Xorble Key Vault KSP User Guide

https://www.xorble.com

Introduction

Many organisations require Cryptographic Key material to be protected using dedicated Hardware Security Modules (HSMs) that help safeguard private key material and protects it from theft. Export of keys from HSMs is generally not possible and this protects from various attacks such as recovery of keys from disks, backup tapes or even computer memory. Normally, HSMs are deployed as network appliances in secure locations by large organisations. The HSM devices are usually quite expensive to purchase and to manage. When moving systems to the cloud providing an HSM service can also be quite expensive with solutions such as the Azure Dedicated HSM solution[1].

Azure Key Vault[2] provides a cloud based cryptographic service including both software and HSM based offerings at a much lower price point than either on-premises HSMs or Azure Dedicated HSMs. Azure Key Vault processes your keys in FIPS 140-2 Level 2 validated HSMs. Azure Key Vault provides modern RESTful API[3] for applications to consume but this means applications have to be written specifically to use Key Vault.

Windows applications use the native CryptoAPI interfaces by default and therefore are unable to consume Key Vault. The Xorble Key Vault KSP provides an Azure cloud-based Hardware Security Module (HSM) protected Key Storage Provider (KSP) for general cryptographic use by Windows. The KSP is designed to provide a solution for organisations that require an HSM backed service for application such as PKI etc. The KSP supports RSA (2048, 3072 and 4096bit) and Elliptical Curve Algorithms (ECDSA and ECDH at 256, 384 and 521 bits).

Windows components tested with the KSP include the following:

- Active Directory Certificate Services (AD CS)
- SSL/TLS
- OCSP Signing

Beta KSP and Pricing

The KSP is currently a beta release and hence should not be used for production workloads. The Beta KSP is also time limited and will not function after December 2020. The beta version is digitally signed with a test signing key rather than a key issued by a public CA. To use the KSP, the test Root certificate will need to be added to the trusted root store on the local machine. The GA release version of the KSP will be digitally signed by a public root CA.

The beta KSP is free to use. The licensing and costs for the GA version of the KSP have not yet been finalised.

---

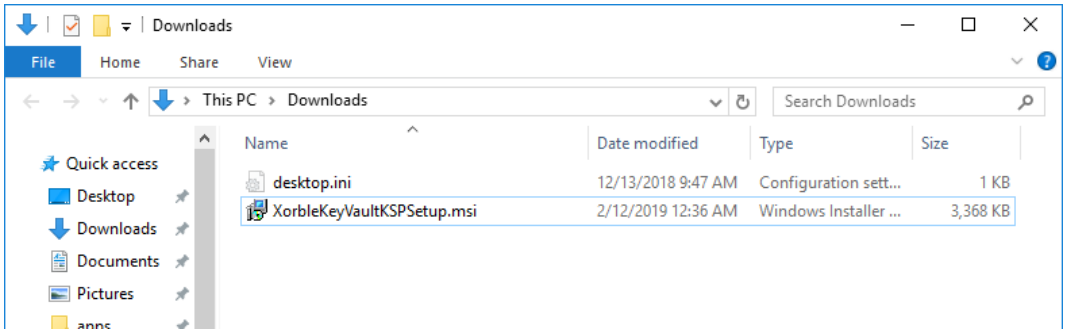
[11] Azure Dedicated HSM - https://azure.microsoft.com/en-us/services/azure-dedicated-hsm/

[2] Azure Key Vault - https://azure.microsoft.com/en-us/services/key-vault/

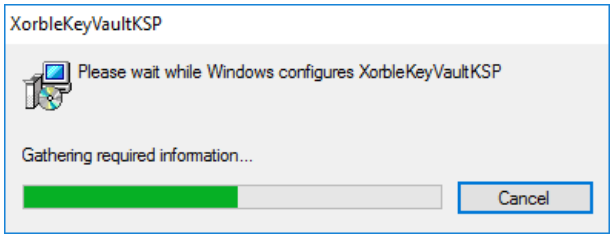
[33] Azure Key Vault REST API - https://docs.microsoft.com/en-us/rest/api/keyvault/

Installing the KSP

Download the KSP installation file and then select and run it.



Wait for the installation to complete (this should only take a few seconds):



Checking the Installation

The installation copies files to various directories, it registers the KSP and also registers an event source.

KSP Installed Files

The installation should copy the installation files to the following

Copies the following files to c:\windows\system32 (%SystemRoot%\System32)

- boost_date_time-vc141-mt-x64-1_68.dll
- SSLEAY32.dll
- XorbleKeyVaultKSP.dll
- boost_system-vc141-mt-x64-1_68.dll
- zlib1.dll
- LIBEAY32.dll
- cpprest_2_10.dll
- XorbleKeyVaultKSPInstall.dll

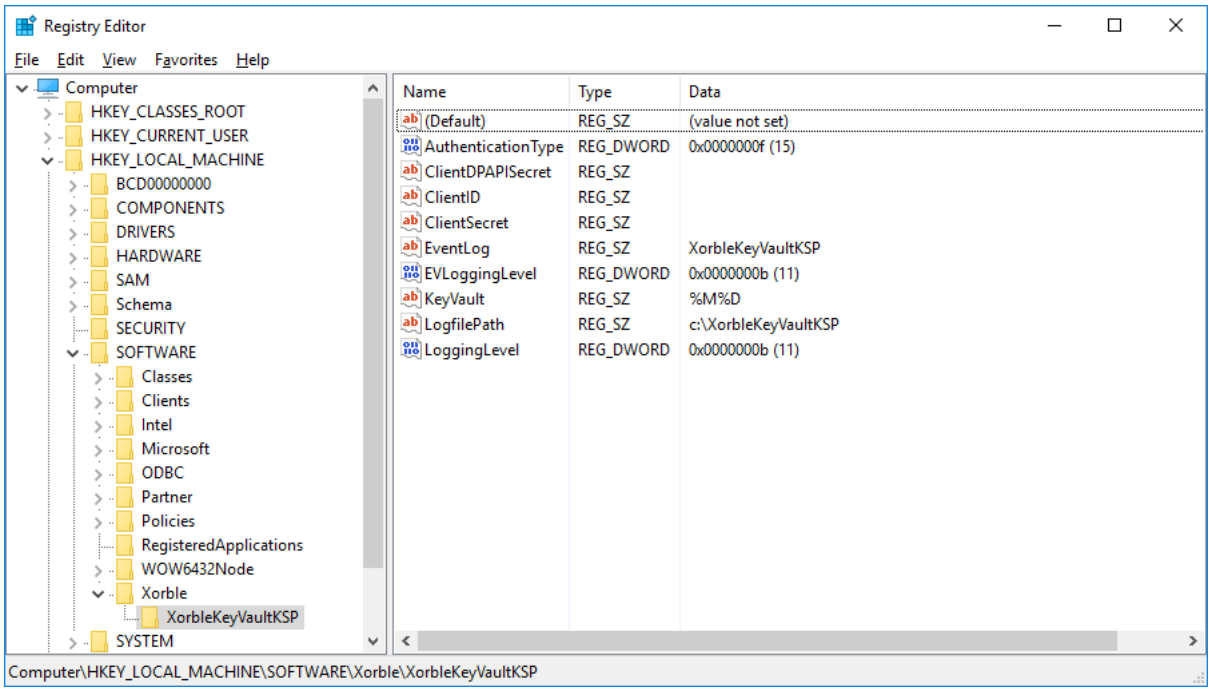
Copies the following file to C:\Program Files\Xorble\XorbleKeyVaultKSP\

- XorbleKVKSPUserConfig.exe

Registry keys

The installation will create a new registry key and values at the following location:

- HKEY_LOCAL_MACHINE\SOFTWARE\Xorble\XorbleKeyVaultKSP



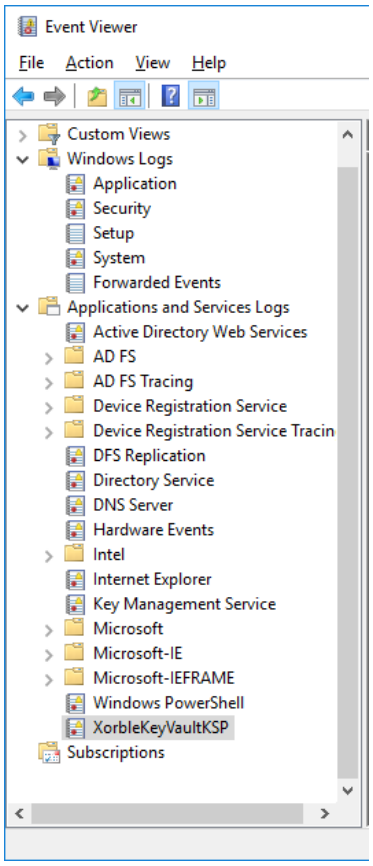
KSP Registration

The provider registration of the Cryptographic Provider in the registry can be seen at the following registry path:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Cryptography\Providers\Xorble Key Vault Key Storage Provider\UM

New Event Provider

The installer also registers a new event source which can be shown in the event viewer as shown below:



Azure Endpoints Accessed by the KSP

The KSP needs to be able to access various endpoints within Azure. Specifically, the KSP needs to be able to communicate with Azure Active Directory and Azure Key Vault. The URLs that it needs to communicate with are as follows:

- https://<vaultname>.vault.azure.net
- https://login.windows.net

Create the Azure Key Vault

Each server should have one or more Azure key Vaults for storing keys. Typically, there should be a machine Key Vault for storing key material associated with the server and potentially zero or more additional Key Vaults for storing keys associated with different users or service accounts. It is expected that in most scenarios only a single Key Vault per server will be required.

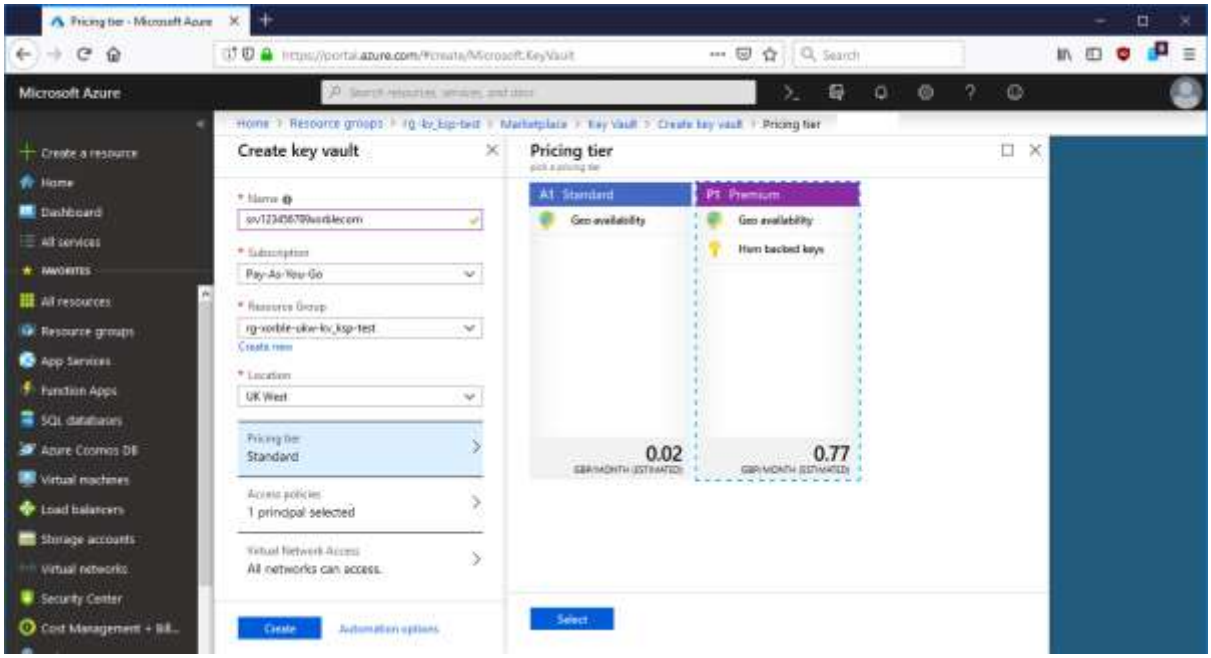
Configure the Key Vault Name

The default Key Vault name used by the KSP is based on the Fully Qualified Domain Name (FQDN) of the server. In order to create a name that is compatible with Azure, the FQDN has all the "." characters removed, is converted to lower case and truncated to 24 characters.

