NuGet Signing Integration Document

CodeSign Secure can sign NuGet packages (.nupkg) using the NuGet command line interface or Encryption Consulting's in-house utility tool, backed by Encryption Consulting's Key Storage Provider (KSP). The signing key remains inside the HSM at all times, and every operation is logged centrally, giving you strong key custody and a complete audit trail without changing your project files.

NuGet does not talk to the HSM directly. It selects a signing certificate from the Windows certificate store by fingerprint, and because that certificate is associated with the Encryption Consulting KSP, the private key operation is transparently routed to the HSM. Three signing methods are documented below — using an imported publicly trusted certificate, using a self-signed certificate issued from the CodeSign Secure portal, or using our utility tool. Sections 1 to 4 apply to all three; choose whichever of Sections 5 to 7 matches your certificate strategy.

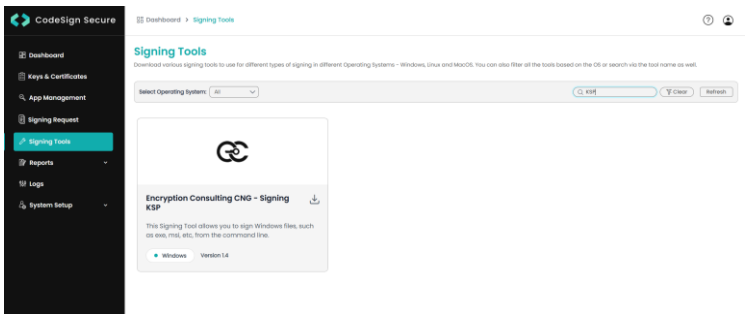
1. Set up CodeSign Secure KSP

The Encryption Consulting Key Storage Provider (KSP) for Windows is a software component that extends the Microsoft Cryptography API: Next Generation (CNG) framework. Its primary purpose is to enable Windows applications, including the NuGet command line interface, to interact seamlessly with the cryptographic keys and certificates stored within an HSM.

Steps:

Step 1: Download the EC KSP

- Log in to the CodeSign Secure portal and navigate to the Signing Tools section to download "Encryption Consulting CNG-SigningKSP" (also listed as "EC KSP for Windows").

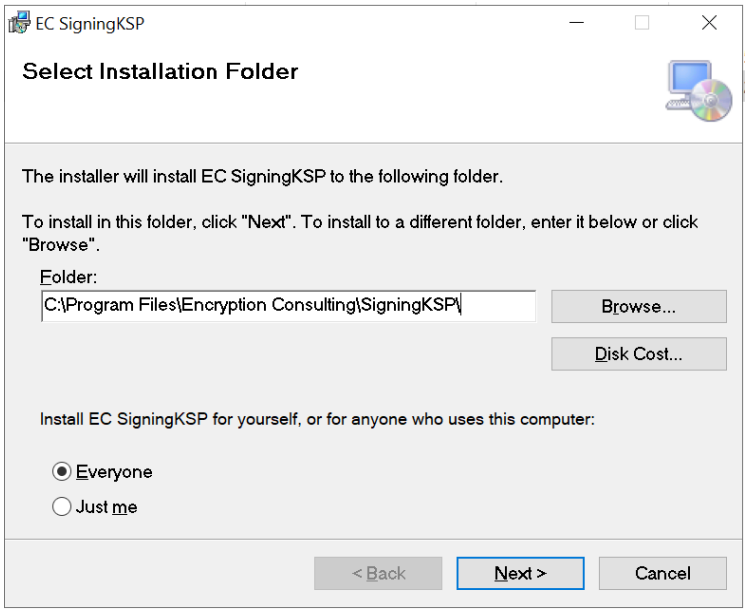


- Extract the zip file to get the "Setup.msi" file.

Step 2: Install the EC KSP

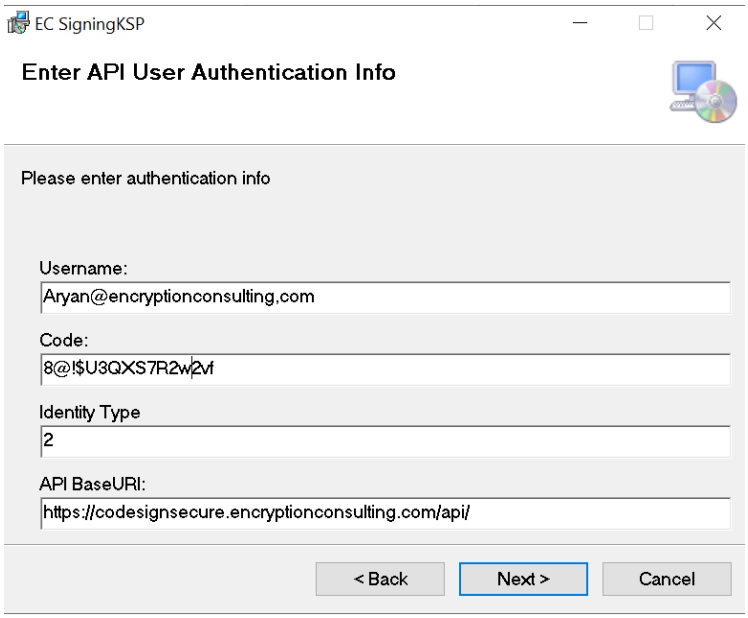
- Run the "Setup.msi" installer with Administrator privileges.

- Follow the on-screen prompts of the installation wizard.
  - Accept the End-User License Agreement.
  - Choose the installation directory (the default is C:\Program Files\Encryption Consulting\SigningKSP).
  - Choose whether you want to install the KSP for Everyone or just for the current user.

