Integrating SQL EKM with Azure Cloud HSM

For additional information about SQL TDE please reference Enable TDE on SQL Server using EKM.

Table of Contents

Summary

Azure Cloud HSM can be used to enhance the security of databases by providing dedicated hardware-based key management and cryptographic operations. It allows customers to encrypt their databases, ensuring that the data remains protected, even if the underlying infrastructure is compromised. Azure Cloud HSM supports SQL EKM integration. With MSSQL TDE, the database software encrypts data before storing it on disk.

The data in the database is encrypted with a key. These keys are encrypted with the TDE master encryption key. Customers can store the TDE master encryption key in our Azure Cloud HSM, which provides additional security.

Important Note: The Azure Cloud HSM SQL EKM provider is a DLL designed to implement the SQL Server EKM interface. This enables SQL Server to establish communication with your Azure Cloud HSM. The Azure Cloud HSM SQL EKM specifically supports SQL IaaS scenarios only, where SQL is running within a virtual machine. If you are working with SaaS/Pass environments, such as Azure SQL DB or Azure SQL Managed Instance, it is necessary to utilize Azure Managed HSM.

Prerequisites

The following prerequisites are required to support the Azure Cloud HSM SQL EKM. Please reference the Azure Cloud HSM Onboarding Guide for SDK Installation and Azure Cloud HSM configuration if you have not completed your HSM deployment.

System Requirements

- Windows Server (2016, 2019, 2022)
- Azure Cloud HSM Client SDK
- Azure Cloud HSM resource has been deployed, initialized, and configured.

SQL EKM Requirements:

- Copy of partition owner certificate “PO.crt” on SQL server.
- Known address of your HSM “hsm1.chsm-<resourcename>-<uniquestring>.privatelink.cloudhsm.azure.net”.
- Knowledge of Crypto User credentials
- SQL Server 2017, 2019, 2022 Image on Windows Server VM

SQL Server Installation

Reference: Overview of SQL Server on Azure Windows Virtual Machines

For simplification in this integration guide, we are using SQL Server Image on Windows Server as the Admin VM and SQL Server VM. For production you may choose to have a separate Admin VM and SQL Server VM. If you operate a separate SQL Server VM then you will need to download and install the Azure SDK onto your SQL Server VM, copy the PO.crt (partition owner certificate), update the %HostName% for the 2 azcloudhsm_resource.cfg file locations under /client and /mgmt_util and ensure that azcloudhsm_client.exe is always running as a service.

Configure the Azure Cloud HSM Client Tools on SQL Server VM

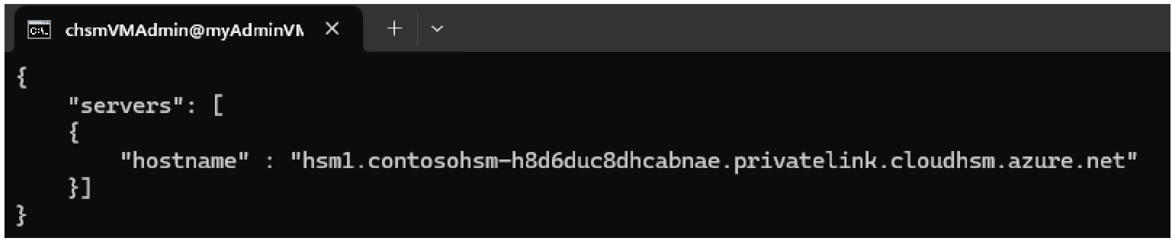
In this instance, we will verify that the Azure Cloud HSM SDK is correctly installed and set up on your Administrative Virtual Machine (SQL Server VM). Ensure that your current Admin VM has the partition owner certificate (PO.crt) as well as the Azure Cloud HSM SDK tools and utilities, previously employed to initialize and configure your Azure Cloud HSM. Following that, we'll proceed to install and configure SQL EKM on your Admin VM.

Important Note: The SQL Server VM should be deployed in the same region as your existing Cloud HSM and existing private VNET and subnet you created earlier. Customers using any version of Windows Server should install the most recent version of the Visual C++ Redistributable.

STEP 1: Validate Azure Cloud HSM SDK is configured.

You will need to update the "hostname" property in two locations, \client\azcloudhsm_resource.cfg and \mgmt_util\azcloudhsm_resource.cfg. The azcloudhsm_resource.cfg is used to initialize the Azure Cloud HSM cluster. It must point to the private link FQDN for "hsm1" before the cluster is fully initialized as only hsm1 is currently running.

