Backup Encryption in SQL Server

Source: https://sqlespresso.com/2018/09/05/back-up-encryption/

1. **Why Encrypt Backups?**
    - Backup encryption adds a security layer so that database backups are protected even if someone gets the .bak file. It's especially recommended in highly sensitive environments. (
2. **Availability:**
    - Native backup encryption has been available since **SQL Server 2014**, making it easier than older methods (like only using TDE in Enterprise editions).

What You Must Do Before Encrypting

3. **Create a Database Master Key (DMK):**
    - Must create a **master key** in the master database with a secure password.
4. **Create a Certificate (or Asymmetric Key):**
    - Create a **certificate** in the master database which will be used to encrypt the backup.
    - **Certificates have expiration dates** — if expired, you *cannot take new encrypted backups* until renewed, though you *can still restore existing ones*.
5. **Grant Permissions:**
    - The user performing backups must have **VIEW DEFINITION** on the certificate.

Performing an Encrypted Backup

6. **Backup Command with Encryption:**
    - Use the BACKUP DATABASE statement with the ENCRYPTION option and specify:
        - **Algorithm** (e.g., AES_256)
        - **SERVER CERTIFICATE** name
    - Example syntax shown in the article.
7. **Encryption Algorithms Supported:**
    - SQL Server supports several encryption algorithms such as **AES_128 / AES_192 / AES_256** and **Triple DES**. AES_256 is usually recommended for strongest security.
    -

https://dataginger.com/2015/07/21/sql-server-new-database-backup-encryption-with-sql-server-2014

Restoring an Encrypted Backup

8. **Certificate Requirement:**
    - **Without the certificate and keys used to encrypt the backup, you *cannot restore*** the encrypted backup. So you must *back up the certificate and private key* to a safe location (and preferably off-site).
9. **Normal Restore Command:**
    - Once the certificate and keys are present on the server, restoring the encrypted backup uses a normal RESTORE DATABASE command.

Best Practices & Important Notes

10. **Backup the Keys:**
    - Always **backup the master key and certificate** right after creation and store them securely — without them, the backup is useless.

11. **Certificate Expiration:**
    - Pay attention to certificate expiration dates; expired certificates block taking new encrypted backups until renewed.
12. **Key Management Matters:**
    - If certificates/keys are lost or not properly managed, **restoring backups becomes impossible** — so include certificate management in your backup strategy.

https://www.sqlshack.com/understanding-database-backup-encryption-sql-server/

T-SQL examples and a **step-by-step checklist** for **SQL Server Backup Encryption**.

SQL Server Backup Encryption – T-SQL Examples

1) Create Master Key (in master)

```
USE master;
GO
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'Str0ng_P@ssw0rd!';
GO
```

2) Create Certificate for Backup Encryption

```
USE master;
GO
CREATE CERTIFICATE BackupEncryptionCert
WITH SUBJECT = 'Backup Encryption Certificate',
    EXPIRY_DATE = '2030-12-31';
GO
```

3) Backup the Certificate & Private Key (VERY IMPORTANT)

```
BACKUP CERTIFICATE BackupEncryptionCert
TO FILE = 'D:\BackupKeys\BackupEncryptionCert.cer'
WITH PRIVATE KEY
(
  FILE = 'D:\BackupKeys\BackupEncryptionCert_PrivateKey.pvk',
  ENCRYPTION BY PASSWORD = 'Cert_Str0ng_P@ss'
);
GO
```


Store these files securely & off-server

4) Take an Encrypted Database Backup

```
BACKUP DATABASE MyDatabase
TO DISK = 'D:\Backups\MyDatabase_Enc.bak'
WITH
  ENCRYPTION
  (
    ALGORITHM = AES_256,
    SERVER CERTIFICATE = BackupEncryptionCert
  ),
  INIT;
GO
```

✓ Backup file is now **encrypted at rest**

5) Restore Encrypted Backup (Same or Another Server)

