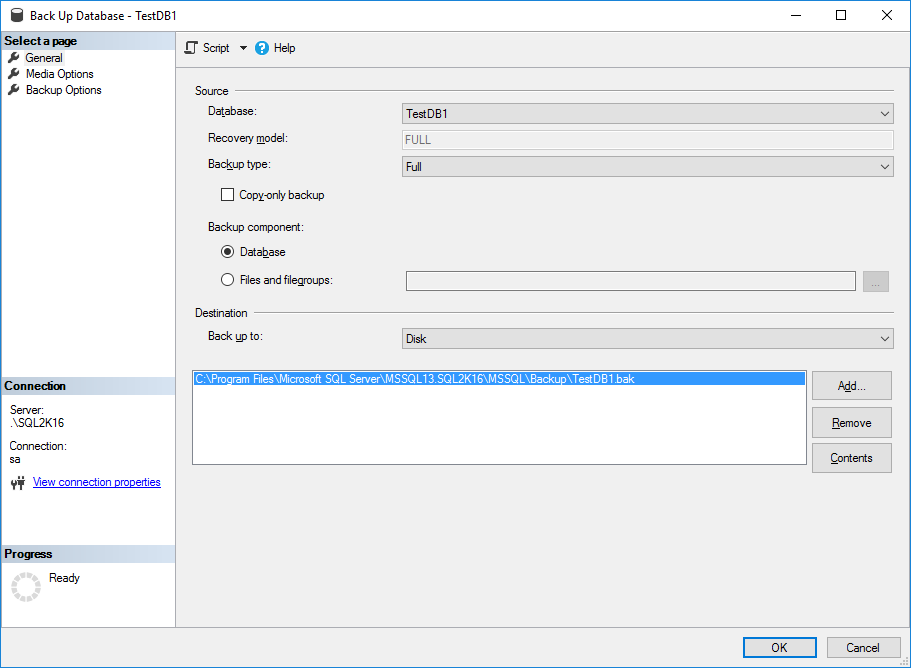
# Encrypting a SQL Server Database Backup

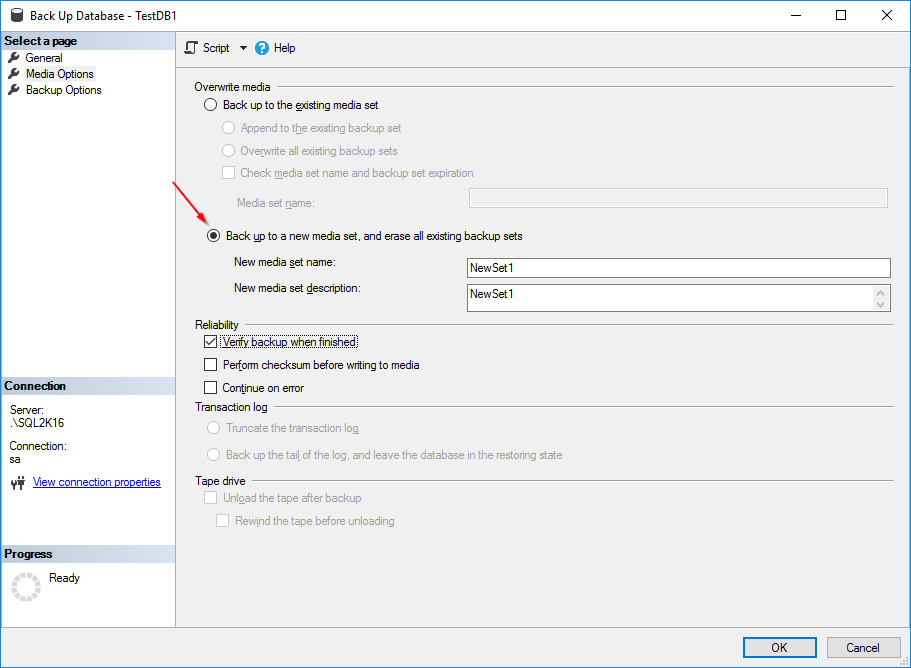
<https://www.sqlnethub.com/blog/encrypting-a-sql-server-backup-set/>

* Encrypting a SQL Server database backup is necessary in many cases, especially when the database has sensitive data.
* SQL Server provides an easy way to encrypt database backups.
* Let’s further examine this functionality with a step-by-step example.
* In this example, we are going to backup a SQL Server 2014 database, encrypt it, and then restore it on a SQL Server 2016 instance. The sample database’s name is “TestDB1” (not quite an original name for a database 🙂

