Member of
Microsoft Intelligent
Security Association

Microsoft Security

Microsoft Authenticode

nShield® HSM Integration Guide

2023-12-12

Table of Contents

Chapter 1. Introduction

The Entrust nShield Hardware Security Module (HSM) integrates with Microsoft Authenticode to enable you to identify the publisher of a software component before it is downloaded from the Internet, and to verify that no one has altered the code after it has been signed. Authenticode relies on proven cryptographic techniques and the use of one or more private keys to sign and time-stamp the published software. It is important to maintain the confidentiality of these keys.

The benefits of using an HSM with Microsoft Authenticode include:

- Protection for the organizational credentials of the software publisher.
- Secure storage of the private key.
- FIPS 140 Level 3 validated hardware.
- Provision of a trusted time-stamp to Authenticode.

1.1. Product configurations

Entrust has successfully tested nShield HSM integration with Microsoft Authenticode in the following configurations:

Product	Version
Base OS	Windows Server 2019 Datacenter
Microsoft .NET Framework	4.8
Windows SDK	10.1

1.2. Supported nShield features

Entrust has successfully tested nShield HSM integration with the following features:

Feature	Support
Operator Card Set (OCS)	Yes
Softcard Protection	Yes

Feature	Support
Module	Yes
nSaaS	Yes
FIPS 140 level 3	Yes

1.3. Supported nShield hardware and software versions

Entrust has successfully tested with the following nShield hardware and software versions:

Product	Security World Software	Firmware	Image	OCS	Softcard	Module
Connect XC	12.80.4	12.50.11 (FIPS Certified)	12.80.4	✓	✓	✓
Connect +	12.80.4	12.50.8 (FIPS Certified)	12.80.4	✓	✓	✓
Connect XC	12.80.4	12.72.1 (FIPS Certified)	12.80.5	✓	✓	✓
Connect +	12.80.4	12.72.0 (FIPS Certified)	12.80.5	✓	✓	✓
nShield 5c	13.2.2	13.2.2 (FIPS Pending)	13.2.2	✓	✓	✓

1.4. Requirements

Entrust recommends that you familiarize yourself with the Microsoft Authenticode documentation and setup process, and have the nShield documentation available.

Entrust also recommends that the following aspects of HSM administration are taken into account:

- The Administration Card Set (ACS) K-of-N and management of the card set.
- The type of protection for the application keys, that is, module protection or Operator Card Set (OCS) protection.
- The Operator Card Set K-of-N and management of the card set.
- Any requirement for a FIPS 140 Level 3 Security World.
- Key attributes, such as the key size, persistence, and time-out.

Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

1.5. This document

This document explains how to set up and configure Microsoft Authenticode with an HSM. The instructions in this document have been tested and provide an integration process. There may be other untested ways to achieve interoperability.

This document may not cover every step in the process of setting up all the software. Entrust assumes you have read the HSM documentation and that you are familiar with the documentation and setup process for Microsoft Authenticode. For more information about installing Microsoft Authenticode, refer to the Microsoft documentation.

1.6. More information

For more information about OS support, contact your Microsoft sales representative or Entrust nShield Support, https://nshieldsupport.entrust.com.

Chapter 2. Procedures

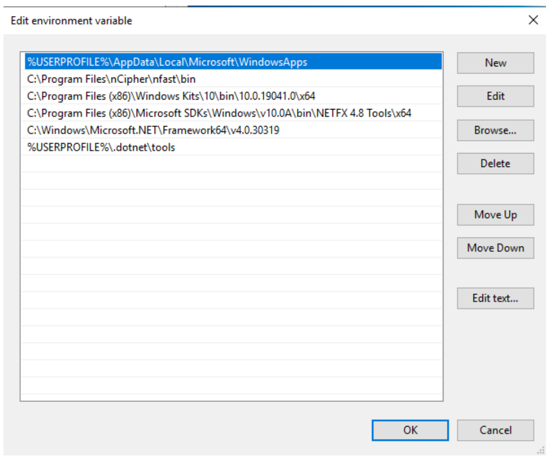
To install and configure Microsoft Authenticode with an HSM:

- Install the Microsoft Windows SDK.
- Install the nShield Security World software and create a Security World.
- Install and register the CNG provider.

The installation procedures outlined here assume, for example purposes, that you are installing an offline root Certificate System. It is also assumed that a new root key is generated during installation, rather than a software key being imported from an existing installation.

