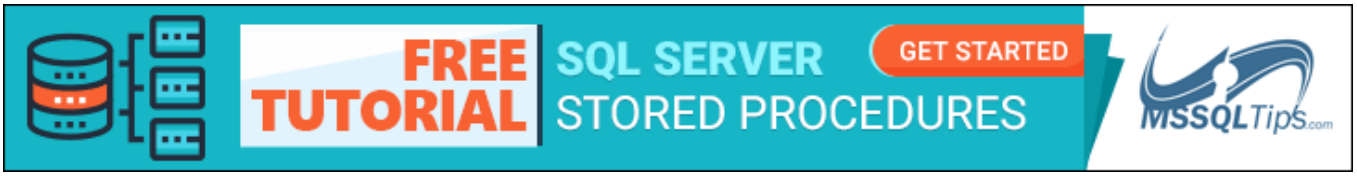
MSSQLTips Free SQL Server Conference Registration + Travel Expenses click to learn more

SQL Server 2008 Transparent Data Encryption getting started



Free Webcast - Simplify SQL Server Administration

Problem

While reviewing the new features in SQL Server 2008, we noticed Transparent Data Encryption. This sounds very interesting. Could you provide us with an explanation and the details to implement it?

Solution

TDE is a new feature in SQL Server 2008; it provides real time encryption of data and log files. Data is encrypted before it is written to disk; data is decrypted when it is read from disk. The "transparent" aspect of TDE is that the encryption is performed by the database engine and SQL Server clients are completely unaware of it. There is absolutely no code that needs to be written to perform the encryption and decryption. There are a couple of steps to be performed to prepare the database for TDE, then the encryption is turned on at the database level via an ALTER DATBASE command.

We've probably all heard of incidents where backup tapes containing sensitive information have been lost or stolen. With TDE the backup files are also encrypted when using just the standard BACKUP command once

encryption is turned on for the database. The data in the backup files (or on disk) is completely useless without also having access to the key that was used to encrypt the data.

Before we dive in to the steps to implement TDE, let's take a minute to discuss encryption at a very high level. The Wikipedia definition of encryption is "the process of transforming information (referred to as plaintext) using an algorithm (called cipher) to make it unreadable to anyone except those possessing special knowledge, usually referred to as a key". To encrypt some data, I choose an available algorithm, supply a key and I now have encrypted data. To decrypt the encrypted data, I choose the same algorithm and supply the key. The security provided by encryption is based on the strength of the algorithm and protection of the key. There are two types of keys - symmetric and asymmetric. With a symmetric key, the same value is used to encrypt and decrypt the data. An asymmetric key has two components - a private key and a public key. I use the private to encrypt data and someone else must use the public key to decrypt the data. To recap, the symmetric key or private key of the asymmetric key pair must be stored securely in order for encryption to be effective.

Now let's walk through an example of how to implement TDE. Books on Line lists the following four steps to implement TDE on a particular database:

- 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

Create a Master Key

A master key is a symmetric key that is used to create certificates and asymmetric keys. Execute the following script to create a master key:

```
USE master;
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'Pass@word1';
GO
```

Note that the password should be a strong one (i.e. use alpha, numeric, upper, lower, and special characters) and you have to backup (use BACKUP MASTER KEY) and store it in a secure location. For additional details on master keys refer to our earlier tip Managing SQL Server 2005 Master Keys for Encryption.

Create a Certificate

Certificates can be used to create symmetric keys for data encryption or to encrypt the data directly. Execute the following script to create a certificate:

```
USE master;
CREATE CERTIFICATE TDECert
```

```
WITH SUBJECT = 'TDE Certificate'
GO
```

Note that certificates also need to be backed up (use BACKUP CERTIFICATE) and stored in a secure location. For additional details on certificates, refer to our earlier tip SQL Server 2005 Encryption - Certificates 101.

Create a Database Encryption Key

A database encryption key is required for TDE. Execute the following script to create a new database and a database encryption key for it:

```
CREATE DATABASE mssqltips_tde
GO
USE mssqltips_tde;
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER CERTIFICATE TDECert
GO
```

In order to work with TDE the encryption key must be encrypted by a certificate (a password will not work) and the certificate must be located in the master database.

Enable TDE

The final step required to implement TDE is to execute the following script:

```
ALTER DATABASE mssqltips_tde
SET ENCRYPTION ON
GO
SELECT [name], is_encrypted FROM sys.databases
GO
```

You can query the is_encrypted column in sys.databases to determine whether TDE is enabled for a particular database.

Next Steps

- Refer to Books on Line for additional details on Transparent Data Encryption. Make sure to carefully review the restrictions imposed when enabling TDE.
- Take a look at our earlier tips on Managing SQL Server 2005 Master Keys for Encryption and SQL Server 2005 Encryption - Certificates 101 for additional information on encryption and certificates in SQL Server.
- Download the sample script here and experiment with TDE. The script was tested using the SQL Server 2008 February CTP which you can download here.
- It is important to emphasize that TDE only encrypts the content of data and log files. It does not encrypt the data as it is being passed between the client and the database server. There are many aspects to securing SQL Server; a good starting point is the Books on Line topic Securing SQL Server.
- Take a look at this tip as well Implementing Transparent Data Encryption in SQL Server 2008

Last Update: 2008-06-06

You're invited
to our next free
SQL Server webcast!

click to learn more

Next Tip

Click Here

About the author

Ray Barley is a Principal Architect at RDA Corporation and a MSSQLTips.com BI Expert.