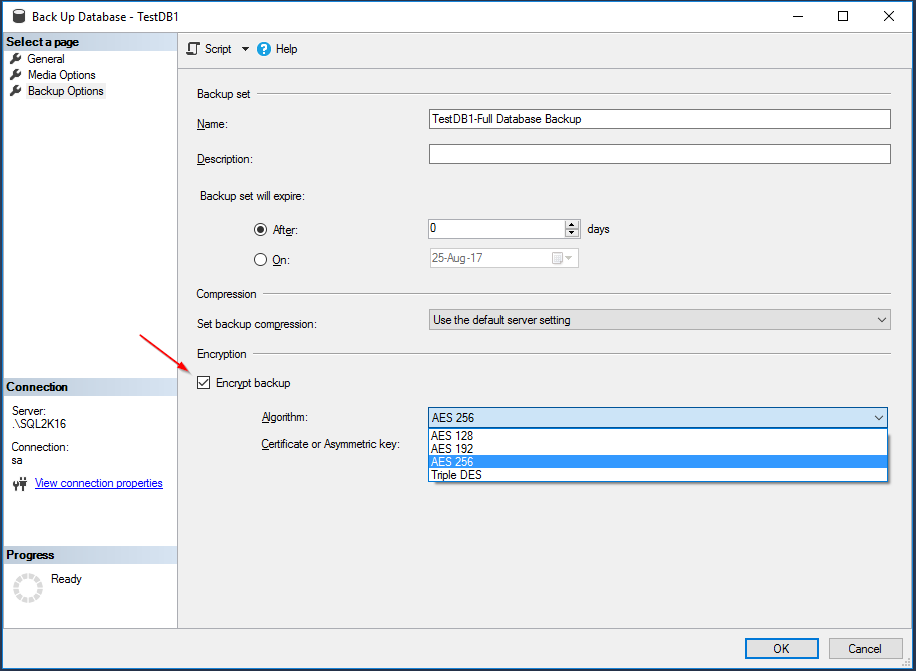
In SQL Server Management Studio, if we right-click on the database and go to “Tasks”, “Back Up…”, we are presented with the well-known backup dialog:



* If we go to “Backup Options” we see that there is a checkbox named “Encrypt Backup”. By default, this is not available as you prior need to set a new media set in “Media Options”.
* It will be made available, only if you set a new media set.
* So, we now navigate to “Media Options”, and select to Backup to a new media set by setting up the new media set’s name and description:



Now, if we navigate back to “Backup Options”, we can see that we can check the “Encrypt Backup” checkbox and select an encryption algorithm, but still we need a backup certificate or asymmetric key:



**So, we go back to the query window and create a backup certificate for the database as per below example (always use strong passwords, oh well, at least stronger than the one in this example 🙂**

**--Create Database Master Key and Encrypt it with a Strong Password**

USE master;

GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'MyComplexMasterKeyPassword';

GO

**--Create Backup Certificate**

USE master;

GO

CREATE CERTIFICATE TestDB1BackupEncryptCert

WITH SUBJECT = 'TestDB1 Backup Encryption Certificate';

GO

--IMPORTANT NOTE: It is critical that you backup the master DB key and the database backup certificate to a secure location

**--Backup Master DB Key**

BACKUP MASTER KEY

TO FILE = 'c:\tmp\MasterKey.key'

ENCRYPTION BY PASSWORD = 'S3curePass!';

GO

**--Export the Backup Certificate to a File**

BACKUP CERTIFICATE TestDB1BackupEncryptCert TO FILE = 'c:\tmp\TestDB1Cert.cert'

