Integrating Active Directory Certificate Services with Azure Cloud HSM

Table of Contents

Summary

To use ADCS backed by a certificate stored in Azure Cloud HSM, the Azure Cloud HSM Client SDK must be installed and configured as a Key Storage Provider. ADCS must be configured to use the Azure Cloud HSM Client Key Storage Provider to store the CA private key.

For general information about Active Directory Certificate Services (ADCS) refer to:
https://learn.microsoft.com/en-us/windows-server/identity/ad-cs/

Prerequisites

System Requirements

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

ADCS Install Prerequisites

- Copy of partition owner certificate “PO.crt” on ADCS server.
- Known address of your HSM “hsm1.chsm-<resourcename>-<uniquestring>.privatelink.cloudhsm.azure.net”.
- Knowledge of Crypto User credentials

Important Note: The Windows Server for ADCS 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.

Important Note: The Azure Cloud HSM SDK must be installed before setting up ADCS. The Cloud HSM SDK registers both KSP and CNG providers during installation. Active Directory Certificate Services (ADCS) enumerates cryptographic providers only at CA installation time and does not update this list dynamically. ADCS also applies aggressive filtering when displaying providers.

If the Cloud HSM SDK is installed after ADCS, its providers will not appear or be selectable in ADCS. To resolve this, uninstall ADCS, install the Azure Cloud HSM SDK first, and then set up ADCS to ensure the KSP and CNG providers are available.

Option 1: ADCS Installation on existing Azure Cloud HSM Admin VM

Customers that deploy ADCS following option 1 will install and configure ADCS on their existing Admin VM which already has the partition owner certificate PO.crt and Azure Cloud HSM SDK tools and utilities which they used to initialize and configure their Azure Cloud HSM. This may be the easier path for customers as all Azure Cloud HSM SDK tools and certs are already present on your Admin VM and it reduces your need for additional Azure resources and configuration steps.

➔ Go to Windows Server Prerequisites for the next steps.

Option 2: ADCS Installation of a new Windows Server VM

Customers that deploy ADCS following option 2 chose to install and configure ADCS on a separate, new Windows Server VM. You will need to complete the prerequisite steps below on your new Windows Server before proceeding with ADCS installation and configuration. You will also need to copy from your Admin VM your PO.crt file to the /Cert directory where you extracted the Cloud HSM SDK on your ADCS server.

Option 2: Azure Cloud HSM SDK Prerequisites

STEP 1: Download and Install Azure Cloud HSM SDK on to your ADCS Server

Customers can download the Azure Cloud HSM SDK and Client Tools from GitHub:
@ [microsoft/MicrosoftAzureCloudHSM: Azure Cloud HSM SDK]

- To install the Azure Cloud HSM SDK for Windows, download and run the Azure Cloud HSM SDK MSI (Windows Installer Package). This installer will automatically configure all necessary environment variables and dependencies and set up the Cloud HSM Client to run as a service. In the example below, running the Cloud HSM MSI deploys the SDK to C:\Program Files\Microsoft Azure Cloud HSM Client SDK. This is the directory for all Azure Cloud HSM management and client utilities including CNG and KSP providers which can be found under \libs\cng and \libs\ksp.

```
PS C:\Program Files\Microsoft Azure Cloud HSM Client SDK\libs> dir .\cng\


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


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/18/2024   2:39 PM         155568 azcloudhsm_cng_config.exe
-a----         6/18/2024   2:40 PM        3334176 azcloudhsm_cng_provider.dll


PS C:\Program Files\Microsoft Azure Cloud HSM Client SDK\libs> dir .\ksp\


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


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         6/18/2024   2:40 PM        3279904 azcloudhsm_ksp.dll
-a----         6/18/2024   2:39 PM         154032 azcloudhsm_ksp_client.exe
-a----         6/18/2024   2:41 PM         298528 azcloudhsm_ksp_import_key.exe
```

STEP 2: Configure the Azure Cloud HSM Client Tools

You will need to update the "hostname" property in the \bin\azcloudhsm_resource.cfg file. 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.

azcloudhsm_mgmt_util.cfg - Notepad

File Edit Format View Help

```
{
        "servers": [
        {
                "hostname" : "hsm1.contoso-aycya2e2fsb9fxhu.privatelink.cloudhsm.azure.net",
                "port" : 2225,
                "certificate": "..\\cert\\cert-c",
                "pkey": "..\\cert\\pkey-c",
                "CAfile": "",
                "CApath": "..\\cert\\ssl",
                "ssl_ciphers": "default",
                "server_ssl" : "yes",
        "enable"    : "yes",
                "owner_cert_path":"..\\cert\\PO.crt",
                "nonce_size" : 128
        }]
}
```

Important Note: Customers that want to run the Azure Cloud HSM SDK on Windows will need to install OpenSSL on their server. We recommend installing the Chocolatey Package Manager on your Windows Server to ease with the installation of OpenSSL. Once installed you may need to close your console window and open a new console window for next steps.

Windows Installation for Chocolatey

1. Open PowerShell prompt as 'Administrator'

2. Run the following command to Set Execution Policy and Download Chocolatey:

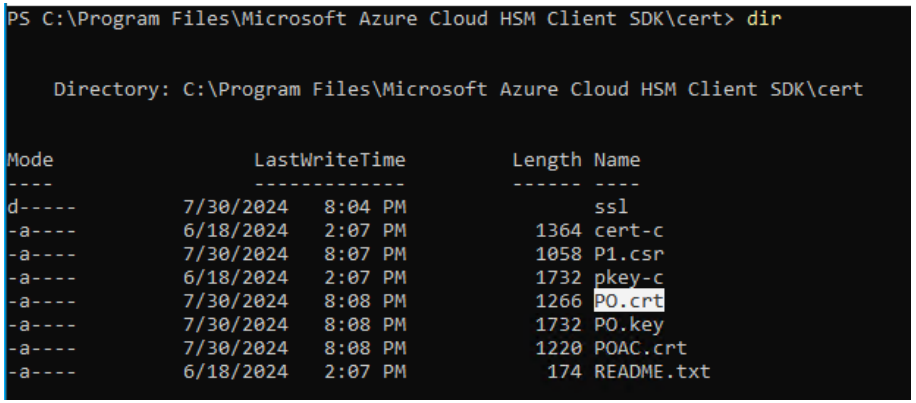
```
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol =
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

3. Run the following command for OpenSSL Installation:

```
choco install openssl --version 1.1.1.1900 -y
```

STEP 3: Validate PO.crt exists on your ADCS Server

If the PO.crt file is missing, you will need to copy the PO.crt file from your Admin VM you initialized your Azure Cloud HSM to your ADCS server under the /cert directory. If you fail to copy, or if PO.crt file is missing or incorrect when you attempt to run azcloudhsm_mgmt_util or azcloudhsm_client you will receive an error.



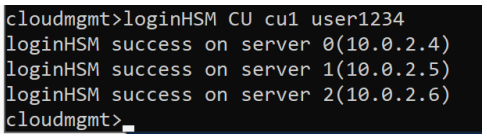
STEP 4: Validate access to your Azure Cloud HSM

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
```



Windows Server Configuration

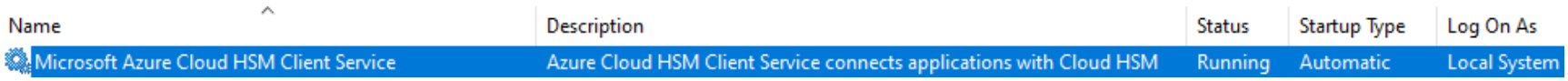
The Windows Server prerequisites are applicable and required for both option 1 and option 2 choices prescribed above.

1. Ensure you have copied over your PO.crt (partition owner cert) file to the ..\cert directory which from this example resides under C:\Program Files\Microsoft Azure Cloud HSM Client SDK\cert

2. 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."

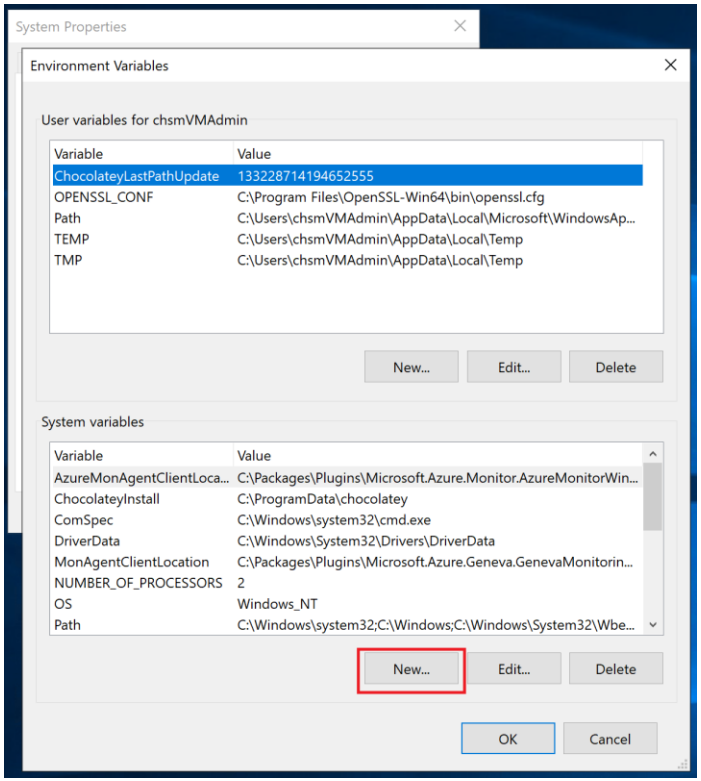
   Important Note: The Azure Cloud HSM client must always be running for ADCS and CNG/KSP providers to be operational.
   Important Note: Windows KSP tooling historically supports environment variables (e.g. CU_USER, CU_PASSWORD). ADCS runs as a Windows service, so it cannot prompt interactively. CU credentials are used once during initial KSP login and CA service startup. After login, session is cached in the client but not persisted after reboot. While ADCS uses authenticated session handles, not credentials.

   It is recommended to treat the ADCS VM as a high-trust, restricted asset and lock it down, as CU credentials supplied via environment variables are sensitive. Enforce Just-In-Time (JIT) access to minimize exposure.



3. Add the following system environment variables to your Windows Server.

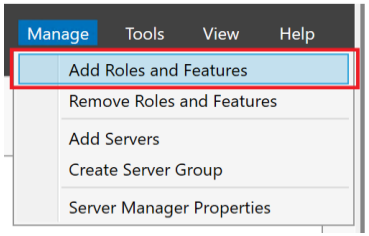
Variable Name	Variable Value	Example
azcloudhsm_username	<crypto user username>	cu1
azcloudhsm_password	<crypto user username>:<crypto user password>	cu1:user1234



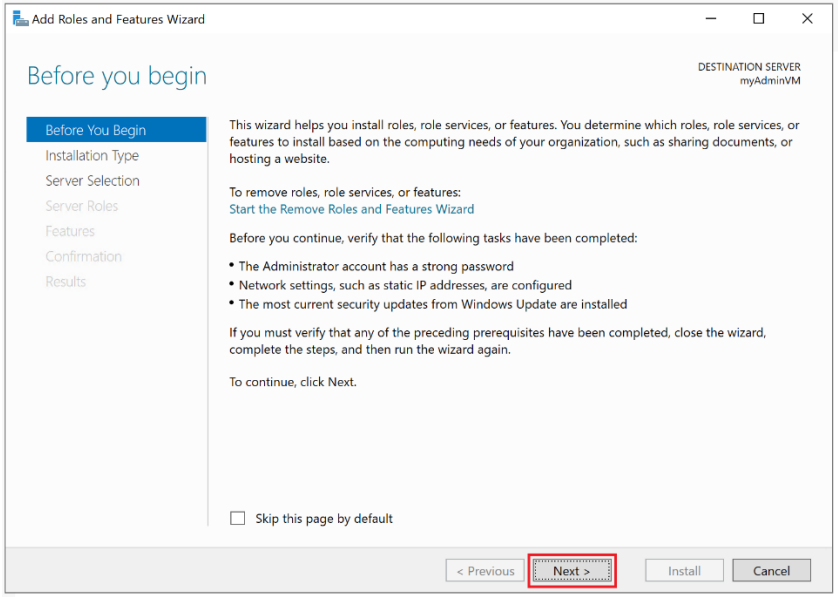
Install Active Directory Certificate Services

1. Open Server Manager
    - Click the **Start > Server Manager** to open the Server Manager Dashboard.

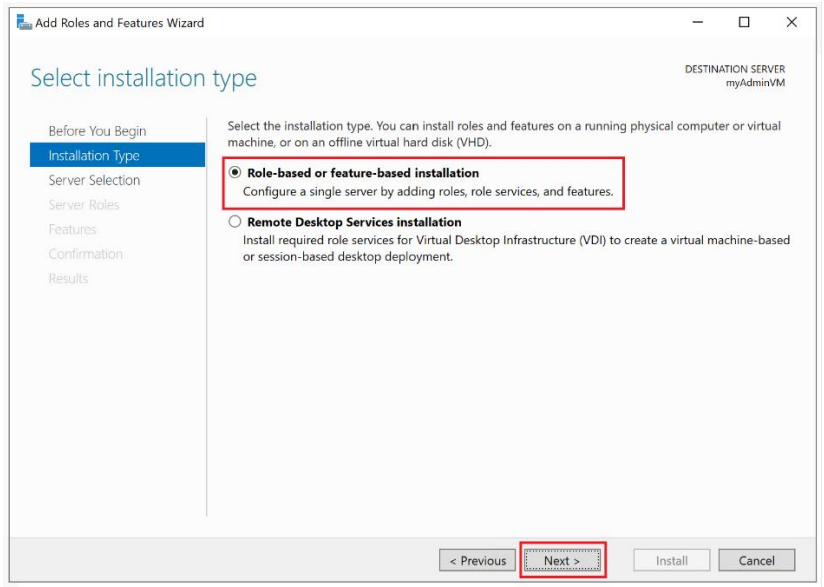
2. Click **Manage** in the upper-right corner, then select **Add Roles and Features** from the context menu.