Hence for a server SRV123456789 in domain Xorble.com the Key Vault name would be "srv123456789xorblecom".

Azure Key Vault Premium and Standard

In order to have HSM backed keys you MUST create a Premium Key Vault. Key Vault Standard uses software backed keys rather than HSM backed. When creating the Key Vault, select Premium as shown below:



The KSP will always attempt to create an HSM based key but if this is rejected (http 403 error) then it will be retried with a software key based request. This is potentially useful in testing scenarios as Standard Key Vaults incur less cost than Premium ones.

Create the Key Vault as close geographically to the server as practical for performance reasons – if the server is a virtual machine (VM) within Azure then create in the same region as the VM.

Networking Considerations for Azure VMs

If the Key Vault is to be used by an Azure VM then the Key Vault should ideally be configured to use a Service Endpoint so that it is only available on the Subnet/VNet of the VM and Internet access to the vault would be blocked. By using a Service Endpoint, access to the vault is limited to only virtual machines on the specified Subnet(s).

Authenticating to the Key Vault

Azure Key Vault uses Azure Active Directory (AAD) as its the source of authentication and hence for every server created there needs to be a security principal created within Azure.

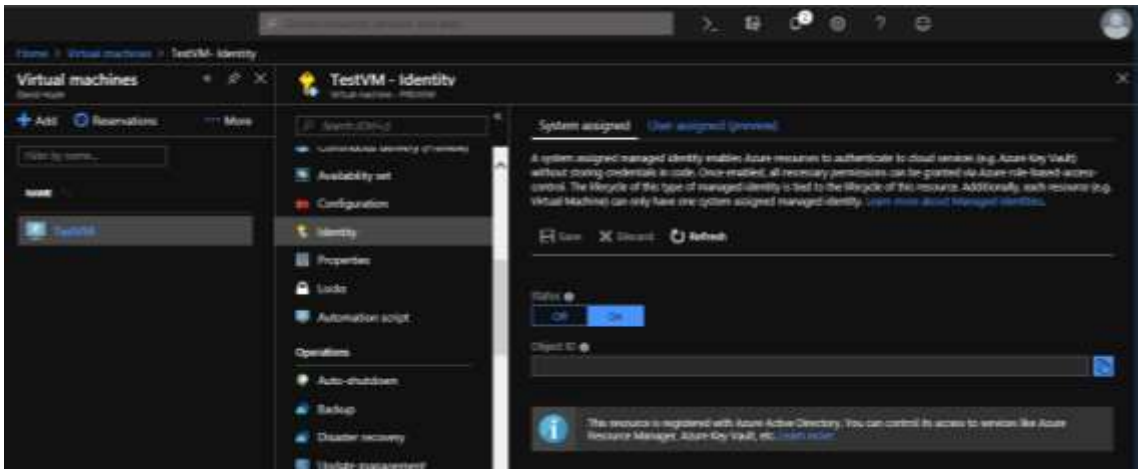
Authentication to AAD from the KSP can use one of the following methods:

- Managed Service Identifier (MSI)
- Managed Service Identifier (MSI) with Client ID (client_id)
- Service Principal and Key
- Service Principal and Certificate (not yet supported)

For Windows Virtual Machines running in Azure, the default will be to use MSI based authentication as that typically provides the strongest security and is the easiest to configure.

Creating the MSI based Service Principal

To create an MSI based Service Principal, simply enable the System Assigned Identity as shown below in the Azure Portal:

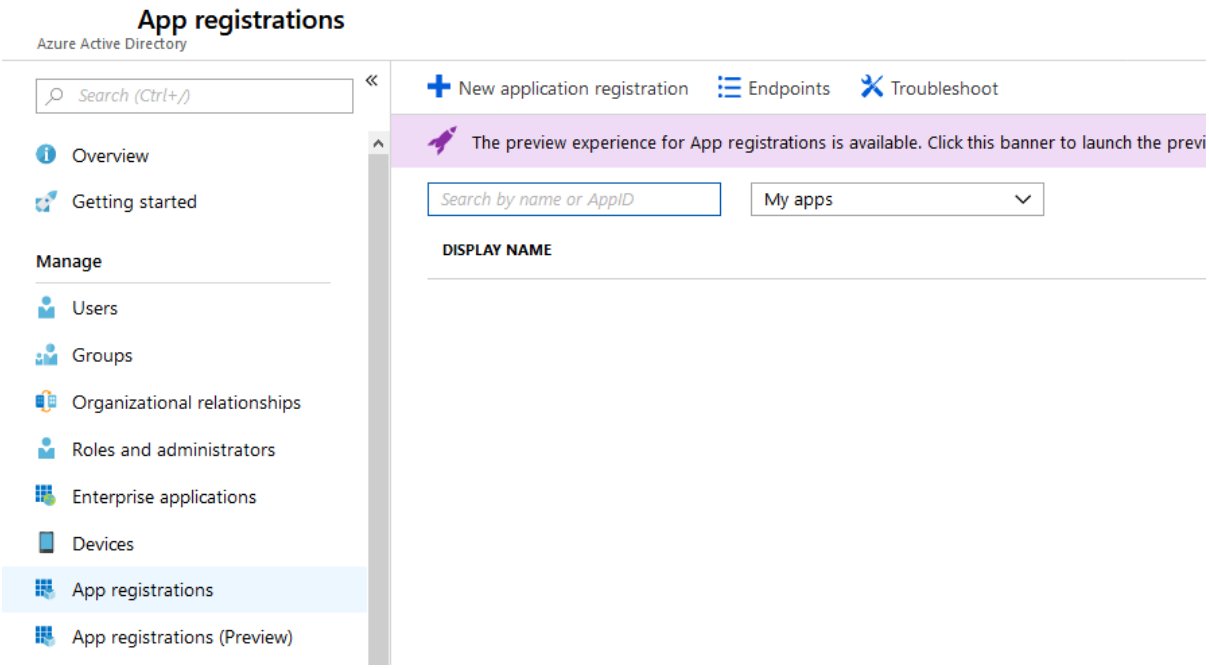


More details on the use of MSI can be found at the following location:

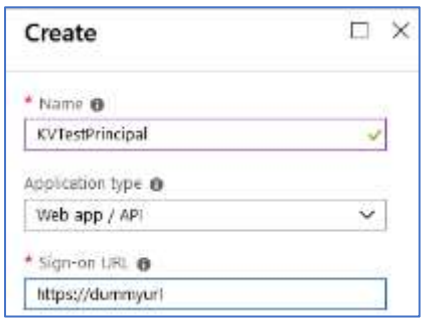
https://docs.microsoft.com/en-us/azure/active-directory/managed-identities-azure-resources/how-to-use-vm-token

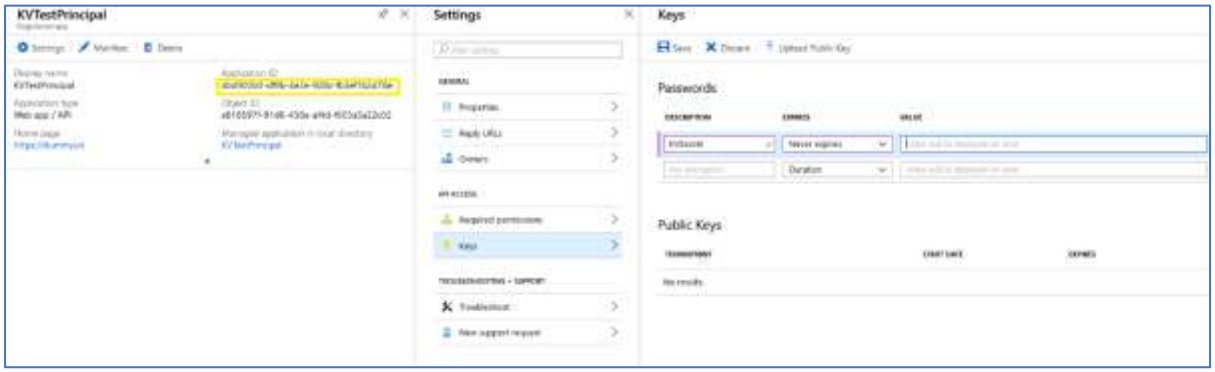
Creating a non MSI Service Principal

Open the Azure portal and select Azure Active Directory and then select ***App registrations*** as shown below. Then select ***New application registration***:



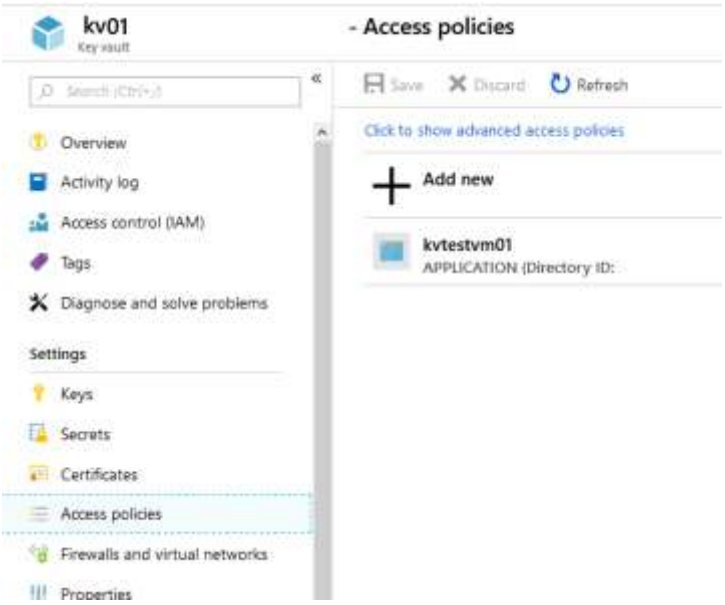
Enter a name for the Service Principal and a dummy URL (this will not be used). A new Principal will be created with a Client Id as shown in yellow below. A Secret Key will be needed for the principal. Select ***Settings*** and then ***Keys*** and then enter a Description, Expiration Date. When the Password (Secret) is saved it will be displayed. Make sure to copy the secret to use later.





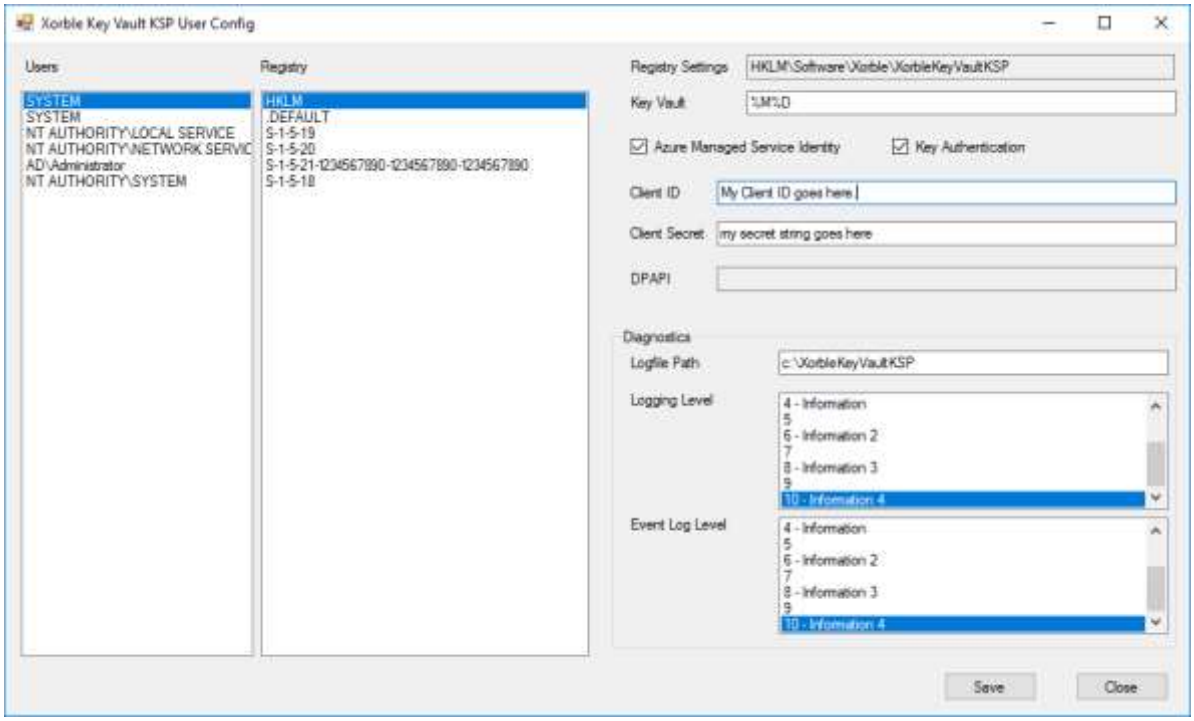
Setting Key Vault Permissions

The Key Vault Access Policies will need to be updated to allow the above Security Principal (either MSI or non MSI) to be able to perform key operations. Select the Access Policies on the Key Vault, select Add new and then add the Security Principal and give all Key and Certificate Permissions as shown below:



Configuring the Provider

Run the “XorbleKVKSPUserConfig.exe” configuration application within “C:\Program Files\Xorble\XorbleKeyVaultKSP”. The below dialog is displayed.