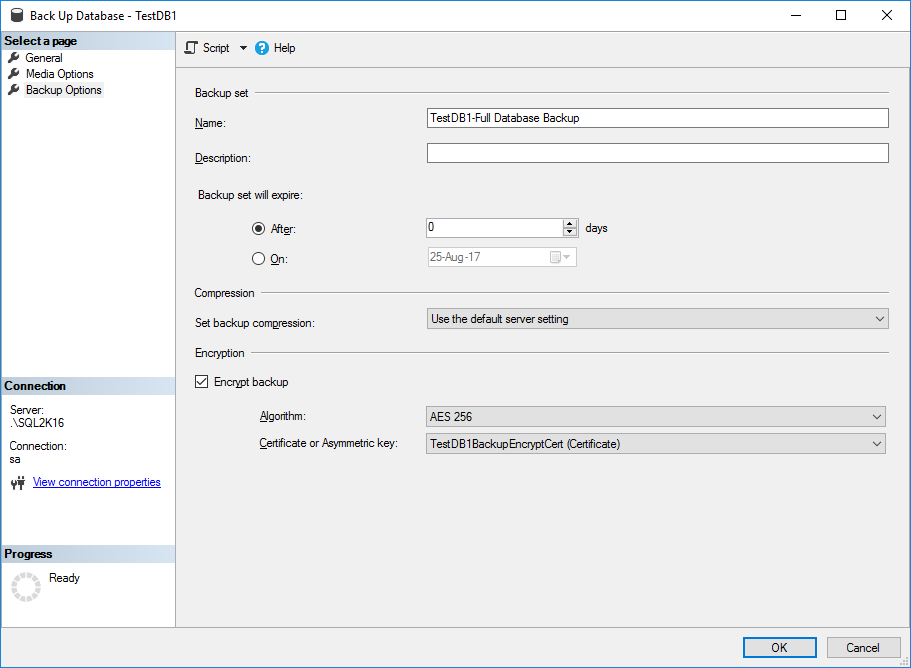
WITH PRIVATE KEY (

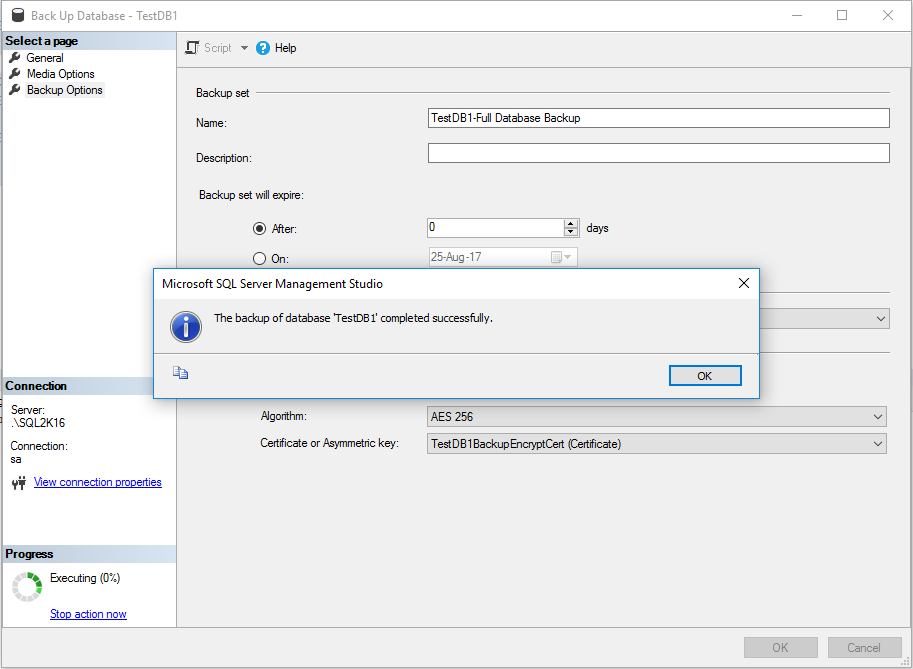
FILE = 'c:\tmp\TestDB1CertKey',

ENCRYPTION BY PASSWORD = 'S3curePassCert!')