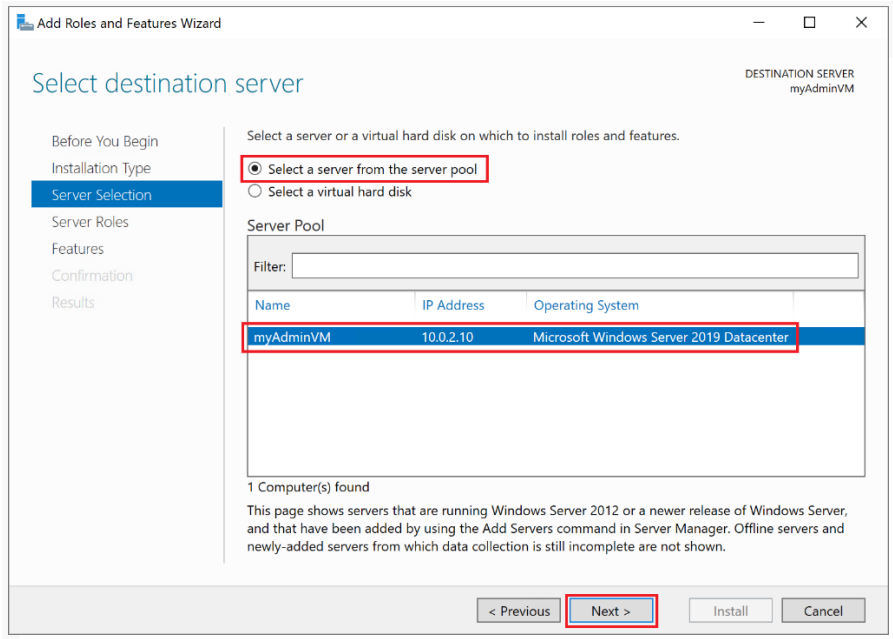
3. The *Before you begin* page displays. Verify that you have completed the tasks listed on the page and click **Next**.



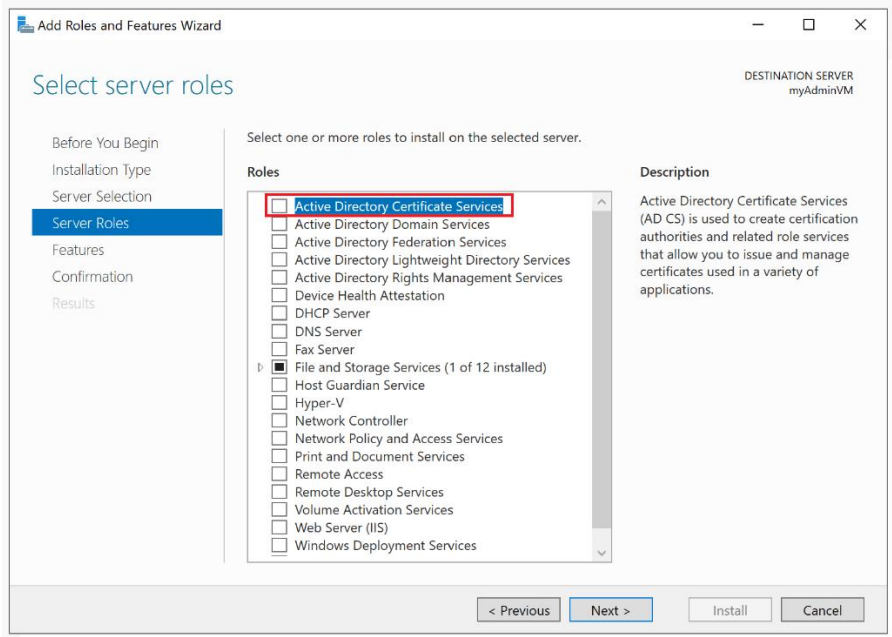
4. On the *Select installation type* page, make sure the default selection of Role Based or Feature Based Installation is selected and click **Next**.



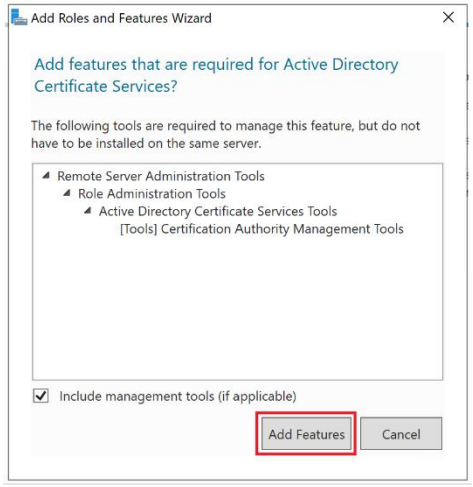
5. On the *Select destination server* page, select a server from the server pool and click **Next**.



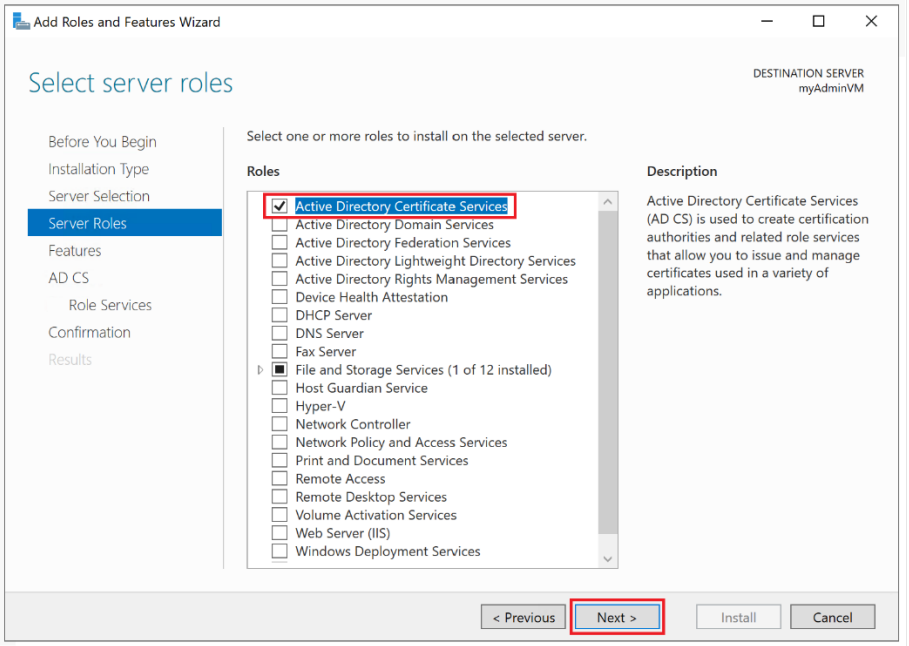
6. On the *Select server roles* page, select the **Active Directory Certificate Services** role.



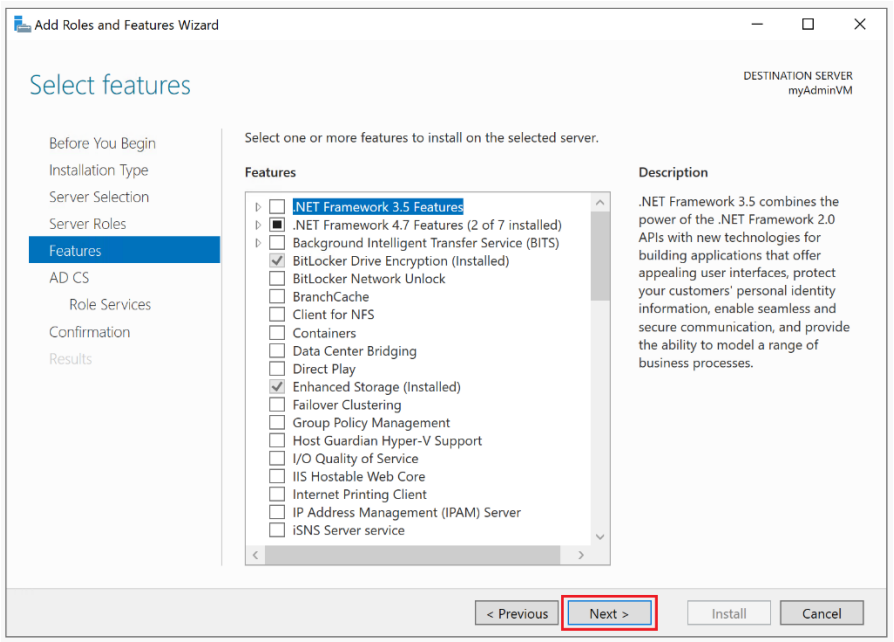
7. When prompted to install the **Remote Server Administration Tools**, click **Add Features**.



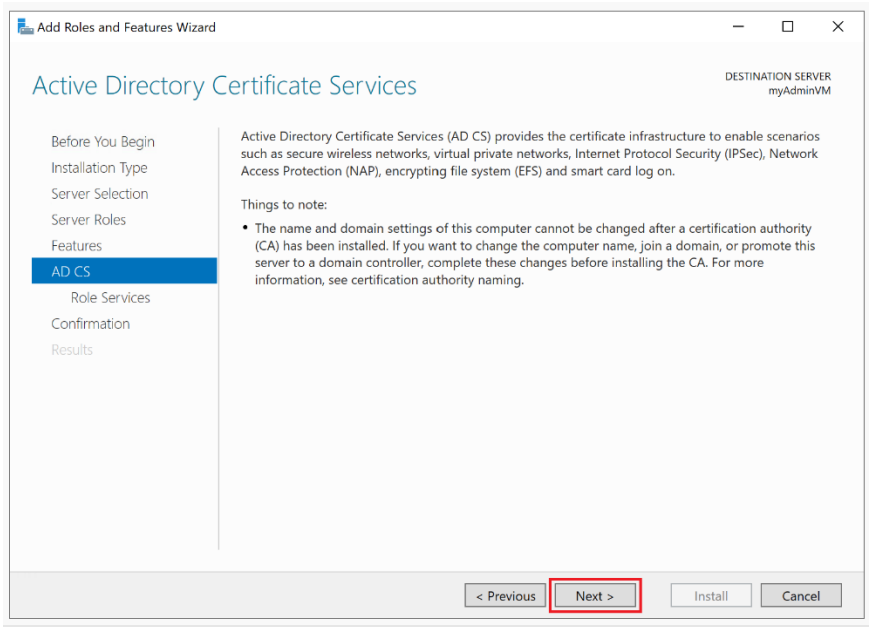
8. On the *Select server roles* page, click **Next**.



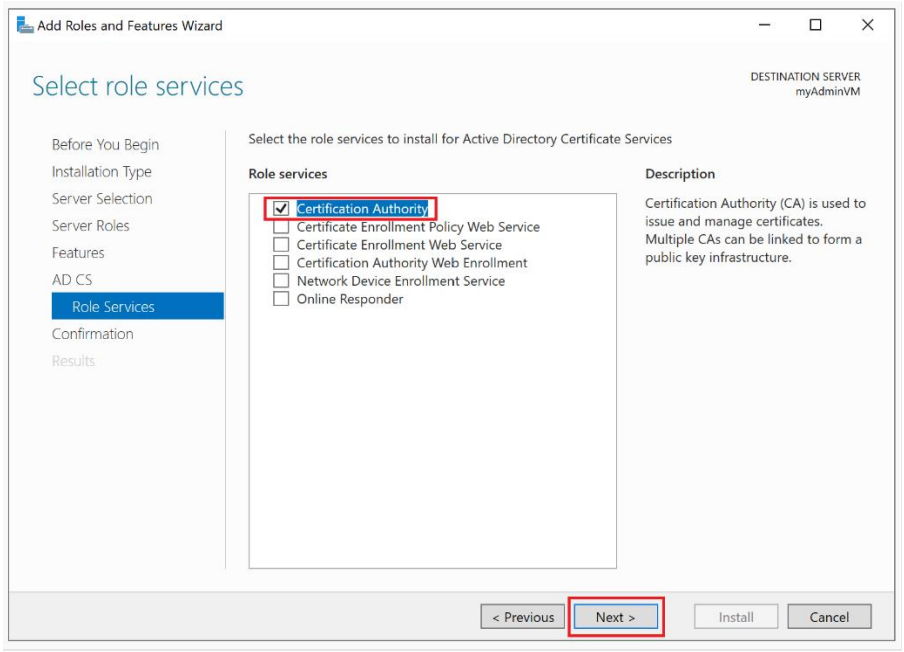
9. On the *Features* page, click **Next.**



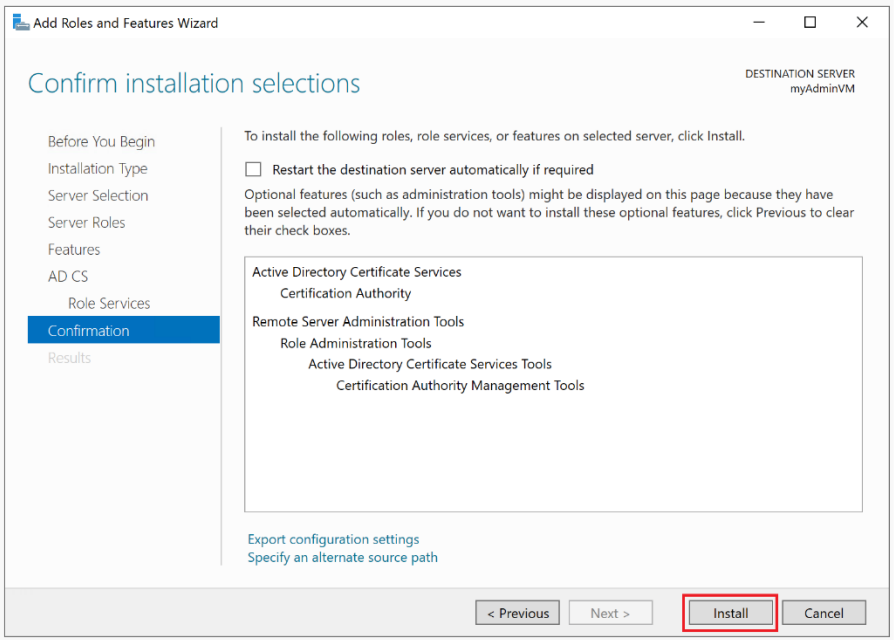
10. On the **Active Directory Certificate Services** page, click **Next.**



11. On the *Select role services* page, the Certification Authority role is selected by default. Leave the selection unchanged and click **Next**.



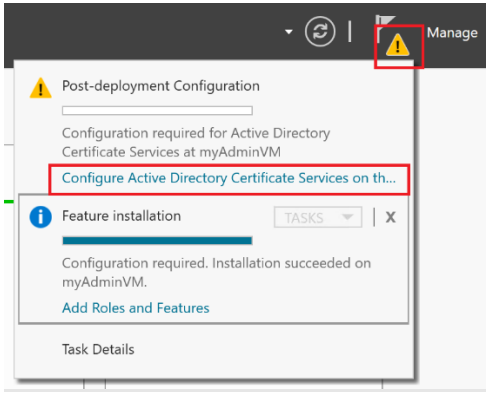
12. On the *Confirm installation selections* page, verify the information, and click **Install**.