View all my tips

Related Resources

- How to simulate transparent data encryption TDE wi...
- Implementing Transparent Data Encryption in SQL Se...
- SQL Server 2008 Transparent Data Encryption gettin...
- Configuring Transparent Data Encryption with SQL S...
- SQL Server Transparent Data Encryption (TDE) Perfo...
- Recovering a SQL Server TDE Encrypted Database Suc...
- More Database Developer Tips...

Post a comment or let the author know this tip helped.

All comments are reviewed, so stay on subject or we may delete your comment. Note: your email address is not published. Required fields are marked with an asterisk (*).

*Name *Email Notify for updates

*** NOTE *** - If you want to include code from SQL Server Management Studio (SSMS) in your post, please copy the code from SSMS and paste the code into a text editor like NotePad before copying the code below to remove the SSMS formatting.

p

Send me SQL tips:

I'm not a robot

reCAPTCHA
Privacy - Terms

Save Comment

Monday, June 09, 2014 - 3:10:34 PM - Ray Barley Back To Top

According to http://msdn.microsoft.com/en-us/library/bb934049%28v=sql.105%29.aspx "A certificate that has exceeded its expiration date can still be used to encrypt and decrypt data with TDE" This is for SQL Server 2008R2. I would be careful relying on this because if you don't have the certificate you cannot access the database.

Monday, June 09, 2014 - 12:52:13 PM - Stuart Housden Back To Top

I am wondering what would happen if the certificate expired. E.g. I encrypt a database that isn't used often (e.g. in a testing environment) and then when I come back to use it again the certificate is expired. Can I still access the database? If not, can I recover the situation by creating a new certificate?

Many thanks,

Stuart

Thursday, January 24, 2013 - 10:20:10 AM - Ray Barley Back To Top

I'm not sure if you have to do that but certainly it would be the conservative approach.

This is an interesting scenario that I will research as time permits.

Wednesday, January 23, 2013 - 9:36:23 AM - Wayne Back To Top

I was wondering if there is anything I need to do when moving a server with encrypted databases from one domain to a new domain. Should I remove encryption of all the databases, move the server to the new domain, and then encrypt the databases again?

Thank you,

Wayne

Sunday, January 06, 2013 - 7:02:45 PM - Ned Back To Top

Hey Ray. This is teriffic. Thanks for the clear explanations. I'm basically a programmer, but a complete newbie when it comes to Security and Encryption. We have a number of at rest databases in the organization I work for which are only used for internal purposes through in house developed applications. We want to provide some level of security since at the moment we have none :-) It sounds like TDE would be a good first step at providing protection to our data without impacting the applications. Is this a correct assumption?

Also and please forgive this stupid question, but are there easy ways to protect data that flows from an application to a database through let's say an update statement? In other words if I have a simple ASP application running on a Server within our organization (no external facing) and enter my name and address in a form and then click OK to submit that data to the database, is my name and address being sent as plain text from the application server to the database server.

Thanks again and sorry for these questions.

Ned

Monday, November 19, 2012 - 3:32:56 PM - Ray Barley Back To Top

The encryption is performed automatically as you insert or update data. The only thing you can do is query to see whether the database is encrypted or not; e.g.

SELECT [name], is_encrypted FROM sys.databases

will show you each database and whether it is encrypted or not

There is nothing needed in the connection string as far as encryption is concerned

Sunday, November 18, 2012 - 11:13:54 PM - Bhudev Arya Back To Top

How i will make a connection string in visyal studio 2010 application to connect Encrypted Database. I want to know that after encrypted database the name of Table, its columns, stored procedure's name and their defination will be encrype or it will as readable by anyone.

Sunday, October 14, 2012 - 8:23:39 AM - Ray Barley Back To Top

This tip covered transparent data encryption which is at the database level.

As best as I can tell there is no way to determine whether a column of data is encrypted when you are using the built-in ENCRYPTBY functions in SQL Server. I think your only choice is a custom solution. You could stored procedures for all of your selects, inserts and updates so that you are certain that the column is encrypted on every insert and update and decrypted on every select. Maybe you could use a trigger to make sure that the column get encrypted on every insert and update.

Saturday, October 13, 2012 - 4:42:16 AM - Shekhar Back To Top

Thanks Ray...

If we encrypt a particular column in a table.. then is there a way to find before running script, data is Encrypted or decrypted in script. Problem is if data is decrypted and use to decrypted it agian it returns null. If perfom update to decrypt data (Encrypt/Decrypt update on same column) on same column all data get lost. So if somehow i can find in script that data is decrypted, we can bypass decryption step so data will not lost. Currenly using 'YES'/'NO' flag in some Reference table.