By default, when using an Azure VM with MSI based authentication there should be no need to change the default settings. The default Key Vault name is set to “%D” which is converted to the use fully qualified domain name (converted to lower case, with “.” characters removed and truncated to 24 characters).

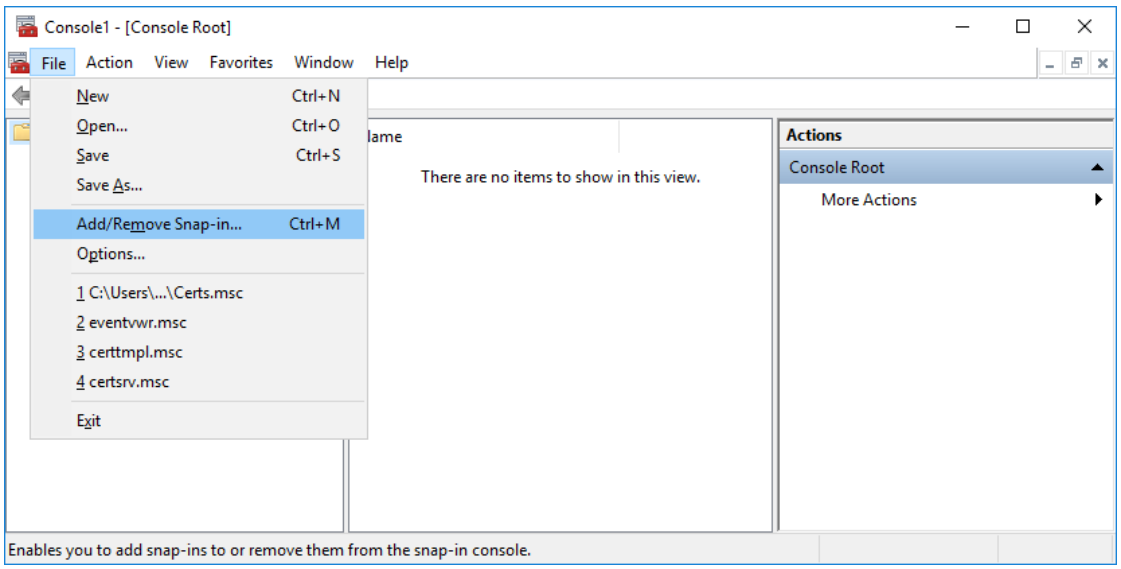
Select the either “Azure Managed Service Identity” or “Key Authentication” as required. If using a Key based Principal then enter both the Client Id and Client Secret as shown above and ***Save***.

When using Key and Secret based authentication, the KSP will protect the secret using the Data Protection API (DPAPI) upon first use. The secret value in the registry is replaced by a DPAPI protected value.

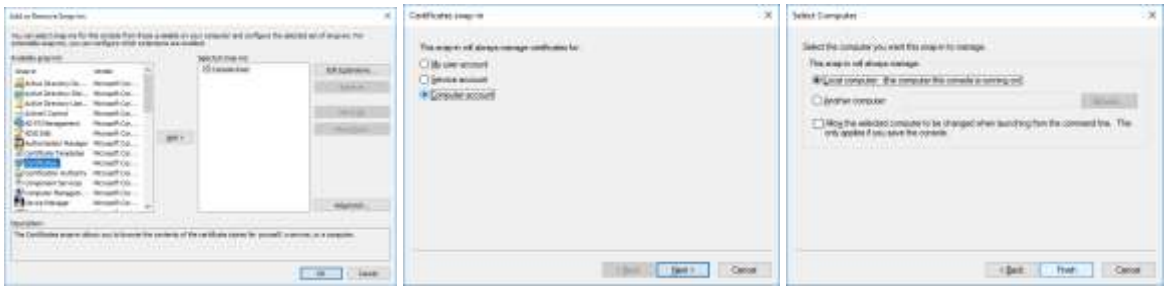
If applications run under different user contexts (rather than as SYSTEM) then additional settings can be added for different service accounts.

Test the Xorble Key Vault KSP

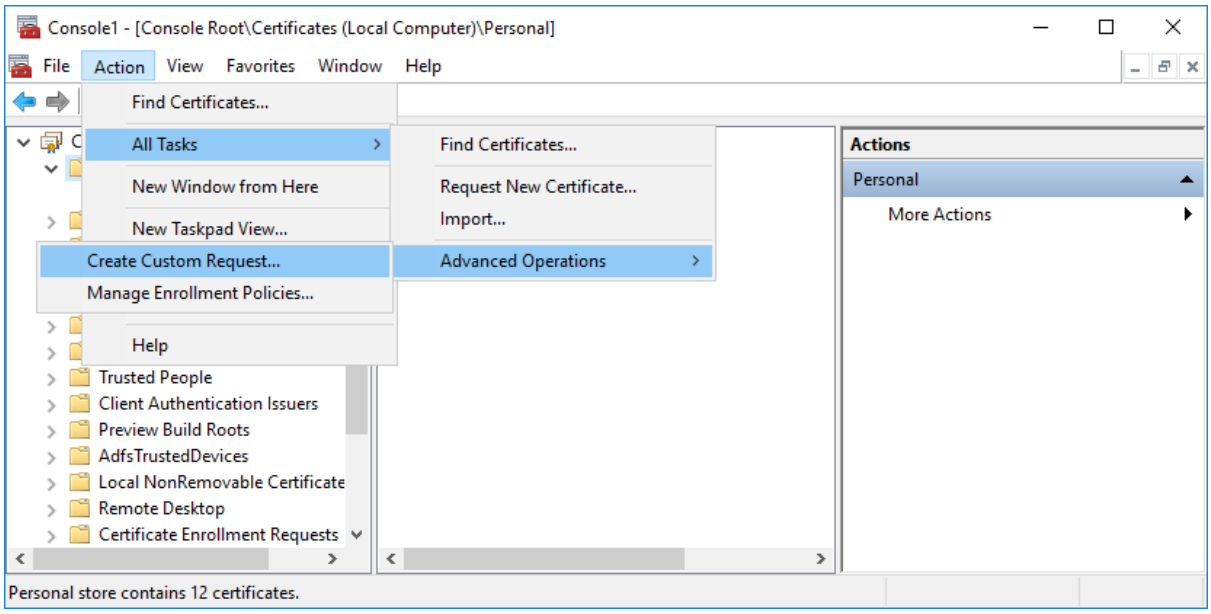
Run MMC.EXE and then select ***File***, ***Add/Remove Snap-in*** as shown below:



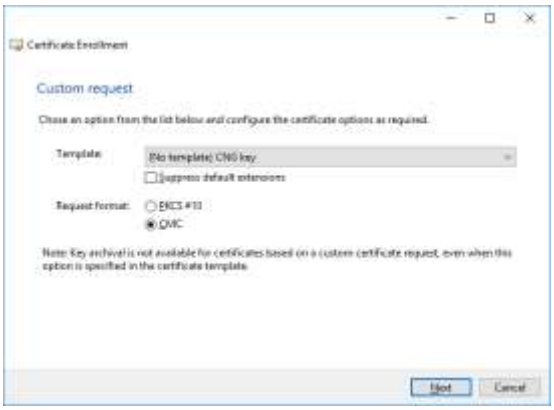
Select ***Certificates*** and then ***Add***. Then ***Computer Account*** and ***Next*** and then ***Finish*** as shown below:



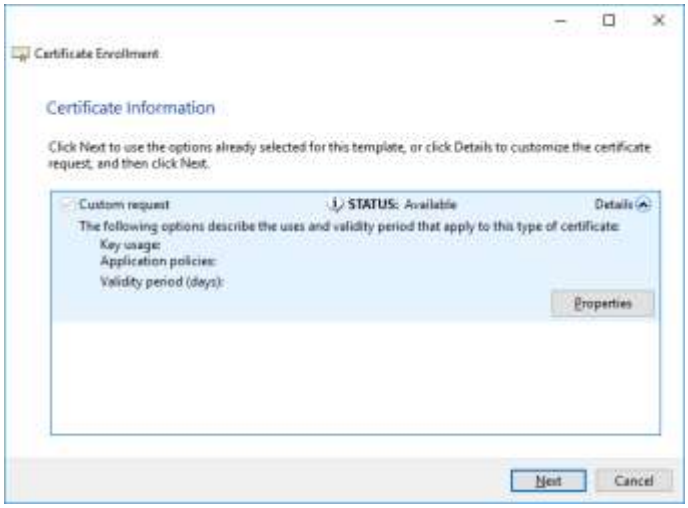
Select the Personal certificate store and then Select ***Action***, ***All Tasks***, ***Advanced Operations***, ***Create Custom Request*** as shown below:

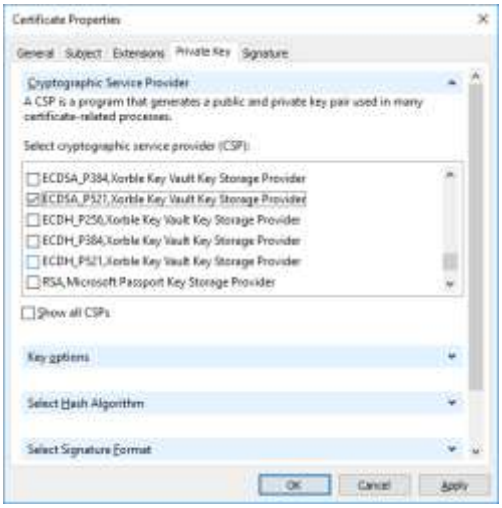


Select ***Next*** and then select ***Proceed without enrolment policy*** and then ***Next, Next*** as shown below:

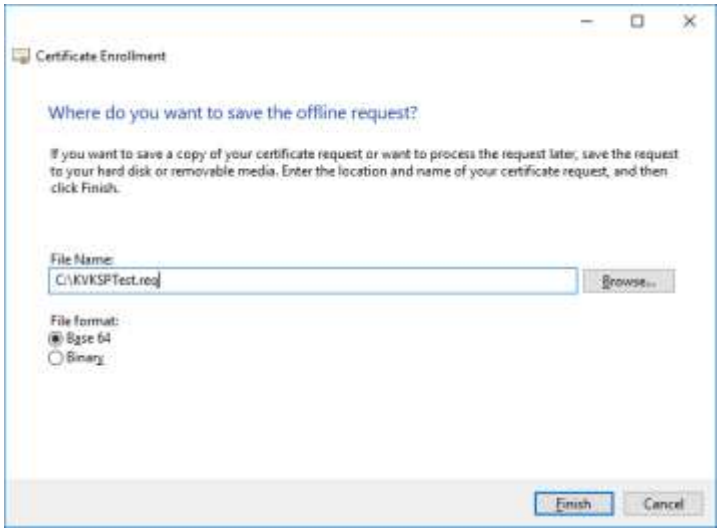


Select ***Details*** and then ***Properties*** and then select the ***Private Key*** tab. Deselect all CSPs apart from the ***ECDSA_P521 Xorble Key Vault Key Storage Provider*** (The Microsoft RSA CSP is selected by default and needs to be deselected) and then ***OK*** as shown below:





Select ***Next*** and then Enter a file name (***C:\KVKSPTest.req***) and ***Finish***.