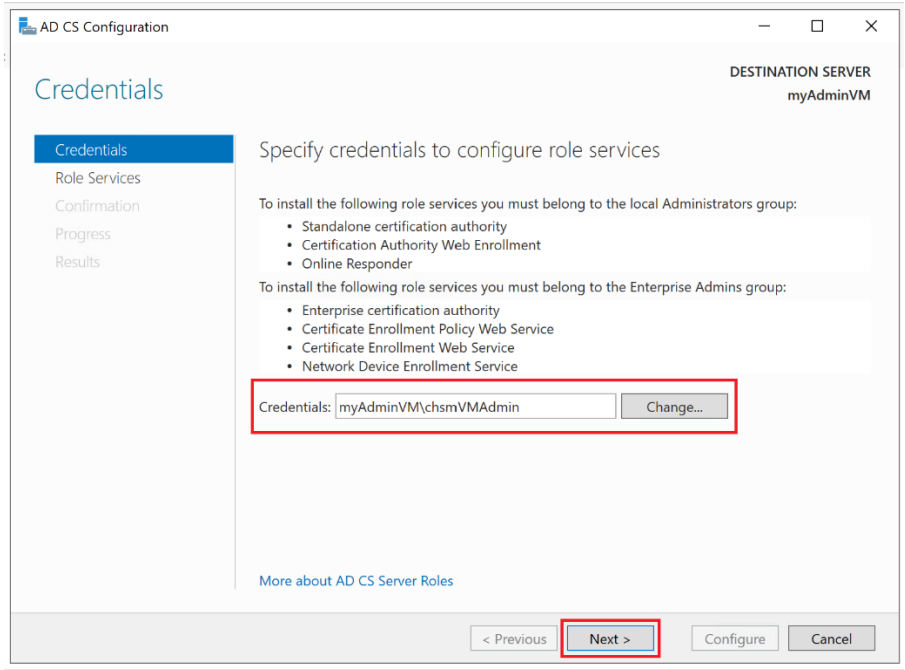
13. When installation is completed, click **Close.**

Configure Active Directory Certificate Services

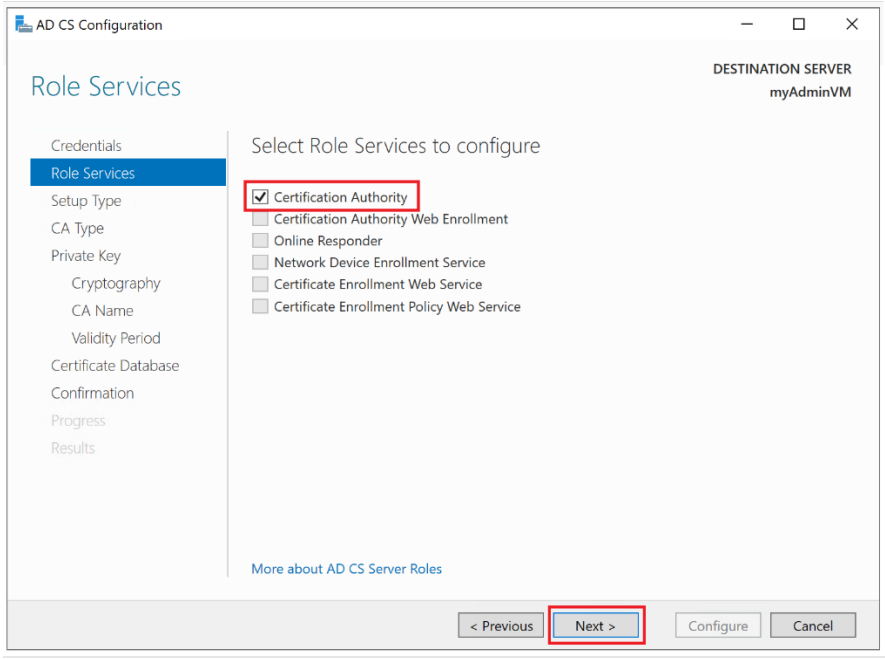
1. When the installation is completed, click the *warning symbol* in the upper-right corner and select **Configure Active Directory Certificate Services on the destination server**.



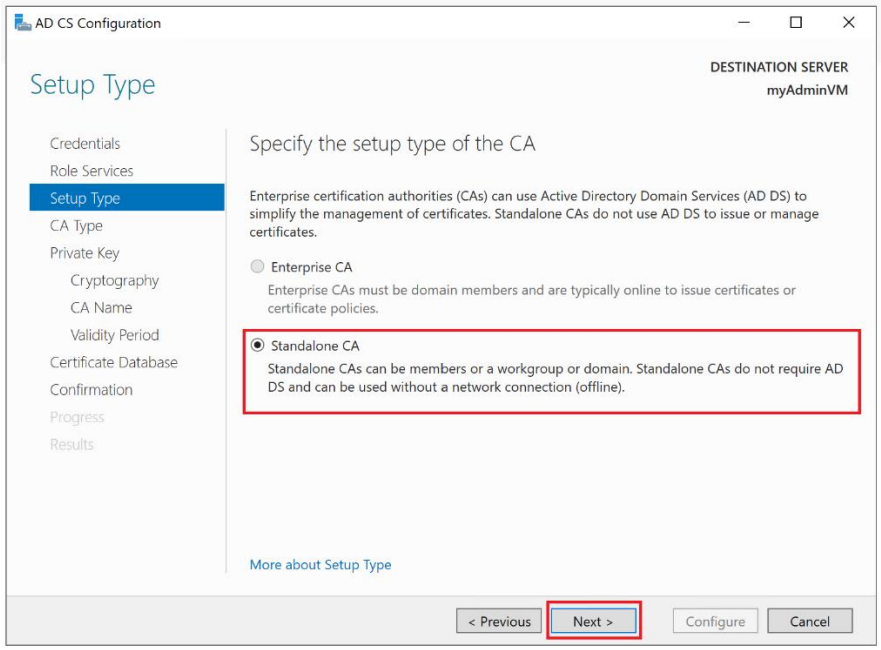
2. On the *Credentials* page, make sure that Administrator credentials are displayed in the **Credentials** box. If not, click **Change** and specify the appropriate credentials. Click **Next**.



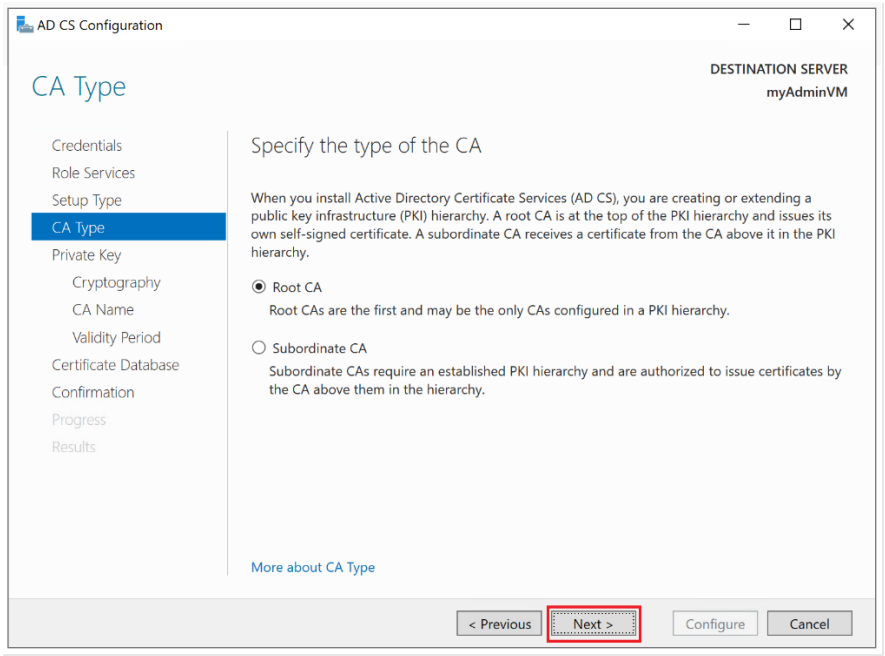
3. On the *Role Services* page, select **Certification Authority**. This is the only available selection when the certification authority role is installed on the server. Click **Next**.
    - It can take several minutes for the **Next** button to become available after selecting Certification Authority.



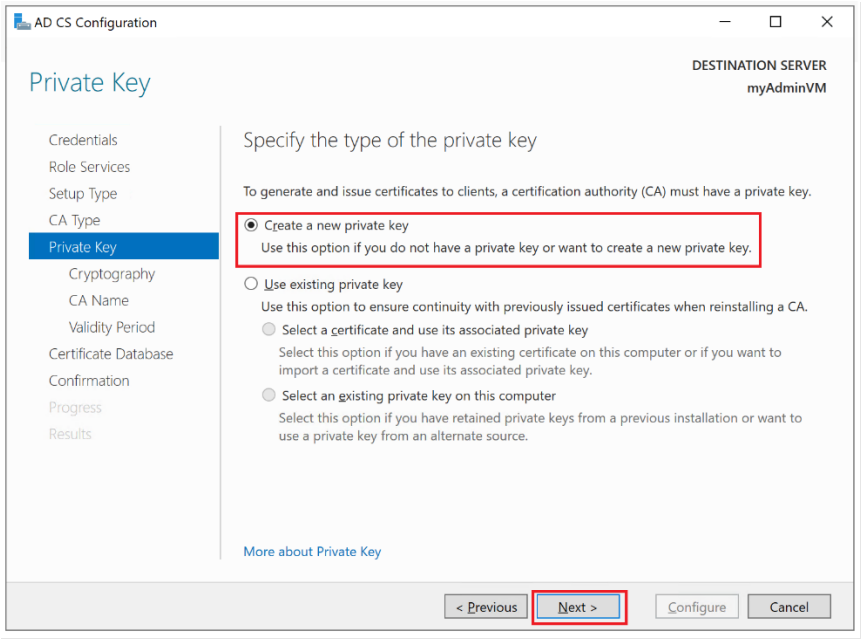
4. In the *Setup Type* page, select the appropriate **CA** setup type for your requirements. Click **Next**.
    - For this configuration example "**Standalone CA**" was selected.



5. On the *CA Type* page, **Root CA** is selected by default. Leave the selection unchanged and click **Next**.

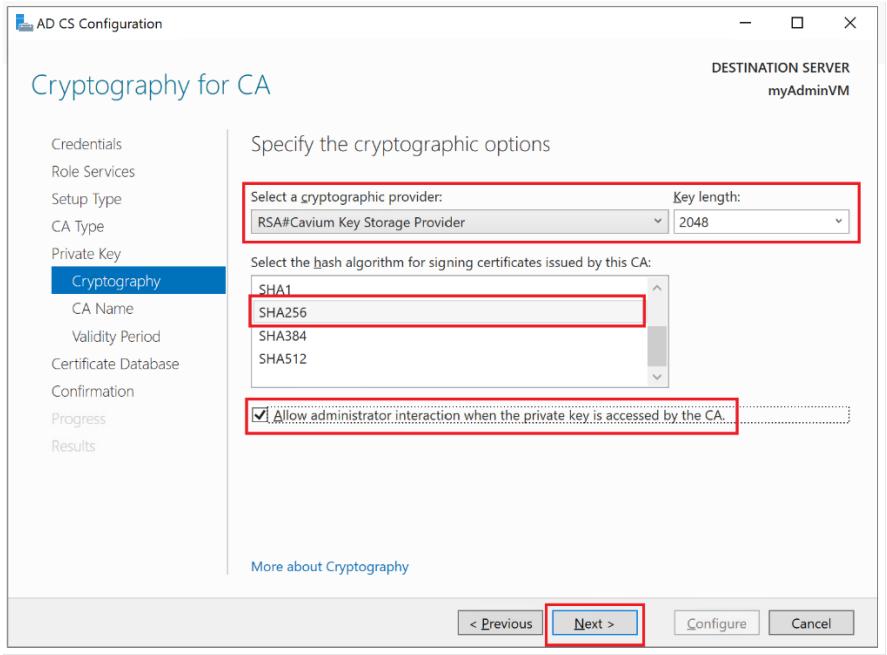


6. On the *Private Key* page, **Create a new private key** is selected by default. Leave the selection unchanged and click **Next**.

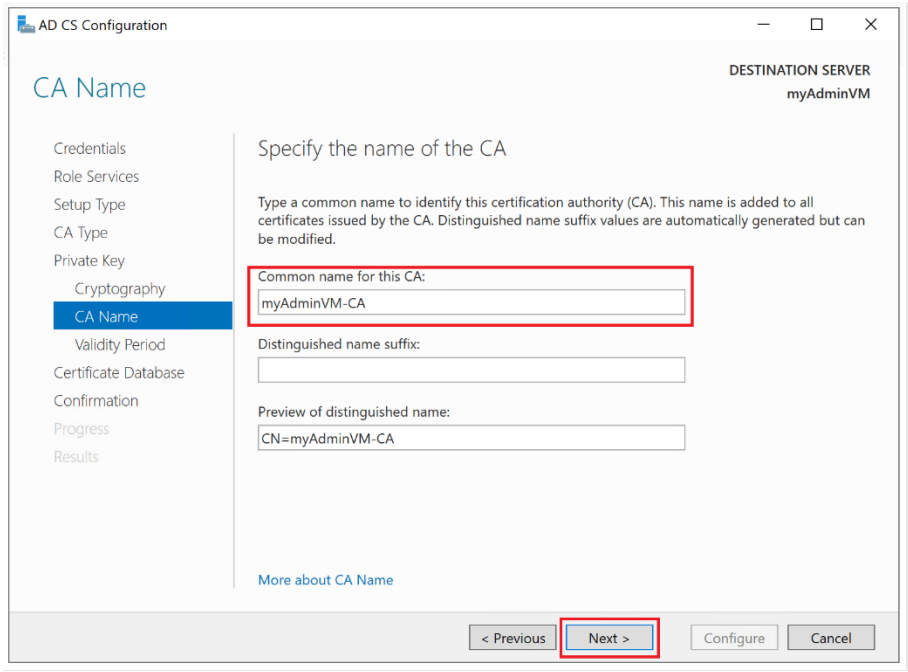


7. On the *Cryptography for CA* page, select the appropriate Azure Cloud HSM cryptographic provider along with the key type, key length, and suitable hash algorithm. Then click **Next**.