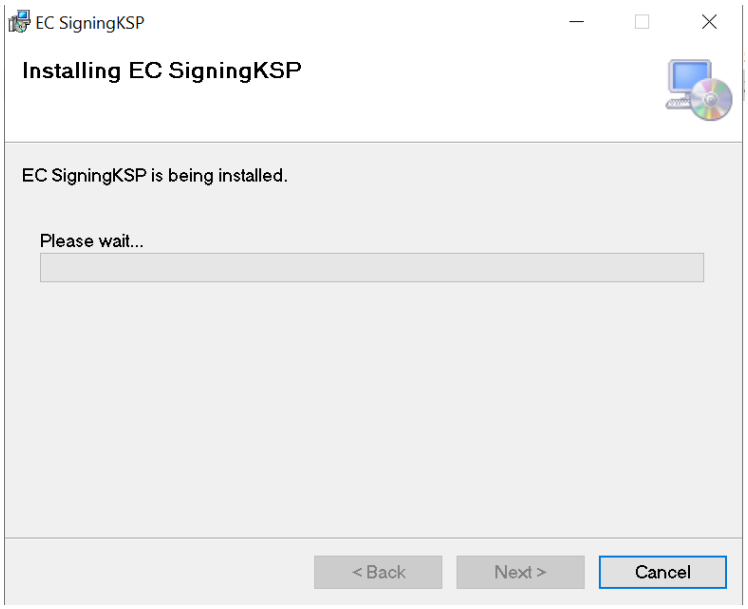


- Enter the prompted details such as:
  - **Username –** The username/email that you use to log in to the CodeSign Secure portal.
  - **Code –** The secret code that you set at the time of setting up the CodeSign Secure solution (this is the code defined in your conf.ini configuration file).

- **IdentityType –** Keep this field as default (2). If your deployment authenticates against a local identity store, set this to 1 as described in the product documentation.
- **CodeSign Secure URL –** The URL to access the portal (remember to add "/api/" at the end of the URL). Leave the API BaseURL unchanged if it is already populated correctly.

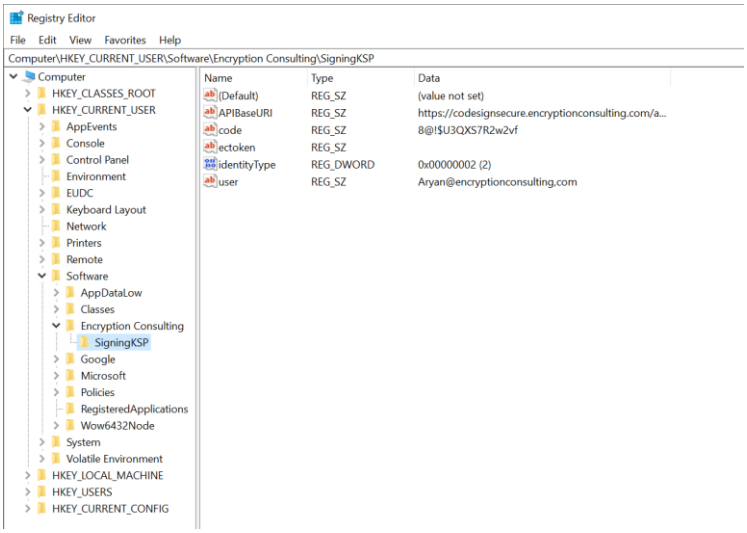


- Click Next and confirm the installation. When Windows asks whether you want to allow this program to make changes to your PC, click Yes.

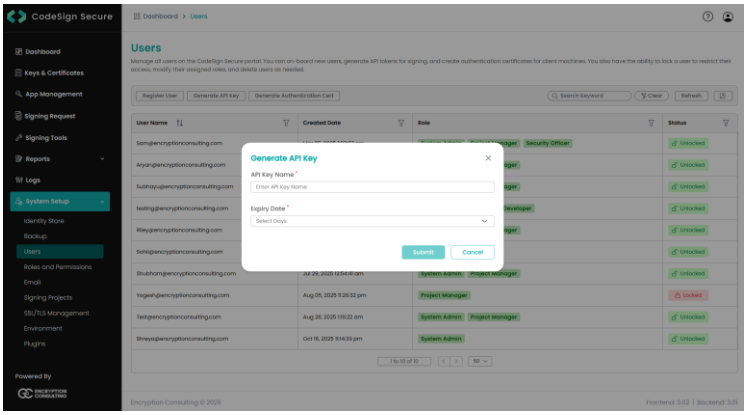


Step 3: Configure the Registry Editor settings

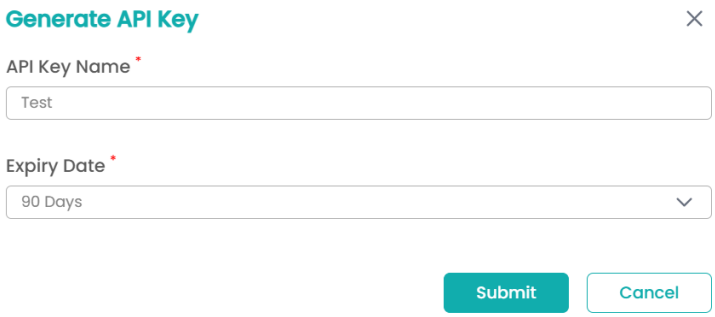
- Open the Registry Editor from the Start menu and navigate to HKEY_CURRENT_USER > Software > Encryption Consulting > SigningKSP.



