Table Contents

[Transparent Data Encryption (TDE) 2](#_Toc404839089)

[Introduction 2](#_Toc404839090)

[TDE Set Up Process 3](#_Toc404839091)

[TDE Revoke Process 5](#_Toc404839092)

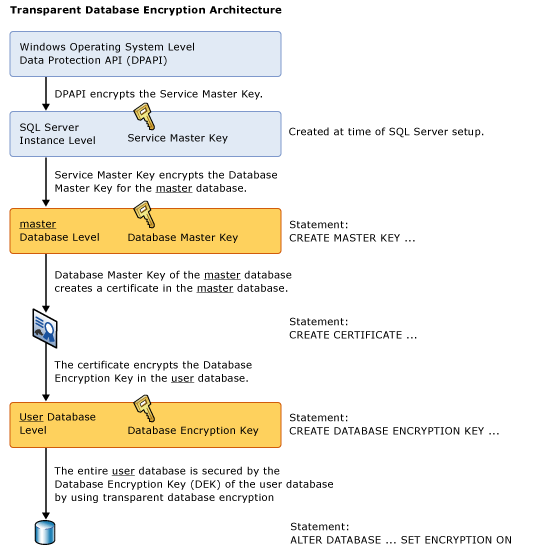
[Case Study 7](#_Toc404839093)

[Summary Points 8](#_Toc404839094)

# Transparent Data Encryption (TDE)

## Introduction

Today’s world is more concerned on security and sensitivity of data. As the data movement is so volatile in the IT industry, there is high risk of misusing the data source without any security measures. In recent days, IT organizations have been working together closely with in-house security team to make sure the security of data has been considered.

This whitepaper presents a native Encryption technique which has been introduced in SQL Server 2008 called - Transparent Data Encryption (TDE).Transparent Data Encryption allows the encryption of the entire database while providing real time encryption of mdf and ldf files.  TDE also ensure encryption on any backups taken while the encryption is enabled. TDE is fairly a straightforward concept as Data is encrypted before it is written to disk and data is decrypted when it is read from disk as it is read into the memory. Hence, there are no special changes needs to be considered on the application code or database queries. This will help in preventing unauthorized access to the data and backup files. TDE will support only in Enterprise and Developer Editions of SQL Server.

## TDE Set Up Process

To use TDE, follow these steps.

* Create a master key
* Create or obtain a certificate protected by the master key
* Create a database encryption key and protect it by the certificate
* Set the database to use encryption

**1. Creating a master key**

It is a good practice to backup the database before proceeding with the encryption. So just in case of any issues during the implementation, we can revert it back to its original state.

|  |
| --- |
| USE MASTER;  CREATE MASTER KEY ENCRYPTION BY PASSWORD='Password@1'  GO |

**2. Creating a Certificate Protected by the Master Key**

|  |
| --- |
| USE MASTER;  CREATE CERTIFICATE TDECert WITH SUBJECT='TDE Certificate'  GO |

**3. Creating a Database Encryption Key**

|  |
| --- |
| USE TestingTDE;  CREATE DATABASE ENCRYPTION KEY  WITH ALGORITHM=AES\_256  ENCRYPTION BY SERVER CERTIFICATE TDECert;  GO |

**4. Backing up the CERTIFICATE with PRIVATE KEY**

It is always required to maintain a backup of private key and certificate to use at the time of restoring. When enabling TDE, you should immediately back up the certificate and the private key associated with the certificate. If the certificate ever becomes unavailable and if you must restore or attach the database on another server, you must have backups of both the certificate and the private key or you will not be able to open the database.

|  |
| --- |
| USE master;  BACKUP CERTIFICATE TDECert1  TO FILE='D:\certificate\_TDE\_Test\_Certificate.cer'  WITH PRIVATE KEY (FILE='D:\certificate\_TDE\_Test\_Key.pvk',  ENCRYPTION BY PASSWORD='Password@1')  GO |

Note: Altering the certificates once used by TDE will cause the database to become inaccessible after a restart.

**5. Turning on Encryption**

|  |
| --- |
| USE TestingTDE  ALTER DATABASE TestingTDE SET ENCRYPTION ON;  GO |

**6. Checking the progress of Encryption using the below script**

|  |
| --- |
| Select db.name,  Case dbe.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'  when 6 then'Protection change in progress (The certificate or asymmetric key that is encrypting the database encryption key is being changed.)'  end EncryptionState  ,dbe.percent\_complete  From sys.dm\_database\_encryption\_keys dbe  Inner join sys.databases db on db.database\_id = dbe.database\_id  WHERE db.name='TestingTDE' |

**7. take a backup of our encrypted database**

|  |
| --- |
| BACKUP DATABASE TestingTDE  TO DISK='D:\TestingTDE\TestingTDE.bak' |