2.1. Install the Microsoft tools

1. Install the Microsoft Windows SDK, which contains the Authenticode programs such as `signtool` and `sn` (Strong Name Tool). See the appropriate Microsoft SDK installation documentation.
2. Add the path to the above programs to the Windows Environment Variable, since it might be long.
   - Second line: path to the nShield utilities.
   - Third line: path to the `signtool` utility.
   - Fourth line: path to the `sn` utility.
   - Fifth line: path to the `csc.exe` compiler.



```
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Program Files\nCipher\nfast\bin
C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x64
C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools\x64
C:\Windows\Microsoft.NET\Framework64\v4.0.30319
%USERPROFILE%\.dotnet\tools
```

3. Install the PowerShell PKI module:

```
PS C:\Users\Administrator> Get-Command -Module PKI

CommandType     Name                                               Version    Source
```

```
-----------     ----                                                ------- 	------
Cmdlet          Add-CertificateEnrollmentPolicyServer               1.0.0.0    PKI
Cmdlet          Export-Certificate                                  1.0.0.0    PKI
Cmdlet          Export-PfxCertificate                               1.0.0.0    PKI
Cmdlet          Get-Certificate                                     1.0.0.0    PKI
Cmdlet          Get-CertificateAutoEnrollmentPolicy                 1.0.0.0    PKI
Cmdlet          Get-CertificateEnrollmentPolicyServer               1.0.0.0    PKI
Cmdlet          Get-CertificateNotificationTask                     1.0.0.0    PKI
Cmdlet          Get-PfxData                                         1.0.0.0    PKI
Cmdlet          Import-Certificate                                  1.0.0.0    PKI
Cmdlet          Import-PfxCertificate                               1.0.0.0    PKI
Cmdlet          New-CertificateNotificationTask                     1.0.0.0    PKI
Cmdlet          New-SelfSignedCertificate                           1.0.0.0    PKI
Cmdlet          Remove-CertificateEnrollmentPolicyServer            1.0.0.0    PKI
Cmdlet          Remove-CertificateNotificationTask                  1.0.0.0    PKI
Cmdlet          Set-CertificateAutoEnrollmentPolicy                 1.0.0.0    PKI
Cmdlet          Switch-Certificate                                  1.0.0.0    PKI
Cmdlet          Test-Certificate                                    1.0.0.0    PKI
```

2.2. Install the Security World software and create a Security World

1. Install the Security World software by double-clicking on the `SecWorld_Windows-xx.xx.xx.iso` file. For detailed instructions, see the *Installation Guide* and the *User Guide* for the HSM available from the installation disc.
2. Add the Security World utilities path `C:\Program Files\nCipher\nfast\bin` to the Windows system path.
3. Open the firewall port 9004 for the HSM connections.
4. Install the nShield Connect HSM locally, remotely, or remotely via the serial console. See the following nShield Support articles, and the *Installation Guide* for the HSM:
   - How to locally set up a new or replacement nShield Connect
   - How to remotely set up a new or replacement nShield Connect
   - How to remotely set up a new or replacement nShield Connect XC Serial Console model

   Access to the Entrust nShield Support Portal is available to customers under maintenance. To request an account, contact nshield.support@entrust.com.
5. Open a command window and run the following to confirm that the HSM is `operational`:

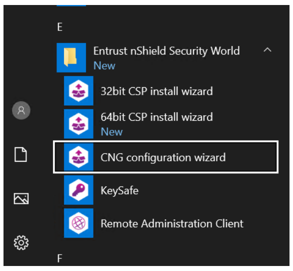
```
>enquiry
Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number       530E-02E0-D947 7724-8509-81E3 09AF-0BE9-53AA 9E10-03E0-D947
 mode                operational
...
Module #1:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        530E-02E0-D947
 mode                 operational
 ...
```

6. Create your Security World if one does not already exist, or copy an existing one. Follow your organization's security policy for this.
7. Confirm that the Security World is `usable`:

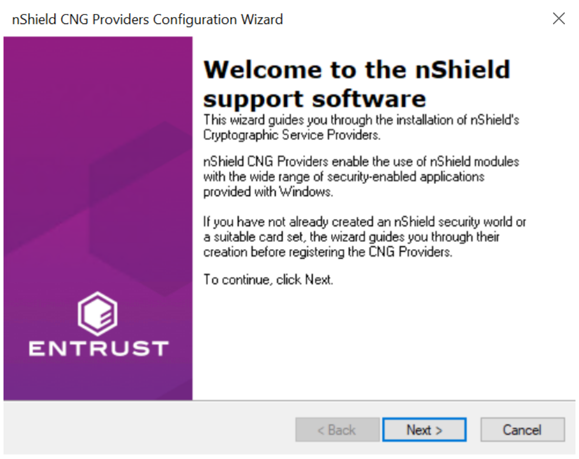
```
>nfkminfoAuthen
 generation  2
 state       0x37270008 Initialised Usable ...
 ...
Module #1
 generation 2
 state      0x2 Usable
 ...
```