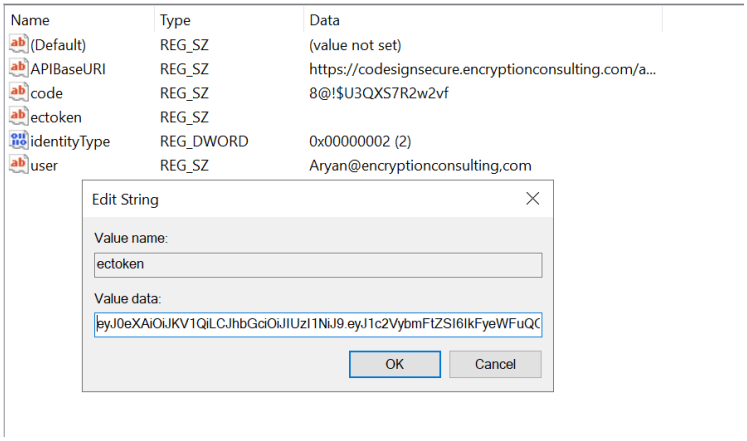
- Now open the CodeSign Secure portal and navigate to System Setup > User. Select the “Generate API Key” option.



- Create a token for your account by providing a name and the validity period. Remember to copy the token as it will be shown only once.



- Add this token to the “ectoken” field in the Registry Editor.



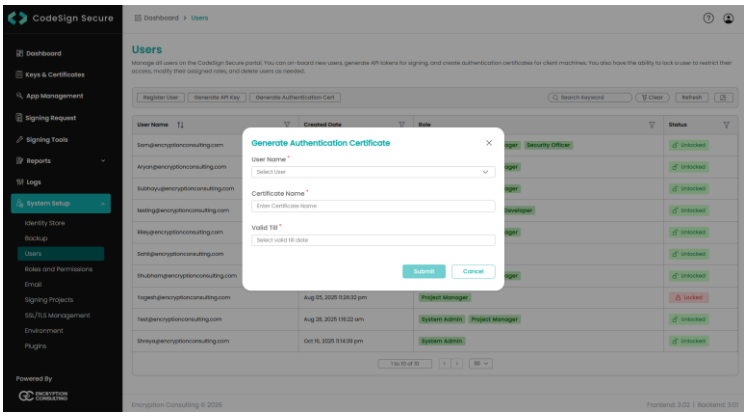
2. Set up P12 Authentication Certificate

Setting up a P12 Certificate involves configuring your environment variables to authenticate your client machine with Encryption Consulting's CodeSign Secure.

Steps:

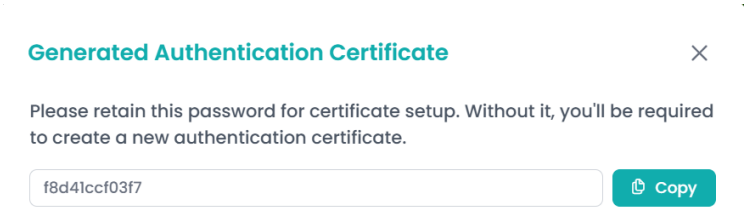
Step 1: Create a Machine Authentication Certificate

- Open the CodeSign Secure portal and navigate to System Setup > User. Select the "Generate Authentication Cert" option.



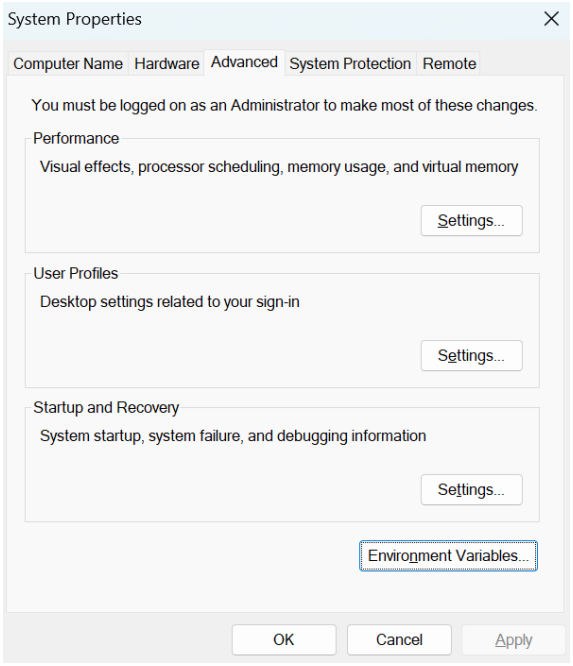
- Select the user name from the drop down and enter the details like certificate name and its expiry date.
- It will then provide you with a .pfx certificate file and also display the password to the certificate file.

NOTE: This password will be displayed only once. So you must copy and store it safely to perform the authentication with the CodeSign Secure server.

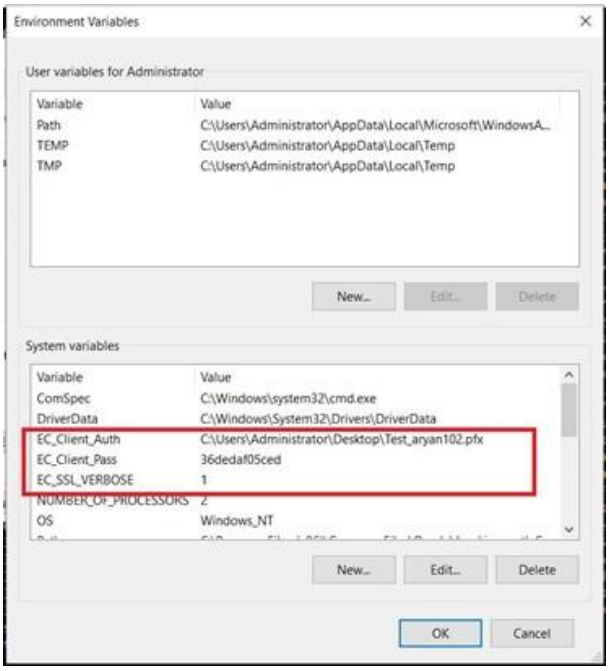


Step 2: Configure the Environment Variables

- Open the Environment Variables from your Start Menu.