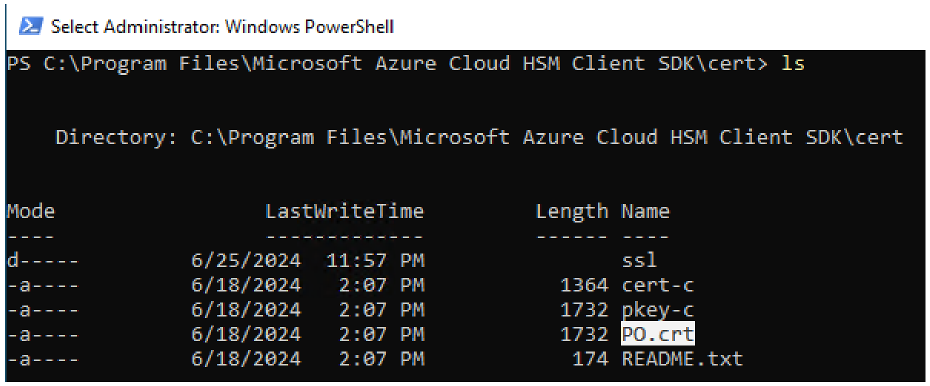
To find the FQDN that includes private link go to your resource group, and select Private Endpoint then select DNS Configuration. You will see configuration name and the FQDN that includes *.privatelink.cloudhsm.azure.net for each HSM instance. Use "hsm1" FQDN as the hostname for the \bin\azcloudhsm_resource.cfg file.



```
{
    "servers": [
    {
        "hostname" : "hsm1.contosohsm-h8d6duc8dhcabnae.privatelink.cloudhsm.azure.net"
    }]
}
```

STEP 2: Validate PO.crt exist on your SQL Server VM

In the /cert directory, ensure the existence of the PO.crt file. If this file is not found, consult the Azure Cloud HSM Onboarding Guide for SDK Installation and Azure Cloud HSM configuration. This step is crucial if your HSM deployment is not initialized, and you are running SQL Server from your Admin VM. Failure to activate your HSM or neglecting to create and copy the PO.crt file post-activation to your VM running SQL Server will result in an error when executing azcloudhsm_mgmt_util or azcloudhsm_client.


Select Administrator: Windows PowerShell

```
PS C:\Program Files\Microsoft Azure Cloud HSM Client SDK\cert> ls


    Directory: C:\Program Files\Microsoft Azure Cloud HSM Client SDK\cert


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         6/25/2024  11:57 PM                ssl
-a----         6/18/2024   2:07 PM           1364 cert-c
-a----         6/18/2024   2:07 PM           1732 pkey-c
-a----         6/18/2024   2:07 PM           1732 PO.crt
-a----         6/18/2024   2:07 PM            174 README.txt
```

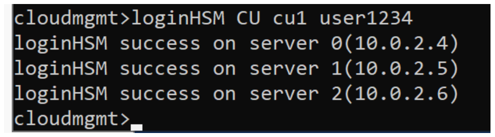
STEP 3: Validate access to your Azure Cloud HSM

1. Start the azcloudhsm_mgmt_util by executing:

```
.\azcloudhsm_mgmt_util.exe .\azcloudhsm_resource.cfg
```

2. Logon as CU using the username and password when you created a ‘user’ with the crypto user role.

```
loginHSM CU cu1 user1234
```



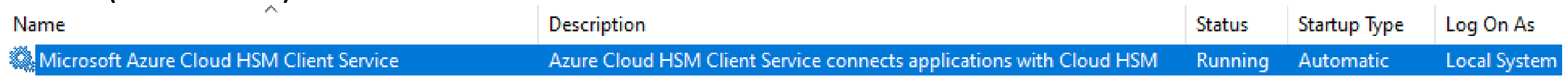
```
cloudmgmt>loginHSM CU cu1 user1234
loginHSM success on server 0(10.0.2.4)
loginHSM success on server 1(10.0.2.5)
loginHSM success on server 2(10.0.2.6)
cloudmgmt>_
```

STEP 4: Start Azure Cloud HSM Client

If you installed the Azure Cloud HSM SDK using MSI, the client is already configured to run as a service. If the client is not running, you can open Services.msc, right-click on the Microsoft Azure Cloud HSM Client Service, and select "Start."

Important Note: The Azure Cloud HSM client must always be running for SQL EKM to be operational.

Windows (Recommended):



Name	Description	Status	Startup Type	Log On As
Microsoft Azure Cloud HSM Client Service	Azure Cloud HSM Client Service connects applications with Cloud HSM	Running	Automatic	Local System

Windows (Manual):
Start the client daemon if it is not running.

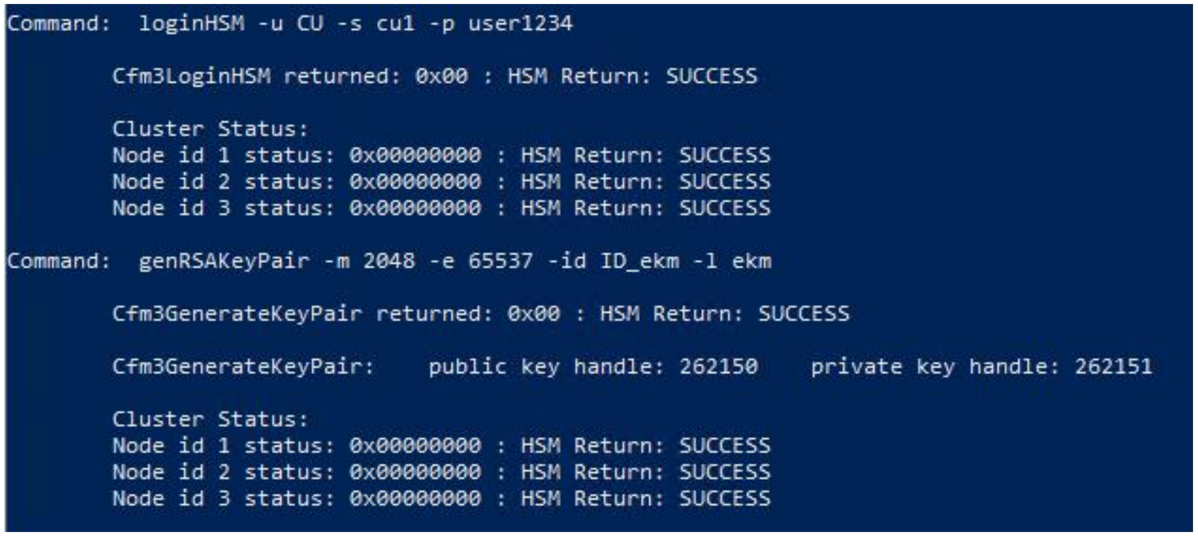
```
cd C:\Program Files\Microsoft Azure Cloud HSM Client SDK
.\azcloudhsm_client.exe .\azcloudhsm_resource.cfg
```