2.3. Install and register the CNG provider

1. Select the **Start** button to access all applications. Look for the recently installed nShield utilities.
2. Double-click the CNG configuration wizard and run it as Administrator.



3. Select **Next** on the **CNG Install** welcome screen.


nShield CNG Providers Configuration Wizard

Welcome to the nShield support software

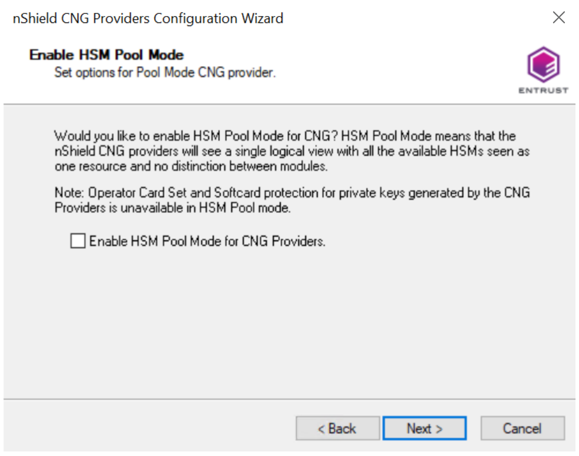
This wizard guides you through the installation of nShield's Cryptographic Service Providers.

nShield CNG Providers enable the use of nShield modules with the wide range of security-enabled applications provided with Windows.

If you have not already created an nShield security world or a suitable card set, the wizard guides you through their creation before registering the CNG Providers.

To continue, click Next.

4. Select **Next** on the **Enable HSM Pool Mode** screen. Leave the **Enable HSM Pool Mode for CNG Providers** check box un-checked.


nShield CNG Providers Configuration Wizard

Enable HSM Pool Mode

Set options for Pool Mode CNG provider.

Would you like to enable HSM Pool Mode for CNG? HSM Pool Mode means that the nShield CNG providers will see a single logical view with all the available HSMs seen as one resource and no distinction between modules.

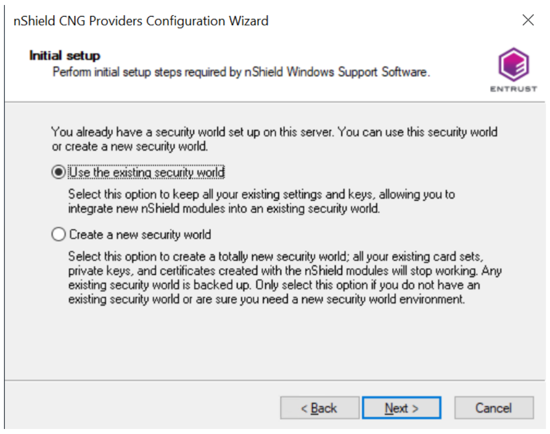
Note: Operator Card Set and Softcard protection for private keys generated by the CNG Providers is unavailable in HSM Pool mode.

☐ Enable HSM Pool Mode for CNG Providers.

5. At the **Security World** screen, select:
    - **Use the existing security world** if you already have a Security World that you intend to use for this integration. The corresponding `world` and `module_xxxx-xxxx-xxxx` files most be present in the `%NFAST_KMDATA%\local` folder. Be prepared to present the quorum of Administrator cards.
    - **Create a new Security World** if you do not currently have a Security World or would like to create a new Security World.

    For the purposes of this *Integration Guide*, an existing Security World is used. For instructions on how to create and configure a new Security World, see the *Installation Guide* and *User Guide* for your HSM.