- Add new system variables by clicking on the New button. Provide the following variable name and its corresponding details.
  - **EC_Client_Auth:** Corresponds to the path of your SSL Authentication certificate, which can be created from CodeSign Secure.
  - **EC_Client_Pass:** Corresponds to the password of your certificate, which is provided at the time of creation of the certificate.
  - **EC_SSL_VERBOSE:** Corresponds to the setting to either enable (1) or disable (0) the debugging output for EC KSP.



3. Install the Signing Prerequisites

The following components must be present on the client machine before signing:

- NuGet installed in your system.
- Windows operating system.
- Encryption Consulting's utility tool installed in the system.
- OpenSSL configured in the system.

Steps:

Step 1: Download the NuGet CLI

- Download nuget.exe from https://www.nuget.org/downloads.
- Place nuget.exe somewhere permanent, such as a dedicated tools directory, and make a note of the folder path.

Step 2: Download and Install OpenSSL

- Download and install OpenSSL on your system from here.

Step 3: Install the Encryption Consulting Utility Tool

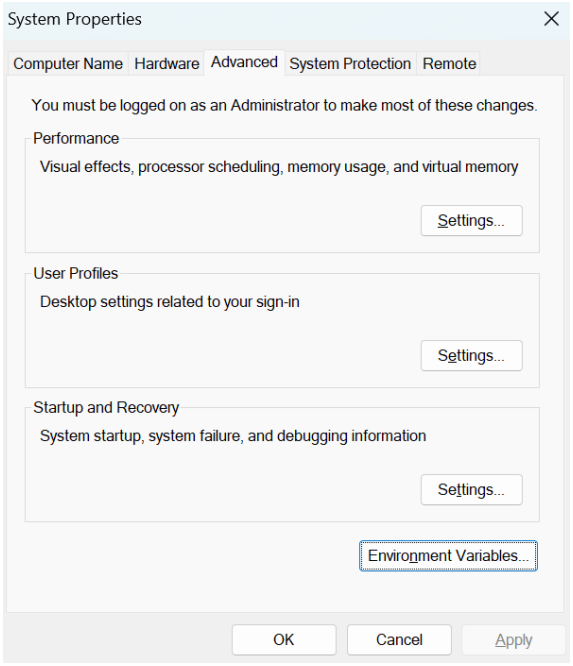
- For installation, you would need to download and run an MSI file from the portal, which will install this tool into your machine.
- Along with the tool, it will also prompt you to install various signing software such as Windows SDK, Java SDK, and Encryption Consulting KSP. These signing softwares will be used by the Utility Tool to perform signing operations on different file types.

NOTE: If you have already installed the Encryption Consulting KSP in Section 1, you can skip that prompt during this installation.

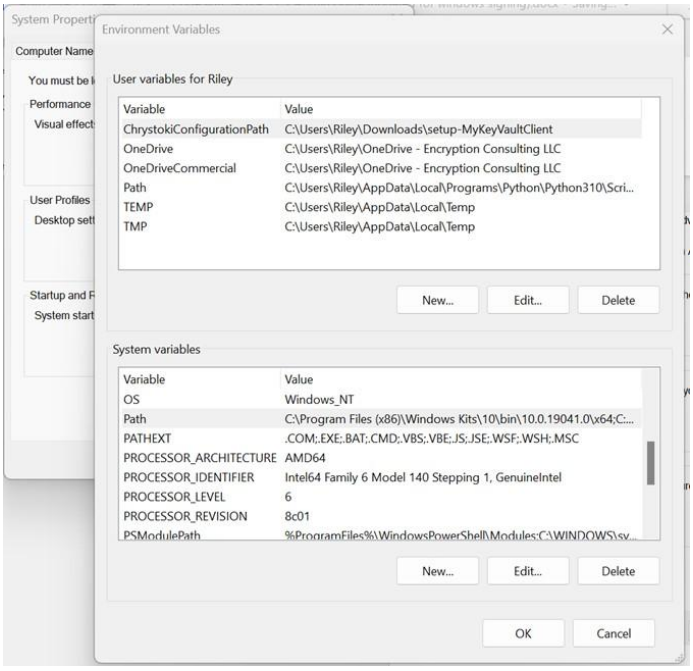
- After successful installation, you will find the Utility Tool.exe in the "C:\Program Files\Encryption Consulting\Utility Tool" directory.
- You will need to change the "BASEURL" field in the config file with the URL of your domain.

Step 4: Add the Executables to the PATH Environment Variable

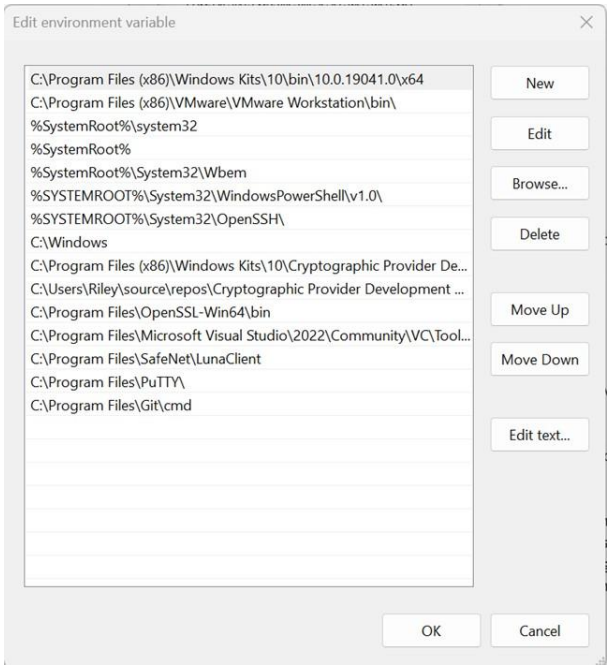
- Set these executable files to the PATH environment variable so that they can be invoked from any directory.
- Open the Environment Variables from the Start Menu.