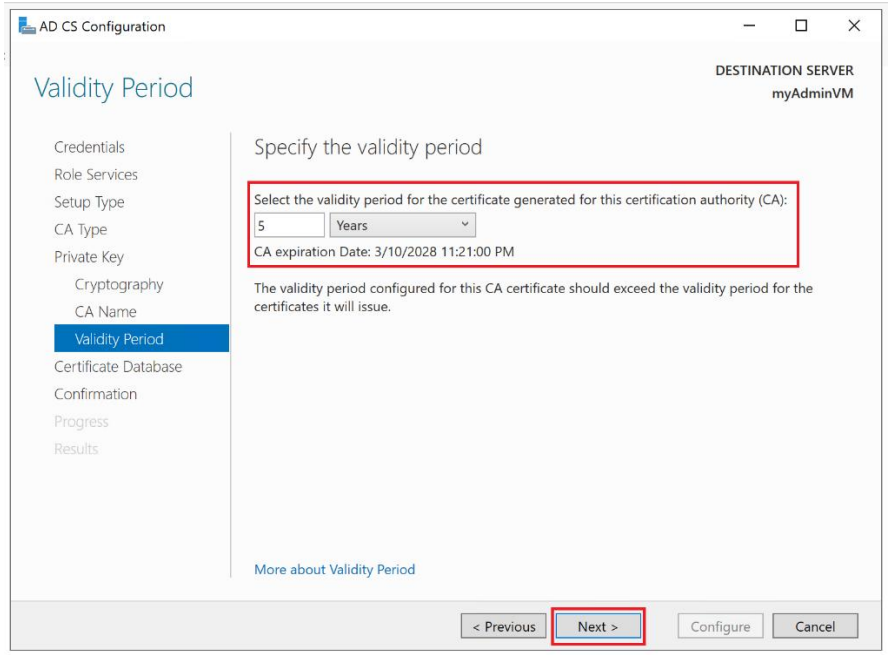
- For this example, we selected *Cryptographic Provider* as **RSA#Cavium Key Storage Provider,** *Key length* as **2048** and *hash algorithm* as **SHA256**.
- Click *check box* to enable, **Allow administrator interaction when the private key is accessed by the CA.**



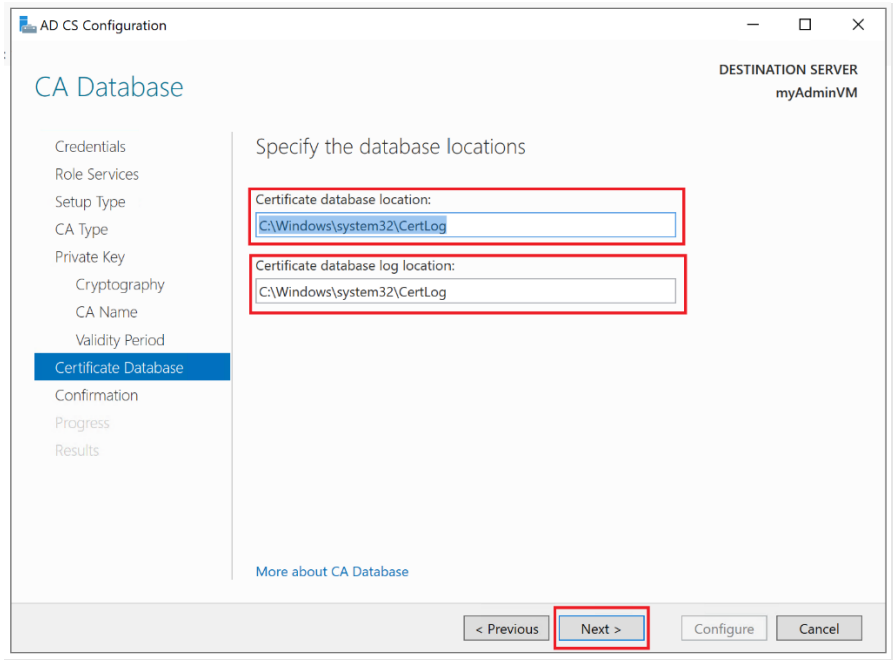
8. On the *CA Name* page, provide the appropriate CA name and click **Next**.



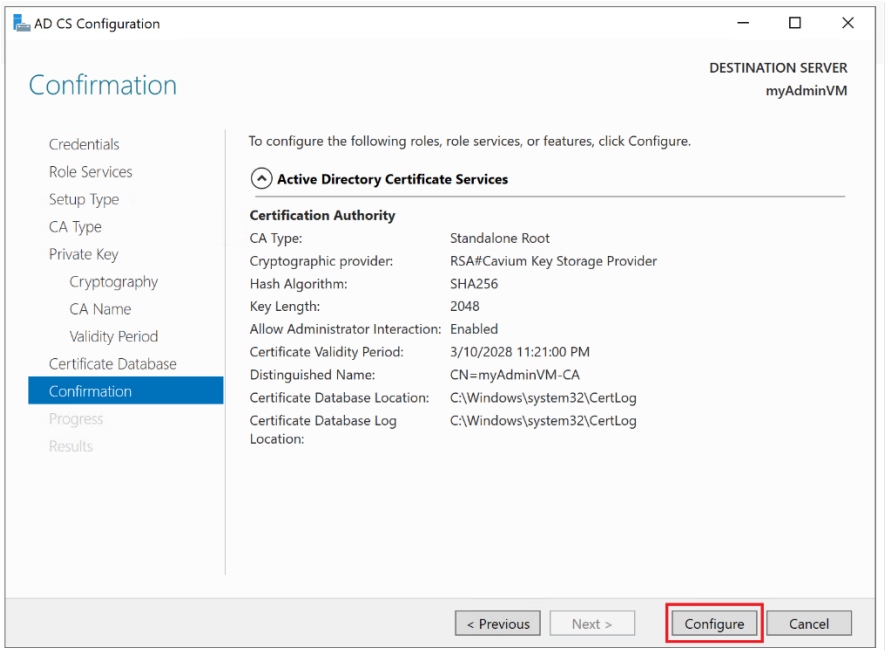
9. On the *Validity Period* page, enter the number of years for the certificate to be valid and click **Next**.



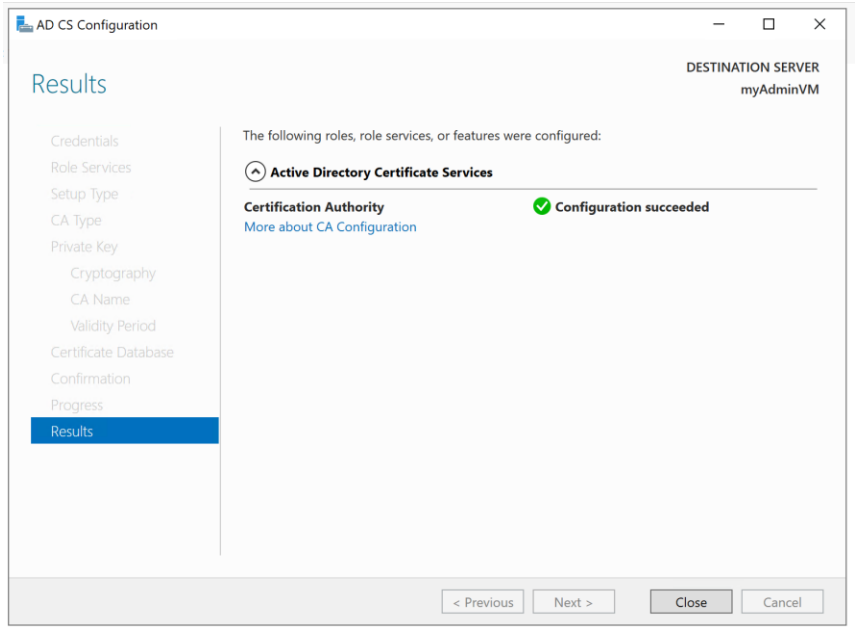
10. On the *Certificate Database* page, provide the locations for both the certificates and logs and click **Next**.



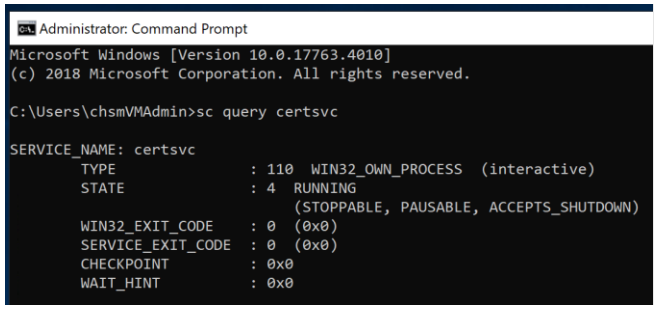
11. On the *Confirmation* page, click **Configure**.



12. When the configuration completes, a success message is displayed. Click **Close**.



*Important Note: You can also check to see whether the CA has been installed correctly by executing **sc query certsvc** from the command line:*



Configuring Certificate Services using an existing private key

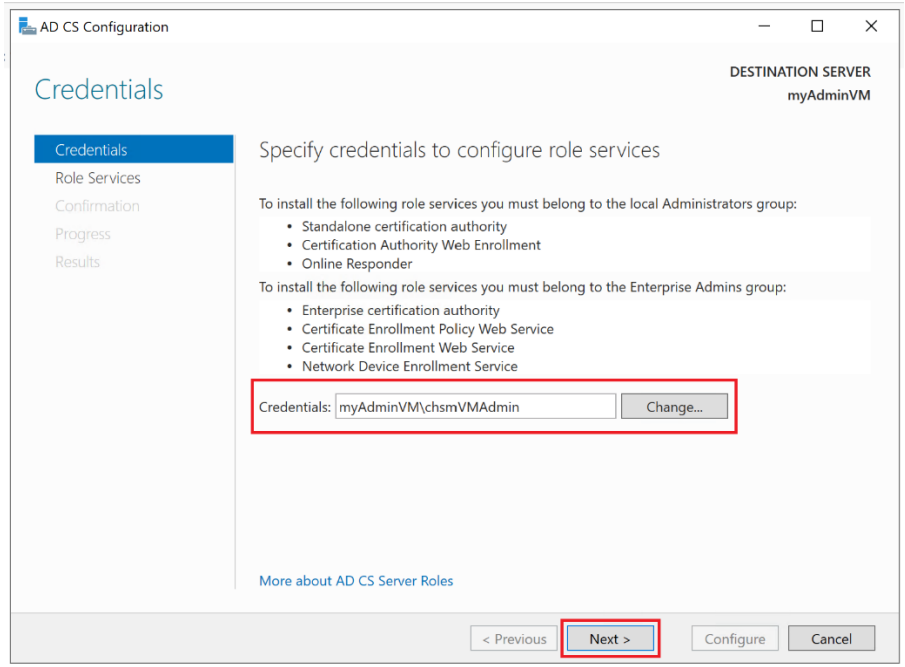
Prerequisites:

- The Install Active Directory Certificate Services steps must have been completed.

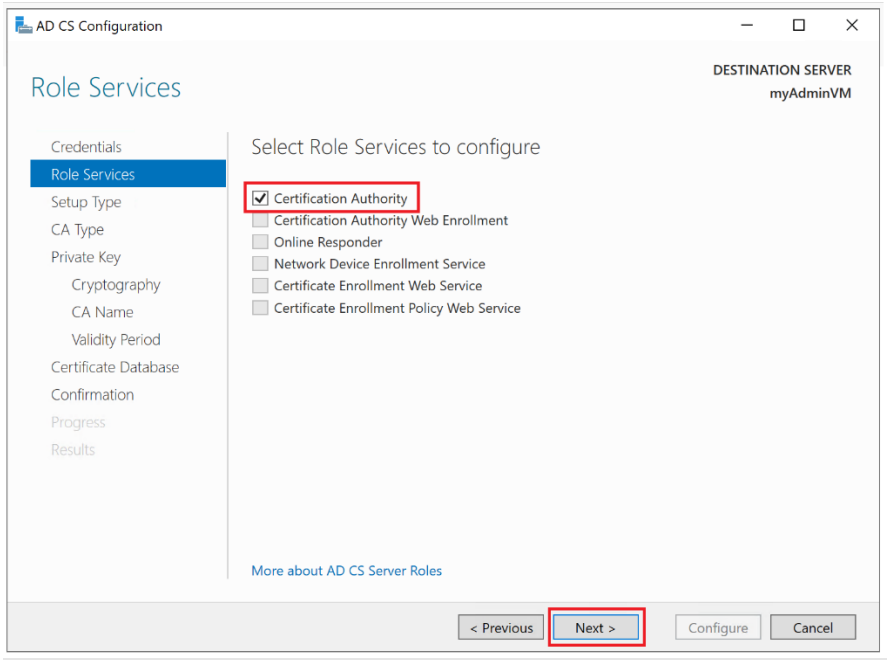
- The user has already generated a key from the KSP (the key label used in the example below is test).

To configure the certificate server using the Azure Cloud HSM KSP with an existing key, complete the following steps.

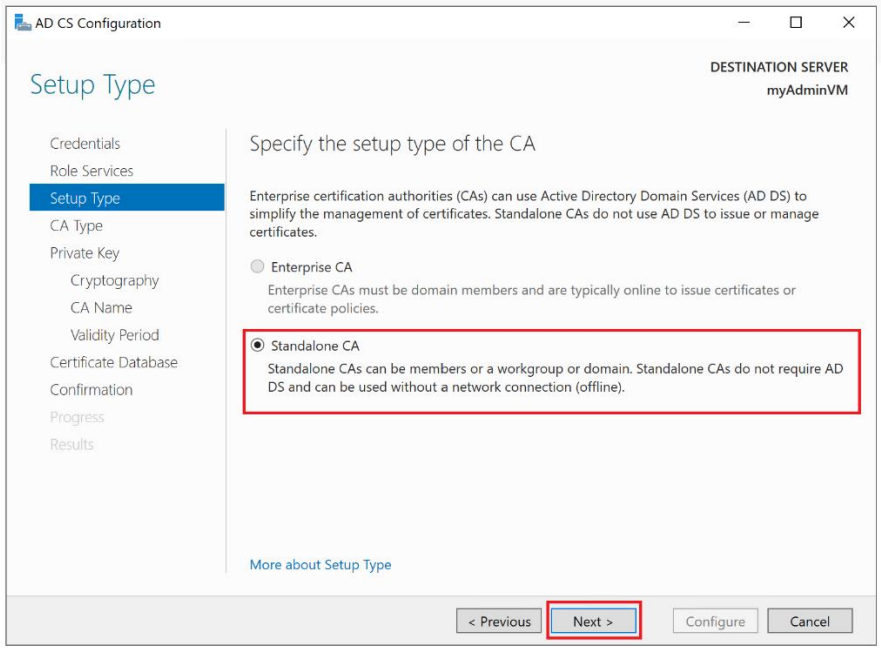
1. Click the **Configure ADCS** link from the Server Manager and then click **Next**.



