

Microsoft Corporation

Database Backup Encryption

SQL Server Technical Article

**Summary:** To manage the security of data which has been backed up to the file system in form of database backup files by using SQL Server 2014 backup encryption feature this document provides information on encryption options for SQL Server database backups. Also includes details of the usage, benefits, and recommended practices for encrypting SQL Server database backup during the backup process.

**Writer:** Sandeep Kumar ([sandeep.kumar@microsoft.com](mailto:sandeep.kumar@microsoft.com)), Microsoft

**Reviewers:** Mandi Ohlinger ([mandi.ohlinger@microsoft.com](mailto:mandi.ohlinger@microsoft.com)), Microsoft, Lori Clark ([lori.clark@microsoft.com](mailto:lori.clark@microsoft.com)), Microsoft

**Published:** July 2014

**Applies to:** Microsoft SQL Server 2014 CTP2, RTM

Copyright

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, can change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You can copy and use this document for your internal reference purposes.

© 2014 Microsoft. All rights reserved.

[SQL Server 2014 Database Backup Encryption 3](#_Toc390547387)

[Overview – What is database backup encryption? 3](#_Toc390547388)

[What is required? 4](#_Toc390547389)

[Benefits of database backup encryption 4](#_Toc390547390)

[Changes in system tables 4](#_Toc390547391)

[msdb.dbo.backupset 4](#_Toc390547392)

[msdb.dbo.backupmediaset 5](#_Toc390547393)

[Prerequisites: Database backup encryption 5](#_Toc390547394)

[Master key 5](#_Toc390547395)

[SQL Server certificate 6](#_Toc390547396)

[Considerations 7](#_Toc390547397)

[Permissions 8](#_Toc390547398)

[Database backup encryption methods 8](#_Toc390547399)

[SQL Server Management Studio 8](#_Toc390547400)

[SQL Server Management Studio Maintenance Plan Wizard 9](#_Toc390547401)

[SQL Transact-SQL statements 14](#_Toc390547402)

[Using PowerShell 15](#_Toc390547403)

[Database backup restores 16](#_Toc390547404)

[Retrieving backup, encryption and certificate information 16](#_Toc390547405)

[Create the database master key 17](#_Toc390547406)

[Restoring a certificate from an existing certificate backup 17](#_Toc390547407)

[Recommended practices 17](#_Toc390547408)

[Backup encryption impact analysis on system performance 18](#_Toc390547409)

[System performance analysis 18](#_Toc390547410)

[The Implementation 27](#_Toc390547411)

[Steps to be performed on source SQL Server Instance 27](#_Toc390547412)

[Steps to be performed on destination SQL Server Instance 28](#_Toc390547413)

[Single certificate approach across the environment 29](#_Toc390547414)

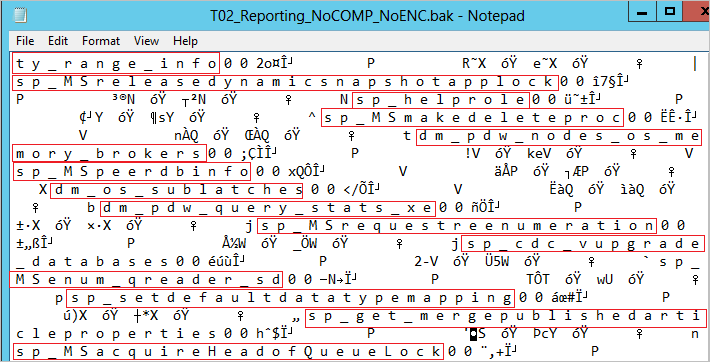
[Appendix: Reference links 30](#_Toc390547415)

SQL Server 2014 Database Backup Encryption

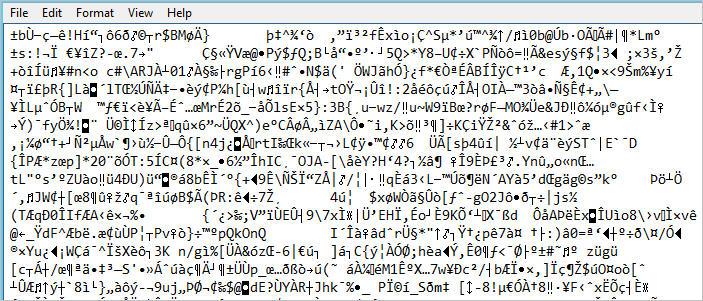
Overview – What is database backup encryption?

With the growing need to keep data secure, be sure to secure your database backup files, especially those on a server file system. With native SQL Server backups, the data in the backup file is stored in plain text on the file system. It is easily read using a text editor. Depending on the data types used in your tables, some data is much easier to view than other data. In the following images, the backup files are opened in Notepad. You can see which database backup file is encrypted:

SQL Server native backup without encryption:



SQL Server native backup with backup encryption:



Starting SQL Server 2014, SQL Server can encrypt the data while creating a backup. By specifying the encryption algorithm and the encryptor (a Certificate or Asymmetric Key) when creating a backup, you can create an encrypted backup file. All storage destinations (on-premises and Microsoft Azure Storage) are supported.

SQL Server backup encryption improves security and works in any context where SQL Server can be used: on-premises, in a Microsoft Azure VM, or in a Hyper-V environment.

What is required?

To encrypt the database backup during backup operation, specify an encryption algorithm and an encryptor to secure the encryption key. Supported encryption options include:

* **Encryption algorithm**: AES\_128, AES\_192, AES\_256, and Triple\_DES\_3Key
* **Encryptor**: A certificate or asymmetric key

|  |
| --- |
| **Caution** |
| It is very important to back up the certificate or asymmetric key, and preferably to a different location than the backup file it was used to encrypt. Without the certificate or asymmetric key, you cannot restore the backup, rendering the backup file unusable. |

Benefits of database backup encryption

* Encrypting the database backup helps secure the data. SQL Server provides the option to encrypt the backup data while creating a backup.
* Database backup encryption can also be used for the databases that are encrypted using TDE (Transparent Data Encryption).
* Encryption is also supported for backups done by SQL Server managed backup to Microsoft Azure which provides additional security for off-site backups.
* Supports multiple encryption algorithms up to AES 256 bit. This gives you the option to select an algorithm that aligns with your requirements.
* You can integrate encryption keys with Extended Key Management (EKM) providers.