- Scroll down through the system variables on the bottom table until you find PATH in the variable names.



- Double-click on PATH in System Variables and select New on the left of the screen. Paste the directory path containing nuget.exe into the new selection. Repeat for the OpenSSL and utility tool directories if they are not already present.



- Select OK at the bottom of each page to exit the Environment Variables page.

NOTE: Open a new command prompt after editing the PATH, as existing windows will not pick up the change.

4. Obtain a Signing Certificate

NuGet signing requires a code signing certificate whose private key is held in the HSM. There are two routes, and which one you choose determines whether you follow Section 5 or Section 6.

Steps:

Option A: Publicly Trusted or MS PKI Certificate (preferably EV)

- Generate a CSR from the CodeSign Secure web portal. The key pair is created inside the HSM, and only the request leaves it.
- Get the CSR signed using a publicly trusted CA (or MS PKI).
- Import that certificate back into the CodeSign Secure web portal.
- Continue with Section 5, which signs using this imported certificate.

Option B: Self-Signed Certificate from the Portal

- Generate a self-signed certificate from the CodeSign Secure web portal.
- Download the certificate and convert it from .pem to .crt format for better operations. This is what OpenSSL is required for.

```
openssl x509 -outform der -in certificate.pem -out certificate.crt
```

- Continue with Section 6, which installs the full certificate chain before signing.

NOTE: A self-signed certificate is only trusted on machines where you install its chain, so Option B suits internal distribution. Public package feeds require Option A.

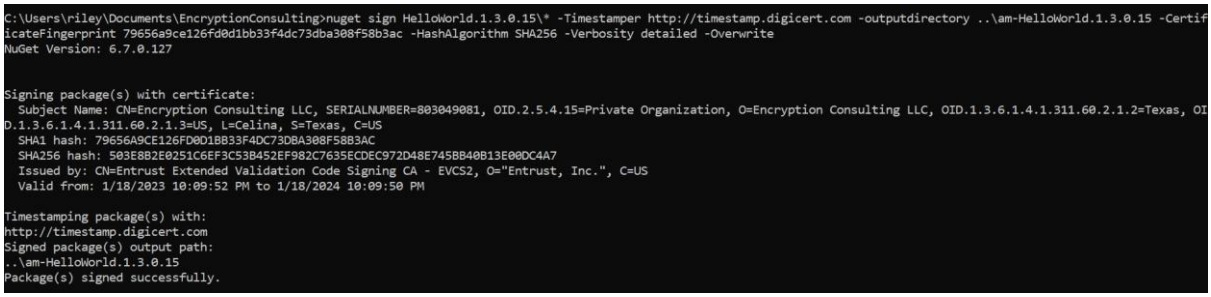
5. Sign Using an Imported Certificate (Command Line)

This method involves leveraging the NuGet command line interface, which provides functionality to create, publish, sign, and manage packages without changing the project files. It assumes you have completed Option A in Section 4, so the signed certificate is already present in your certificate store.

Steps:

Step 1: Run the NuGet Sign Command

- Now, use nuget.exe to sign your package.



- The command is:

```
nuget sign HelloWorld.1.3.0.15*
  -Timestamper http://timestamp.digicert.com
  -outputdirectory ..\am-HelloWorld.1.3.0.15
  -CertificateFingerprint 79656a9ce126fd0d1bb33f4dc73dba308f58b3ac
  -HashAlgorithm SHA256
  -Verbosity detailed
  -Overwrite
```

NOTE: Replace the fingerprint with the thumbprint of your own certificate, taken from the Certificate Manager. Remove any spaces from the value, as the Certificate Manager displays it in space-separated pairs.

Step 2: Parameter Reference

- The following are the parameters and their meaning in the command:

Flag	Description
sign	This command tells NuGet to sign the selected/specified package.
HelloWorld.1.3.0.15*	This specifies the package file(s) to be signed. The asterisk (*) is a wildcard character that indicates all files starting with HelloWorld.1.3.0.15 should be signed.
-Timestamper	Specifies the URL of the timestamping service to be used, for example http://timestamp.digicert.com.
-outputdirectory	This parameter directs NuGet to place the signed package(s) into the specified directory relative to the current directory.
-CertificateFingerprint	Specifies the SHA-1 fingerprint of the signing certificate. This uniquely identifies which certificate should be used from the store if multiple certificates are available.
-HashAlgorithm	This option defines the hash algorithm to use for creating the signature. SHA256 is a recommended hash algorithm for security.
-Verbosity	Tells NuGet how much information to output to the console during the signing process. “detailed” means it will output detailed information useful for debugging or logging.
-Overwrite	This flag indicates that if the package is already signed, the existing signature should be overwritten with the new one. Without this flag, if the package is already signed, NuGet will not sign it again and will throw an error instead.

- Once the command completes, verify the signature as described in Section 8.