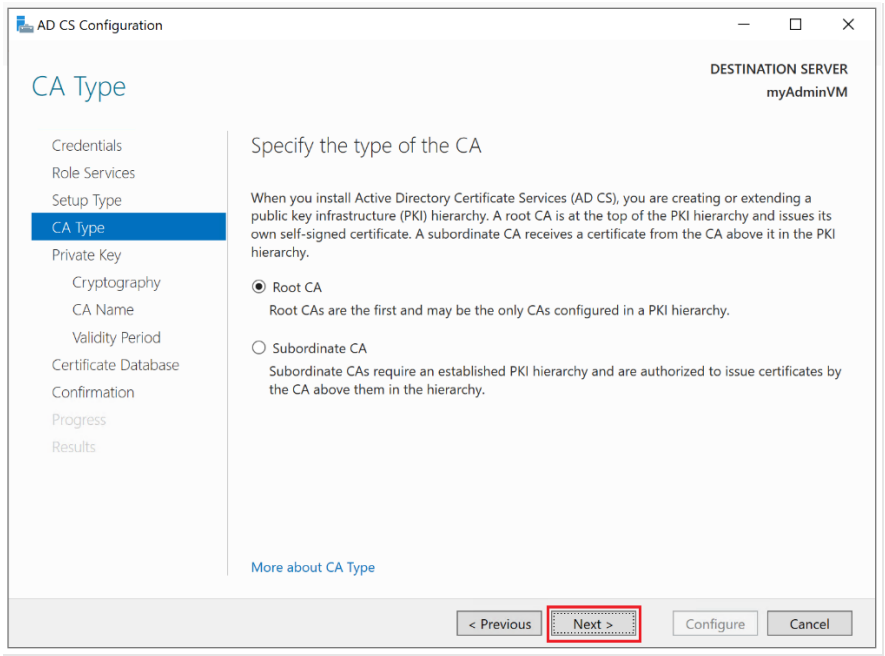
2. On the *Role Services* page, select the role services to configure and click **Next**.



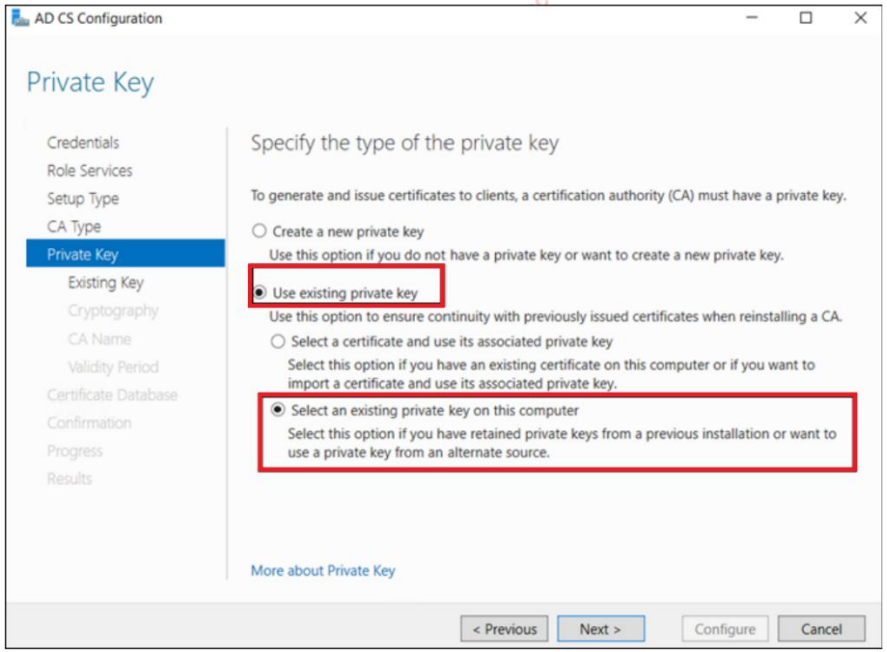
3. In the *Setup Type* page, select the appropriate **CA** setup type for your requirements. Click **Next**.
    - For this configuration example "**Standalone CA**" was selected.



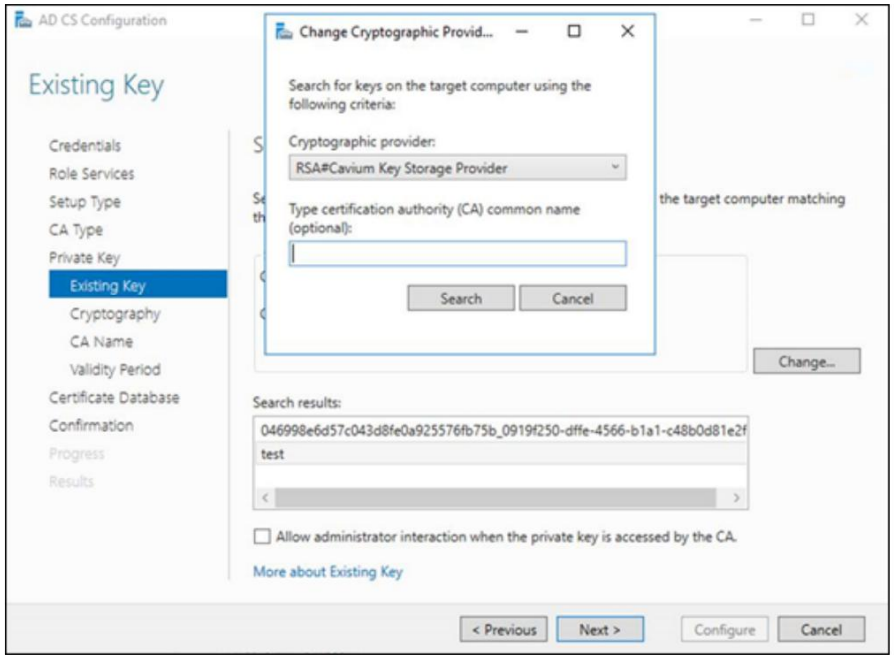
4. On the *CA Type* page, **Root CA** is selected by default. Leave the selection unchanged and click **Next**.



5. On the *Private Key* page, select **Use existing private key** and then select the option **Select the existing private key on this computer.**

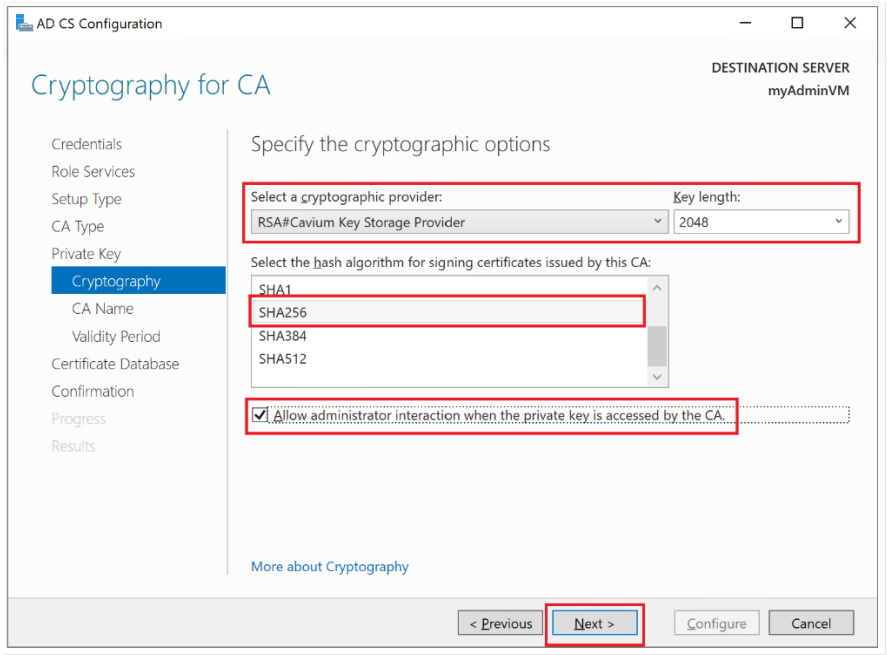


6. On the **Change Cryptographic Provider** dialog, select the CSP that contains the created key. Delete the contents of the field **CA common name** and click **Search**. The search function finds the existing private key. Select the key, then select **Allow administrator interaction when the private key is accessed by the CA**. Click **Next.**

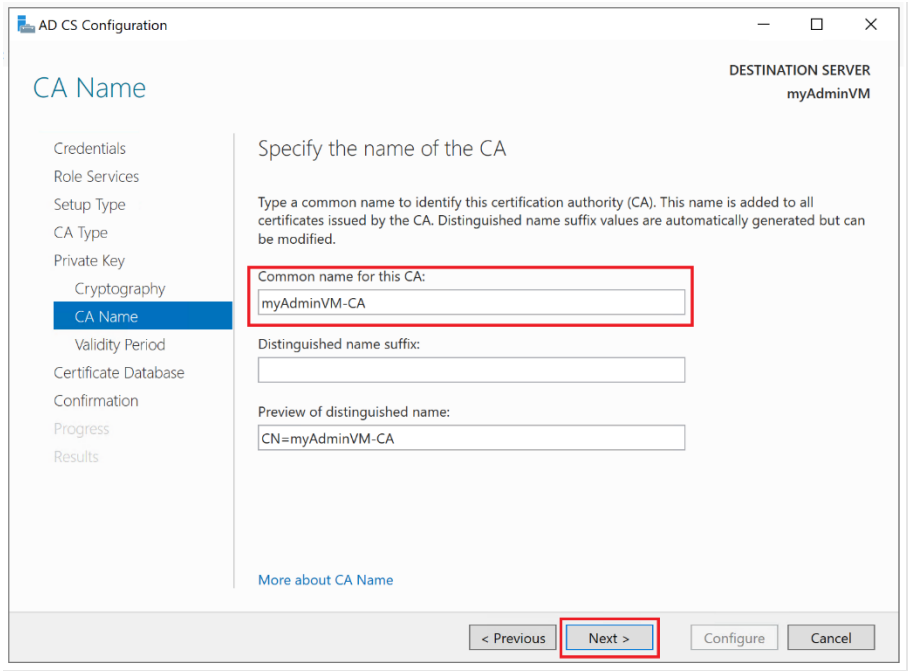


7. On the *Cryptography for CA* page, select the appropriate Azure Cloud HSM cryptographic provider along with the key type, key length, and suitable hash algorithm. Then click **Next**.

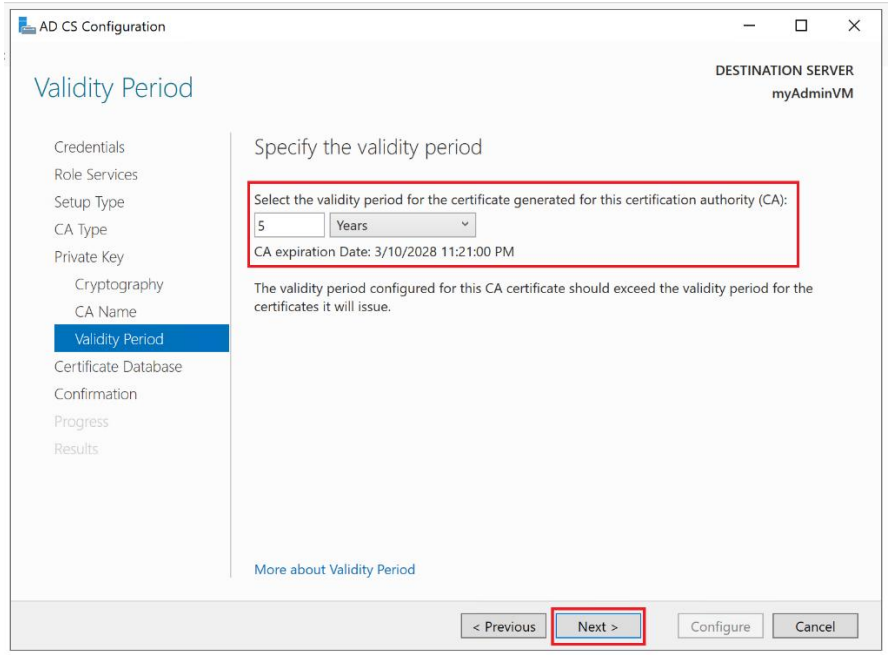
- For this example, we selected *Cryptographic Provider* as **RSA#Cavium Key Storage Provider,** *Key length* as **2048** and *hash algorithm* as **SHA256**.
- Click *check box* to enable, **Allow administrator interaction when the private key is accessed by the CA.**



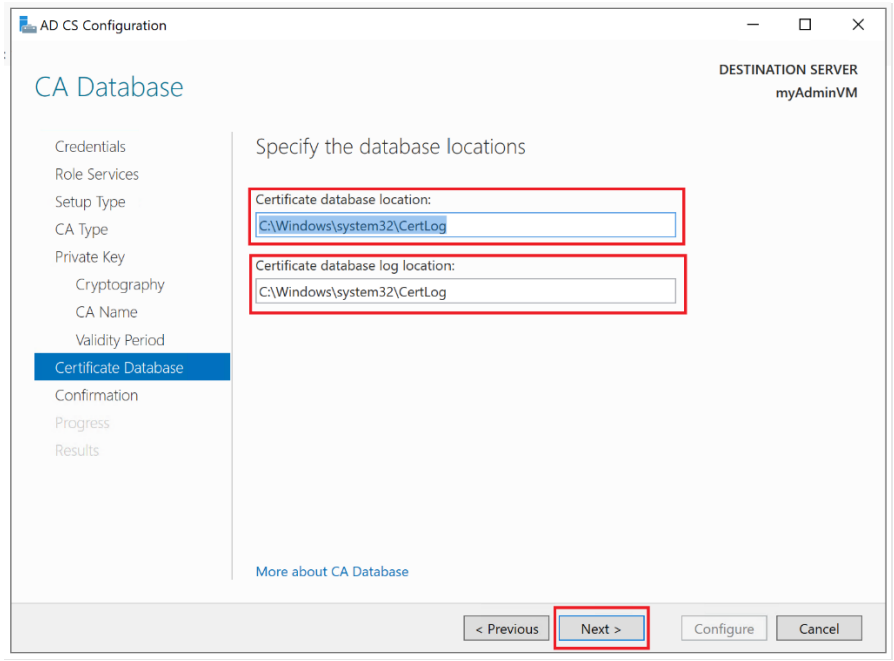
8. On the *CA Name* page, provide the appropriate CA name and click **Next**.