Configure SQL EKM for Azure Cloud HSM

STEP 5: Create a Key for use by SQL EKM.

A key must be created for use by SQL EKM. From a new PowerShell window, navigate to C:\Program Files\Microsoft Azure Cloud HSM Client SDK\utils\azcloudhsm_util and execute azcloudhsm_util.exe to generate a new key for SQL EKM. Below will create an asymmetric key with the id ID_ekm and label ekm.

```
    .\azcloudhsm_util.exe
    loginHSM -u CU -s cu1 -p user1234
    genRSAKeyPair -m 2048 -e 65537 -id ID_ekm -l ekm
```



```
Command:  loginHSM -u CU -s cu1 -p user1234

        Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS

        Cluster Status:
        Node id 1 status: 0x00000000 : HSM Return: SUCCESS
        Node id 2 status: 0x00000000 : HSM Return: SUCCESS
        Node id 3 status: 0x00000000 : HSM Return: SUCCESS

Command:  genRSAKeyPair -m 2048 -e 65537 -id ID_ekm -l ekm

        Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 262150    private key handle: 262151

        Cluster Status:
        Node id 1 status: 0x00000000 : HSM Return: SUCCESS
        Node id 2 status: 0x00000000 : HSM Return: SUCCESS
        Node id 3 status: 0x00000000 : HSM Return: SUCCESS
```

STEP 6: Enable SQL EKM provider.

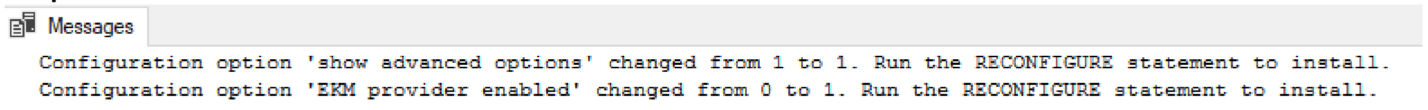
If you installed the Azure Cloud HSM SDK using MSI, the client is already configured to run as a service and the Azure Cloud HSM SQL EKM provider has been automatically registered. Next steps are to enable the EKM provider within Microsoft SQL Server.

The following enables the display of advanced options, then sets the EKM provider to be enabled, and finally applies these changes. The ability to enable advanced options and the EKM provider is used when configuring SQL Server for specific features or integrations, such as External Key Management for encryption operations. Always exercise caution when modifying advanced configurations and ensure that it aligns with your system's security and operational requirements.

Open SQL Server Management Studio (SSMS): Launch SSMS and connect to your SQL Server instance by providing the necessary server name, authentication method, and credentials. Then proceed forward with enabling the SQL EKM provider.

```
-- Enable advanced options.
USE master;
GO

EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE;
GO

-- Enable EKM provider
EXEC sp_configure 'EKM provider enabled', 1;
GO
RECONFIGURE;
```

Output:



Messages

```
Configuration option 'show advanced options' changed from 1 to 1. Run the RECONFIGURE statement to install.
Configuration option 'EKM provider enabled' changed from 0 to 1. Run the RECONFIGURE statement to install.
```

STEP 7: Restart SQL Server

Important Note: *After registering and enabling SQL EKM provider you must restart SQL Server. If you fail to restart SQL Server, you will get an error message when you attempt to Create EKM provider in the next step.*

STEP 8: Create an EKM Cryptographic provider.

You will need to create a cryptographic provider named 'CloudHsm_EKM' and specify the location of the Azure Cloud HSM SQL EKM DLL file that implements the Cloud HSM SQL EKM functionality. This is a crucial step in setting up SQL Server to work with an external key management system for cryptographic operations.

```
-- Create a Cloud HSM EKM cryptographic provider
CREATE CRYPTOGRAPHIC PROVIDER CloudHsm_EKM
FROM FILE = 'C:\Program Files\Microsoft Azure Cloud HSM Client SDK\libs\ekm\azcloudhsm_ekm.dll';
GO
```

STEP 9: Create an EKM Credential

Next you will establish a credential for the Azure Cloud HSM SQL EKM provider, providing the necessary identity and secret. It then associates this credential with a specified SQL Server login associated with a domain account. This is a common practice for integrating SQL Server with external key management systems for cryptographic operations.