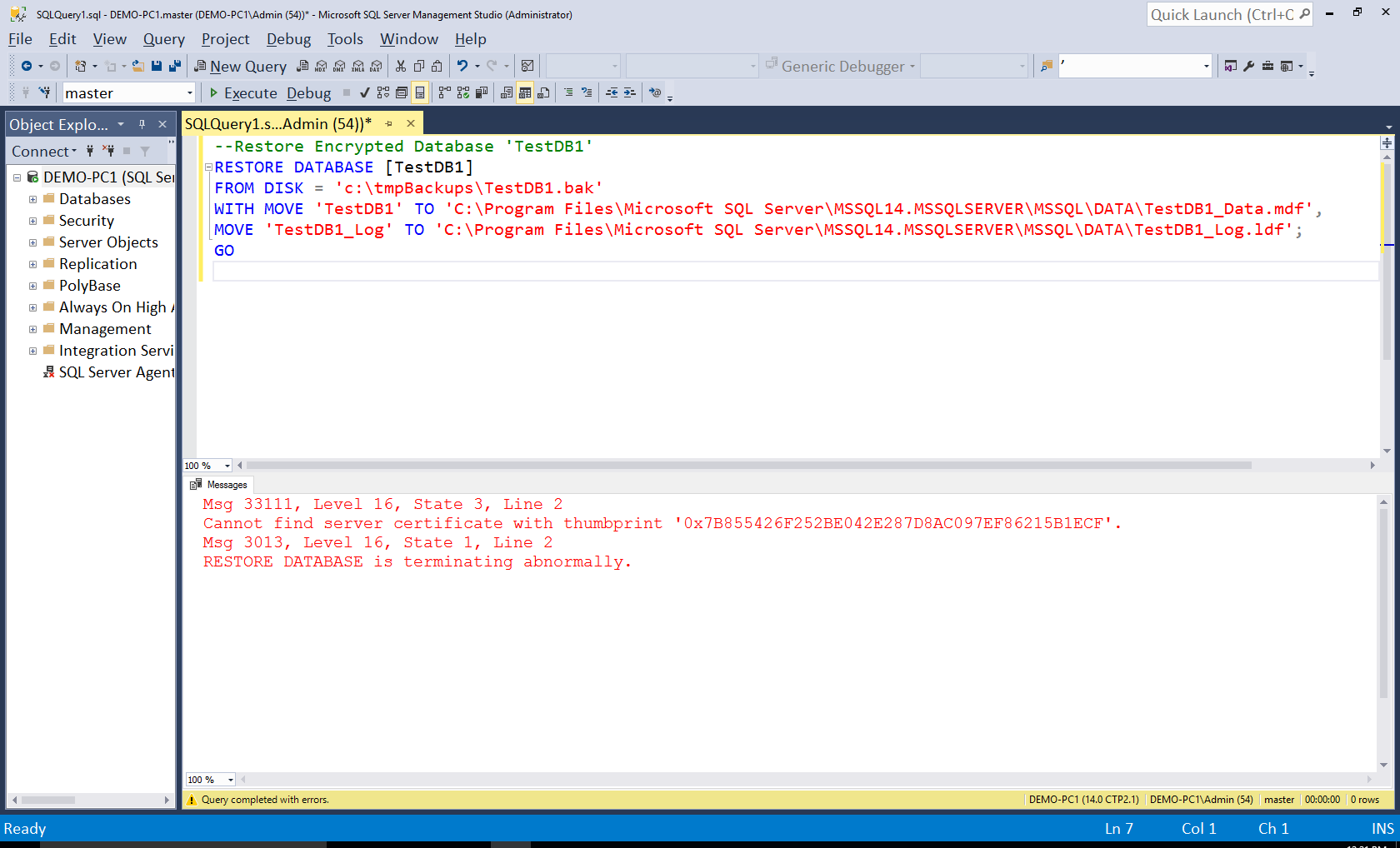
Note that the above file keys are created by the service account that runs SQL Server Database Engine and it is the only user that has full access. In order to get access to these files, if you are a local administrator on the machine running SQL Server, you can do so by editing the permissions (via Advanced dialog).

Now, let’s try again to take an encrypted backup of the database:





* As you can see, backup was successful. Now, the backup file create is encrypted with the newly created backup certificate.
* Now, let’s try to restore the encrypted database backup set to another instance of SQL Server (instance name: “DEMO-PC1”, it is a default instance name).
* First, for illustration purposes, let’s just try to restore it without creating a backup certificate on the destination SQL Server instance:



As you  can see, it was not possible to restore the encrypted database.

Now, let’s try again but this time prior to running the restore command, we re-create the master DB key as well as the database backup certificate (based on the exported cert/key) on the destination SQL Server instance:

--Recreate master DB key on destination SQL Server instance

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'S3curePass!';

GO

**--Restore the Certificate Based on the Previously Exported Key/Cert files**

CREATE CERTIFICATE TestDB1BackupEncryptCert

FROM FILE = 'c:\tmpBackups\keys\TestDB1Cert.cert'

WITH PRIVATE KEY (FILE = 'c:\tmpBackups\keys\TestDB1CertKey',

DECRYPTION BY PASSWORD = 'S3curePassCert!');

GO

**--Restore Encrypted Database 'TestDB1'**

RESTORE DATABASE [TestDB1]