Finally, open a CMD prompt and show the contents of the request file using the following command:

certutil -dump c:\KVKSPTest.req

The output should include the following fragment (in red):

```
  Attribute[3]: 1.3.6.1.4.1.311.13.2.2 (Enrollment CSP)
    Value[3][0], Length = 54
    CSP Provider Info
    KeySpec = 0
    Provider = Xorble Key Vault Key Storage Provider
    Signature: UnusedBits=0
```

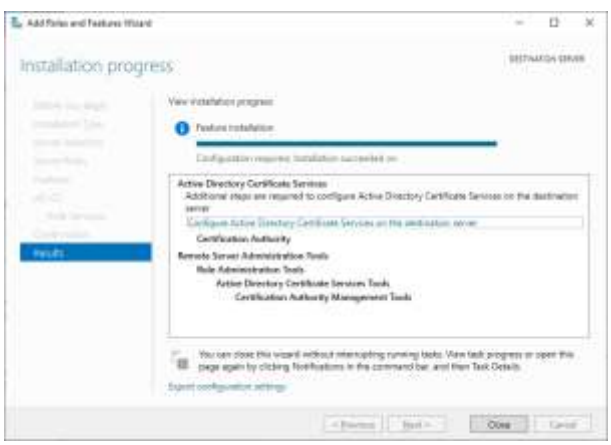
Using the Xorble Key Vault KSP

After successfully installing and configuring the KSP, it can be used for a variety of purposes including the following:

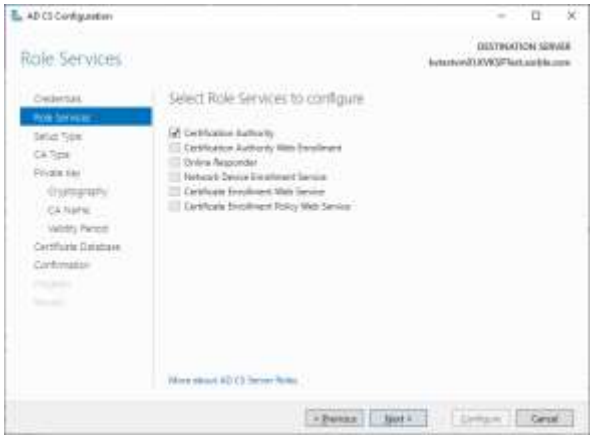
- Active Directory Certificate Services (AD CS) – CA Certificate Keys
- Internet Information Service (IIS) – Web Server Certificates
- Computer Certificates
- Active Directory Certificate Services – OCSP Responder signing certificate

Install and Configure AD CS to use the KSP.

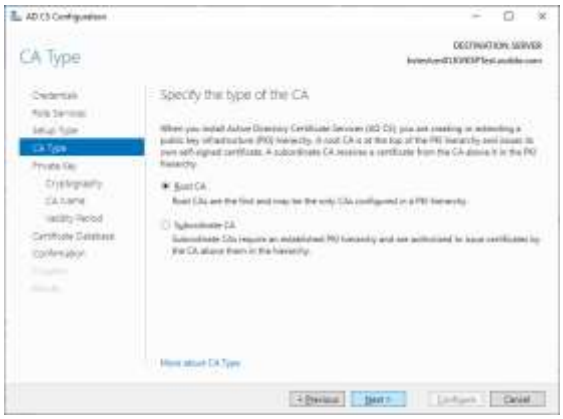
First add the Certification Authority role to the server, then select configure and ***Next***:



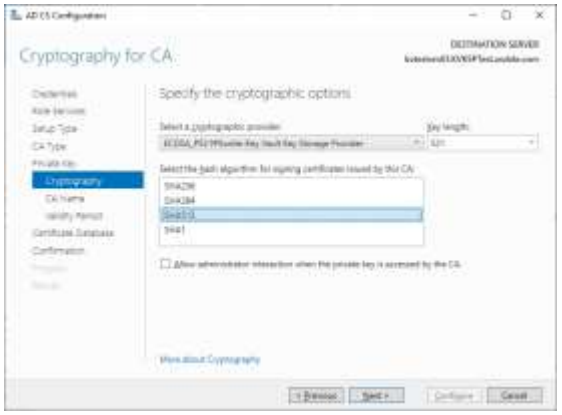
Select ***Certification Authority*** and ***Next***, ***Next***:



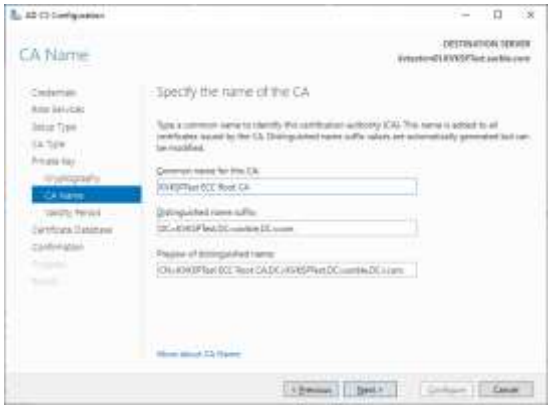
Select either ***Root CA*** or Subordinate CA (in this sample a new Root CA is selected) and ***Next*** and then select ***Create a new private key*** and ***Next***:



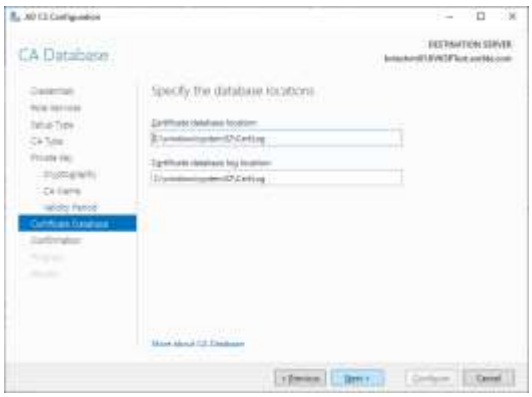
Select the Xorble KSP and then pick the required key type (RSA or ECC), Key Length and Hash Algorithm (SHA 256, 384 or 512) and ***Next***.

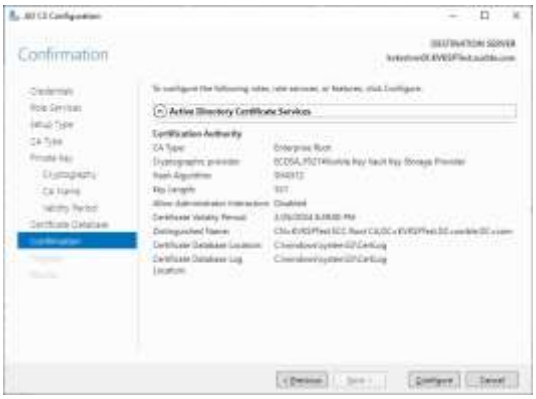


Enter the name of the CA and ***Next***, and then specify the lifetime and ***Next***:

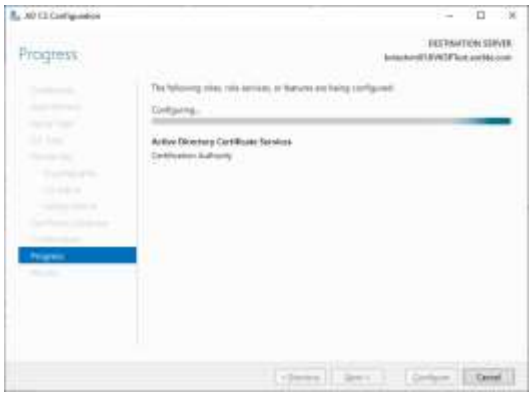


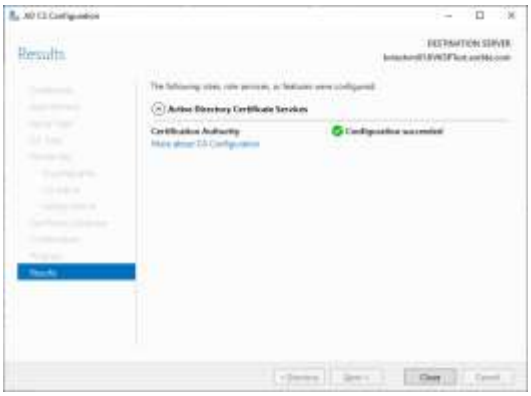
The specify the location of the CA database and then ***Next*** and finally select ***Configure***:





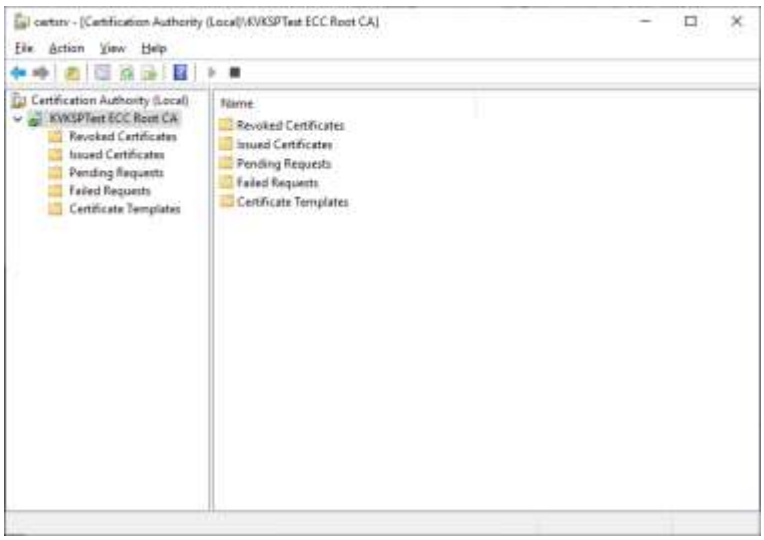
Wait fior the CA to complete configuration:





Check the CA is installed correctly with the correct KSP.

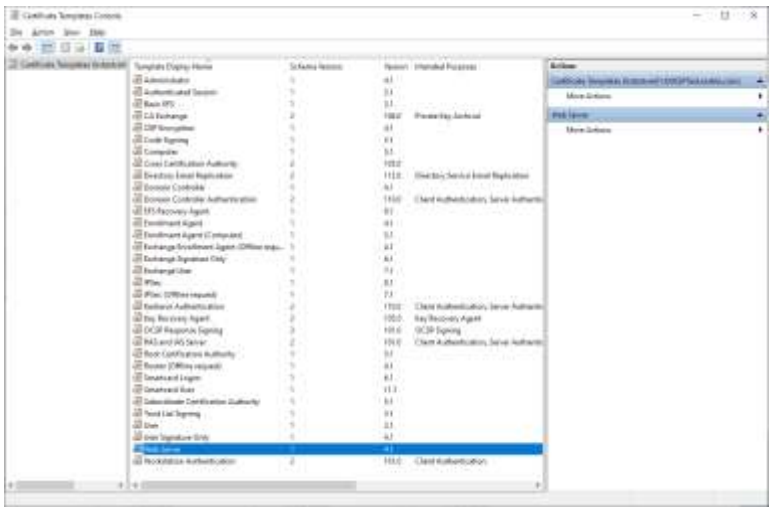
First open the Certification Authority management console and then display the properties of the CA as shown below to confirm the Key Vault KSP has been selected:



Create Web Server Certificate to use the KSP

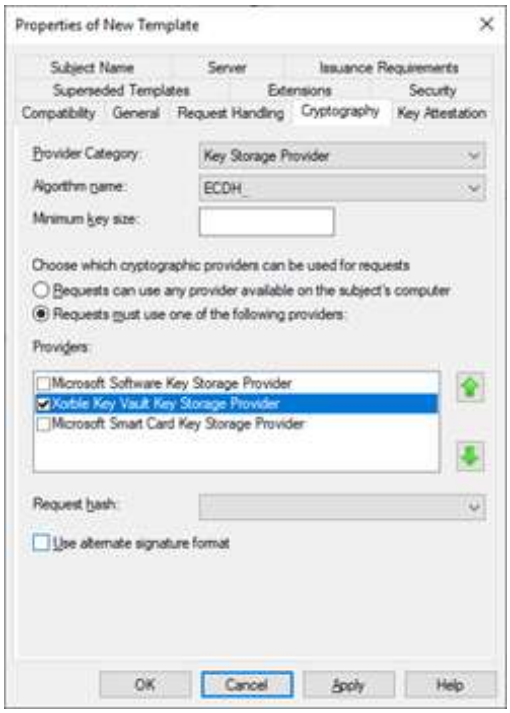
Open the Certificates Templates MMC. This allows you to create and configure templates to use the KSP.