Replace {domain\username} with your domain and username and run the following to create a Cloud HSM EKM provider. In this example we are using cu1:user1234 as our Identity and Secret.

Important Note: *You can only assign one credential to a login / SQL connection.*

```
-- Create Cloud HSM EKM credential
USE master;
CREATE CREDENTIAL sysadmin_cloudhsm_ekm_cred
   WITH IDENTITY = 'cu1',
   SECRET = 'user1234' -- password goes here
FOR CRYPTOGRAPHIC PROVIDER CloudHSM_EKM;

-- Add the credential to the SQL Server administrator's domain login
ALTER LOGIN [{domain\username}]
```

```
ADD CREDENTIAL sysadmin_cloudhsm_ekm_cred;
```

STEP 10: Create an Asymmetric Key from an Existing HSM Key

The following sets up an asymmetric key in SQL Server by opening an existing key from the Azure Cloud HSM SQL EKM provider. It creates a credential associated with the key, a login associated with the asymmetric key, and then associates a credential with the login. These steps are part of configuring SQL Server to work with external key management for cryptographic operations. Below an asymmetric key in SQL is created from an existing HSM key with the ID ID_ekm.

```
-- Open your Azure Cloud HSM key in your SQL Server instance.
CREATE ASYMMETRIC KEY CloudHSMEKMSampleASYKey
FROM PROVIDER [CloudHSM_EKM]
WITH PROVIDER_KEY_NAME = 'ID_ekm',
CREATION_DISPOSITION = OPEN_EXISTING;

-- Create Cloud HSM EKM credential to use with the key
USE master;
CREATE CREDENTIAL sysadmin_cloudhsm_ekm_cred_for_key
   WITH IDENTITY = 'cu1',
   SECRET = 'user1234' -- password goes here
FOR CRYPTOGRAPHIC PROVIDER CloudHSM_EKM;

-- Create a Login that will associate the asymmetric key to this login
CREATE LOGIN CloudHSM_TDE_Login
FROM ASYMMETRIC KEY CloudHSMEKMSampleASYKey;

-- Now add the credential mapping to the new Login
ALTER LOGIN CloudHSM_TDE_Login
ADD CREDENTIAL sysadmin_cloudhsm_ekm_cred_for_key;
```

Database Encryption

SQL Server's Transparent Data Encryption (TDE) safeguards a database's encryption key through the utilization of an asymmetric key stored in an extensible key management (EKM) module using Transact-SQL. TDE secures the storage of an entire database by employing a symmetric key known as the database encryption key.

Enable Database Encryption on SQL Server

The T-SQL below is setting up a new database, creating a table within that database, creating an encryption key for TDE using an Azure Cloud HSM asymmetric key, and then enabling TDE for the database as TDE provides an additional layer of security by encrypting the data files and backups of the database.

```sql
--Create a database that will be encrypted with your Azure Cloud HSM key
CREATE DATABASE CloudHSM_TestTDE;
GO

CREATE TABLE [CloudHSM_TestTDE].[dbo].[TestTdeTable](
    [test] [nchar](10) NULL
) ON [PRIMARY]
GO

USE CloudHSM_TestTDE;
--Create an ENCRYPTION KEY using the ASYMMETRIC KEY (EKMSampleASYKey)
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_256
ENCRYPTION BY SERVER ASYMMETRIC KEY CloudHSMEKMSampleASYKey;
GO

--Enable TDE by setting ENCRYPTION ON
ALTER DATABASE CloudHSM_TestTDE
SET ENCRYPTION ON;
```

Appendix

Frequently Asked Questions

- **Does Azure Cloud HSM SQL EKM Provider support Symmetric Keys?**
  No. Microsoft SQL Server does not support symmetric keys for SQL TDE. Only Asymmetric keys are supported!

- **Does Azure Cloud HSM SQL EKM Provider support SQL Server on Linux?**
  No. Azure Cloud HSM SQL EKM provider only supports Windows Server.

- **Does Azure Cloud HSM SQL EKM Provider support SQL Server SaaS/PaaS?**
  No. Azure Cloud HSM is IaaS only. Support for SaaS/PaaS customers should use Azure Managed HSM which supports SQL TDE for Azure SQL DB and Azure SQL Managed Instance.

- **Does Azure Cloud HSM SQL EKM support multiple logins, credentials, connections?**
  Azure Cloud HSM SQL EKM the login or SQL connection is limited and can only have a single credential assigned.

- **Why am I getting an *INF: shutdown_ssl_socket: SSL_shutdown sent close_notify alert* when running azcloudhsm_client?**
  This error commonly occurs when a customer created and initialized their Azure Cloud HSM from another VM and attempts to install the Cloud HSM Client SDK and run from another VM that is missing or does not have the correct PO.crt from the Admin VM you initialized from. If you copy the PO.crt from your Admin VM to your new VM and rerun the azcloudhsm_client you should see a successful connection to your HSM.