Changes in system tables

In SQL Server 2014, there are changes to system tables that provide information about the key algorithm, encryption type, and encryption thumbprint used while the database backup is encrypted.

msdb.dbo.backupset

This table in *msdb* system database contains a row for each backup set. A backup set contains the backup for a single, successful backup operation. RESTORE, RESTORE FILELISTONLY, RESTORE HEADERONLY, and RESTORE VERIFYONLY statements operate on a single backup set within the media set on the specified backup device(s). ***key\_algorithm***, ***encryptor\_thumprint***, ***encryptor\_type*** columns of this DMV provide information on whether the backup is encrypted, what encryptor type is used, and the encryptor thumbprint.

SELECT TOP 5

name,

key\_algorithm,

encryptor\_thumbprint,

encryptor\_type,

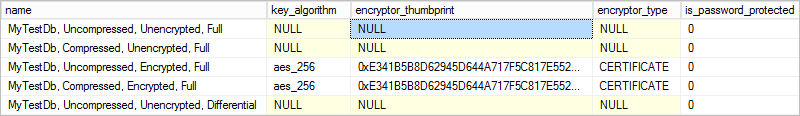
is\_password\_protected

FROM msdb.dbo.backupset AS backupset with (NOLOCK)

WHERE type IN ('D', 'I') AND database\_name = 'MyTestDb'

ORDER BY backupset.backup\_start\_date DESC

GO



msdb.dbo.backupmediaset

This table in *msdb* system database contains one row for each backup media set. The *is\_encrypted* column indicates whether the backup is encrypted or not. ***0*** indicates not-encrypted and ***1*** means encrypted. Initially this value is set to *NULL* which indicates non-encrypted backupmediaset.

SELECT TOP 2

name,

media\_set\_id,

is\_encrypted,

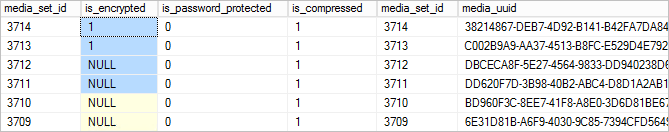
is\_password\_protected,

is\_compressed, \*

FROM msdb.dbo.backupmediaset AS mediaset with (NOLOCK)

ORDER BY mediaset.media\_set\_id DESC

GO



Prerequisites: Database backup encryption

Encrypting a database backup requires the following prerequisites:

* *Create a database master key for the master database*: The database master key is a symmetric key that protects the private keys of certificates and asymmetric keys that are present in the database. For more information, see [SQL Server and Database Encryption Keys (Database Engine)](http://msdn.microsoft.com/library/bb964742(v=sql.120).aspx) (http://msdn.microsoft.com/library/bb964742(v=sql.120).aspx).
* *Create a certificate or asymmetric key* to use for backup encryption. For more information on creating a certificate, refer to the following topics.
  + [CREATE CERTIFICATE (Transact-SQL)](http://msdn.microsoft.com/library/ms187798(v=sql.120).aspx)

(http://msdn.microsoft.com/library/ms187798(v=sql.120).aspx)

* + [CREATE ASYMMETRIC KEY (Transact-SQL)](http://msdn.microsoft.com/library/ms174430(v=sql.120).aspx)

(http://msdn.microsoft.com/library/ms174430(v=sql.120).aspx)

|  |
| --- |
| **Caution** |
| Only asymmetric keys residing in an Extended Key Management (EKM) are supported. |

Master key

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. When created, the master key is encrypted using the AES\_256 algorithm and a user-supplied password. 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 both the database and in master. Typically, the copy stored in master is silently updated whenever the master key is changed. This default behavior can be changed using the DROP ENCRYPTION BY SERVICE MASTER KEY option of ALTER MASTER KEY. A master key that is not encrypted by the service master key must be opened using the OPEN MASTER KEY statement and a password.

The *is\_master\_key\_encrypted\_by\_server* column of the *sys.databases* catalog view in *master* database indicates whether the database master key is encrypted by the service master key. Information about the database master key is visible in the *sys.symmetric\_keys* catalog view. The following scripts create, back up, and restore the database master key:

-- is\_master\_key\_encrypted\_by\_server indicates whether the database master key is encrypted by the service master key

SELECT is\_master\_key\_encrypted\_by\_server, \* FROM sys.databases with (NOLOCK)

WHERE [name] = 'MyTestDb'

GO

-- Information about database master key

SELECT \* FROM sys.symmetric\_keys with (NOLOCK)

GO

-- Create the master key

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'doodle$7'

GO

-- Opening the master key

OPEN MASTER KEY DECRYPTION BY PASSWORD = 'doodle$7'

GO

-- Backing up the master key

BACKUP MASTER KEY TO FILE = 'H:\MSSQL\Backup\Keys\exported\_master\_key'

ENCRYPTION BY PASSWORD = 'doodle$7'

GO

-- Restoring the master key

RESTORE MASTER KEY FROM FILE = 'H:\MSSQL\Backup\Keys\exported\_master\_key'

DECRYPTION BY PASSWORD = 'doodle$7'

ENCRYPTION BY PASSWORD = 'doodle$7'

GO

-- Altering the master key

ALTER MASTER KEY REGENERATE WITH

ENCRYPTION BY PASSWORD = 'doodle$7'

GO

-- Closing up the master key

CLOSE MASTER KEY

GO

SQL Server certificate

A certificate is a database-level securable that follows the X.509 standard and supports X.509 V1 fields. CREATE CERTIFICATE can load a certificate from a file or assembly. This statement can also generate a key pair and create a self-signed certificate.

The private key must correspond to the public key specified by *certificate\_name*. When SQL Server generates a self-signed certificate, the private key is always created. By default, the private key is encrypted using the database master key. If the database master key does not exist and no password is specified, the CREATE CERTIFICATE statement fails.

You do not have to specify a decryption password when the private key is encrypted with the database master key. The following scripts create, back up, and restore the certificate:

-- Create database backup certificate

CREATE CERTIFICATE BackupEncryptCert

WITH SUBJECT = 'Certificate for Database Backup Encryption'

GO

-- Backup the certificate

BACKUP CERTIFICATE BackupEncryptCert TO FILE = 'H:\MSSQL\Backup\Keys\BackupEncryptCert.cer'

WITH PRIVATE KEY (FILE = 'H:\MSSQL\Backup\Keys\BackupEncryptCert.PrivateKey.pvk',

ENCRYPTION BY PASSWORD = 'doodle$7')

GO

-- Restoring a certificate from existing certificate backup

CREATE CERTIFICATE BackupEncryptCert

FROM FILE = 'H:\MSSQL\Backup\Keys\BackupEncryptCert.cer'

WITH PRIVATE KEY (FILE = 'H:\MSSQL\Backup\BackupEncryptCert.PrivateKey.pvk',

DECRYPTION BY PASSWORD = 'doodle$7')

GO

The *ENCRYPTION BY PASSWORD* option is not required when the private key is encrypted with the database master key. Use this option only when the private key is encrypted with a password. If no password is specified, the private key of the certificate is encrypted using the database master key. Omitting this clause causes an error if the master key of the database cannot be opened.