6. Sign Using a Self-Signed Certificate from the Portal

This method also involves leveraging the NuGet command line interface. It assumes you have completed Option B in Section 4. Because the certificate is self-signed, its full chain must be installed into the correct Windows certificate stores before signing will succeed.

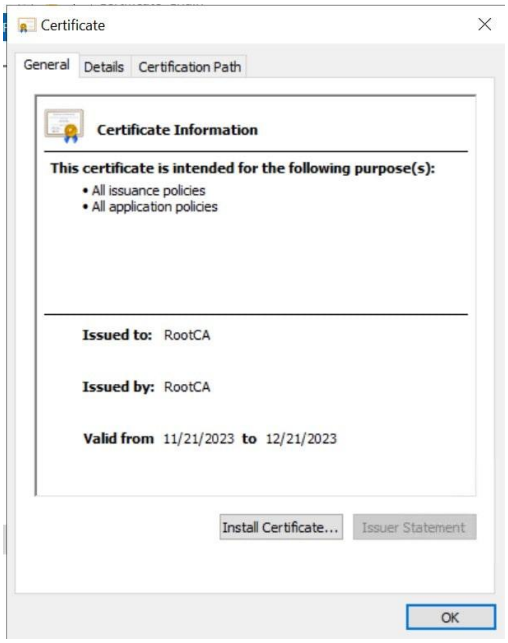
Steps:

Step 1: Download the Certificate Chain

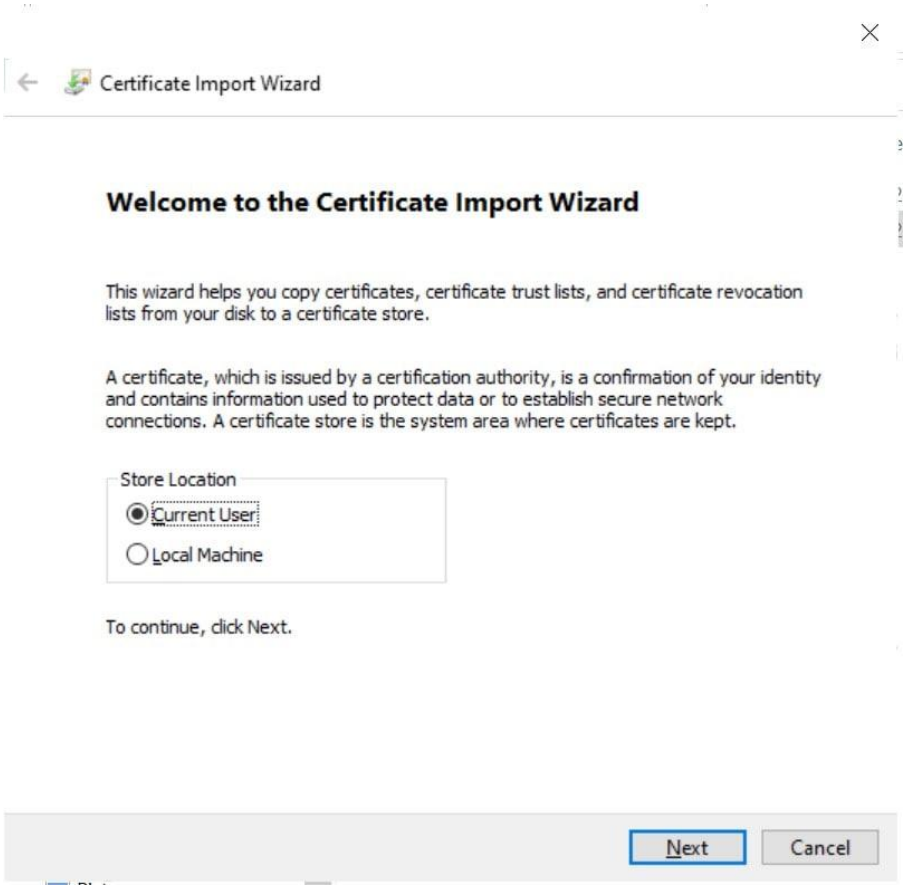
- Download the Certificate Chain from the “Signing Tools” screen of the CodeSign Secure portal, and unzip it on your client machine.

Step 2: Install the Root CA Certificate

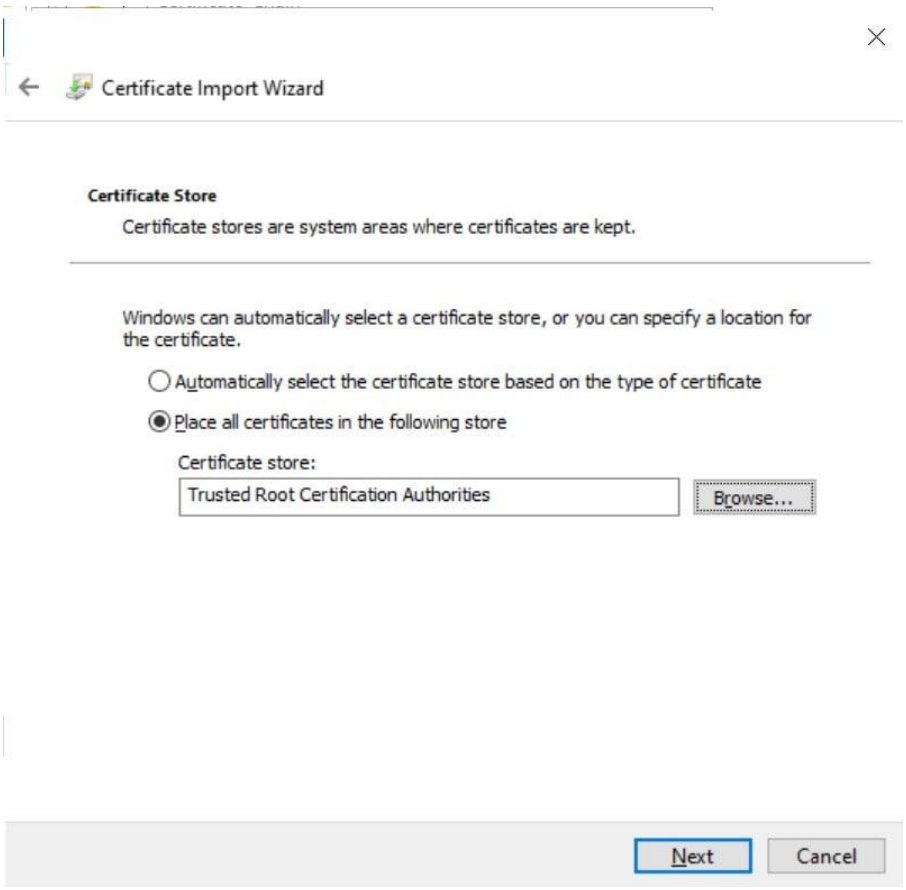
- Click on the INSTALL CERTIFICATE button.