- **Why am I getting Error: Cannot initialize cryptographic provider. Provider error code: 1. (Failure - Consult EKM Provider for details)?**
  This error commonly occurs when the ‘Azure Cloud HSM MSI’ is run after SQL Server is installed. This error goes away once you restart SQL Server as it will then pick up the environment variables and settings created when the Azure Cloud HSM MSI was executed.

- **Why do I get an Access Violation and 'Error: Cannot continue execution, session in kill state' when creating an Asymmetric Key from an HSM key?**
  This error occurs if the Visual C++ Redistributable is missing or outdated on your host, for which the Azure Cloud HSM SQL EKM provider is expecting. Upgrading Visual C++ Redistributable and restarting the host should resolve it.

Common T-SQL Task Examples

Check TDE encryption state of a specific database.

The T-SQL below queries the sys.databases view to retrieve the name and encryption state (is_encrypted) of the database named 'CloudHSM_TestTDE'. The result will indicate whether the specified database is encrypted using Transparent Data Encryption (TDE) or not. The is_encrypted column will have a value of 1 if the database is encrypted and 0 if it is not encrypted.

```
-- Check TDE encryption state of a specific database
SELECT name, is_encrypted
FROM sys.databases
WHERE name = 'CloudHSM_TestTDE';
```

View TDE status of databases.

The T-SQL below queries the dynamic management view to get a list of databases and their respective encryption states. The result will include the name of each database (DatabaseName) and the encryption state (encryption_state). The encryption state can have values like 0 (No database encryption), 1 (Database is encrypted), and 2 (Change in progress).

```
-- View TDE status of databases
SELECT DB_NAME(database_id) AS DatabaseName, encryption_state
FROM sys.dm_database_encryption_keys;
```

Encrypting and Decrypting Data with an Asymmetric Key

The T-SQL below showcases how to use asymmetric key-based encryption and decryption functions in SQL Server. In this example it uses the 'CloudHSMEKMSampleASYKey' asymmetric key to encrypt and then decrypt a sample data value (0x010203). This process is often used for securing sensitive information within the database.

```
DECLARE @encrypted_data varbinary(256)

-- Encrypt data
USE [master]
SELECT @encrypted_data = EncryptByAsymKey(AsymKey_Id('CloudHSMEKMSampleASYKey'), 0x010203)
SELECT @encrypted_data

```

```
-- Decrypt data
SELECT DecryptByAsymKey(AsymKey_Id('CloudHSMEKMSampleASYKey'), @encrypted_data)
GO

-- or --
SELECT DecryptByAsymKey(AsymKey_Id('CloudHSMEKMSampleASYKey'), EncryptByAsymKey(AsymKey_Id('CloudHSMEKMSampleASYKey'), 0x010203))
GO
```

Disable Database Encryption on SQL Server

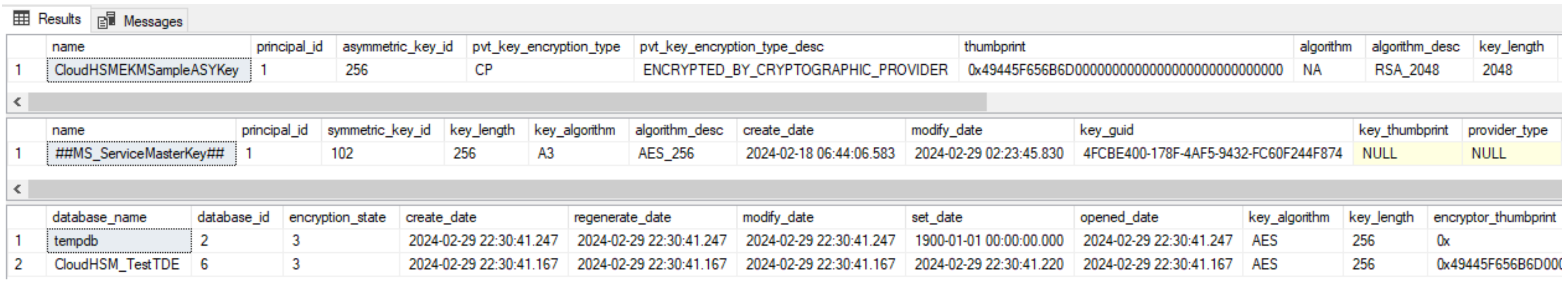
The T-SQL below disables TDE for the specified database. It's important to note that disabling TDE involves decrypting the entire database, and this process might take some time, depending on the size of the database. Additionally, it's crucial to ensure that proper security measures are in place, as disabling encryption may expose previously encrypted data in an unencrypted state. Always exercise caution when making changes to encryption settings and be sure to follow best practices and security guidelines.

```
--Disable TDE by setting ENCRYPTION OFF
ALTER DATABASE CloudHSM_TestTDE
SET ENCRYPTION OFF;
```

List all Symmetric and Asymmetric Keys

The T-SQL below provides a comprehensive view of the keys present in the 'master' database, which is the system database in SQL Server. The results will include information about any asymmetric or symmetric keys that have been created in the database, including those used for various cryptographic operations, and information about database encryption keys associated with databases in the SQL Server instance.

```
-- Asymmetric keys, symmetric keys, and database encryption keys
SELECT * FROM master.sys.asymmetric_keys
SELECT * FROM master.sys.symmetric_keys
SELECT dbs.name AS [database_name], deks.* FROM master.sys.dm_database_encryption_keys AS deks LEFT JOIN master.sys.databases AS dbs ON
deks.database_id = dbs.database_id
GO
```