Considerations

The following restrictions apply to the encryption options:

* If you are using the asymmetric key to encrypt the backup data, only asymmetric keys residing in the EKM provider are supported.
* Stronger encryption (depending on the encryption algorithm chosen) consumes more CPU resources that weak encryption.
* SQL Server Express Edition and SQL Server Web Edition do not support encryption during backup. However, restoring from an encrypted backup to an instance of SQL Server Express Edition or SQL Server Web Edition is supported.
* Previous versions of SQL Server cannot read encrypted backups.
* Appending to an existing backup set option is not supported for encrypted backups. If an encrypted backup file using WITH INIT has been generated, it is not possible to append another encrypted backup to the same file. Doing so returns the following error:

**Error**: Msg 3095, Level 16, State 1, Line 11

The backup cannot be performed because 'ENCRYPTION' was requested after the media was formatted with an incompatible structure. To append to this media set, either omit 'ENCRYPTION' or create a new media set by using WITH FORMAT in your BACKUP statement. If you use WITH FORMAT on an existing media set, all its backup sets will be overwritten.

Msg 3013, Level 16, State 1, Line 11

BACKUP DATABASE is terminating abnormally.

* Presently there is no way to read backup information from an encrypted backup file. Even RESTORE HEADERONLY requires the corresponding certificate availability on the target SQL Server Instance.

Permissions

* To encrypt a database backup during backup or to restore from an encrypted backup, **VIEW DEFINITON** permission is required on the certificate or asymmetric key that is used to encrypt the database backup.
* BACKUP DATABASE and BACKUP LOG permissions default to members of the *sysadmin* fixed server role and the *db\_owner* and *db\_backupoperator* fixed database roles.
* The account performing the restore should have **VIEW DEFINITION** permissions on the certificate or the asymmetric key used to encrypt during backup.

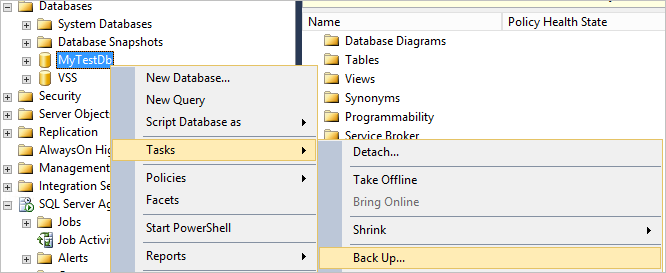
Database backup encryption methods

The following sections list the detailed steps to encrypt the database backup during backup:

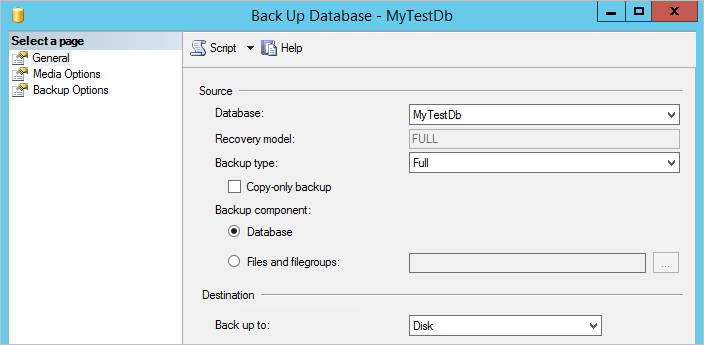
SQL Server Management Studio

In SQL Server Management Studio, on the Backup Options page or Backup Database wizard, you can select encryption, and specify the encryption algorithm and the certificate or asymmetric key to use for the encryption. Steps:

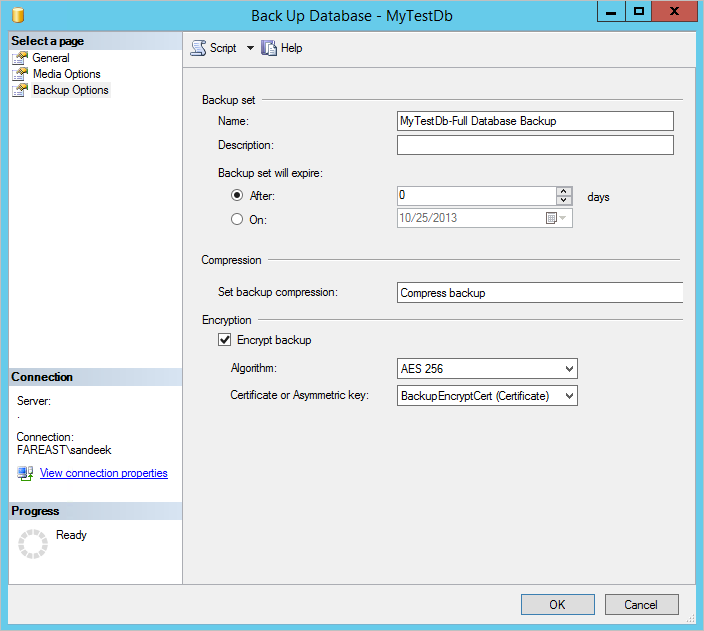
1. Connect to the appropriate instance of Microsoft SQL Server Database Engine. In Object Explorer, expand server name, and then the Databases.
2. Right click on the Database, point to ***Tasks***, and then click on ***Back Up...***



1. The Back Up Database dialog box appears:



1. On the ***General*** and ***Media Options*** pages, enter the database backup options as you would in a normal scenario while taking a database backup using Backup Database dialog box.
2. For database backup encryption, the options are listed in ***Backup Options*** page of Backup Database dialog box:



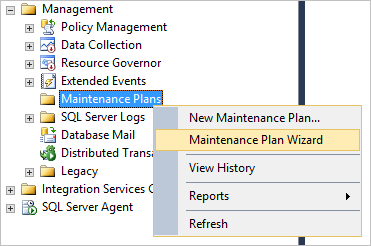
1. To create an encrypted backup, check ***Encrypt backup***.
2. Select an encryption ***Algorithm*** from the drop down list.
3. Provide a ***Certificate or Asymmetric key*** from a list of existing certificates or asymmetric keys.

|  |
| --- |
| **Information** |
| * The encryption option is disabled if you select to append to existing backup set option on Media Options page of Backup Database wizard. Choose backup to a new media set, and erase all existing backup sets. * Create the certificate/key before starting the process. Certificate or asymmetric key created before initiating the backup database wizard will be listed in the drop-down. |

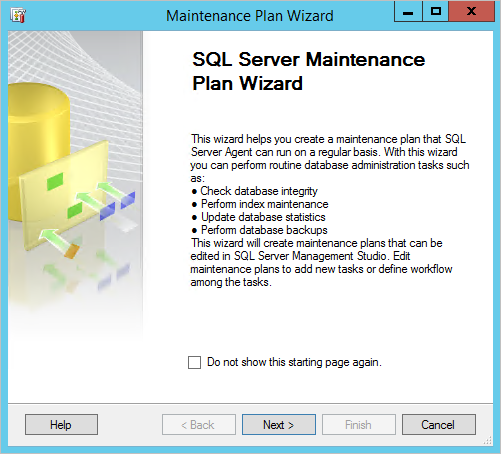
SQL Server Management Studio Maintenance Plan Wizard

SQL Server Management Studio includes the Maintenance Plan Wizard to automate and schedule management and maintenance tasks. Backup is available and can be automated using maintenance plans:

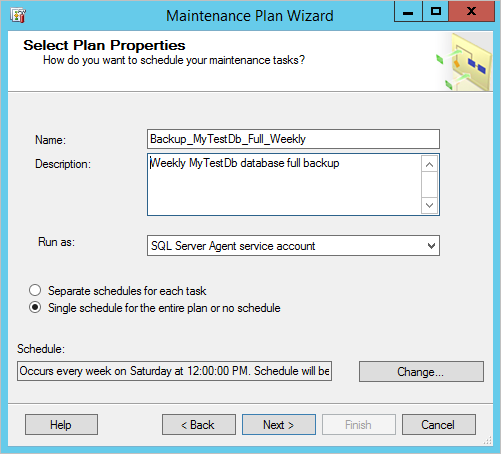
1. Connect to the appropriate instance of Microsoft SQL Server Database Engine. In object explorer, expand Management, and right click on ***Maintenance Plans***. Choose ***Maintenance Plan Wizard***:



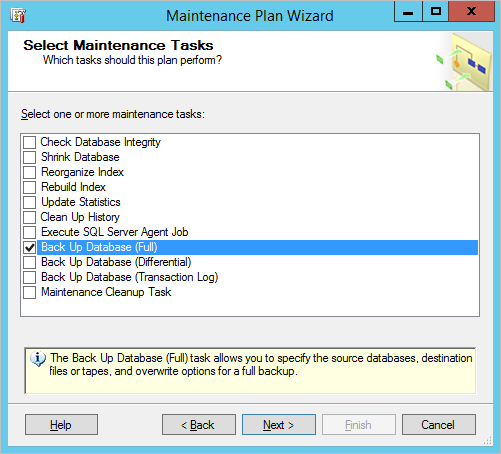
1. In SQL Server Maintenance Plan Wizard, click ***Next***:



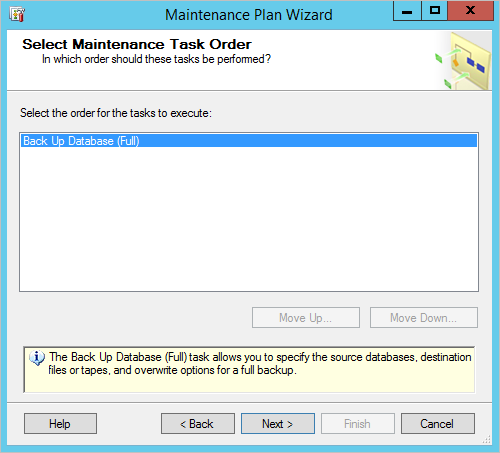
1. In *Select Plan Properties*, enter a meaningful *Name*, *Description*, and required options for the maintenance task. Click ***Next***:



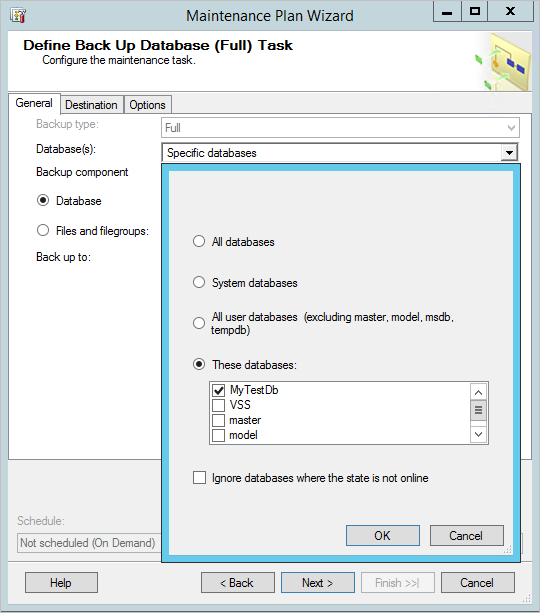
1. Select *Back Up Database (Full)*, *Back Up Database (Differential)*, or *Back Up Database (Transaction Log)* based on your requirements. Click ***Next***:



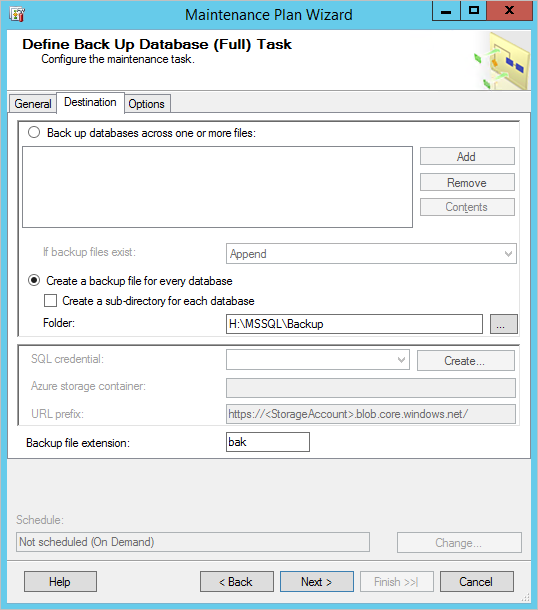
1. If multiple tasks are selected in the previous window, *Select Maintenance Task Order* lets you specify the order that tasks should execute. Click ***Next***:



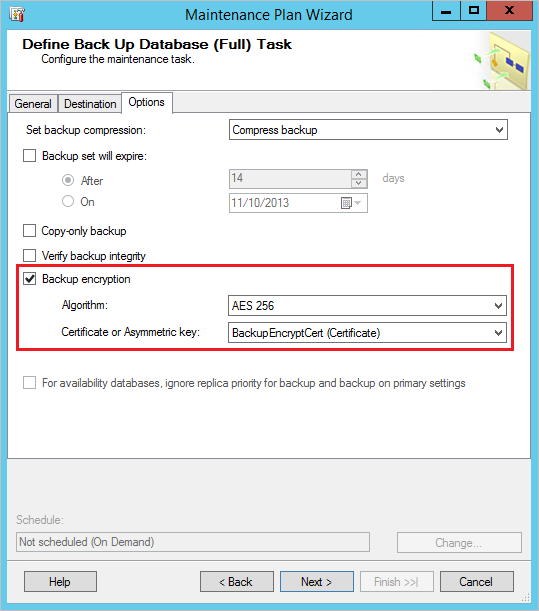
1. In *Define Back Up Database (Full) Task*, *General* tab, select Database from the available Databases list, and select ***OK***:



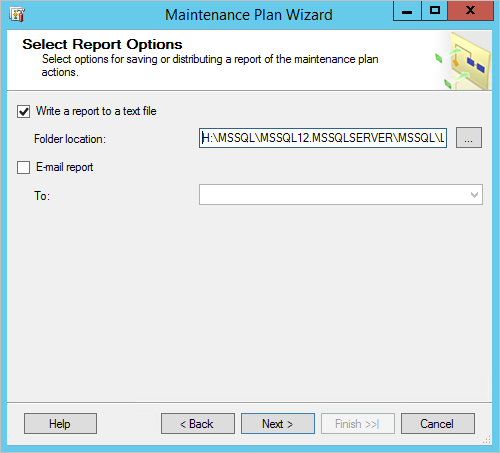
1. In the *Destination* tab, enter the Database backup file details and select ***Next***:



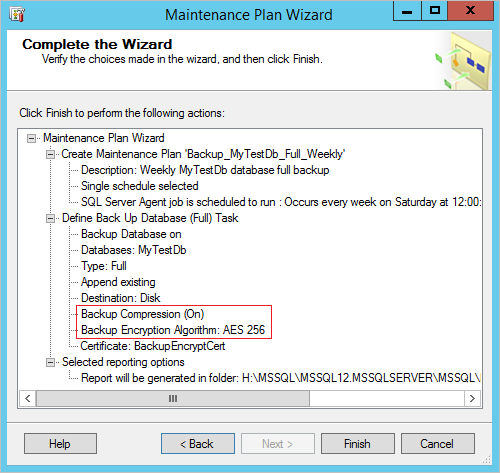
1. *Options* tab provides the options for Backup encryption. To create an encrypted backup, check the ***Encrypt backup*** check box.
2. Select an encryption ***Algorithm*** to use for the encryption.
3. Provide a ***Certificate*** *or* ***Asymmetric*** *key* from a list of existing certificates or asymmetric keys. Select ***Next***:



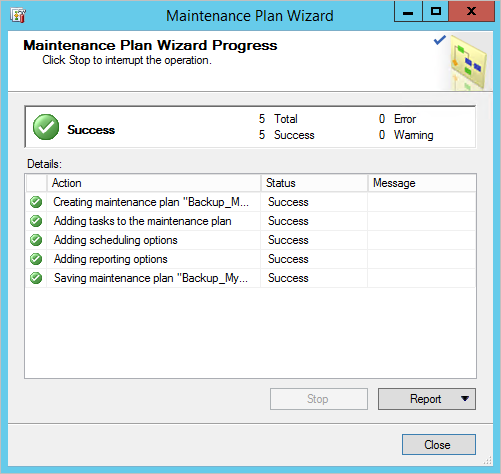
1. In *Select Report Options*, logging can be enabled for Database Backup Maintenance plans. Select ***Next***:



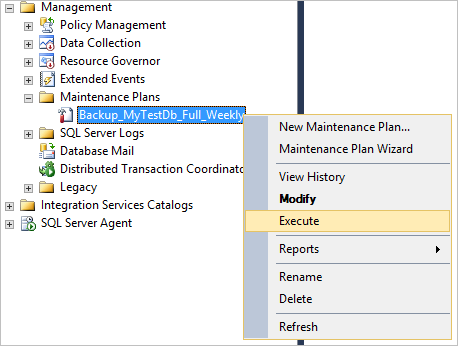
1. In *Complete the Wizard*, verify the summary and select ***Finish***:



1. When *Maintenance Plan Wizard Progress* shows all Green and Maintenance task has been completed, select ***Close***. You can verify the created task under ***Management*** 🡪 ***Maintenance Plans*** in SQL Server Management Studio:



1. To execute the created task, expand Management 🡪 Maintenance Plans, right click on the created task, and then select Execute:



1. Once the task has completed, verify the successful backup of the database and try to open the file in a text editor. For an encrypted database backup, you do not see plain text in the file.

SQL Transact-SQL statements

BACKUP DATABASE backs up a complete database to create a backup file, one or more files or filegroups of the database to create a backup file depending on the options specified.

BACKUP LOG backs up the transaction log of the database to create a log backup under the full recovery model or bulk-logged recovery model. Refer to the following topic.

* Database Backup (TSQL): <http://msdn.microsoft.com/library/ms186865(v=sql.120).aspx>

|  |
| --- |
| **Information** |
| If the certificate used for encryption has never been backed up, the following warning occurs when the backup completes. Make sure you take a backup of certificate along with the private key associated with the certificate:  **Warning**: The certificate used for encrypting the database encryption key has not been backed up. You should immediately back up the certificate and the private key associated with the certificate. If the certificate ever becomes unavailable or 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. |

Full database backup

A full database backup backs up the entire database. This includes part of the transaction log so that the full database can be recovered after a full database backup is restored. Full database backups represent the database at the time the backup finished.