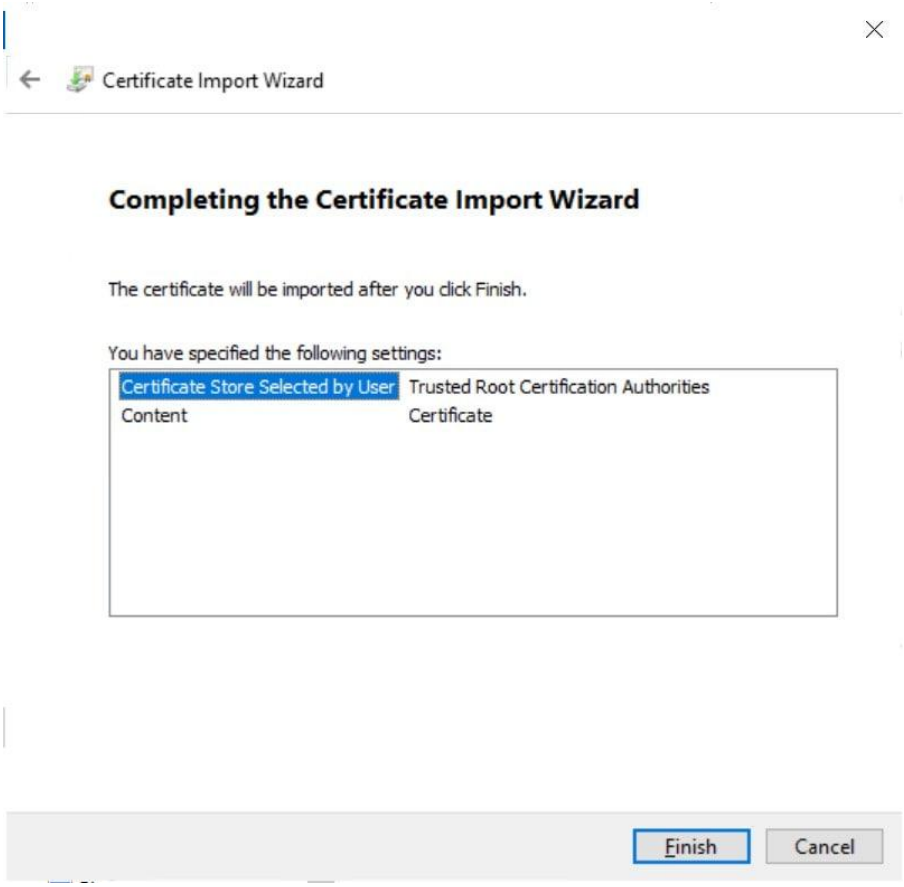
- On the “Import Wizard” popup, select Current User and then hit NEXT.



- On the “Certificate Store” popup, choose Trusted Root Certification Authorities for the Root CA certificate and click on NEXT.



- After verifying all the details, click on FINISH.



Step 3: Install the Issuing CA Certificate

- After installing the Root CA, install the Issuing CA from the Certificate Chain folder (intermediate.crt).
- Follow the same procedure as Step 2, except install this certificate in Intermediate Certification Authorities.

Step 4: Install the End Entity Certificate

- After successfully installing the Issuing CA, install the End Entity certificate (or the self-signed certificate from our portal) into your system.
- The process is the same, except this certificate will be installed in the Personal store.

Step 5: Run the Repairstore Command

- Run the repairstore command using the thumbprint of this End Entity certificate. Like this:

```
certutil -f -repairstore
  -csp "Encryption Consulting Key Storage Provider"
  -user "My" <thumbprint of the certificate>
```

```
C:\Users\riley\Documents\EncryptionConsulting>certmgr

C:\Users\riley\Documents\EncryptionConsulting>certutil -f -repairstore -csp "Encryption Consulting Key Storage Provider" -user "My" 9d2b76971a8a209cec353748e257a8f92b7ff7d5

My "Personal"
================ Certificate 4 ================
Serial Number: 1037
Issuer: E=subhayu@encryptionconsulting.com, CN=IssuingCA, OU=Dev, O=Encryption Consulting LLC, L=Dallas, S=Texas, C=US
 NotBefore: 11/21/2023 8:56 AM
 NotAfter: 11/15/2024 8:56 AM
Subject: CN=nug , OU=Development , O=Encryption Consulting LLC, L=Dalas , S=Texas, C=US
Non-root Certificate
Cert Hash(sha1): 9d2b76971a8a209cec353748e257a8f92b7ff7d5
  Key Container = nugetTesting
  Provider = Encryption Consulting Key Storage Provider
Private key is NOT exportable
Encryption Consulting Key Storage Provider: KeySpec=0
AES256+RSAES_OAEP(RSA:CNG) test skipped
Signature test passed
CertUtil: -repairstore command completed successfully.
```

NOTE: This command is what associates the certificate with the Encryption Consulting KSP, so that NuGet's private key operation is routed to the HSM. Remove any spaces from the thumbprint before running it.