Results | Messages

	name	principal_id	asymmetric_key_id	pvt_key_encryption_type	pvt_key_encryption_type_desc	thumbprint	algorithm	algorithm_desc	key_length
1	CloudHSMEKMSampleASYKey	1	256	CP	ENCRYPTED_BY_CRYPTOGRAPHIC_PROVIDER	0x49445F656B6D0000000000000000000000000000	NA	RSA_2048	2048

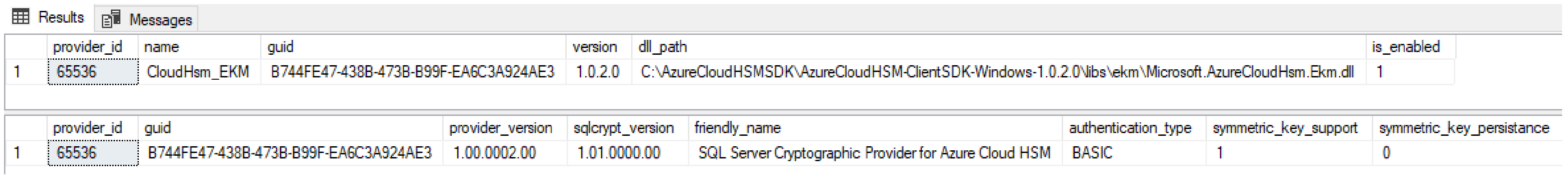
	name	principal_id	symmetric_key_id	key_length	key_algorithm	algorithm_desc	create_date	modify_date	key_guid	key_thumbprint	provider_type
1	##MS_ServiceMasterKey##	1	102	256	A3	AES_256	2024-02-18 06:44:06.583	2024-02-29 02:23:45.830	4FCBE400-178F-4AF5-9432-FC60F244F874	NULL	NULL

	database_name	database_id	encryption_state	create_date	regenerate_date	modify_date	set_date	opened_date	key_algorithm	key_length	encryptor_thumbprint
1	tempdb	2	3	2024-02-29 22:30:41.247	2024-02-29 22:30:41.247	2024-02-29 22:30:41.247	1900-01-01 00:00:00.000	2024-02-29 22:30:41.247	AES	256	0x
2	CloudHSM_TestTDE	6	3	2024-02-29 22:30:41.167	2024-02-29 22:30:41.167	2024-02-29 22:30:41.167	2024-02-29 22:30:41.220	2024-02-29 22:30:41.167	AES	256	0x49445F656B6D00(

List of all cryptographic providers.

The T-SQL below queries system views and dynamic management views to retrieve detailed information about cryptographic providers and their properties within the SQL Server instance. The results will provide insights into the currently configured cryptographic providers and their associated properties.

```
-- List the providers and provider properties
SELECT * FROM sys.cryptographic_providers;
SELECT * FROM sys.dm_cryptographic_provider_properties
GO
```


Results | Messages

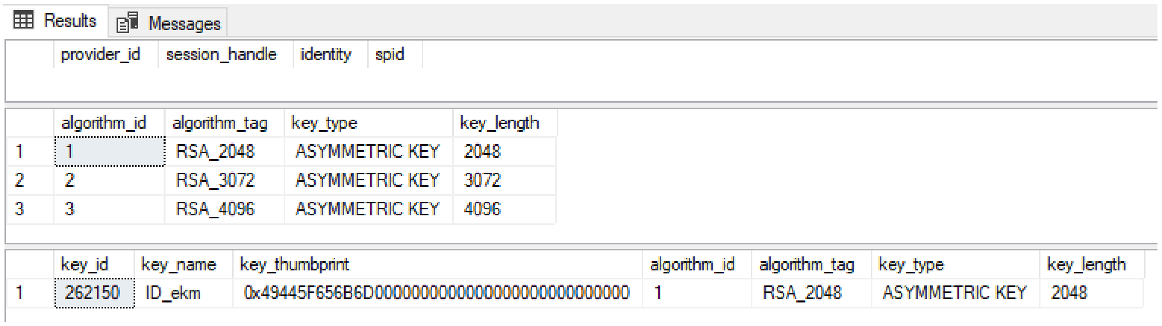
	provider_id	name	guid	version	dll_path	is_enabled
1	65536	CloudHsm_EKM	B744FE47-438B-473B-B99F-EA6C3A924AE3	1.0.2.0	C:\AzureCloudHSMSDK\AzureCloudHSM-ClientSDK-Windows-1.0.2.0\libs\ekm\Microsoft.AzureCloudHsm.Ekm.dll	1

	provider_id	guid	provider_version	sqlcrypt_version	friendly_name	authentication_type	symmetric_key_support	symmetric_key_persistance
1	65536	B744FE47-438B-473B-B99F-EA6C3A924AE3	1.00.0002.00	1.01.0000.00	SQL Server Cryptographic Provider for Azure Cloud HSM	BASIC	1	0

List all detailed information about sessions, algorithms, and keys.