The following script initiates a Full database ***encrypted*** backup at the provided location using the specified certificate and encryption algorithm.

BACKUP DATABASE MyTestDb

TO DISK = 'H:\MSSQL\Backup\MyTestDb\_COMP\_ENC.bak'

WITH NAME = N'MyTestDb, Compressed, Encrypted, Full',

FORMAT, INIT, SKIP, NOREWIND, NOUNLOAD, STATS = 5, COMPRESSION,

ENCRYPTION

(

ALGORITHM = AES\_256, SERVER CERTIFICATE = BackupEncryptCert

)

GO

Differential database backup

Differential database backup makes sure that the database or file backup consists of only the portions of the database or file changed since the last full backup. A differential backup usually takes up less space than a full backup.

The following script initiates a differential database ***encrypted*** backup at the provided location using the specified certificate and encryption algorithm.

BACKUP DATABASE MyTestDb

TO DISK = 'H:\MSSQL\Backup\MyTestDb\_COMP\_ENC\_DIFF.bak'

WITH NAME = N'MyTestDb, Compressed, Encrypted, Differential',

FORMAT, INIT, SKIP, NOREWIND, NOUNLOAD, STATS = 5, COMPRESSION, DIFFERENTIAL,

ENCRYPTION

(

ALGORITHM = AES\_256, SERVER CERTIFICATE = BackupEncryptCert

)

GO

Database transaction log backup

SQL Server database has a transaction log file that records all transactions and the database modifications made by each transaction. The transaction information gets collected only in the databases where database recovery mode is set to *full* recovery or *bulk-logged* recovery.

Backing up transaction logs makes sure that the changes which have occurred after last successful transaction log backup are backed up, and it clears up the inactive portion of the transaction log file.

The following script initiates database transaction log ***encrypted*** backup at the provided location using the specified certificate and encryption algorithm.

BACKUP LOG MyTestDb

TO DISK = 'H:\MSSQL\Backup\MyTestDb\_COMP\_ENC\_LOG.trn'

WITH NAME = N'MyTestDb, Compressed, Encrypted, TLog',

FORMAT, INIT, SKIP, STATS = 5, COMPRESSION, ENCRYPTION

(

ALGORITHM = AES\_256, SERVER CERTIFICATE = BackupEncryptCert

)

GO

Using PowerShell

SQL Server 2014 provides the ability to back up the database using Windows Powershell commandlets. To create an encrypted database backup file, the following example creates the encryption options and uses it as a parameter value in the Backup-SqlDatabase commandlet:

$encryptionOption=New-SqlBackupEncryptionOption –Algorithm Aes256 –EncryptorType ServerCertificate –EncryptorName “BackupEncryptCert”

Backup-SqlDatabase –ServerInstance vmServer0004 –Database “MyTestDb” –BackupFile “H:\MSSQL\Backup\MyTestDb\_PS.bak” –CompressionOption On –EncryptionOption $encryptionOption

Database backup restores

There are no changes to the way Database backups are restored from a backup that is encrypted. There are no changes to the RESTORE commands or the steps. The only important additional step is to have the certificates and keys available on the target server where an encrypted database backup is being restored. You must have the certificates and key available that were used for encrypting the database backup.

If you try to restore the encrypted database backup without having the certificate installed on the SQL Server Instance, the following error occurs. This applies for all the RESTORE, RESTORE FILELISTONLY, RESTORE HEADERONLY, and RESTORE VERIFYONLY statements:

Msg 33111, Level 16, State 3, Line 1

Cannot find server certificate with thumbprint 0xDCC07D539XXXXXXXDF586796D45D82XXX'

Msg 3013, Level 16, State 1, Line 1

RESTORE HEADERONLY is terminating abnormally.

Retrieving backup, encryption and certificate information

Execute the following script on source SQL Server Instance to find out information about the database backups. The output provides information on the encryption, certificate used along with algorithm, and the type of encryptor respective to the particular backup file. Based on the database backup, once you get the information about certificate used for encrypting the backup, then backup the certificate key, and copy it over to the target server where the backup file is being restored. Restore the certificate on the target server.

SELECT TOP 25

bs.database\_name as [DatabaseName],

bs.server\_name as [SQLServerInstance],

cast(convert(numeric,cast(bs.compressed\_backup\_size/1024/1024 as money),1) as decimal(18,2)) as [BackupSize(MB)],

bs.name as [Name], bs.type as [BackupType],

is\_encrypted, bs.encryptor\_type, bs.key\_algorithm,

c.name as [CertificateName],

bmf.physical\_device\_name,

bs.backup\_start\_date, bs.backup\_finish\_date,

convert(varchar(5),DateDiff(s, bs.backup\_start\_date, bs.backup\_finish\_date)/3600)+':'+convert(varchar(5),DateDiff(s, bs.backup\_start\_date, bs.backup\_finish\_date)%3600/60)+':'+convert(varchar(5),(DateDiff(s, bs.backup\_start\_date, bs.backup\_finish\_date)%60)) as [hh:mm:ss]

FROM msdb.dbo.backupmediaset bms

LEFT OUTER JOIN msdb.dbo.backupmediafamily bmf ON bms.media\_set\_id = bmf.media\_set\_id

LEFT OUTER JOIN msdb.dbo.backupset bs ON bmf.media\_set\_id = bs.media\_set\_id

LEFT OUTER JOIN sys.certificates c ON c.thumbprint = bs.encryptor\_thumbprint

WHERE [database\_name] = 'MyTestDb'

ORDER BY bs.backup\_finish\_date DESC

GO

|  |
| --- |
| **Information** |
| You are required to restore the certificate only once for particular database on the target SQL Server Instance. If there are changes to the certificate or the algorithm used at the source SQL Server Instance where the database is being backed up, use the same procedure to back up and restore the certificate on the target SQL Server Instance. |

Create the database master key

To create the master key at the source SQL Server Instance, use the following script. Use an appropriate password as per your security policy.

-- Create the master key

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'doodle$7'

GO

Restoring a certificate from an existing certificate backup

Use the following script to back up the certificate that was used to encrypt the database backup on the source SQL Server Instance. Use an appropriate password as per your security policy.

