

TDE and Customer Managed Key (CMK)

Last updated by | Vitor Tomaz | May 17, 2022 at 10:50 AM PDT


Contents

- Self-help content presented in Azure Portal
 - Learn about Transparent Data Encryption (TDE) and Custo...
 - Service-managed key
 - Customer-managed key - Bring Your Own Key
 - get, wrapKey, unwrapKey
 - How to check encryption state at instance and databases l...
 - Backup and restore of TDE-encrypted database
 - Resources
- Public Doc Reference
 - Transparent data encryption for SQL Database and Data W...
 - Azure SQL Transparent Data Encryption with customer-ma...
 - Manage Transparent Data Encryption in a Managed Instan...
 - Rotate the Transparent Data Encryption (TDE) protector usi...
 - Remove a Transparent Data Encryption (TDE) protector usi...
 - Migrate certificate of TDE protected database to Azure SQ...

Self-help content presented in Azure Portal

(This content was shown to the customer during case submission. It's also visible on 'Diagnose and solve problems' blade.)


Learn about Transparent Data Encryption (TDE) and Customer Managed Key (CMK) in SQL Managed Instance

[Transparent data encryption](#)  (TDE) helps protect Azure SQL Managed Instance against the threat of malicious offline activity by encrypting data at rest. It performs real-time encryption and decryption of the database, associated backups, and transaction log files at rest without requiring changes to the application.

You can use a service-managed key or use your own key stored in Azure Key Vault (customer managed key/BYOK).

Scan and select one or more of the following headings to learn more about Transparent Data Encryption (TDE) and Customer Managed Key (CMK) in SQL Managed Instance.

Service-managed key

When you choose [service-managed key](#) , Azure will automatically generate a key to encrypt your databases and manage key rotations.

In Azure, the default setting for TDE is that the DEK is protected by a built-in server certificate. The built-in server certificate is unique for each server and the encryption algorithm used is AES 256. If a database is in a geo-replication relationship, both the primary and geo-secondary databases are protected by the primary database's parent server key. If two databases are connected to the same server, they also share the same built-in certificate. Microsoft automatically rotates these certificates in compliance with the internal security policy and the root key is protected by a Microsoft internal secret store. Microsoft also seamlessly moves and manages the keys as needed for geo-replication and restores.

Customer-managed key - Bring Your Own Key

Azure SQL transparent data encryption (TDE) with customer-managed key enables Bring Your Own Key (BYOK) scenario for data protection at rest and allows organizations to implement separation of duties in the management of keys and data. With customer-managed TDE, the customer is responsible for and in a full control of a key lifecycle management (key creation, upload, rotation, deletion), key usage permissions, and auditing of operations on keys.

For those using service-managed TDE who would like to start using customer-managed TDE, data remains encrypted during the process of switching over, and there is no downtime nor re-encryption of the database files. Switching from a service-managed key to a customer-managed key only requires re-encryption of the DEK, which is a fast and online operation.

See more at [Azure SQL transparent data encryption with customer-managed key](#) ☐

get, wrapKey, unwrapKey

Managed Instance requires access to the key vault (get, wrapKey, unwrapKey) using its Azure Active Directory identity. The instance identity can be a system-assigned managed identity, or a user-assigned managed identity assigned to the server.

When using the Azure portal, the Azure AD identity gets automatically created when the server is created. When using PowerShell or Azure CLI, the Azure AD identity must be explicitly created and should be verified. See [Configure TDE with BYOK for SQL Managed Instance](#) ☐ for detailed step-by-step instructions when using PowerShell.

Depending on the permission model of the key vault (access policy or Azure RBAC), key vault access can be granted either by creating an access policy on the key vault, or by creating a new Azure RBAC role assignment with the role [Key Vault Crypto Service Encryption User](#) ☐.

When using firewall with AKV, you must enable option Allow trusted Microsoft services to bypass the firewall.

How to check encryption state at instance and databases level

By default, TDE is enabled at the instance level and newly created databases, with the following exceptions:

- Databases created through restore inherit encryption status from the source.
- Existing databases created before February 2019 are not encrypted by default.

For Azure SQL Managed Instance, the TDE protector is set at the instance level, and it is inherited by all encrypted databases on that instance.


TDE cannot be used to encrypt system databases, such as the master database, in Azure SQL Managed Instance. The master database contains objects that are needed to perform the TDE operations on the user databases. It is recommended to not store any sensitive data in the system databases.

The DMV `sys.dm_database_encryption_keys` returns information about the encryption state of a database and its associated database encryption keys.


The following query can be used:

```
SELECT DB_NAME(database_id) AS [database],
       encryption_state = CASE 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 t
           ELSE 'No State'
       END,
       encryption_scan_state = CASE encryption_scan_state
           WHEN '0' THEN 'No scan has been initiated, TDE is not enabled'
           WHEN '1' THEN 'Scan is in progress'
           WHEN '2' THEN 'Scan is in progress but has been suspended, user can resume'
           WHEN '3' THEN 'Scan was aborted for some reason, manual intervention is required. Contact Microsoft
           WHEN '4' THEN 'Scan has been successfully completed, TDE is enabled and encryption is complete'
           ELSE 'No State'
       END,
       percent_complete, encryptor_type, key_algorithm, key_length, modify_date
FROM sys.dm_database_encryption_keys
```





Backup and restore of TDE-encrypted database

To restore an existing TDE-encrypted database, the required TDE certificate must first be [imported](#)  into the SQL Managed Instance.

When using customer-managed key (BYOK):

- Keep previous versions of the key in the key vault when rotating keys, so older database backups can be restored. When the TDE protector is changed for a database, old backups of the database are not updated to use the latest TDE protector. At restore time, each backup needs the TDE protector it was encrypted with at creation time.
- Keep all previously used keys in AKV even after switching to service-managed keys. It ensures database backups can be restored with the TDE protectors stored in AKV. TDE protectors created with Azure Key Vault must be maintained until all remaining stored backups have been created with service-managed keys. Make recoverable backup copies of these keys using [Backup-AzKeyVaultKey](#) .

Resources

- [Transparent Data Encryption](#) 
- [BYOK Transparent Data Encryption](#) 
- [Configure TDE with BYOK for SQL Managed Instance](#) 
- [Migrate a certificate of a TDE-protected database to Azure SQL Managed Instance](#) 

Public Doc Reference

Transparent data encryption for SQL Database and Data Warehouse

<https://docs.microsoft.com/en-us/azure/sql-database/transparent-data-encryption-azure-sql?tabs=azure-portal>

Azure SQL Transparent Data Encryption with customer-managed key

<https://docs.microsoft.com/en-us/azure/sql-database/transparent-data-encryption-byok-azure-sql>

Manage Transparent Data Encryption in a Managed Instance using your own key from Azure Key Vault

<https://docs.microsoft.com/en-us/azure/sql-database/scripts/transparent-data-encryption-byok-sql-managed-instance-powershell?toc=%2fpowershell%2fmodule%2ftoc.json>

Rotate the Transparent Data Encryption (TDE) protector using PowerShell

<https://docs.microsoft.com/en-us/azure/sql-database/transparent-data-encryption-byok-azure-sql-key-rotation?tabs=azure-powershell>

Remove a Transparent Data Encryption (TDE) protector using PowerShell

<https://docs.microsoft.com/en-us/azure/sql-database/transparent-data-encryption-byok-azure-sql-remove-tde-protector?tabs=azure-powershell>

Migrate certificate of TDE protected database to Azure SQL Database Managed Instance

<https://docs.microsoft.com/en-us/azure/sql-database/sql-database-managed-instance-migrate-tde-certificate>

How good have you found this content?