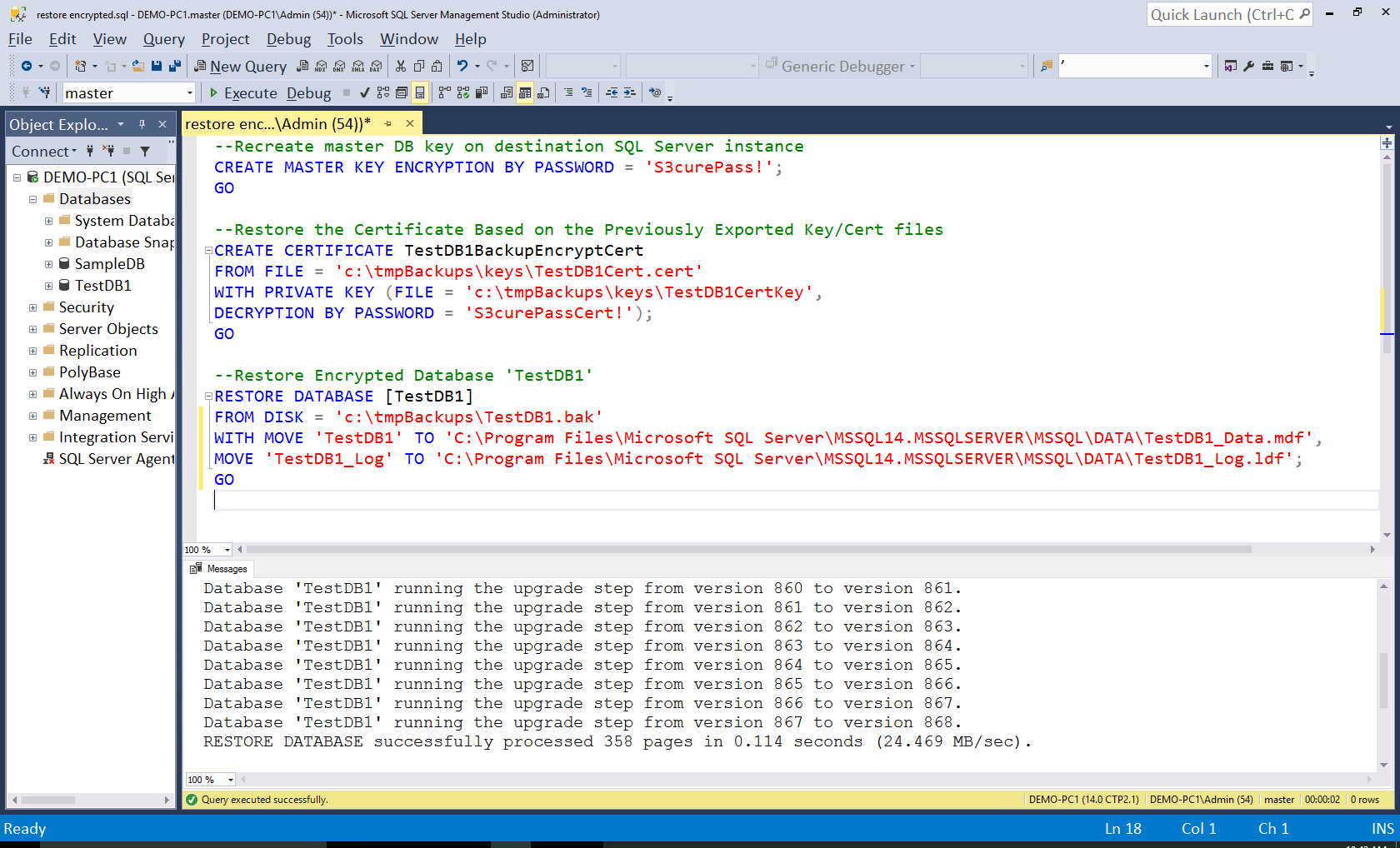
FROM DISK = 'c:\tmpBackups\TestDB1.bak'

WITH MOVE 'TestDB1' TO 'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\TestDB1\_Data.mdf',

MOVE 'TestDB1\_Log' TO 'C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\TestDB1\_Log.ldf';

GO

As you can see, now the encrypted database has been successfully restored on the destination SQL Server instance:



* This article explained by example, a way of encrypting a SQL Server database backup using SQL Server’s built-in security mechanisms and restoring it in another SQL Server instance.
* Encryption is a very powerful feature in SQL Server and must be used wisely.
* SQL Server is a powerful data platform that provides all the necessary mechanisms for achieving almost anything you would like to do with your data.
* All these mechanisms are built-in and can be easily used just like in this example, where we encrypted a database backup and then restored it on another SQL Server instance.