**8. Create a Master Key in destination server**

You can either backup the master key from the source DB and restore to the destination or create a fresh NEW master key on the destination server. In my POC, I have created a new key and tested.

|  |
| --- |
| USE master  GO  CREATE MASTER KEY ENCRYPTION BY PASSWORD='Password@2' |

**9. Import the CERTIFICATE to the destination server**

After the master key is created, create a certificate by importing the certificate that we backed up from the source. Here the **‘Decryption By Password’** parameter is same as that provided to export the certificate from the source.

|  |
| --- |
| CREATE CERTIFICATE TDECert2  FROM FILE ='D:\TestingTDE\certificate\_TDE\_Test\_Certificate.cer'  WITH PRIVATE KEY (FILE ='D:\TestingTDE\certificate\_TDE\_Test\_Key.pvk',  DECRYPTION BY PASSWORD ='Password@1') |

**10. Restore the encrypted database**

|  |
| --- |
| USE [master]  RESTORE DATABASE [TestingTDE] FROM DISK = N'D:\TestingTDE\TestingTDE.bak'  WITH REPLACE |

TDE implementation is complete with these 10 steps.

## TDE Revoke Process

Following below are the steps to revert back the database to its original form with no encryption.

Please be noted to follow the hierarchy of encryptionrevoking and do not violate it.

1. SET DATABASE ENCRYPTION OFF
2. DROP ENCRYPTION KEY
3. DROP CERTIFICATE
4. DROP MASTER KEY

**1. Source server: SET DATABASE ENCRYPTION OFF**

|  |
| --- |
| USE TestingTDE  ALTER DATABASE TestingTDE  SET ENCRYPTION OFF;  GO |

**2. Source server: DROP ENCRYPTION KEY**

|  |
| --- |
| USE TestingTDE  DROP DATABASE ENCRYPTION KEY |

**3. Source server: DROP CERTIFICATE**

|  |
| --- |
| USE Master  DROP CERTIFICATE TDECert |

**4. Source server: DROP MASTER KEY**

|  |
| --- |
| USE Master  DROP MASTER KEY |

**5. Destination server: SET DATABASE ENCRYPTION OFF**

|  |
| --- |
| USE TestingTDE  ALTER DATABASE TestingTDE  SET ENCRYPTION OFF;  GO |

**6. Destination server: DROP ENCRYPTION KEY**

|  |
| --- |
| USE TestingTDE  DROP DATABASE ENCRYPTION KEY |

**7. Destination server: DROP CERTIFICATE**

|  |
| --- |
| USE Master  DROP CERTIFICATE TDECert2 |

**8. Destination server: DROP MASTER KEY**

|  |
| --- |
| USE Master  DROP MASTER KEY |

The database will now become normal with no encryption.

## ****Case Study****

The performance test on TDE has been carried out to assess the impact of encryption for SELECT/UPDATE and INSERT operations. The test script has been created to simulate the OLTP workload in SQL Server instance. Elapsed Time has been collected for equal and varied distribution of transaction as below.

**Environment Details**

4 64-bit Dual-Core   
8 GB RAM  
Windows Server 2003 R2 Enterprise x64 Edition  
SQL Server 2008 (RTM)

**Test Results**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Iteration #** | **SELECT** | **UPDATE** | **INSERT** | **ELAPSED TIME(ms)** | | **OVERHEAD** |
| **NORMAL** | **TDE** |
| 1 | 10 | 0 | 0 | 327690 | 337029 | 2.849949648 |
| 2 | 6 | 2 | 2 | 457100 | 480457 | 5.109822796 |
| 3 | 4 | 3 | 3 | 345726 | 373453 | 8.019934862 |
| 4 | 0 | 10 | 0 | 544639 | 590061 | 8.339836112 |
| 5 | 0 | 0 | 10 | 134404 | 149309 | 11.08969971 |

Test results has been concluded the following:

The average overhead for iteration is observed as 7.08. This indicated a varied overhead on the performance with TDE enablement. Interestingly, WRITE operations are having more impact on the performance than the READ operations.

## ****Summary Points****

* Ensures Security of database at master files (mdf & ldf) and backup files.
* TDE does not increase the size of the encrypted database.
* Encrypts the Entire Database and no granular level of encryption possible.
* When one of the databases on an instance has been set for TDE, then, TEMPDB on that instance also get under the TDE enablement. This might have a performance issue on other databases as TEMPDB is common for that instance.
* The data and logs are encrypted and decrypted to memory real time and that can have performance impact.
* No protection for data in memory.
* FILESTREAM Data will not be encrypted though TDE is enabled.
* TDE does not provide encryption across communication Channels.