-- Backup the certificate

BACKUP CERTIFICATE BackupEncryptCert TO FILE = 'H:\MSSQL\Backup\Keys\BackupEncryptCert.cer'

WITH PRIVATE KEY (FILE = 'H:\MSSQL\Backup\Keys\BackupEncryptCert.PrivateKey.pvk',

ENCRYPTION BY PASSWORD = 'doodle$7')

GO

Copy both the certificate and private key files to the target SQL Server Instance and use the following script to create the certificate. Use an appropriate password as per your security policy.

-- Restoring a certificate from existing certificate backup

CREATE CERTIFICATE BackupEncryptCert

FROM FILE = 'H:\MSSQL\Backup\Keys\BackupEncryptCert.cer'

WITH PRIVATE KEY (FILE = 'H:\MSSQL\Backup\BackupEncryptCert.PrivateKey.pvk',

DECRYPTION BY PASSWORD = 'doodle$7')

GO

When the certificate is restored, you can restore the database as you would for a native unencrypted backup.

Recommended practices

Create a backup of the encryption certificate and keys to a location other than the machine where SQL Server Instance is installed. To account for disaster recovery scenarios, consider storing a backup of the certificate or key to an off-site location. You cannot restore an encrypted backup without the certificate used to encrypt the backup.

To restore an encrypted backup, the original certificate used when the backup was taken with the matching thumbprint should be available on the instance you are restoring to. Therefore, the certificate should not be renewed on expiration or changed in any way. Renewal can update the certificate, triggering the change of the thumbprint, and therefore making the certificate invalid for the backup file.

If the database is TDE enabled, choose different certificates or asymmetric keys for encrypting the database and the backup to increase security.

Backup encryption impact analysis on system performance

System performance analysis

This section provides information on the system performance counters collected before implementing backup encryption and after backup encryption implementation when the backup is taking place. We have analyzed the impact of implementing backup encryption and evaluated the collected system performance counters. Performance counters like Logical Disk, Physical Disk, Memory, and Processor were analyzed.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | MyTestDb Backup Runtime and Backup File Size Analysis | | | | |
| Database Size: 9,252,640 (MB)  Free Space in Database: 3,987,294 (MB)  Recovery Model: Full  Compatibility Level: 120  AlwaysOn Configured: Yes  SQL LogShipping: Yes | | | | | |
| Backup execution time *before* implementing backup encryption and *after* implementing backup encryption with various offered encryption algorithms: | | | | | |
| Encryption Algorithm | | **Backup Start Time** | **Backup End Time** | **Duration (hh:mm:ss)** | **Backup File Size** |
| SQL Server Native Backup | | 2013-12-20 09:39:29 | 2013-12-20 13:18:47 | 03:39:18 | 1,645,773 MB |
| AES\_128 Encrypted | | 2013-12-26 04:26:56 | 2013-12-26 08:09:13 | 03:42:17 | 1,646,812 MB |
| AES\_192 Encrypted | | 2013-12-26 08:31:00 | 2013-12-26 12:13:48 | 03:42:48 | 1,646,815 MB |
| AES\_256 Encrypted | | 2013-12-26 21:23:13 | 2013-12-27 01:03:49 | 03:40:36 | 1,647,258 MB |
| TRIPLE\_DES\_3KEY Encrypted | | 2013-12-27 01:19:47 | 2013-12-27 05:06:01 | 03:46:13 | 1,647,279 MB |
|  | | | | | |
| Observation: From the gathered data for backup file size and backup time taken with various encryption algorithms, there is a slight increase in the time taken for backup and slightest increase in the backup file size. For this size of the database (nearly 9TB), an increase of 2GB in backup file size and 6 minutes of increase in backup time is acceptable. | | | | | |

|  |  |
| --- | --- |
| Performance Counter | \memory\available bytes |
| Available Mbytes stands for free unallocated RAM and displays the amount of physical memory, in MB, available to processes running on the computer. | |
| SQL Server Native Backup |  |
| AES\_128 Encryption Algorithm |  |
| AES\_192 Encryption Algorithm |  |
| AES\_256 Encryption Algorithm |  |
| TRIPLE\_DES\_3KEY Encryption Algorithm |  |
| Observation: Encrypting backup while backup is being written to the file system did not cause any memory overhead on the system. As seen in the charts, there is no extra memory utilization. | |

|  |  |
| --- | --- |
| Performance Counter | \process(\*)\% processor time\sqlservr |
| This analysis checks all of the processes to determine if any of the processes are consuming a large amount of CPU. | |
| SQL Server Native Backup |  |
| AES\_128 Encryption Algorithm |  |
| AES\_192 Encryption Algorithm |  |
| AES\_256 Encryption Algorithm |  |
| TRIPLE\_DES\_3KEY Encryption Algorithm |  |
| Observation: There is a noticeable CPU overhead when AES\_192 or TRIPLE\_DES\_3KEY algorithm is used for backup encryption. | |

|  |  |
| --- | --- |
| Performance Counter | \process(\*)\IO data operations/sec\sqlservr |
| The rate at which the process is issuing reads and writes I/O operations. This counter counts all I/O activity generated by the process to include file, network and device I/O. | |
| SQL Server Native Backup |  |
| AES\_128 Encryption Algorithm |  |
| AES\_192 Encryption Algorithm |  |
| AES\_256 Encryption Algorithm |  |
| TRIPLE\_DES\_3KEY Encryption Algorithm |  |
| Observation: Encrypting backup while backup is being written to the file system did not cause IO overhead on the system. | |

|  |  |
| --- | --- |
| Performance Counter | \processor(\*)\% processor time\ \_total |
| The percentage of elapsed time that the processor spends to execute a non-Idle thread. This counter is the primary indicator of processor activity, and displays the average percentage of busy time observed during the sample interval. | |
| SQL Server Native Backup |  |
| AES\_128 Encryption Algorithm |  |
| AES\_192 Encryption Algorithm |  |
| AES\_256 Encryption Algorithm |  |
| TRIPLE\_DES\_3KEY Encryption Algorithm |  |
| Observation: Slight increase on processor time \_total while using TRIPLE\_DES\_3KEY algorithm for backup encryption. | |

The Implementation

Steps to be performed on source SQL Server Instance

The following steps are needed to create an encrypted backup using Transact-SQL statement. Before proceeding with implementing Backup Encryption, you need:-

* A database master key for the master database and a certificate for backup encryption
* Identified encryption algorithm

Create master database master key

Create the master database master key if it does not exist and if it does open the master key by following the next section. Choose a password for encrypting the copy of the master key that is stored in the database. To create the master database master key, connect to the source database engine, and execute the following script:

USE master

GO

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'doodle$7'

GO

Open up the master key

A master key that is not encrypted by the service master key must be opened by using the OPEN MASTER KEY statement and a password. To open the master key, connect to the source database engine, and execute the following script:

USE master

GO

OPEN MASTER KEY DECRYPTION BY PASSWORD = 'doodle$7'

GO

Create a certificate on source

This certificate is used to encrypt the backup file while the backup is being generated. If no expiration date is specified when creating the certificate, by default the certificate is valid for one year from the creation date. You have the option to specify certificate expiry date. The other way of encrypting the backup is by using asymmetric key.