Step 6: Run the NuGet Sign Command

- When all of these steps have been completed successfully, try running the NuGet command line command using this certificate's thumbprint:

```
nuget sign HelloWorld.1.3.0.15*
  -Timestamper http://timestamp.digicert.com
  -outputdirectory ..\am-HelloWorld.1.3.0.15
  -CertificateStoreLocation CurrentUser
  -CertificateStoreName My
  -CertificateFingerprint <thumbprint of the certificate>
  -HashAlgorithm SHA256
  -Verbosity detailed
  -Overwrite
```

```
C:\Users\riley\Documents\EncryptionConsulting>nuget sign HelloWorld.1.3.0.15\* -Timestamper http://timestamp.digicert.com -outputdirectory ..\am-HelloWorld.1.3.0.15 -CertificateStoreLocation CurrentUser -Certif
icateStoreName My -CertificateFingerprint 9d2b76971a8a209cec353748e257a8f92b7ff7d5 -HashAlgorithm SHA256 -Verbosity detailed -Overwrite
NuGet Version: 6.7.0.127


Signing package(s) with certificate:
  Subject Name: CN="nug ", OU="Development ", O=Encryption Consulting LLC, L="Dalas ", S=Texas, C=US
  SHA1 hash: 9D2B76971A8A209CEC353748E257A8F92B7FF7D5
  SHA256 hash: FCD2279096A81DC59500A43A74598CF2CA70202648583C60135F6087374747F
  Issued by: E=subhayu@encryptionconsulting.com, CN=IssuingCA, OU=Dev, O=Encryption Consulting LLC, L=Dallas, S=Texas, C=US
  Valid from: 11/21/2023 8:56:58 AM to 11/15/2024 8:56:58 AM

Timestamping package(s) with:
http://timestamp.digicert.com
Signed package(s) output path:
..\am-HelloWorld.1.3.0.15
WARNING: NU3018: RevocationStatusUnknown: The revocation function was unable to check revocation for the certificate.
Package(s) signed successfully.
```

- The two additional parameters used by this method are:

Flag	Description
-CertificateStoreLocation	The certificate store location to search. CurrentUser matches the store location chosen during the import wizard in Step 2.
-CertificateStoreName	The name of the certificate store to search. My corresponds to the Personal store, where the End Entity certificate was installed in Step 4.

7. Sign Using the Encryption Consulting Utility Tool

This method involves leveraging our in-house utility tool designed to perform all kinds of Windows signing, including NuGet signing. This tool streamlines the signing process, making it efficient and hassle-free.

Steps:

Step 1: Launch the Utility Tool

- Open the Encryption Consulting utility tool that was installed as part of the prerequisites in Section 3.

Step 2: Enter the Signing Details

- Enter the appropriate details when prompted to get a signed package. This method allows customization and increases efficiency without compromising quality.

8. Verify the Signature

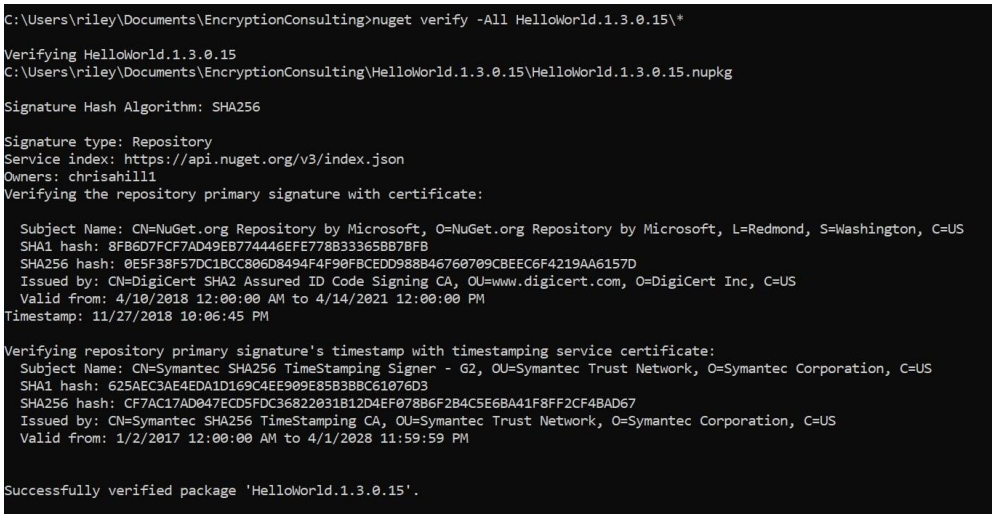
Whichever method you used, confirm that the package now carries a valid signature.

Steps:

Step 1: Run the NuGet Verify Command

- To verify whether the package has been signed or not, the command is:

```
nuget verify -All HelloWorld.1.3.0.15*
```



- A successfully signed package reports that the signature is valid, along with the signing certificate and timestamp details.

Step 2: Cross-Check in CodeSign Secure

- Open the CodeSign Secure portal and navigate to Reports > Signing Request Report, where every signing request performed through the KSP is recorded for audit purposes. Confirm that a signing request appears for the certificate you used.