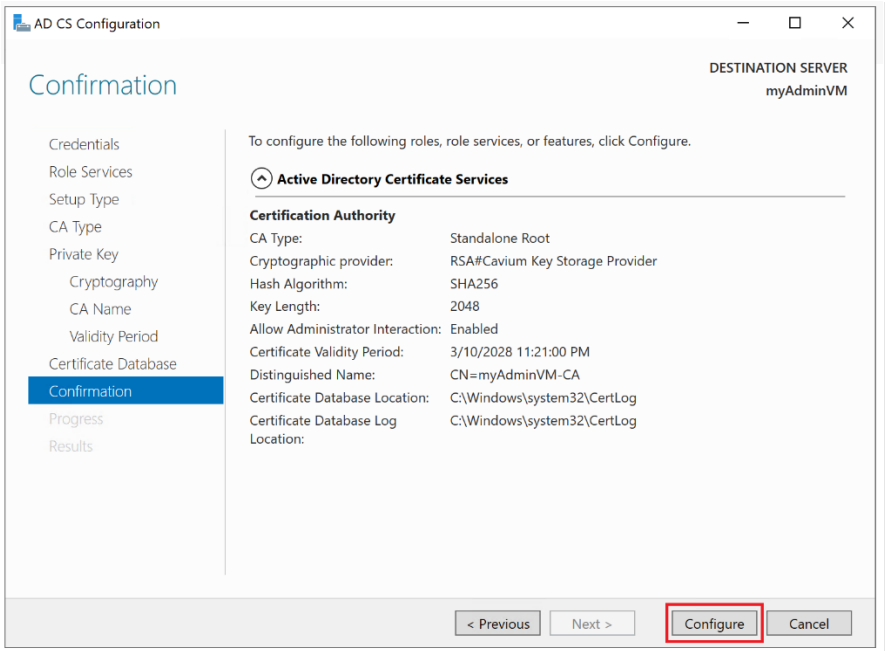
9. On the *Validity Period* page, enter the number of years for the certificate to be valid and click **Next**.



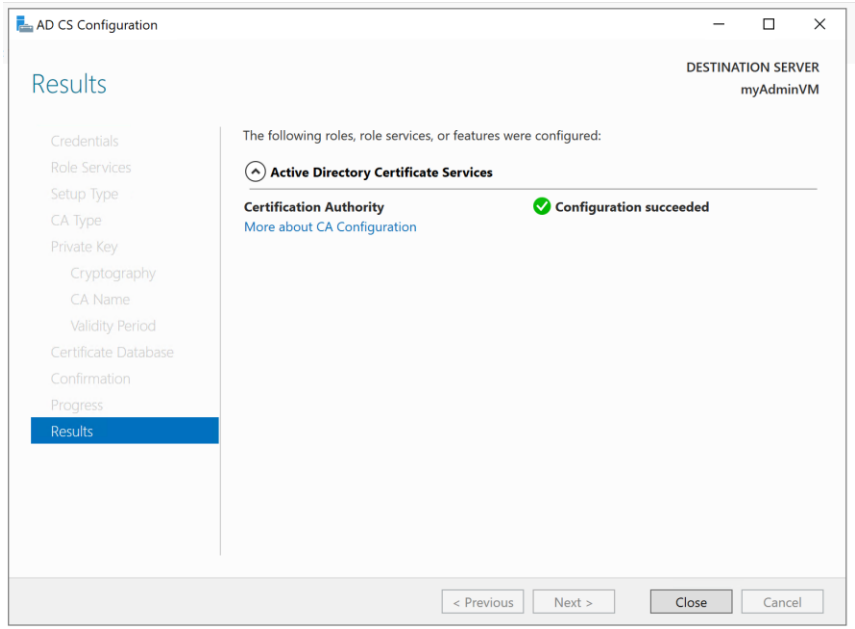
10. On the *Certificate Database* page, provide the locations for both the certificates and logs and click **Next**.



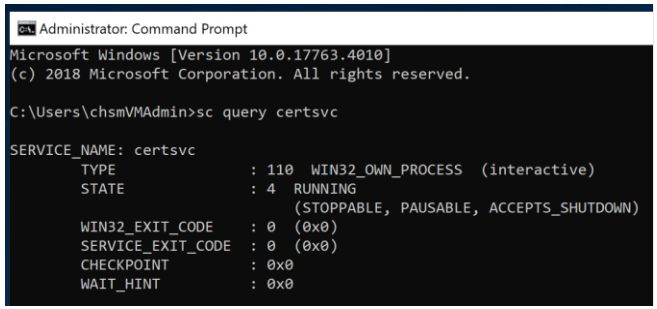
11. On the *Confirmation* page, click **Configure**.



12. When the configuration completes, a success message is displayed. Click **Close**.



*Important Note: You can also check to see whether the CA has been installed correctly by executing **sc query certsvc** from the command line:*



Alternative to CU_USER, CU_PASSWORD Environment Variables

Windows KSP tooling supports environment variables (e.g. CU_USER, CU_PASSWORD). Alternatively, credentials can be protected using DPAPI backed registry storage with deterministic startup automation. Customers may store the CU secret protected by DPAPI and inject it into the process environment at CA service startup, as the KSP reads credentials only from the process environment.

Configuration Steps:

1. Store CU credentials in the registry encrypted with DPAPI.
2. Use a startup script that runs under the CA service account.
3. Set process-level environment variables for the CA service.
4. Start the Certificate Services (CertSvc).
5. The Marvell (Cavium) KSP authenticates to Azure Cloud HSM.

Step 1: Decide the security context (important)

DPAPI encryption is identity scoped.

- If CA runs as **LocalSystem** → encrypt using LocalMachine scope.
- If CA runs as **domain or local user** → encrypt as *that user*

Check which account your CA service is running under.

```
sc qc CertSvc
```

Step 2: Encrypt the CU password with DPAPI

Run PowerShell as Administrator (and as the CA service account if it is not running as LocalSystem). This command outputs a DPAPI encrypted value. Copy the resulting blob.

```
$PlainText = Read-Host "Enter CU_PASSWORD" -AsSecureString
$Encrypted = ConvertFrom-SecureString -SecureString $PlainText -Scope LocalMachine
```

Step 3: Store encrypted secret in the registry

Create a dedicated registry key for storing the credentials:

```
New-Item -Path "HKLM:\SOFTWARE\AzureCloudHSM" -Force | Out-Null
```

Store the username and DPAPI-encrypted password. At rest, the password is now DPAPI protected and securely stored.

```
Set-ItemProperty -Path "HKLM:\SOFTWARE\AzureCloudHSM" -Name "CU_USER" ` -Value "cu1"
Set-ItemProperty -Path "HKLM:\SOFTWARE\AzureCloudHSM" -Name "CU_PASSWORD" ` -Value "<PASTE_ENCRYPTED_BLOB>"
```

Step 4: Create the startup script

This script sets process level environment variables for the KSP login. These variables exist only for the script's process and are not written to disk or persisted across reboots. Use this method if you are not using system environment variables for KSP authentication.

Important Note: *Any restart of CertSvc requires the startup script to be rerun, as KSP sessions do not persist across reboots or service restarts.*

1. Open Notepad and save the script as Start-CertSvc.ps1 (e.g., C:\AzureCloudHSM\Start-CertSvc.ps1).
2. Copy the following contents into the script:

Makefile

```
.\Start-CertSvc.ps1
```

Script Contents

```
$RegPath = "HKLM:\SOFTWARE\AzureCloudHSM"
$CUUser = (Get-ItemProperty $RegPath).CU_USER $EncryptedPwd = (Get-ItemProperty $RegPath).CU_PASSWORD

#Decrypt using DPAPI
$SecurePwd = ConvertTo-SecureString $EncryptedPwd $PlainPwd = [System.Net.NetworkCredential]::new("", $SecurePwd).Password

#Set PROCESS-LEVEL environment variables
[System.Environment]::SetEnvironmentVariable("CU_USER", $CUUser, "Process")
[System.Environment]::SetEnvironmentVariable("CU_PASSWORD", $PlainPwd, "Process")

#Start ADCS
Start-Service CertSvc
```

Optional: Restrict Registry Access (ACLs)

Limit access to your DPAPI-encrypted credentials so only the SYSTEM account and the CA service account can read them.

Optional: Obfuscate Registry Path and Value Names

Avoid obvious names like CU_PASSWORD or AzureCloudHSM and use a generic key path (e.g. HKLM:\SOFTWARE\Microsoft\PolicyCache)

Step 5: Stop automatic CA startup (critical)

To ensure the startup script controls the CA service, set CertSvc to start manually. This prevents the service from starting automatically before the script sets the required process level environment variables.

```
Set-Service CertSvc -StartupType Manual
```

Step 6: Run the script manually (test)

Execute the startup script to launch Certificate Services using the DPAPI protected CU password. This will set the process level environment variables and start CertSvc for testing purposes.

```
powershell.exe -ExecutionPolicy Bypass -File Start-CertSvc.ps1
```

Check that the Certificate Services are running.

```
Get-Service CertSvc
```

Step 7: Validate KSP is working

From an elevated PowerShell or Command Prompt, verify that the KSP is working. If configured correctly the KSP authenticates using the DPAPI protected CU credentials. No CU login prompt appears. No authentication errors are logged in the Application Event Log. You can also perform a test signing operation to further confirm functionality.

```
certutil -csplist
```

Step 8: Automate at boot. (Task Scheduler - Recommended)

Create a startup scheduled task to run the startup script automatically. This ensures environment variables are set only at CA service startup. Credentials are never persisted in plaintext. Reboot and service restart behavior is predictable and automatic.

This setup ensures secure, automated, and reliable startup for the CA service with KSP authentication.

Trigger: At startup
Run as: SYSTEM (or CA service account)
Run whether user is logged on or not

Optional: Add triggers for “On service failure” or “On unexpected stop” to maintain KSP and ADCS availability

Action:

```
powershell.exe -ExecutionPolicy Bypass -File .\Start-CertSvc.ps1
```

Validate Azure Cloud HSM KSP with ADCS

Customers typically use Key Storage Providers (KSP) rather than Cryptographic Next Generation (CNG), providers with Active Directory Certificate Services (ADCS). This preference is primarily due to the enhanced security and regulatory compliance requirements often associated

with ADCS deployments. KSP, especially those implemented with hardware security modules (Azure Cloud HSM), offer several advantages for ADCS environments. The Azure Cloud HSM KSP implementation bestows users with the necessary privileges to establish communication with their Azure Cloud HSM.

1. Generate a CSR File
   You will need to ensure the azcloudhsm_application.cfg is in the directory you are running certreg from, else you will receive a azcloudhsm_application.cfg is not present exception.
   certreq -new request.inf request.req

   - *Sample request.inf file.*
     - *KeySpec 1 indicates AT_KEYEXCHANGE, 2 indicates AT_SIGNATURE.*
     - *ProviderName value "Cavium Key Storage Provider" for KSP,*
     - *ProviderType value 32 for KSP (Cavium Key Storage Provider).*

```
[Version]
Signature="$Windows NT$"

[NewRequest]
Subject = "CN=Test, OU=Security, O=Microsoft, L=Redmond, S=WA, C=US"
KeySpec = 2
KeyLength = 2048
ProviderName = "Cavium Key Storage Provider"
ProviderType = 32
RequestType = PKCS10
HashAlgorithm = SHA256
```

2. Verify CSR file generated by Azure Cloud HSM using “Cavium Key Storage Provider”.
   certutil -dump .\request.req

   Output:

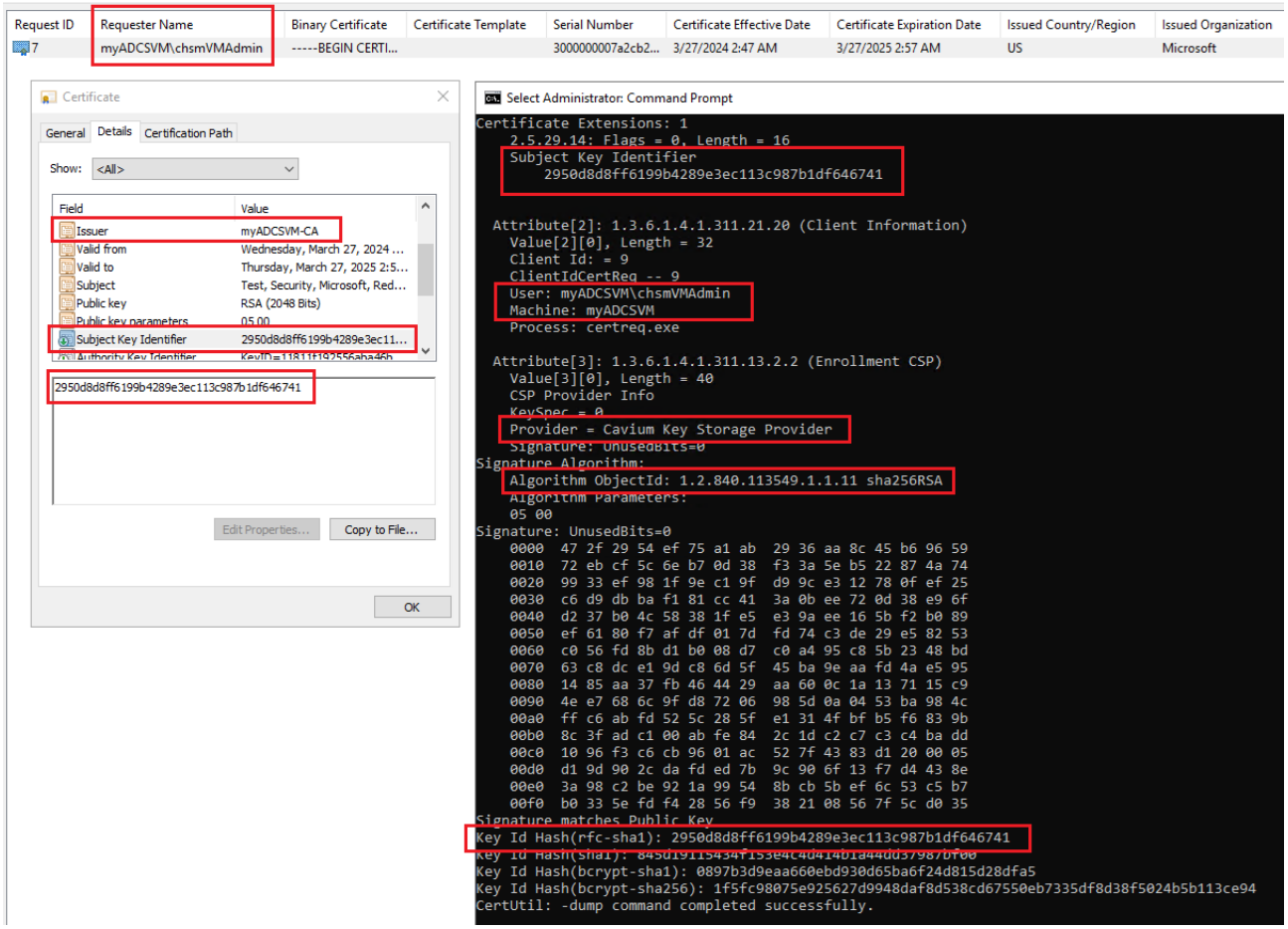
```
PKCS10 Certificate Request:
Version: 1
Subject:
    CN=Test
```

```
    OU=Security
    O=Microsoft
    L=Redmond
    S=WA
    C=US
…
Provider = Cavium Key Storage Provider
…
```

3. Request for a New Certificate.
    a. In the Certification Authority MMC snap-in, expand the Certification Authority node to reveal your CA server.
    b. Right-click on the "CA", select "All Tasks" > "Submit New Request."