The T-SQL below retrieves detailed information about sessions, algorithms, and keys for the Cloud HSM EKM provider in SQL Server, leveraging dynamic management views specific to cryptographic providers. The results provide insights into the state and configuration of the cryptographic operations managed by the Cloud HSM EKM provider.

```
-- Sessions, algorithms, and keys for the Cloud HSM EKM provider
DECLARE @provider_id int
SELECT @provider_id = provider_id FROM sys.cryptographic_providers WHERE guid = 'B744FE47-438B-473B-B99F-EA6C3A924AE3';
SELECT * FROM sys.dm_cryptographic_provider_sessions(@provider_id);
SELECT * FROM sys.dm_cryptographic_provider_algorithms(@provider_id);
SELECT * FROM sys.dm_cryptographic_provider_keys(@provider_id);
GO
```



Results | Messages

	provider_id	session_handle	identity	spid

	algorithm_id	algorithm_tag	key_type	key_length
1	1	RSA_2048	ASYMMETRIC KEY	2048
2	2	RSA_3072	ASYMMETRIC KEY	3072
3	3	RSA_4096	ASYMMETRIC KEY	4096

	key_id	key_name	key_thumbprint	algorithm_id	algorithm_tag	key_type	key_length
1	262150	ID_ekm	0x49445F656B6D0000000000000000000000000000	1	RSA_2048	ASYMMETRIC KEY	2048

List all cryptographic providers associated with credentials and logins.

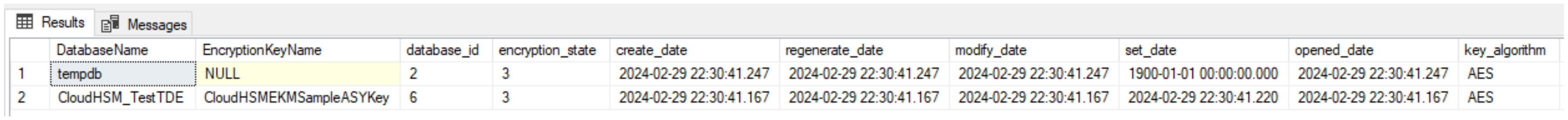
The T-SQL below provides insights into credentials, cryptographic providers associated with credentials, and logins in the SQL Server instance. The left join in the first query allows you to see which cryptographic providers are associated with each credential. The second query lists information about logins, which are crucial for controlling access to the SQL Server instance.

```
-- List credentials, logins
SELECT creds.*, provs.name AS [target_provider_name] from sys.credentials as creds LEFT JOIN master.sys.cryptographic_providers AS provs ON
creds.target_id = provs.provider_id
SELECT * FROM sys.syslogins
GO
```

List all SQL Database encryption keys.

The T-SQL below retrieves information about database encryption keys, including the associated database name, the name of the asymmetric key used for encryption, and other details. The left join allows you to include information about the asymmetric key associated with each encryption key.

```
-- Get all db encryption keys
SELECT DB_NAME(dbk.database_id) AS DatabaseName, keys.name AS EncryptionKeyName, dbk.* FROM master.sys.dm_database_encryption_keys as dbk
LEFT JOIN master.sys.asymmetric_keys AS keys ON dbk.encryptor_thumbprint = keys.thumbprint
```



	DatabaseName	EncryptionKeyName	database_id	encryption_state	create_date	regenerate_date	modify_date	set_date	opened_date	key_algorithm
1	tempdb	NULL	2	3	2024-02-29 22:30:41.247	2024-02-29 22:30:41.247	2024-02-29 22:30:41.247	1900-01-01 00:00:00.000	2024-02-29 22:30:41.247	AES
2	CloudHSM_TestTDE	CloudHSMEKMSampleASYKey	6	3	2024-02-29 22:30:41.167	2024-02-29 22:30:41.167	2024-02-29 22:30:41.167	2024-02-29 22:30:41.220	2024-02-29 22:30:41.167	AES

Rotate the TDE encryption key.

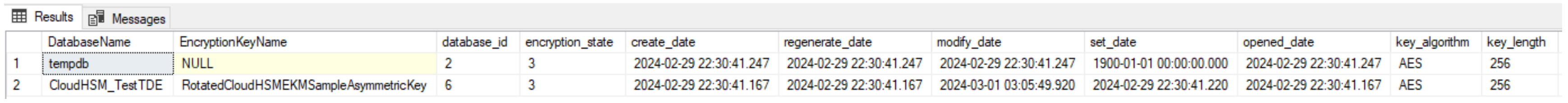
Firstly, as best practice when rotating the TDE encryption key you will create a new HSM key. From a new PowerShell window, navigate to C:\Program Files\Microsoft Azure Cloud HSM Client SDK\utils\azcloudhsm_util and execute azcloudhsm_util.exe to generate a new key for SQL EKM. Below will create an asymmetric key with the id ID_rotated_ekm and label rotated_ekm.

```
.\azcloudhsm_util.exe
loginHSM -u CU -s cu1 -p user1234
genRSAKeyPair -m 2048 -e 65537 -id ID_rotated_ekm -l rotated_ekm
```

The T-SQL below executes a key rotation process with CloudHSM_EKM for TDE in SQL Server, including creating a new asymmetric key, associating it with a login, and updating the database encryption key. Always ensure that you follow the best practices for key management and security, and thoroughly test any changes in a controlled environment before applying them to production systems when performing key rotations.