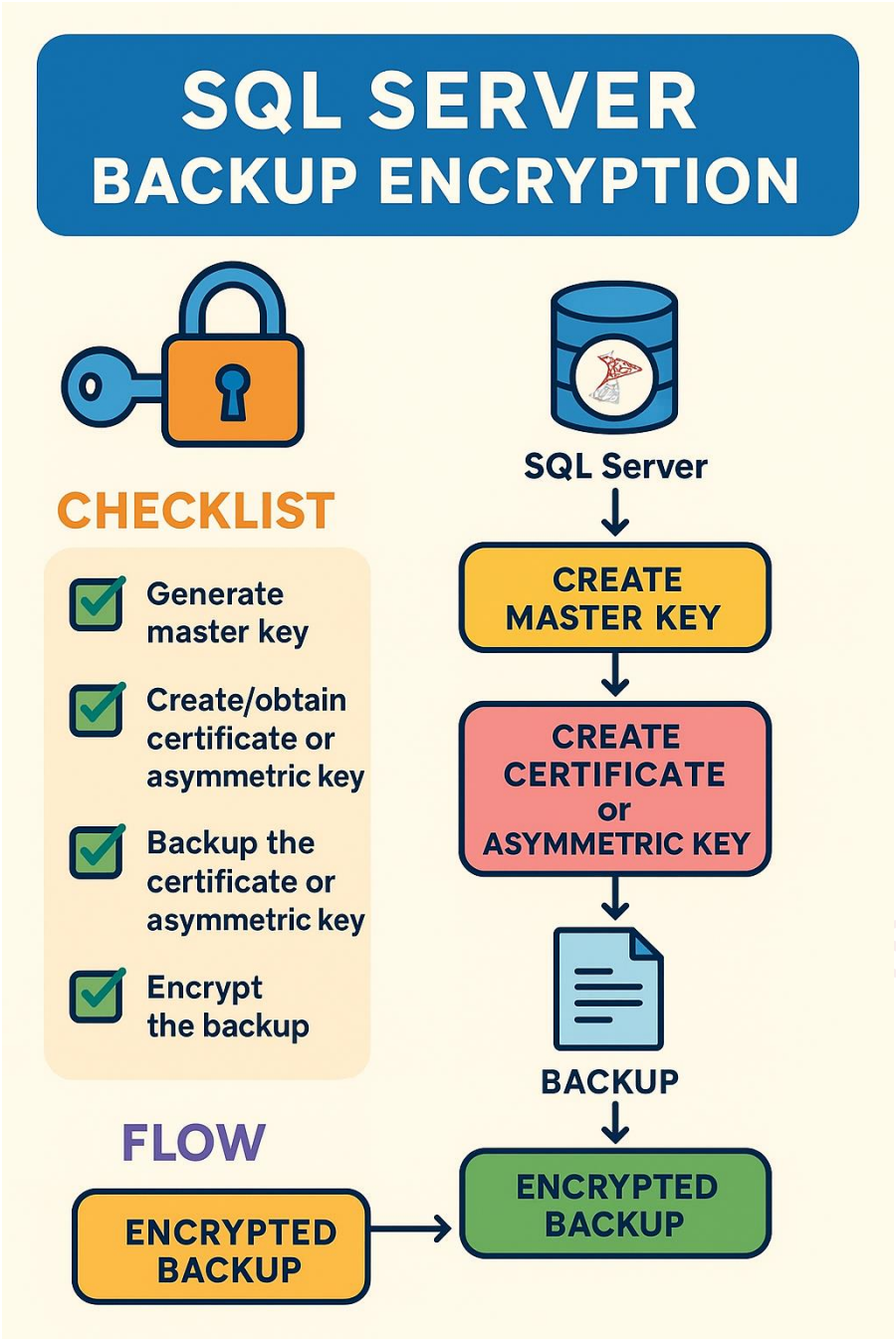
```
RESTORE DATABASE MyDatabase
FROM DISK = 'D:\Backups\MyDatabase_Enc.bak';
GO
```

⚠ **Restore will FAIL if certificate & private key are missing**

6) Restore Certificate on Another Server (If needed)

```
USE master;
GO
CREATE MASTER KEY
ENCRYPTION BY PASSWORD = 'Str0ng_P@ssw0rd!';
GO

CREATE CERTIFICATE BackupEncryptionCert
FROM FILE = 'D:\BackupKeys\BackupEncryptionCert.cer'
WITH PRIVATE KEY
(
   FILE = 'D:\BackupKeys\BackupEncryptionCert_PrivateKey.pvk',
   DECRYPTION BY PASSWORD = 'Cert_Str0ng_P@ss'
);
GO
```


SQL SERVER
BACKUP ENCRYPTION
CHECKLIST
Generate master key
Create/obtain certificate or asymmetric key
Backup the certificate or asymmetric key
Encrypt the backup
FLOW
ENCRYPTED BACKUP
SQL Server
CREATE MASTER KEY
CREATE CERTIFICATE or ASYMMETRIC KEY
BACKUP
ENCRYPTED BACKUP

SQL Server Backup Encryption – Checklist

🔷 **Pre-Requisites**

☑ SQL Server 2014 or later

☑ Access to master database

☑ Strong passwords for keys

🔷 **Configuration Steps**

☑ Create **Master Key** in master

☑ Create **Certificate / Asymmetric Key**

☑ Backup certificate & private key

☑ Securely store key files

🔷 **Backup Process**

☑ Use BACKUP DATABASE with ENCRYPTION

☑ Choose strong algorithm (**AES_256 recommended**)

☑ Specify certificate

🔷 **Restore Considerations**

☑ Certificate must exist on restore server

☑ Private key must be available

☑ Normal RESTORE DATABASE syntax

🔷 **Best Practices**

✓ Always back up certificates immediately

✓ Monitor certificate expiry

✓ Store keys off-server & offline

✓ Document encryption details

✓ Test restore regularly