4. Approve the Certificate Request
    - In the Certification Authority MMC snap-in, navigate to the "Pending Requests" folder.
    - Right-click on the pending certificate request corresponding to the one you just submitted.
    - Select "All Tasks" > "Issue."

5. Retrieved the Issued Certificate.
    - Once the request is approved and the certificate is issued, it will move from the "Pending Requests" folder to the "Issued Certificates" folder.
    - You can now retrieve the issued certificate by locating it in the "Issued Certificates" folder. Double-click on it to view its details and compare it against the certutil -dump you performed earlier they should match.



Representing Cloud HSM Keys in the KSP

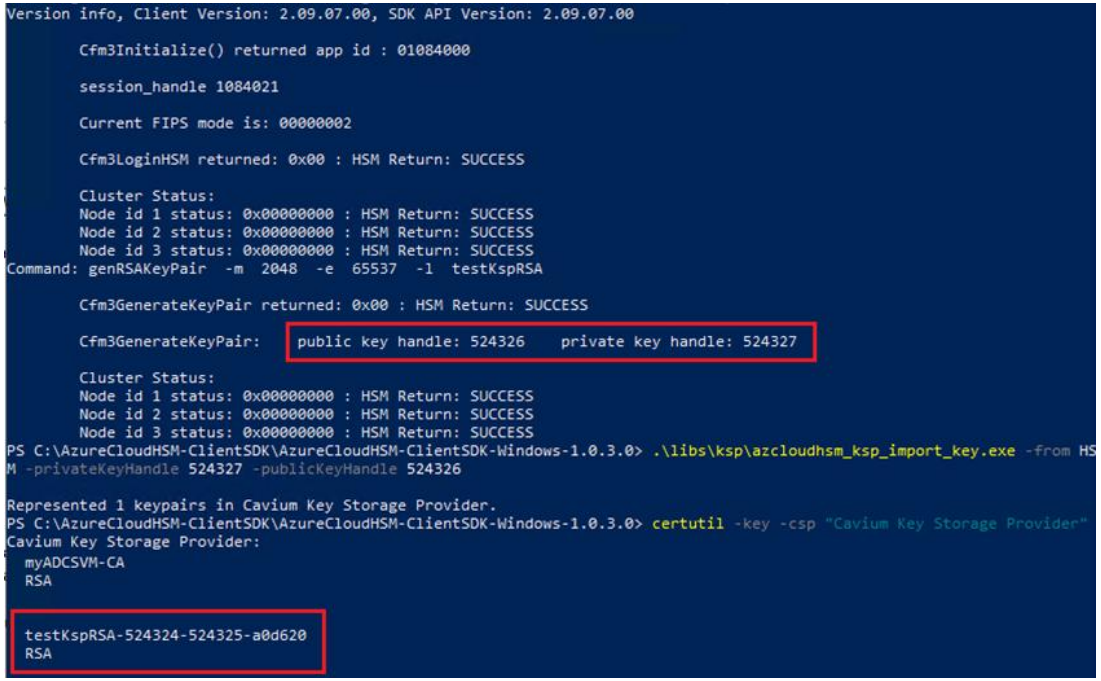
This section shows the process of representing keys generated within Azure Cloud HSM and putting them into the KSP using the Cloud HSM SDK tool azcloudhsm_ksp_import_key.exe.

Creating a single key pair in the HSM and representing It in the KSP.

1. Generate an RSA key pair. Make notes of the private and public key handles. Thes are used in the next steps.
   azcloudhsm_util.exe singlecmd loginHSM -u CU -s cu1 -p user1234 genRSAKeyPair -m 2048 -e 65537 -l testKspRSA

2. Import keys from your Cloud HSM into the Cloud HSM Key Storage Provider (KSP).
   azcloudhsm_ksp_import_key.exe -from HSM -privateKeyHandle {privateKeyHandle} -publicKeyHandle {publicKeyHandle}

3. List keys stored in the Azure Cloud HSM Key Storage Provider (KSP)
   certutil -key -csp "Cavium Key Storage Provider"

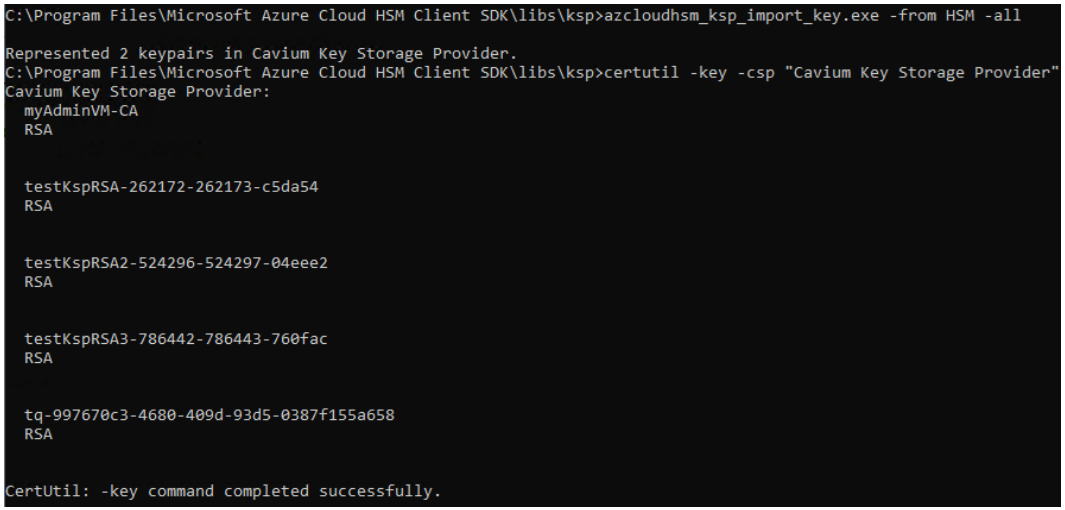


Representing All Cloud HSM Keys in the KSP

1. Generate multiple RSA key pairs within the Azure Cloud HSM.
   azcloudhsm_util.exe singlecmd loginHSM -u CU -s cu1 -p user1234 genRSAKeyPair -m 2048 -e 65537 -l testKspRSA2
   azcloudhsm_util.exe singlecmd loginHSM -u CU -s cu1 -p user1234 genRSAKeyPair -m 2048 -e 65537 -l testKspRSA3

2. Import all keys from your Cloud HSM into the Cloud HSM Key Storage Provider (KSP).
   azcloudhsm_ksp_import_key.exe -from HSM -all

3. List keys stored in the Azure Cloud HSM Key Storage Provider (KSP)
   certutil -key -csp "Cavium Key Storage Provider"



Migrating a Certificate from Microsoft KSP to Cloud HSM KSP

The following sections explain how to migrate certificate and keys created with Microsoft Software Key Storage Provider to the Azure Cloud HSM Key Storage Provider (KSP). For this example, we will be following Option 2.

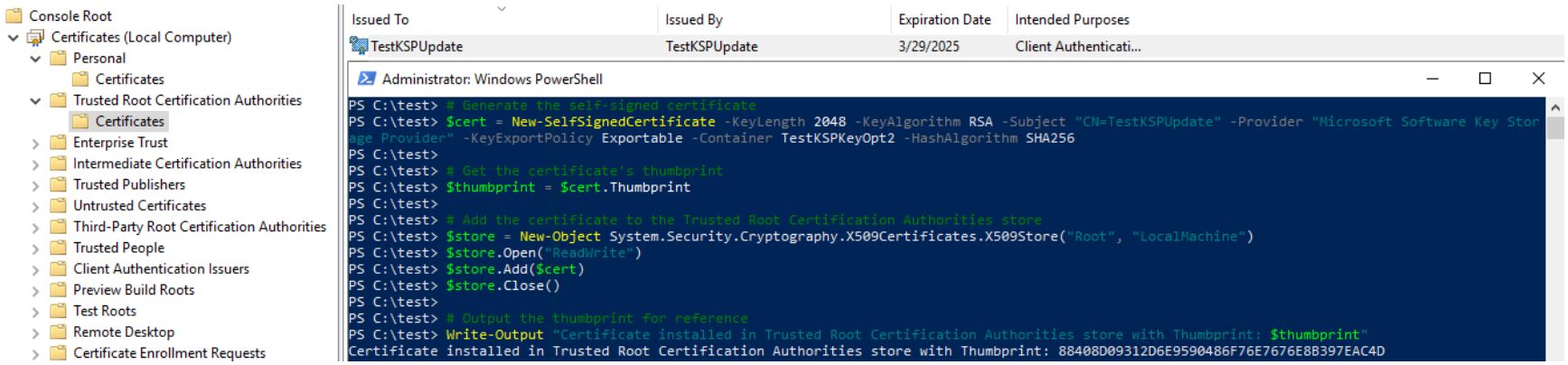
Important Note: To migrate a certificate from Microsoft KSP or another 3rd Party KSP to Azure Cloud HSM KSP, you must first create a User-Generated KEK. This KEK is crucial for the migration process. When transitioning to Cloud HSM KSP, migration involves an import operation, which establishes private and public key handles in Cloud HSM linked to the migrated certificate. If you haven't yet created a User-Generated KEK, please refer to the provided instructions for creating a User-Generated KEK before proceeding with the migration.

1. **Generate a self-signed certificate.**
   Make note of the thumbprint for your certificate. It will be used in the next steps.
   a. **Option 1: Generate self-signed certificate (Personal)**

```
# Generate the self-signed certificate
New-SelfSignedCertificate -KeyLength 2048 -KeyAlgorithm RSA -Subject "CN=TestKSPUpdate" -
Provider "Microsoft Software Key Storage Provider" -KeyExportPolicy Exportable -Container
TestKSPKeyOpt1 -CertStoreLocation Cert:\LocalMachine -HashAlgorithm SHA256
```

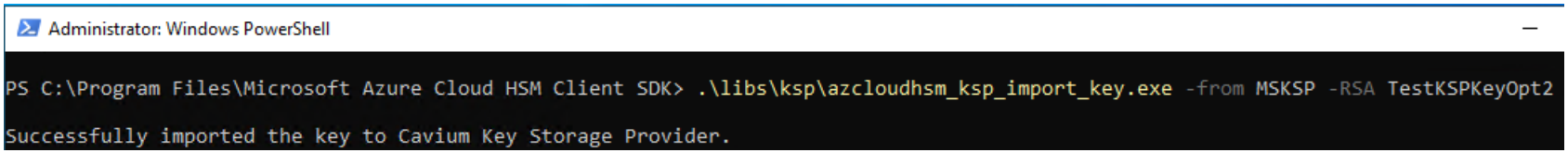
a. **Option 2: Generate self-signed RootCA certificate (Trusted Root Certification Authorities)**

```
# Generate the self-signed certificate
$cert = New-SelfSignedCertificate -KeyLength 2048 -KeyAlgorithm RSA -Subject "CN=TestKSPUpdate"
-Provider "Microsoft Software Key Storage Provider" -KeyExportPolicy Exportable -Container
TestKSPKeyOpt2 -HashAlgorithm SHA256

# Get the certificate's thumbprint
$thumbprint = $cert.Thumbprint

# Add the certificate to the Trusted Root Certification Authorities store
$store = New-Object System.Security.Cryptography.X509Certificates.X509Store("Root",
"LocalMachine")
$store.Open("ReadWrite")
$store.Add($cert)
$store.Close()

# Output the thumbprint for reference
Write-Output "Certificate installed in Trusted Root Certification Authorities store with Thumbprint:
$thumbprint"
```



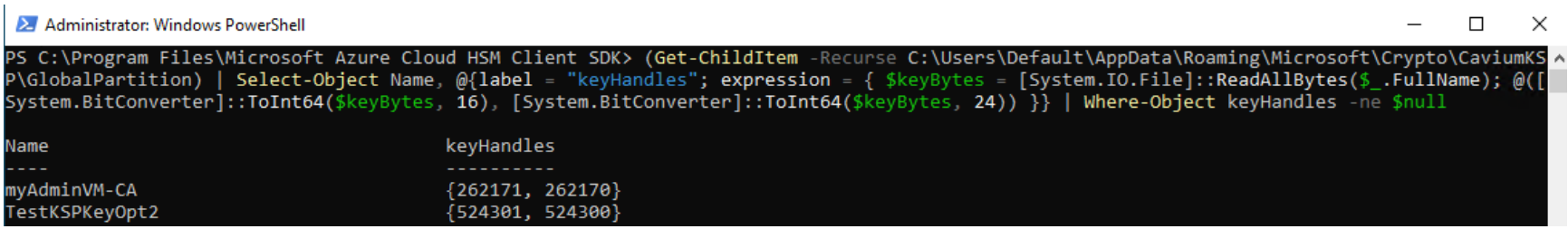
2. Import keys from Microsoft KSP into the Cloud HSM Key Storage Provider (KSP).

```
.\libs\ksp\azcloudhsm_ksp_import_key.exe -from MSKSP -RSA TestKSPKeyOpt2
```