Important Note: For Azure Cloud HSM SQL EKM the login or SQL connection is limited to having a single credential assigned. Currently, SQL Server already has an association when TDE is enabled on your database. When rotating the TDE encryption key using a new key, you must create a new credential name and a new login name for the process to take effect.

```sql
-- Create new Asymmetric key
CREATE ASYMMETRIC KEY RotatedCloudHSMEKMSampleAsymmetricKey
FROM PROVIDER [CloudHSM_EKM]
WITH PROVIDER_KEY_NAME = 'ID_rotated_ekm',
CREATION_DISPOSITION = OPEN_EXISTING;
GO

-- Create Cloud HSM EKM credential to use with the key
USE master;
CREATE CREDENTIAL sysadmin_cloudhsm_ekm_rotated_cred_for_key
   WITH IDENTITY = 'cu1',
    SECRET = 'user1234' -- password goes here
FOR CRYPTOGRAPHIC PROVIDER CloudHSM_EKM;
GO

-- Create a new login that will associate the asymmetric key to this login
CREATE LOGIN CloudHSM_TDE_Rotated_Login
FROM ASYMMETRIC KEY RotatedCloudHSMEKMSampleAsymmetricKey;
GO

-- Now add the credential mapping to the new Login
ALTER LOGIN CloudHSM_TDE_Rotated_Login
ADD CREDENTIAL sysadmin_cloudhsm_ekm_rotated_cred_for_key;
GO

USE CloudHSM_TestTDE;
ALTER DATABASE ENCRYPTION KEY ENCRYPTION BY SERVER ASYMMETRIC KEY RotatedCloudHSMEKMSampleAsymmetricKey;
GO

-- Validate rotation by viewing all db encryption keys
SELECT DB_NAME(dbk.database_id) AS DatabaseName, keys.name AS EncryptionKeyName, dbk.* FROM master.sys.dm_database_encryption_keys as dbk
LEFT JOIN master.sys.asymmetric_keys AS keys ON dbk.encryptor_thumbprint = keys.thumbprint
```



Results | Messages

	DatabaseName	EncryptionKeyName	database_id	encryption_state	create_date	regenerate_date	modify_date	set_date	opened_date	key_algorithm	key_length
1	tempdb	NULL	2	3	2024-02-29 22:30:41.247	2024-02-29 22:30:41.247	2024-02-29 22:30:41.247	1900-01-01 00:00:00.000	2024-02-29 22:30:41.247	AES	256
2	CloudHSM_TestTDE	RotatedCloudHSMEKMSampleAsymmetricKey	6	3	2024-02-29 22:30:41.167	2024-02-29 22:30:41.167	2024-03-01 03:05:49.920	2024-02-29 22:30:41.220	2024-02-29 22:30:41.167	AES	256

Remove Asymmetric Key

The T-SQL below is cleaning up various security-related objects in the SQL Server instance. It starts by dropping the asymmetric key 'CloudHSMEKMSampleASYKey,' then removes the credential mapping associated with the login 'CloudHSM_TDE_Login,' drops the login itself, and finally removes the credential 'sysadmin_cloudhsm_ekm_cred_for_key.' These actions are often part of security maintenance or cleanup processes. Always exercise caution when dropping security-related objects, as they may be critical for certain functionalities or encryption mechanisms.

```
DROP ASYMMETRIC KEY CloudHSMEKMSampleASYKey

-- Now drop the credential mapping from the original association
ALTER LOGIN [CloudHSM_TDE_Login]
DROP CREDENTIAL sysadmin_cloudhsm_ekm_cred_for_key;
GO

-- and drop the login
DROP LOGIN [CloudHSM_TDE_Login]

-- and the credential
DROP CREDENTIAL sysadmin_cloudhsm_ekm_cred_for_key;
GO
```

Remove SQL EKM provider.

The T-SQL below DROP CRYPTOGRAPHIC PROVIDER CloudHsm_EKM is used to drop (remove or delete) the Azure Cloud HSM SQL EKM provider named 'CloudHsm_EKM' from the SQL Server instance.

```
DROP CRYPTOGRAPHIC PROVIDER CloudHsm_EKM
```

Important Note: When you drop a cryptographic provider, you are essentially removing its association with SQL Server. This could be done for various reasons, such as discontinuing the use of a specific external key management system or changing the configuration of the SQL Server instance. Before executing this statement, it's important to ensure that dropping the cryptographic provider won't negatively impact any databases or features relying on it. If there are databases using keys managed by the Azure Cloud HSM SQL EKM provider, those keys might become inaccessible or unusable after dropping the provider.

It's recommended to carefully review all documentation, backup critical data, and coordinate any changes to cryptographic providers with your organization's security and database administration policies. Also, ensure you have the necessary permission to perform this action, as it involves making significant changes to the SQL Server configuration.