioned path. **MyTDECert** is the name of the certificate. Replace the name of the certificate with yours.

```
BACKUP CERTIFICATE MyTDECert
  TO FILE = 'C:\MyTDECert'
  WITH PRIVATE KEY
  (
     FILE = 'C:\MyTDECert_PrivateKeyFile',
     ENCRYPTION BY PASSWORD = ' UseAStrongPasswordHere!f$7'
  );
  GO
```

We backup the certificate in case we ever need to move or restore our database to a different server or SQL instance. It's critical if we want to be able to access our encrypted data.

When you back it up, you specify a password to encrypt the key. What happens is SQL grabs the stored version of the Private Key (which is encrypted by the DMK) decrypts it, then re-encrypts it with the password. This means that you would be able to restore it to a different SQL instance where the DMK didn't exist.

This covers us against the scenarios explained above regarding why we use a certificate rather than just relying on the DMK. It should also make it clear that if we need to migrate or recover the database all we need is:

Creating the certificate from the file

Copy the backup file and the private key file to the server where you are going to restore the Transparent data encryption (TDE) enabled database backup.

Check if you have a master key on the master database already, create one if you do not have it. In this case, I do not have the master database key on the destination server.

Execute the following script on the destination server to create the master key. replace it with the password of your choice.

```
USE MASTER
GO
CREATE MASTER KEY ENCRYPTION
BY PASSWORD = ' UseAStrongPasswordHere!f$7'
GO
```

Once the master key is created, restore the certificate using backup file and the private key. Execute the following T-SQL script to restore the certificate from the backup file. Please note that the password should be the same which was used to back up the certificate.

```
USE MASTER;
GO
CREATE CERTIFICATE TDECert
  FROM FILE = ' path_to_file/MyProduct_Cert.cer'
  WITH PRIVATE KEY (
    FILE = N' path_to_file/MyProduct_Cert.pvk',
DECRYPTION BY PASSWORD = '' UseAStrongPasswordHere!f$7”
  );
GO
```

Now let us restore the TDE enabled database backup on the destination server. Please refer to the below image that shows the restore backup is successful after restoring the certificate that is used to create the database encryption key.

How to Move a TDE Encryption Key to Another SQL Server Instance

If you have a database backup of a Transparent Data Encryption (TDE) enabled database, the database backup will contain encrypted data. Because the database backup contains encrypted data you can't just restore it to any instance. You can only restore the database backup to an instance that contains the same certificate and key used to originally encrypt the database.

If you want to restore an encrypted database backup to a new instance you need to import the certificate from the source instance where the encrypted backup was created. Here are the steps it takes to copy the original certificate to the instance where the TDE enabled backup will be restored.

Step 1: Verify that there is a Database Master Key

In this step you need to verify that the target server for the restore has a Database Master Key created. To verify that the Database Master key exists you can run the following TSQL code:

```
USE MASTER;
SELECT name FROM sys.symmetric_keys
WHERE name LIKE '%DatabaseMasterKey%';
```

If a Database Master Key exists, then the above code will return the *name* of the Database Master key. If the Database Master Key doesn't exist, then you can create it with the following TSQL code:

```
USE MASTER;
GO
CREATE MASTER KEY ENCRYPTION
     BY PASSWORD='Provide Strong Password Here for Database Master Key';
GO
```

Step 2: Generate the Certificate Backup from Source Instance

In order to move a TDE encrypted database to another instance you need to have a backup of the certificate that was used to encrypt the TDE enabled database being moved. Hopefully when TDE was set up on the source server a certificate backup was taken. If not, then you can run this TSQL code on the source instance to create a certificate backup and a private key file:

```
USE master;
GO
BACKUP CERTIFICATE TDE_CERT_For_MyData
TO FILE = '' path_to_file\TDE_Cert_For_MyData.cer'
WITH PRIVATE KEY (file=' path_to_file\TDE_CertKey.pvk',
ENCRYPTION BY PASSWORD=' UseAStrongPasswordHere!f$7' );
```

This code backs up the certificate name *TDE_CERT_for_MyData* and creates two files. The first file *TDE_Cert_For_MyDate.cer* contains the backup of the certificate. The second file *TDE_CertKey.pvk* contains the private key.

Step 3: Restore Certificate & Key to the other SQL instance

This code can be used to restore the certificate backup.

```
USE MASTER;
GO
CREATE CERTIFICATE TDECert
  FROM FILE = '' path_to_file\TDE_Cert_For_MyData.cer'
  WITH PRIVATE KEY (
    FILE = N'' path_to_file\TDE_CertKey.pvk',
 DECRYPTION BY PASSWORD = ''EnterStrongPassword'
  );
GO
```

Once the target instance contains the certificate that was used to encrypt the database being restored, then you will be able to restore your TDE enabled database backup to the target instance.

Conclusion

Enable Transparent Data Encryption (TDE) on a database in SQL Server and move the Transparent Data Encryption (TDE) enabled databases to a different server by restoring the backup.

17520 engle lake drive
middelburg hts., ohio 44130
p:440.239.8611 | f. 440.239.8621
oxcyon.com