3. Validate the creation of new Key Handles in Cloud HSM.

```
(Get-ChildItem -Recurse
C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP\GlobalPartition) | Select-Object
Name, @{label = "keyHandles"; expression = { $keyBytes = [System.IO.File]::ReadAllBytes($_.FullName);
@([System.BitConverter]::ToInt64($keyBytes, 16), [System.BitConverter]::ToInt64($keyBytes, 24)) }} |
Where-Object keyHandles -ne $null
```



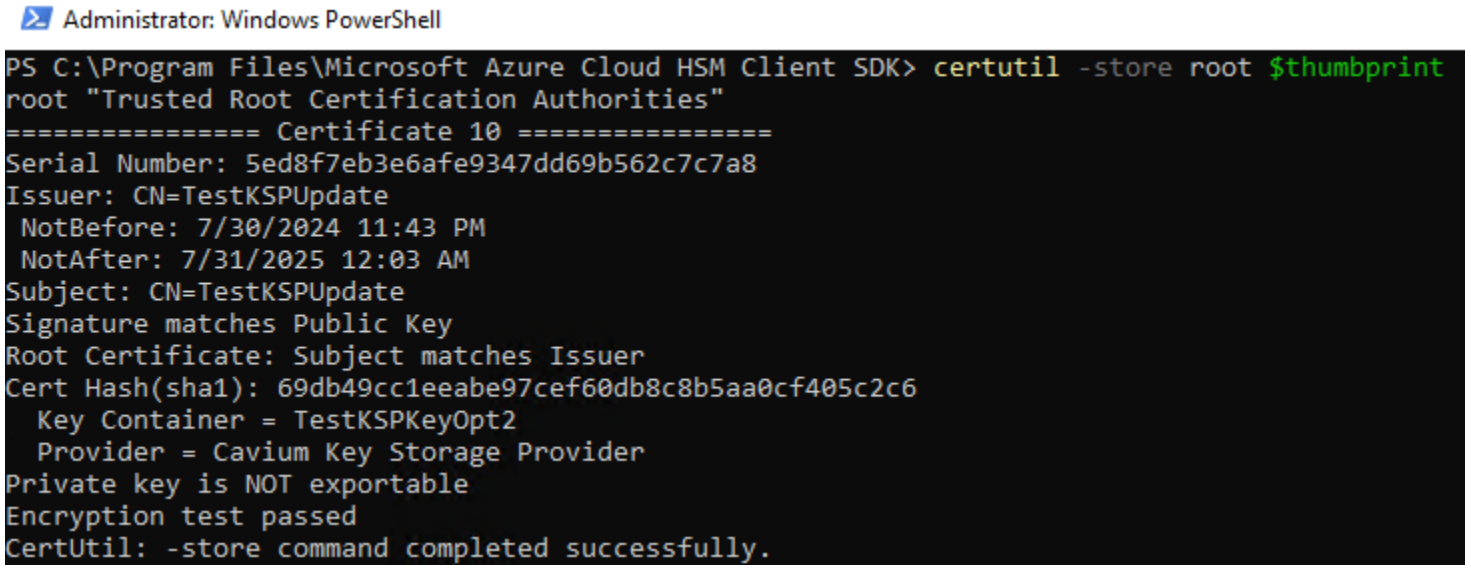
4. Import keys from Microsoft KSP into the Cloud HSM Key Storage Provider (KSP).
   a. If you selected option 1, generate self-signed certificate execute the following command.
      ```
      certutil -store my {CertificateThumbprint}
      ```
   b. If you selected option 2 generate self-signed RootCA certificate execute the following command.
      ```
      certutil -store root {CertificateThumbprint}
      ```

5. Initiate the repair operation for the specified certificate, to update and utilize the "Cavium Key Storage Provider" for Cloud HSM.
   a. If you selected option 1, generate self-signed certificate execute the following command.
      ```
      certutil.exe -f -csp "Cavium Key Storage Provider" -repairstore my {CertificateThumbprint}
      ```

c. If you selected option 2 generate self-signed RootCA certificate execute the following command.
certutil.exe -f -csp "Cavium Key Storage Provider" -repairstore root {CertificateThumbprint}

6. Verify “Cavium Key Storage Provider" for Cloud HSM was applied.
   a. If you selected option 1, generate self-signed certificate execute the following command.
   certutil -store my {CertificateThumbprint}

   d. If you selected option 2 generate self-signed RootCA certificate execute the following command.
   certutil -store root {CertificateThumbprint}



Creating a User Generated KEK

Creating a User-Generated KEK is necessary only for ADCS customers who are trying to migrate certificates from Microsoft KSP, or other 3rd Party KSP to Azure Cloud HSM KSP.

1. Creating a user KEK handle involves initial steps where customers execute the azcloudhsm_client, connect to their Cloud HSM using the azcloudhsm_util, and subsequently log in to their HSM. The azcloudhsm_client must be running for azcloudhsm_util to execute. The

provided example illustrates the process of generating a user KEK and obtaining its handle. On Linux, azcloudhsm_util is located in /bin; on Windows, it is in utils\azcloudhsm_util.

Important Note: Please keep track of your Key Handle ID generated. You'll require this Key Handle ID to finalize the process. A User-Generated KEK can be created with extractable and trusted, or non-extractable but trusted.

Option 1: User KEK with extractable and trusted. *The example below we're setting -l (key label) as userkek, -t (key type) and -s (key size) as AES, 32 bytes.*

```
./azcloudhsm_util.exe
loginHSM -u CU -s cu1 -p user1234
genSymKey -l userkek -t 31 -s 32 -wrap_with_trusted 1
```

```
Command:  genSymKey -l userkek -t 31 -s 32 -wrap_with_trusted 1

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 262259

        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
```

Option 2: User KEK with non-extractable but trusted. *The example below we're setting -l (key label) as userkek, -t (key type) and -s (key size) as AES, 32 bytes and setting the key as non-extractable.*

```
./azcloudhsm_util.exe
loginHSM -u CU -s cu1 -p user1234
genSymKey -l userkek -t 31 -s 32 -nex
```

```
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:  genSymKey -l userkek -t 31 -s 32 -nex

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 262150

        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
```

2. After the key has been generated, customers will need to set the correct attributes on their key so that it can be used as a KEK. Firstly, you will need to end your azcloudhsm_util session. Customers will then run the management util and login as the Crypto Officer. You must be logged in as the Crypto Officer when setting the attributes on the Key.

```
.\azcloudhsm_mgmt_util.exe .\azcloudhsm_resource.cfg
loginHSM CO admin adminpassword
```

3. Upon assuming the role of the Crypto Officer and logging in, proceed to configure the attributes of the previously generated key. Obtain the key handle from the previous step and execute the following command to establish its attributes, utilizing your KeyHandleID.

   Usage: setAttribute <KeyHandle> <AttributeID> <AttributeValue>. AttributeID 134 sets OBJ_ATTR_TRUSTED. AttributeValue 1 sets OBJ_ATTR_AUTH_FACTOR which is 1FA. Customers must use 134 1. No other values are supported as we only support 1FA.

```
setAttribute <KeyHandleId> 134 1
```

```
cloudmgmt>setAttribute 262150 134 1
************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. Cav server does NOT synchronize these changes with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)?y
setAttribute success on server 0(10.0.2.4)
setAttribute success on server 1(10.0.2.5)
setAttribute success on server 2(10.0.2.6)
cloudmgmt>
```

4. After configuring the attributes for the generated key, you can utilize it as a KEK (Key Encryption Key) by modifying the USER_KEK_HANDLE in your azcloudhsm_application.cfg file with the corresponding KeyHandleID.

```
DAEMON_ID=1
SOCKET_TYPE=UNIXSOCKET
PORT=1111
USER_KEK_HANDLE=262150
```

Appendix

Frequently Asked Questions

- **Why am I getting a *Certificate ERROR: [certificate signature failure]* when running azcloudhsm_mgmt_util?**
  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_mgmt_util you should see a successful connection to your HSM.

- **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 NCryptImportKey failed:- C0000001. Failed to import the key to Cavium Key Storage Provider.**
  This error commonly occurs when a user-generated KEK has not been created and the USER_KEK_HANDLE in the azcloudhsm_application.cfg file has not been updated. To enable key import functionality, you must create a user-generated KEK.

- **Why am I getting a can't open openssl-ca.cnf file when trying to migrate a certificate from Microsoft KSP from provided example?**
  This error commonly occurs when the openssl-ca.cnf does not exist. You can create one yourself or obtain it from another source.

- **Can we use our own signing servers in our corporate network which consumes Azure Cloud HSM as SaaS solution?**
  Azure Cloud HSM is IaaS only. You can use your own signing servers in your corporate network, however. You can configure Site-to-Site or Point-to-Site VPN connection from your local network gateway. This is the most common method, and it's suitable for most use cases. Set up a VPN connection between your corporate network's on-premises and Azure to where Azure Cloud HSM is deployed.
  - Tutorial - Connect an on-premises network and a virtual network: S2S VPN: Azure portal - Azure VPN Gateway | Microsoft Learn
  - You can have ADCS for your signing servers on-prem. What is required is SDK must be on your on-prem host and the client be reachable to your Azure VNET Private IPs.
    i. The customer needs to run the azcloudhsm_client on the same machine where Cavium CNG provider exists.
    ii. The azcloudhsm_client should be able to reach out to the Customers Azure vNET private IPs.

- **When testing ADCS we generated an RSAKeyPair then used CFM to Sign.**
  - Are the only accepted methods to sign/verify through Cloud HSM / LS providers (i.e azcloudhsm_util, etc.)?
    - No. Our recommendation is to use the Azure Cloud HSM SDK and the interfaces (PKCS#11, CNG, KSP, JCE, OpenSSL Engine, Etc.) it provides. You can use the Sign Tool for sign/verify operations as well.

  - ADCS is configured to use Azure Cloud HSM CNG/KSP. Can we use the signing / verify from Cloud HSM via signtool.exe?
    - Yes. The Sign Tool will automatically go to your Cloud HSM. There is no need to use azcloudhsm_util for that. We just included azcloudhsm_util in the instructions above as a way for customers to quickly test and validate their ADCS configuration.

- **Can we use self-signed certs based on the key-pair in Cloud HSM for testing or have it be issued?**
  - Yes. You can use self-signed certificates based on the key-pair for testing purposes.

- **Is there an integration or way to make available the certs via the local windows certificate store in windows?**
  - There is a tool called certreq.exe that can be used for this purpose.

- **Why am I getting an error message when trying to use the CNG and KSP provider?**
    - The Azure Cloud HSM SDK MSI package automatically registers the CNG and KSP provider for Windows. In the event the Azure Cloud HSM CNG and KSP provider failed to register or where removed you can run the following commands to register and validate configuration.
        - Register the Azure Cloud HSM CNG provider.
            - Open PowerShell and cd to the .\libs\cng directory under "C:\Program Files\Microsoft Azure Cloud HSM Client SDK"
            - Run: .\azcloudhsm_cng_config.exe -register
            - Validate CNG was registered by running .\azcloudhsm_cng_config.exe -enum which will show as 'Cavium CNG Provider'.

        - Register the Azure Cloud HSM KSP provider.
            - From the same open PowerShell window cd to the ..\libs\ksp directory under C:\AzureCloudHSM-ClientSDK
            - Run .\azcloudhsm_ksp_client.exe -register
            - Validate KSP was registered by running .\azcloudhsm_ksp_client.exe -enum which will show as 'Cavium Key Storage Provider'.

- **Why don't I see the Cavium KSP and CNG providers in ADCS?**
    - The Azure Cloud HSM SDK must be installed before setting up ADCS. The SDK registers both KSP and CNG providers during installation. ADCS enumerates cryptographic providers only at CA installation and does not update this list dynamically. It also applies strict filtering when displaying providers. If the SDK is installed after ADCS, its providers will not appear or be selectable. To resolve this, uninstall ADCS, install the Azure Cloud HSM SDK first, and then reinstall ADCS to ensure the KSP and CNG providers are available.

- **What algorithms do the Azure Cloud HSM CNG and KSP providers support?**
    - RSA encryption/decryption.
    - RSA signing with SHA1, SHA256, SHA384, SHA512, and MD5 Hash algorithms.
    - ECC curves ECDSA_P256, ECDSA_P384, and ECDSA_P521.

- **How do I change or manage private key permissions to non-administrative users on Windows?**
    - When generating a key using the Azure Cloud HSM KSP or CNG, only the Administrator is granted access to the private key. You may encounter difficulties in viewing security information or modifying permissions via MMC. This behavior is intentional. By default, when utilizing KSP or CNG, access to the private key file for a new key is restricted to SYSTEM and Administrators. If

there is a need to provide access to a non-administrative user for a specific private key file, you can achieve this by identifying the private key filename and assigning permissions directly. The private keys are in the directory C:\Users\Default\AppData\Roaming\Microsoft\Crypto\CaviumKSP on Windows for Cloud HSM keys.

- **What does azcloudhsm_ksp_import_key.exe do?**
  azcloudhsm_ksp_import_key.exe allows customers to import or represent an asymmetric key/keys in Cloud HSM KSP.
  - Supported algorithms are RSA(2048 to 4096 in multiples of 256), ECDSA and ECDH. For ECDSA and ECDH supported curves are P256, P384 and P521. Once a key is imported or represented in Cloud HSM KSP, the key should only be managed from Cloud HSM KSP.

- **What is the recommended way of "connection" to Cloud HSM from the Corporate Network (Operation/Admin tasks)?**
  e. Customers can connect through their VNET.
    - azcloudhsm_mgmt_util executes operations (admin tasks on the HSM) which requires PCO (partition crypto officer) credentials.
    - azcloudhsm_client (client daemon) connects via PCU (partition crypto user).
      - The azcloudhsm_client and customers application need to both run on the same VM within that VNET. This is because the application connects to the client process via RPC.
        a. The Azure Cloud HSM service listens on port 443 (azcloudhsm_client requests), 444 (azcloudhsm_mgmt_util Requests) and 445 (server-server communication).
        b. Front end ports are 2224, 2225
           i. TCP over TLS protocol and Ports 2224 and 2225
           ii. Ports 2224 and 2225 are for customers only.

Recommended Readings on Azure Security Best Practices

- Security best practices for IaaS workloads in Azure
- Enable just-in-time access on Virtual Machines
- Adopt a Zero Trust approach