Select the Web template and then Duplicate it. Set the compatibility settings to at least 2008 R2:

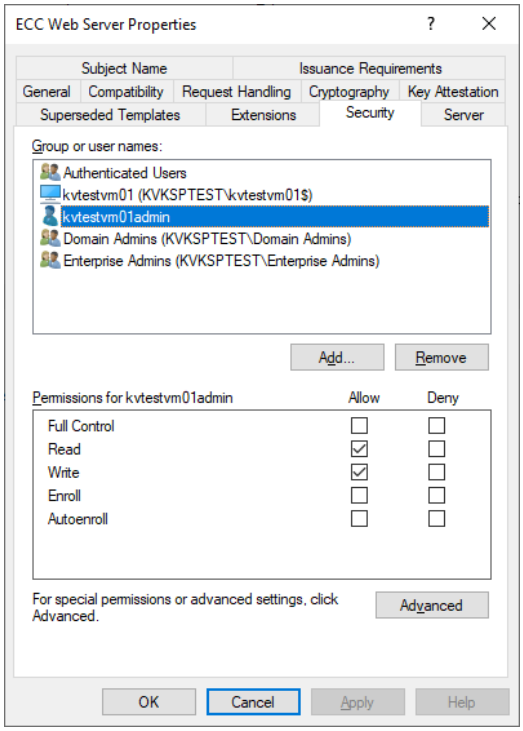


Set the name of the template (in this case for an ECC or RSA certificate):

Then specify the KSP and the algorithm to be used together with key length. For a web server select ECDH (256 or384bit) or RSA (2048 or 4096 bit) as shown below:



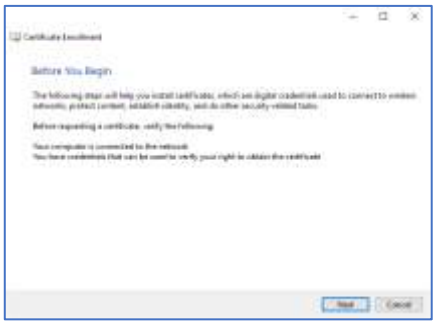
Then update the permissions on the template to allow web servers to enrol for the template. Typically the best way to do this would be to create a group in AD for web servers, give this group permission to enrol andf then add all web servers into this group.



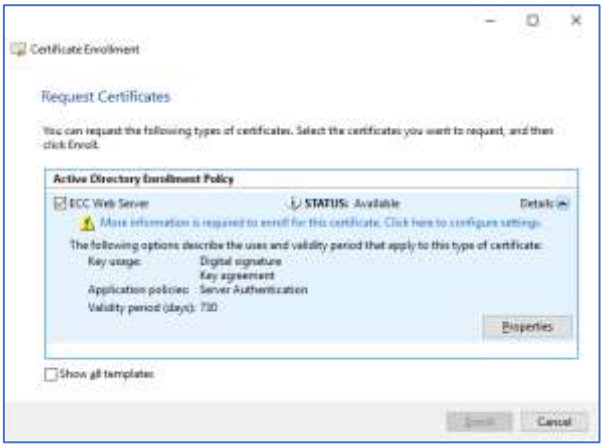
Finally add the new template to be issued by the CA.

Issuing a new Web Server Certificate

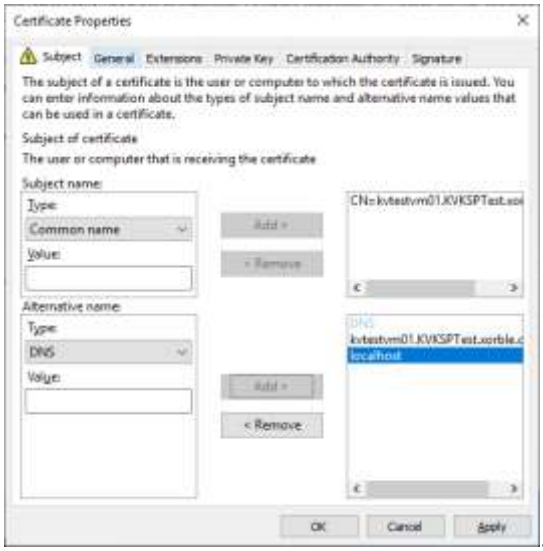
Start the local computerCertificates MMC and select the personal store. Select ***request new certificate*** and then ***Next***.

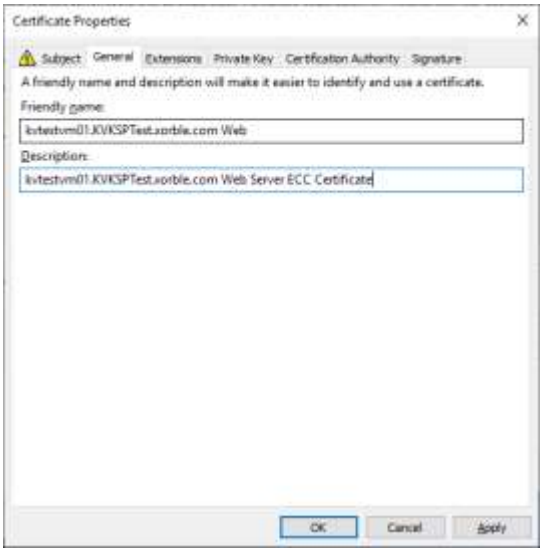


Select ***Next*** and then click on ***Details***:

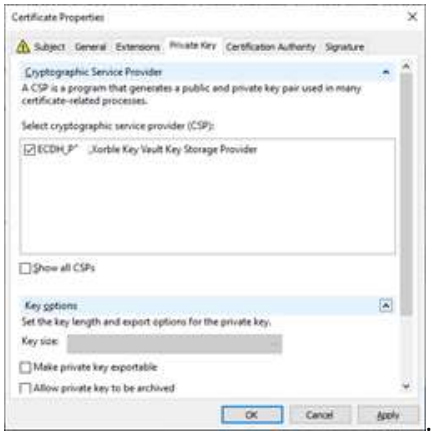


Add the Web server FQDN as the common name and as a DNS Alternative Name. Add a friendly name:

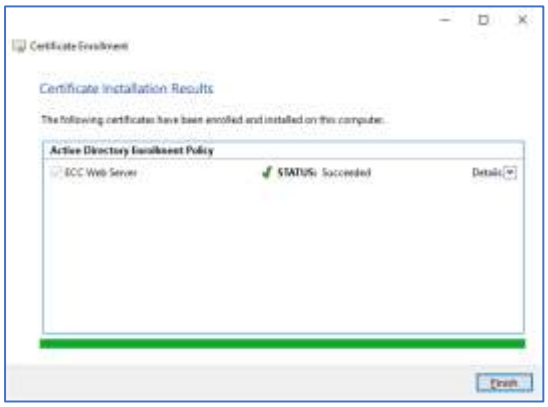




Select private key properties and KSP with the correct key strength and ***OK*** and then ***Enroll***:

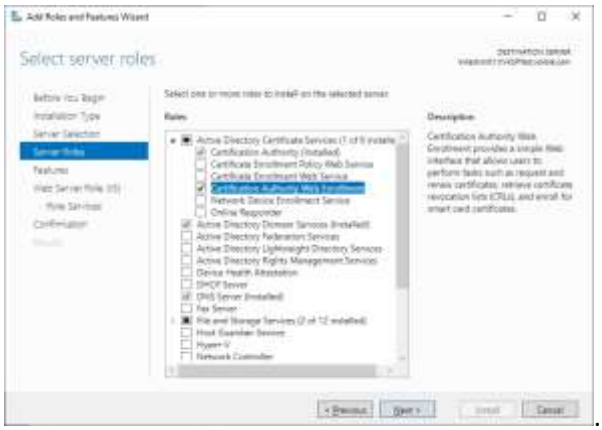


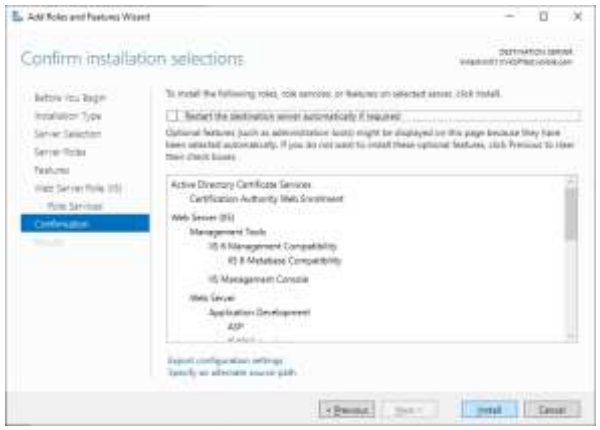
Wait for the enroll to complete.



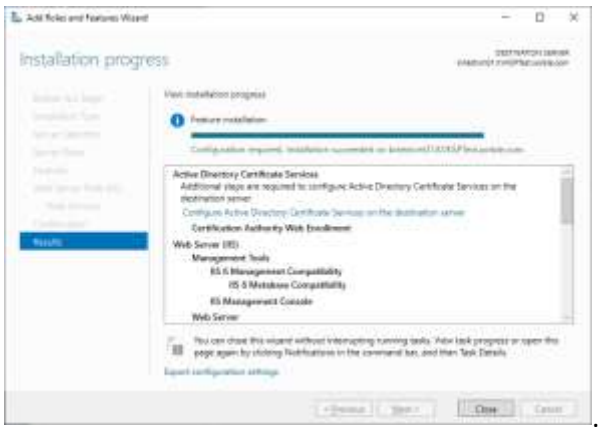
Add Certificate Authority Web Enrollment Role

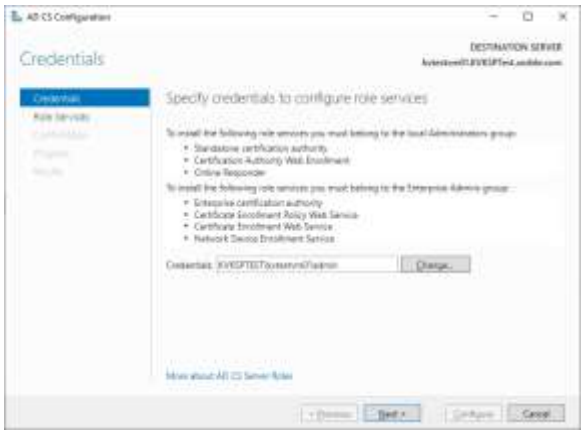
Start add roles and features and select the Web enrollment component and ***Next*** and then ***Install.***



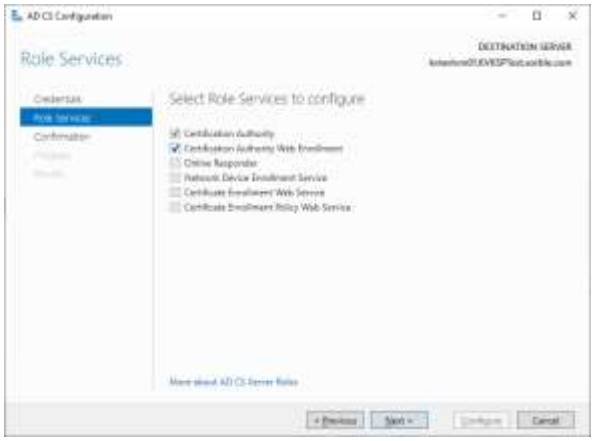


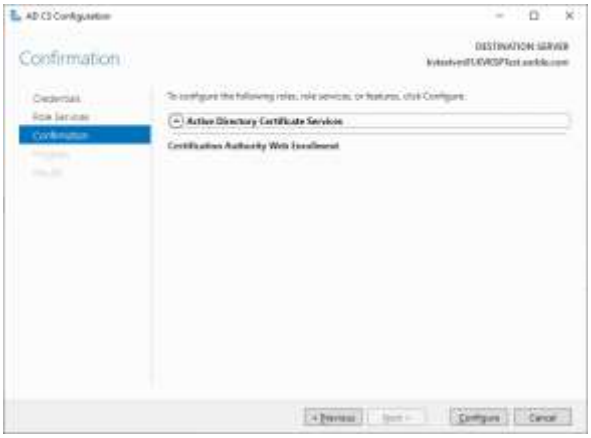
After it installs, select configureand then ***Next***:



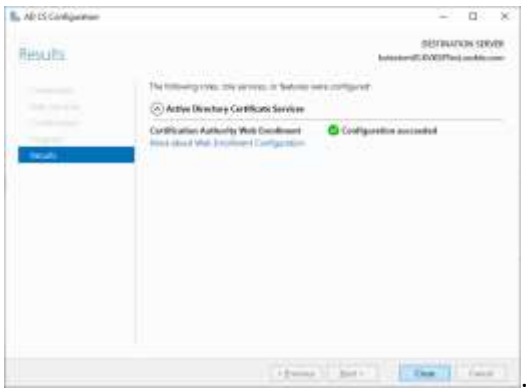


Select the Web enrollment role and ***Next***, ***Configure***:



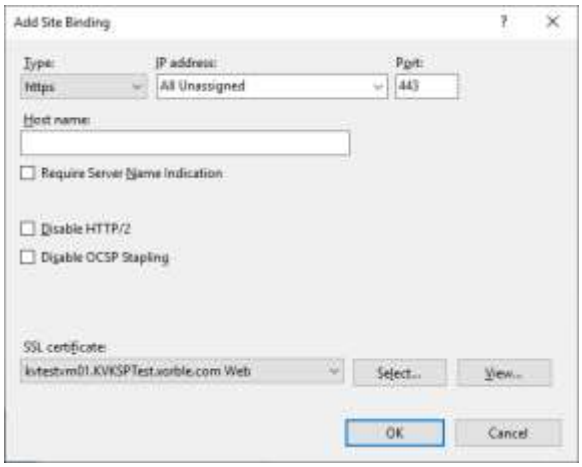


Wait for the configuration to complete and then start IIS Manager and select server certificates:




Select ***Add bindings*** and then ***Add***:

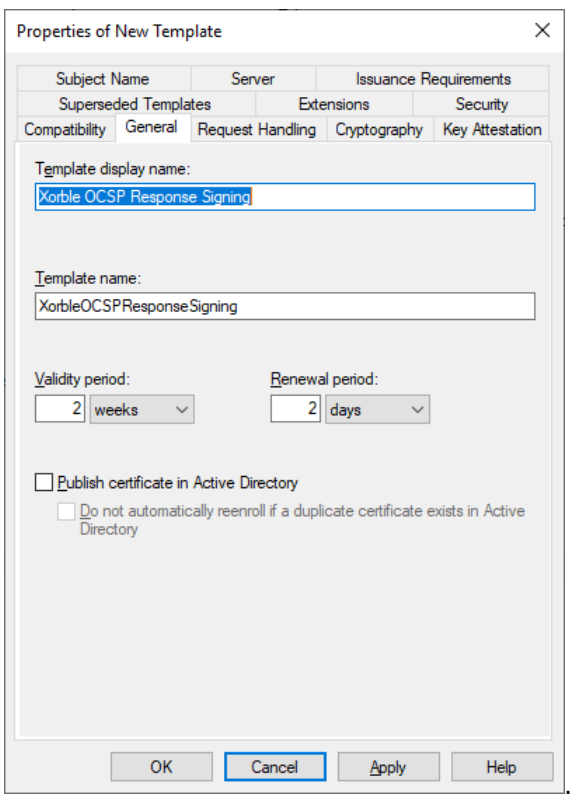

Select ***https*** and then select the Web server certificate and ***OK***.

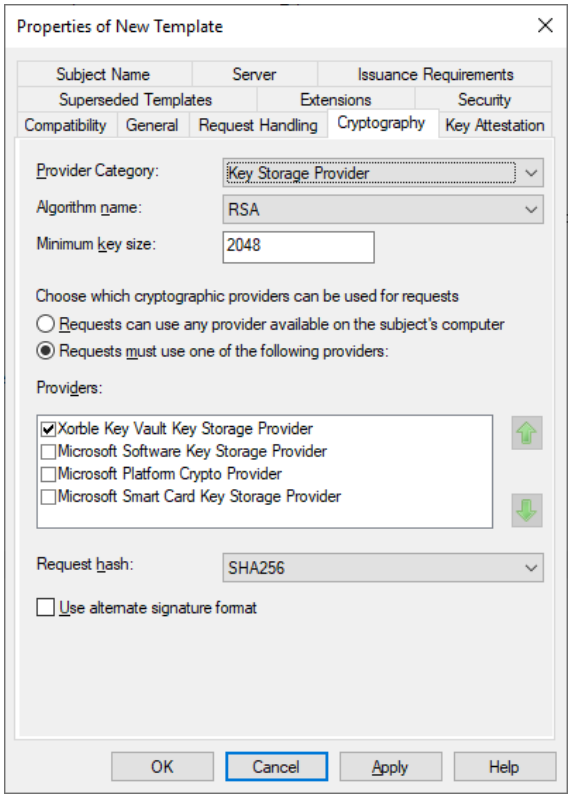


The Web server should now use the KSP certificate for SSL/TLS communications.

Configure OCSP Responder to use the KSP

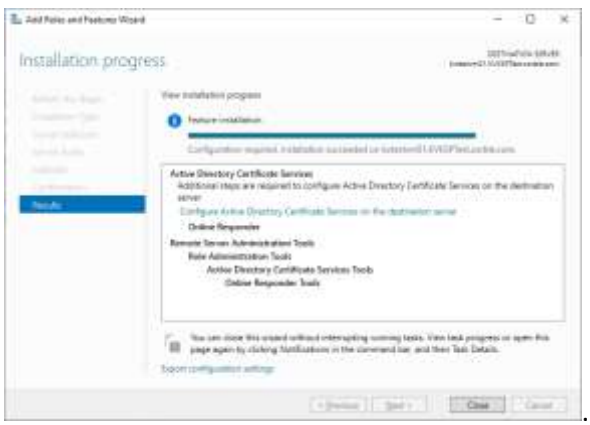
Create a new OCSP template by duplicating the exiting one. Change the display name as shown below. Select cryptography and then select the Xorble RSA algorithm as shown below and ***OK***:

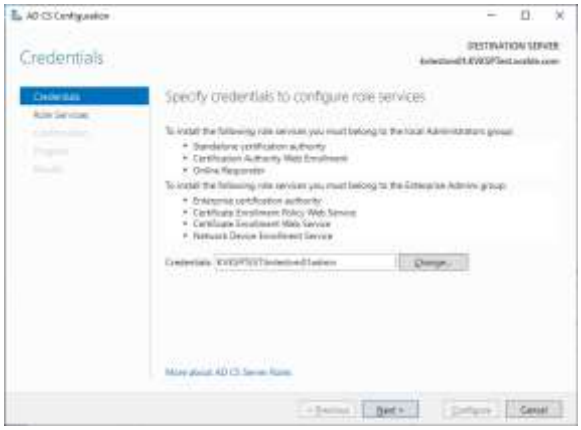




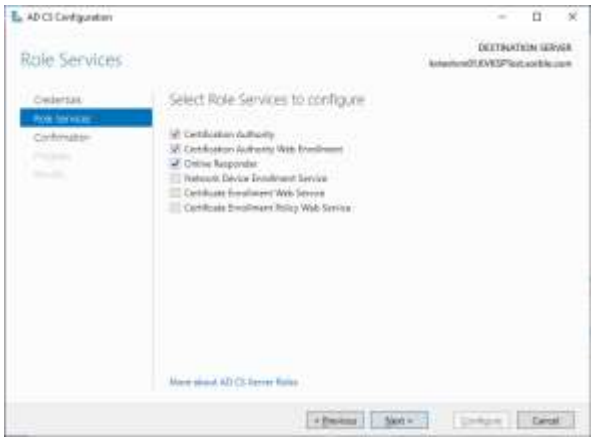
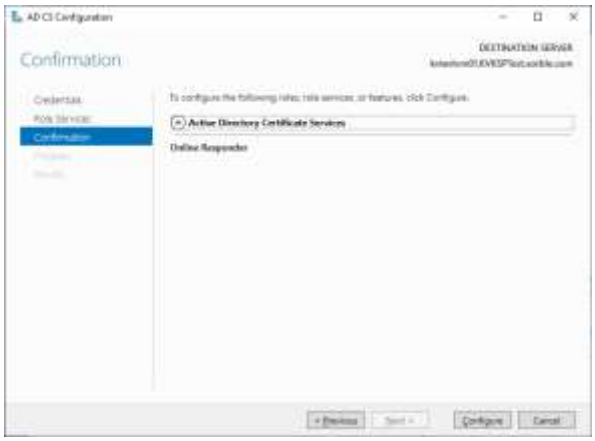
Add this new template to be issued be the CA.

Add the online responder Certification Authority role and select Configure and ***Next***:

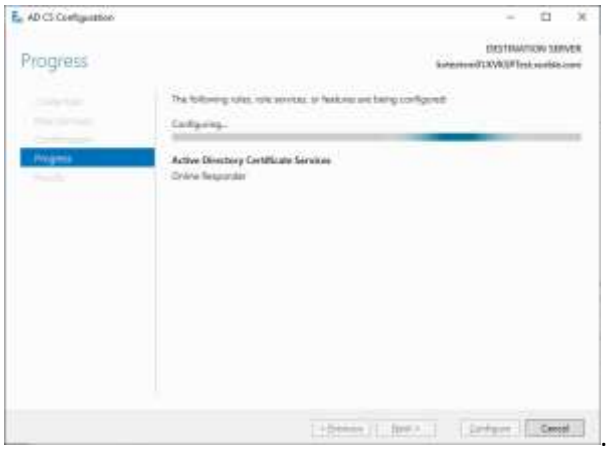
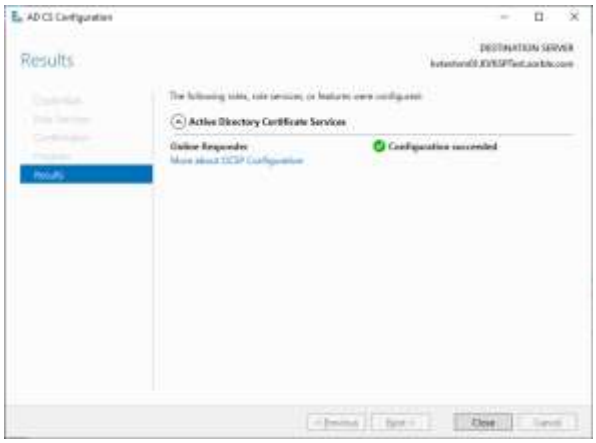




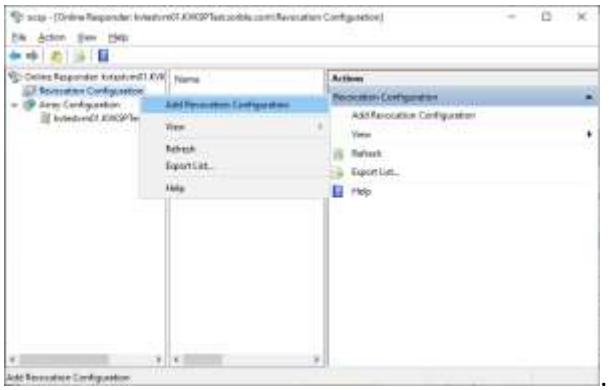
Select the responder and ***Next*** and then ***Configure***:

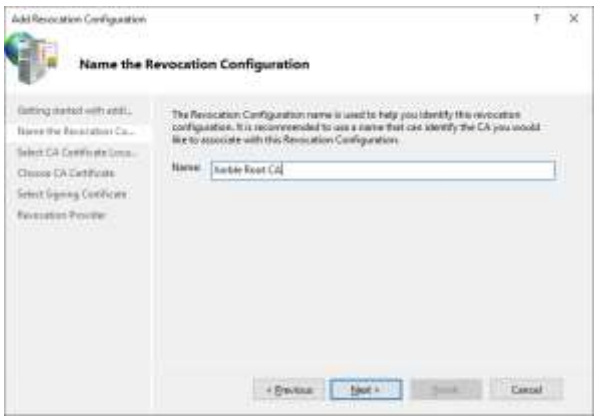
Wait for the configuration. To complete and ***Close***:

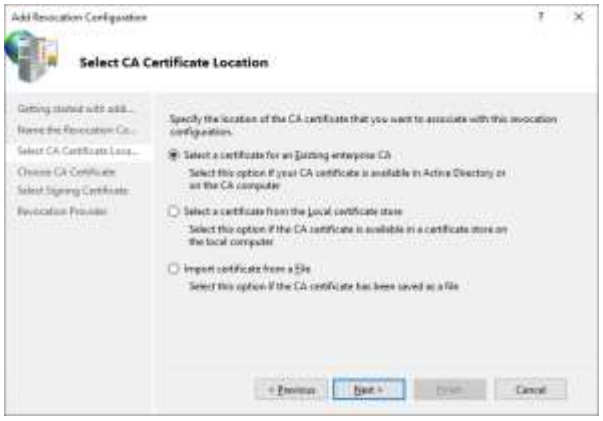



Start the online responder management console and select ***add revocation configuration*** and then ***Next***

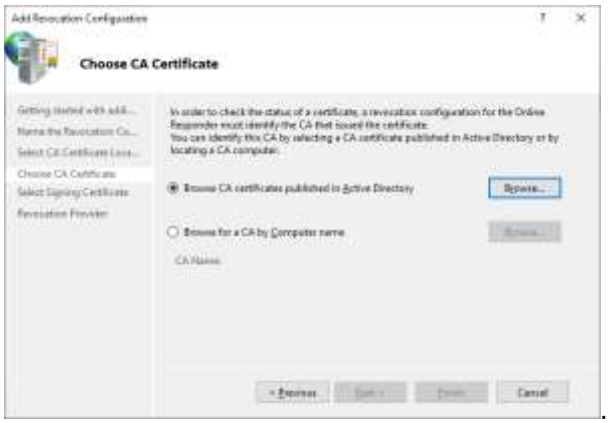


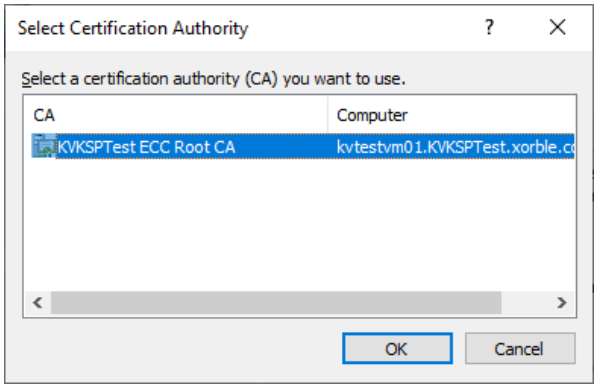

Add a name for the configuration based on the CA name and ***Next***:



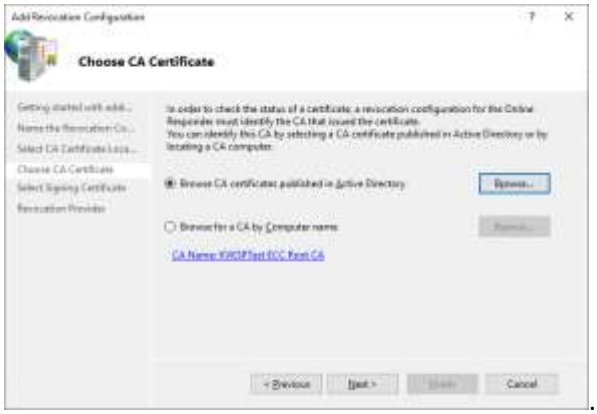


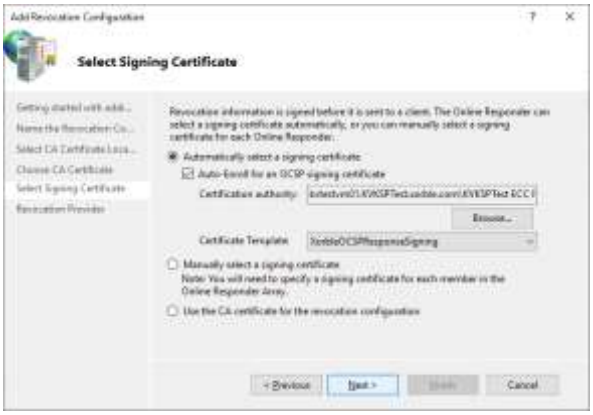
Browse fir the CA, select it and ***OK***:



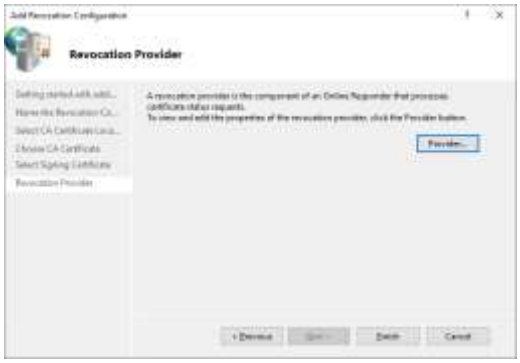


Then select ***Next*** and then select ***Auto Enroll for an OCSP signing certificate*** and ***Next***:



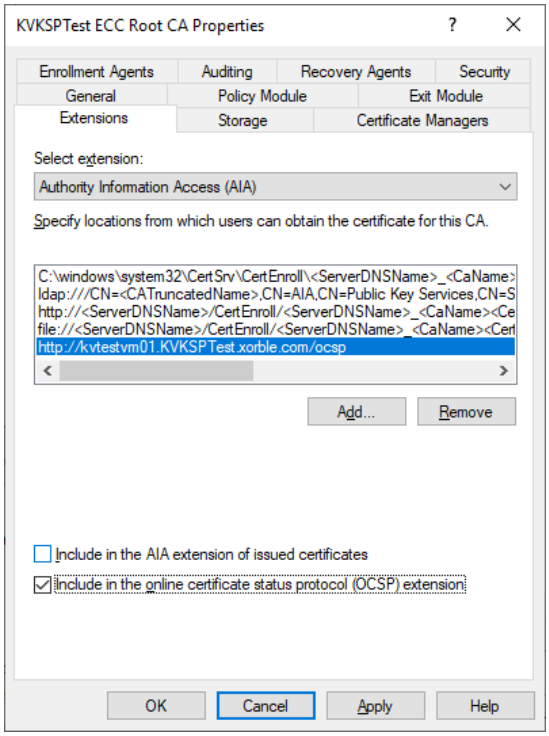


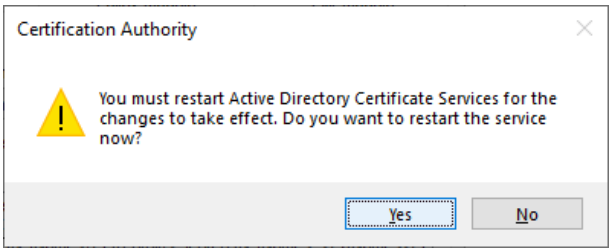
Select Finish to complete the OCSP configuration.



Configure CA to include OCSP URL in issued certificates

Start the Certification Authority management console and select CA properties. Select the ***Extensions*** tab and then select ***Add***. Enter the URL of the OCSP responder ending in /ocsp and also select the ***Include in the online certificate status protocol (OCSP) extension*** and ***OK***. The CA will need to be restarted for this to complete.



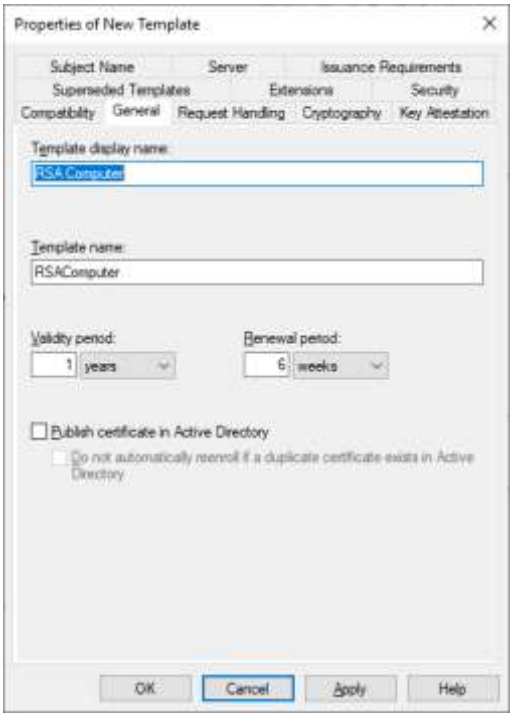


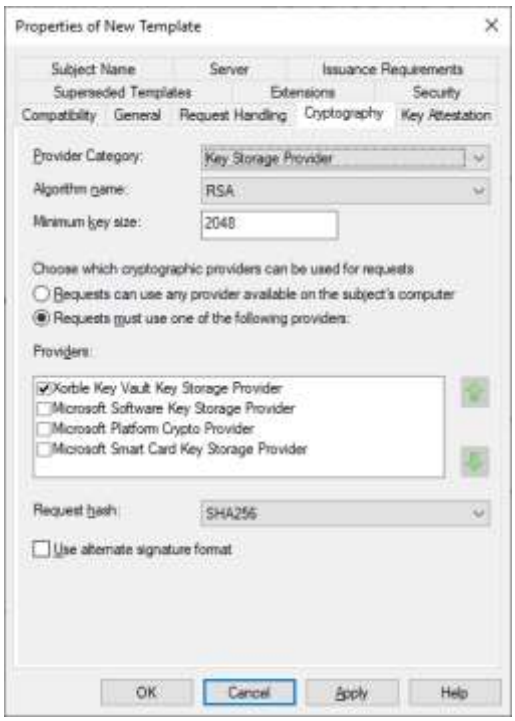
The CA should now issue certificates include an OCSP URL that will use a KSP signing certificate.

The OCSP responder can be checked by running the Certutil -fetchurl -verify <certfile> command which should perform n OCSP operation.

Configure AD CS Templates to use KSP

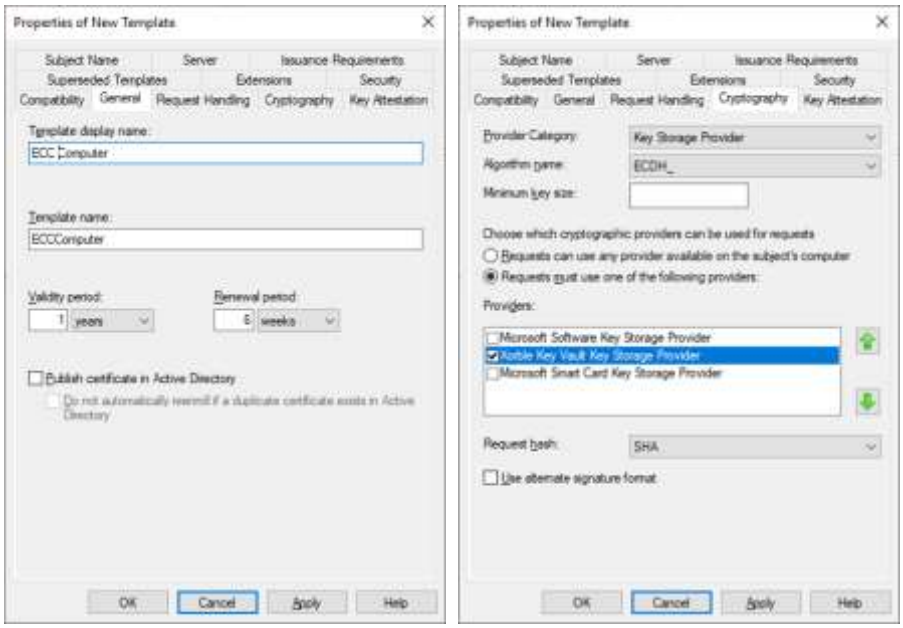
Xorble RSA Computer Certificate Template





Xorble ECC Computer Certificate Template

Similarly, ECC Computer certificates can be issued with 256 and 384-bit ECC keys:



Xorble Key Vault KSP Performance

The KSP performance has been tested only from an Azure VM with the Key Vault located in the same region as the VM (Azure D1_V2). The testing is therefore a best-case scenario and the performance numbers when accessing a remote Key Vault over the Internet are likely to be worse. The test case was run against a test standard Key Vault (software based) rather than Premium which uses HSMs.