USE master

GO

CREATE CERTIFICATE Prod\_BackupEncryptCert

WITH SUBJECT = 'Production Certificate for Database Backup Encryption',

START\_DATE = '2014-01-01', EXPIRY\_DATE = '2019-12-31'

GO

Backup the database

Specify the encryption algorithm and the certificate to be used for backup encryption. To create the database encryption backup, execute the following script:

BACKUP DATABASE MyTestDb

TO

DISK = 'H:\MSSQL\Backup\MyTestDb\_COMP\_ENC.bak'

WITH

NAME = N'MyTestDb, Compressed, Encrypted, AES\_256, Full',

FORMAT, INIT, SKIP, NOREWIND, NOUNLOAD, STATS = 5, COMPRESSION,

ENCRYPTION

(

ALGORITHM = AES\_256, SERVER CERTIFICATE = Prod\_BackupEncryptCert

)

GO

Backup the certificate and private key

Create a backup of the encryption certificate and keys to the file system. To restore an encrypted backup, backup the certificate and its private key, and copy them over to the target SQL Server Instance where backup will be restored:

BACKUP CERTIFICATE Prod\_BackupEncryptCert TO FILE = 'H:\Keys\Prod\_BackupEncryptCert.cer'

WITH PRIVATE KEY (FILE = 'H:\Keys\Prod\_BackupEncryptCert\_PrivateKey.pvk',

ENCRYPTION BY PASSWORD = 'doodle$7')

GO

Steps to be performed on destination SQL Server Instance

The following steps restore an encrypted backup using Transact-SQL statement. Before proceeding, you need the following on the instance where you are restoring:

* A Database Master Key for the master database
* The original certificate used when the backup was taken with the matching thumbprint and private key

Create master database master key on target SQL Server Instance

Create the master database master key if it does not exist and if it does open the master key by following the next section. Provide a password for encrypting the master key that is stored in the database. Connect to the source database engine, and execute the following script.

CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'doodle$7'

GO

Open up the master key on target SQL Server Instance

A master key that is not encrypted by the service master key must be opened using the OPEN MASTER KEY statement and a password. To open the master key, connect to the source database engine, and execute the following script.

OPEN MASTER KEY DECRYPTION BY PASSWORD = 'doodle$7'

GO

Create (Restore) the certificate from the file

Create the certificate in the master database from the certificate and private key files backed up from source SQL Server Instance. This certificate is needed to restore the encrypted backup file; as SQL Server engine looks for the original certificate on the instance where backup is being restored.

CREATE CERTIFICATE Prod\_BackupEncryptCert

FROM FILE = 'H:\mssql\Backups\Keys\Prod\_BackupEncryptCert.cer'

WITH PRIVATE KEY (FILE = 'H:\mssql\Backups\Keys\Prod\_BackupEncryptCert\_PrivateKey.pvk',

DECRYPTION BY PASSWORD = 'doodle$7')

GO

Restore the database on target SQL Server Instance

There are no changes to the way restore of the database is being done. There are no specific parameters required to be specify for restoring an encrypted backup:

USE master

GO

RESTORE DATABASE MyTestDbCopy

FROM

DISK = N'H:\mssql\Backups\MyTestDb\_COMP\_ENC.bak'

WITH

FILE = 1,

MOVE N'MyTestDb' TO N'H:\mssql\MyTestDbData.mdf',

MOVE N'MyTestDb\_log' TO N'H:\mssql\MyTestDbLog.ldf',

NOUNLOAD, REPLACE, STATS = 5

GO

Single certificate approach across the environment

It can be a tedious and complex manageability task to maintain multiple certificates for multiple databases running on different SQL Server Instances. We analyzed an approach where we explored the possibilities of using a single certificate across multiple SQL Server Instances for encrypting database backups.

* It is possible to generate\create a certificate on an identified primary SQL Server Instance and use the same for backup encryption as well as for restoration purposes.
* One of the easily managed and controlled approach can be:-
  + Create a certificate with a validity period of 1 year and have the certificate restored on all the SQL Server Instances. Certificate restoration is one time activity and is not needed every time you restore an encrypted backup encrypted using respective certificate.
  + Use the certificate to encrypt the backups on all the SQL Server Instances and the respective certificate automatically gets determined by SQL Server engine at the time of restoration.
  + Each generated certificate needs to be archived and saved in secure store to comply with organizations’ data retention policies.
  + An expired certificate can still be used to restore the encrypted database backup given the backup was encrypted using the same certificate.

Appendix: Reference links

Database Backup Encryption

<http://msdn.microsoft.com/en-us/library/dn449489(v=sql.120).aspx>

Create an Encrypted Backup

<http://msdn.microsoft.com/en-us/library/dn425035(v=sql.120).aspx>

Create a Full Database Backup

<http://msdn.microsoft.com/en-us/library/ms187510(v=sql.120).aspx>

Create a Differential Database Backup

<http://msdn.microsoft.com/en-us/library/ms188248(v=sql.120).aspx>

Backup (Transact-SQL)

<http://msdn.microsoft.com/en-us/library/ms186865(v=sql.120).aspx>

Backup Database (Backup Option Page)

<http://msdn.microsoft.com/en-us/library/ms189728(v=sql.120).aspx>

SQL Server and Database Encryption Keys (Database Engine)

<http://msdn.microsoft.com/en-us/library/bb964742(v=sql.120).aspx>

Create Certificate (Transact-SQL)

<http://msdn.microsoft.com/en-us/library/ms187798(v=sql.120).aspx>

Create Asymmetric Key (Transact-SQL)

<http://msdn.microsoft.com/en-us/library/ms174430(v=sql.120).aspx>

Encryption Hierarchy

<http://msdn.microsoft.com/en-us/library/ms189586(v=sql.120).aspx>