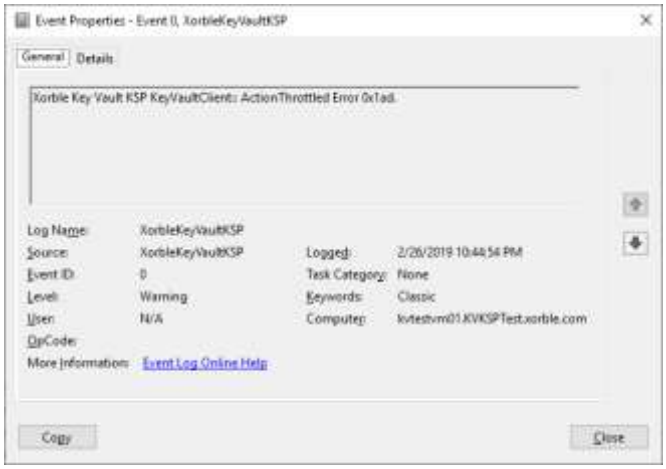
The performance observed was broadly in line with "Azure Key Vault service limits[4]" as the following is reproduced from that - Max transactions allowed in 10 seconds, per vault per region:

Key type	HSM-key CREATE Key	HSM-key All other transactions	Software-key CREATE Key	Software-key All other transactions
RSA 2048-bit	5	1000	10	2000
RSA 3072-bit	5	250	10	500
RSA 4096-bit	5	125	10	250
ECC P-256	5	1000	10	2000
ECC P-384	5	1000	10	2000
ECC P-521	5	1000	10	2000
ECC SECP256K1	5	1000	10	2000

There is also a subscription-wide limit for all transaction types, that is 5x per key vault limit. For example, HSM- other transactions per subscription are limited to 5000 transactions in 10 seconds per subscription and therefore for high throughput scenarios deployment of Key Vaults into separate subscriptions may be required.

When running on a single thread, the KSP is able to sustain approximately 50 2048bit RSA signing operations/sec while using around 20% CPU (on an Azure D1_V2). When the same test is run with 10 threads the KSP is able to sustain a total of approximately 200 signing RSA operations/sec while consuming nearly 100% of the server CPU (D1_V2). During this test, the Key Vault started throttling the KSP (returning a try again return code) which forces the KSP to pause and then retry. This is part of the normal Key Vault DOS protections and therefore the maximum expected operations per Key Vault is expected to be around 200/sec for software keys regardless of the number of virtual CPU cores available and for HSM based keys are half this and hence an expectation of around 100/sec for HSM based keys - the performance of the KSP will ultimately be limited by Key Vault.

When running this test, several warning events were generated by the KSP which shows the throttling by Key Vault as shown to the right:



One thing to note that because of the potential of throttling, multiple servers should generally not share a single vault as this is likely to result in the servers simply waiting for Key Vault operations to be retried.

When using the KSP for protecting SSL/TLS connections a single signing operation is performed for each handshake and hence potentially a single server could handle up to 100 connections/sec.

[4] Azure Key Vault service limits - https://docs.microsoft.com/en-us/azure/key-vault/key-vault-service-limits

Xorble Key Vault KSP Automation within Azure

When using the KSP from an Azure VM, most of the configuration can be automated so that a VM can simply be deployed and it will be correctly configured to use an Azure Key Vault without any manual input.

There are many different ways to automate these processes. One suggested approach is to do the following:

- Create an event hub to listen for virtual machine creation events
- When a virtual machine is created:
    - Update the VM to use an MSI based identity (if not already)
    - Update the VM to automatically install the KSP (see below for details)
- Create a Key Vault based on the name of the virtual machine
    - Make sure that soft delete is enabled on the Key Vault
- Permission the Key Vault with the virtual machine MSI name

Xorble Key Vault KSP Automated Installation within Azure

The KSP can be automatically installed within a VM by running a script such as the following (AzureVMXorbleKSPInstall.ps1):

```
#
# Xorble KSP Installation Script and Certificate Propagation to Machine My Store from
Key Vault Certificates
#
#   Add the Signing Certificate CA into Root Store (required for beta KSP)
#   Install the KSP together with Debug/Logging folder.
#   Add NuGet, Az and Azure Key Vault PowerShell Packages/Modules
#   Set the Key Vault name to DNS name in lower case with "." characters removed and
truncated to 24 characters.
#   Logon to Azure with MSI account
#   Read all certificates from Key Vault and add to local Machine Store
#   Finally run a CertUtil -RepairStore to reassociate the Certificate with Key Vault
KSP based private key
#
#
# Install Signing Certificate CA into Root Store
$rootstore = Get-Item cert:\LocalMachine\Root
$rootstore.Open( [System.Security.Cryptography.X509Certificates.OpenFlags]::ReadWrite
)
$wc = New-Object system.Net.WebClient;
$wc.DownloadFile("https://0xorbleukwstg01kvksp0.blob.core.windows.net/xorblekvksp/Test
SigningRoot.cer","c:\SigningCA.cer")
$SigningCertCA=New-Object
System.Security.Cryptography.X509Certificates.X509Certificate2("c:\SigningCA.cer")
$rootstore.Add($SigningCertCA )
$rootstore.Close()
del "c:\SigningCA.cer"

# Install the new KSP
mkdir "C:\XorbleKeyVaultKSP\"
msiexec -q -i
"https://0xorbleukwstg01kvksp0.blob.core.windows.net/xorblekvksp/XorbleKeyVaultKSPSetu
p.msi" /lv* c:\KSPInstall.log

# Add NuGet to the machine so that we can then install the PowerShell modules for
Azure and Key Vault.
$pac = Get-PackageProvider -Name NuGet -Force
if ( $pac -eq $null )
{
    Install-PackageProvider -Name NuGet -Force
}

# Install the Azure PowerShell bits
$mod = Get-InstalledModule -name Az
```

```
if ( $mod -eq $null )
{
    Install-Module -Name Az -AllowClobber -Force
}

# Install the Key Vault PowerShell bits
$mod = Get-InstalledModule -name AzureRM.KeyVault
if ( $mod -eq $null )
{
    Install-Module -Name AzureRM.KeyVault -AllowClobber -Force
}


# Get the Key Vault name based on lower case hostname with . characters removed.
$kvname = [System.Net.Dns]::GetHostName().Replace(".", "" ).ToLower()
$kvname = $kvname.Substring(0,[System.Math]::Min($kvname.Length, 24))


# Logon to Azure with MSI
Connect-AzureRmAccount -MSI


# Get all the certificates from the Key Vault
$certs=Get-AzureKeyVaultCertificate -VaultName $kvname

# Install all the certificates into the local machine store
$store = Get-Item cert:\LocalMachine\My
$store.Open( [System.Security.Cryptography.X509Certificates.OpenFlags]::ReadWrite )
foreach ( $cert in $certs)
{
    $thecert=Get-AzureKeyVaultCertificate -VaultName $kvname -Name $cert.Name
    $x509cert = $thecert.Certificate
    $store.Add( $x509cert )
}
$Store.Close()


# Finally, go through the machine store and repair so that the private key points at
the Key Vault KSP. The -f option is to force the repair when the certificate cannot be
used to decrypt data.
for ($i = 0; $i -le 30; $i++ )
{
    Certutil -f -csp "Xorble Key Vault Key Storage Provider" -RepairStore My $i
}
```

Xorble Key Vault KSP Key Naming

Key stored within Azure Key Vault can only be named using alphanumeric characters and dashes. A KSP needs to support Unicode key names which means that a translation mechanism is need to convert between Unicode key names and the characters supported by Key Vault.

For the Xorble KSP, it was decided to represent the key name in a base 50 form with each Unicode character represented by three characters within the Azure key name. The characters ‘A’-‘Z’ are used to represent values 0-25 and characters ‘a’-‘x’ represent values 26-50. Thus, the string “A” is represented by the key name “PBA” (‘P’=15,’B’=1,’A’=0 which together make 65 ‘A’ (15+50*1+50*50*0).

A key name in Azure Key Vault can have up to 127 characters and hence the KSP has a key name limit of 42 (1/3 of 127).

When creating keys, the KSP adds a tag to the key of the Unicode Key Name (KeyName tag) to provide a convenient way to display the actual key names.

Alternatively, the key names can be converted by to Unicode using PowerShell or similar. The following PowerShell script provides functions to both encode and decode the base50 key names.

```
# Base 50 Encode and Decode functions for Xorble Key Vault KSP
Function Base50-Decode
{
    [cmdletbinding()]
    Param(
    [Parameter(ValueFromPipelineByPropertyName)]
    $Name
)
process {

    $out = ""

    for ( $i=0; $i -lt $Name.Length; $i += 3 )
    {
        $val = 0
        for ( $j=$i+2; $j -ge $i; $j -- )
        {
            if ( $j -lt $Name.Length )
            {
                $ch = $Name[$j]
            }
            else
            {
                $ch = 'A'
            }

            if ( $ch -le 'Z' )
            {
                $c = [int]$ch -[int][char]'A';
            }
            else
            {
                $c = [int]$ch - [int][char]'a' + 26;
            }

            $val = $val * 50 + $c
        }
        if ($val -gt 65535 -or $val -lt 0 )
        {
            return
        }
        $out = $out + [char] $val
    }
    $out
    }
}

Function Base50-Encode
{
[cmdletbinding()]
Param(
    [Parameter(Position=0,Mandatory,HelpMessage="Enter a string of text")]
    [ValidateNotNullorEmpty()]
    [string]$Text,
    [int]$Start=0,
    [int]$End=0
)
    $out = ""

    for ( $i=0; $i -lt $Text.Length; $i++)
    {
        $val = [int] $Text[$i]
```

```
        for( $j=0; $j -lt 3; $j++ )
        {
            $c = $val % 50
            $val = [int] (($val-$c) / 50)

            if ( $c -lt 26 )
            {
                $out = $out + [char] ( $c + [int][char] 'A' )
            }
            else
            {
                $out = $out + [char] ( $c - 26 + [int][char] 'a' )
            }
        }
    }
    $out
}
```

The Decode function can accept input from the pipeline and decode the names, so for instance all keys in a vault can be listed using the following command:

```
Get-AzureKeyVaultKey  -VaultName <vaultname> |  Base50-Decode
```

Running the following fragment of PowerShell will generate the encoded and decoded values for the following strings (“Cryptographic Key Name”, “密码学密钥名称” and “क्रिप्टोग्राफी कुंजी नाम”).

```
$str = "Cryptographic Key Name"
echo "Encode and then Decode $str"
$txt = Base50-Encode $str
echo $txt
$txt = Base50-Decode $txt
echo "$txt = $str"

$str = "密码学密钥名称"
echo "Encode and then Decode $str"
$txt = Base50-Encode $str
echo $txt
$txt = Base50-Decode $txt
echo "$txt = $str"

$str = "क्रिप्टोग्राफी कुंजी नाम"
echo "Encode and then Decode $str"
$txt = Base50-Encode $str
echo $txt
$txt = Base50-Decode $txt
echo "$txt = $str"
```

The output from the above is:

```
PS C:\Users\XXXX > C:\Users\David\Desktop\Base50-Example.ps1
Encode and then Decode Cryptographic Key Name
RBAOCAVCAMCAQCALCADCAOCAvBAMCAECAFCAxBAgAAZBABCAVCAgAAcBAvBAJCABCA
Cryptographic Key Name = Cryptographic Key Name

Encode and then Decode 密码学密钥名称
sTJVOMwRJsTJDLPReIQYM
密码学密钥名称 = 密码学密钥名称

Encode and then Decode क्रिप्टोग्राफी कुंजी नाम
ZuAfvACvARvAuuAfvAjuAdvAbuAfvACvAQvAvuASvAgAAZuATvAGuAguASvAgAAsuAQvAAvA
क्रिप्टोग्राफी कुंजी नाम = क्रिप्टोग्राफी कुंजी नाम
```