    Select **Next**.


nShield CNG Providers Configuration Wizard

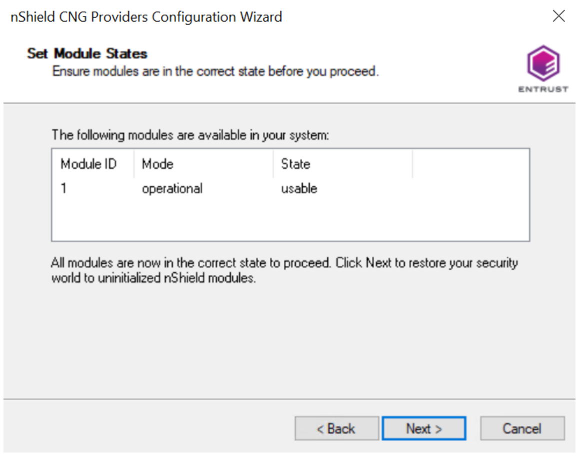
Initial setup
Perform initial setup steps required by nShield Windows Support Software.

You already have a security world set up on this server. You can use this security world or create a new security world.

(●) Use the existing security world
Select this option to keep all your existing settings and keys, allowing you to integrate new nShield modules into an existing security world.

( ) Create a new security world
Select this option to create a totally new security world; all your existing card sets, private keys, and certificates created with the nShield modules will stop working. Any existing security world is backed up. Only select this option if you do not have an existing security world or are sure you need a new security world environment.

6. The **Set Module States** pop-up shows the available HSM(s). Select the desired HSM. The state of the selected HMS should be `operational`. Select **Next**.


nShield CNG Providers Configuration Wizard

Set Module States
Ensure modules are in the correct state before you proceed.

The following modules are available in your system:

Module ID	Mode	State
1	operational	usable

All modules are now in the correct state to proceed. Click Next to restore your security world to uninitialized nShield modules.

7. Select the protection method.
   - Module protection:
     a. In **Key Protection Setup**, select **Softcard protection**, then select **Next**.
   - Operator Card Set protection:
     a. In **Key Protection Setup**, select **Operator Card Set**, then select **Next**.
     b. Enter the OCS name, K of N values, select **Persistent** and **Usable remotely**, then select **Next**.
     c. Insert a blank Operator Card in the HSM.
     d. In **Insert Next Card**, enter a name to for the OCS card, enter the passphrase for the OCS card, then select **Next**.
   - Softcard protection:
     a. In **Key Protection Setup**, select **Softcard protection** , then select **Next**.
     b. In **Token for Key Protection**, select **Create a new Softcard named**, enter the name of the Softcard and the passphrase for the Softcard,

then select **Next**.

8. In **Software installation**, select **Next**.

   The nShield CNG providers are installed and the key Storage Provider is registered.

9. Select **Finish**.
10. Open a command window as administrator and type the following to confirm that the KSP has been successfully registered. Look for **nCipher Security World Key Storage Provider**.

    ```
    >cnglist.exe --list-providers
    Microsoft Key Protection Provider
    Microsoft Passport Key Storage Provider
    Microsoft Platform Crypto Provider
    Microsoft Primitive Provider
    Microsoft Smart Card Key Storage Provider
    Microsoft Software Key Storage Provider
    Microsoft SSL Protocol Provider
    Windows Client Key Protection Provider
    nCipher Primitive Provider
    nCipher Security World Key Storage Provider
    ```

11. Check the registry in `CNGRegistry`:

    ```
    HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Cryptography\Providers\nCipherSecurityWorldKeyStorageProvid
    er
    ```

2.4. Generate the Authenticode master key

This key will be protected by the HSM.

1. Launch PowerShell as Administrator, and run `Generate_Authenticode_MasterKey.ps1`:

   ```
   > .\Generate_Authenticode_MasterKey.ps1
   ```

   Script:

   ```
   $cngProviderName = "nCipher Security World Key Storage Provider"

   $cngAlgorithmName = "RSA"

   $cngKeySize = 2048

   $cngKeyName = "Authenticode_MasterKey"

   $cngProvider = New-Object System.Security.Cryptography.CngProvider($cngProviderName)
   ```

```
$cngKeyParams = New-Object System.Security.Cryptography.CngKeyCreationParameters

$cngKeyParams.provider = $cngProvider

$cngKeyParams.KeyCreationOptions =
[System.Security.Cryptography.CngKeyCreationOptions]::OverwriteExistingKey

$keySizeProperty = New-Object System.Security.Cryptography.CngProperty(
"Length",[System.BitConverter]::GetBytes($cngKeySize),
[System.Security.Cryptography.CngPropertyOptions]::None);

$cngKeyParams.Parameters.Add($keySizeProperty)

$cngAlgorithm = New-Object System.Security.Cryptography.CngAlgorithm($cngAlgorithmName)

$cngKey = [System.Security.Cryptography.CngKey]::Create($cngAlgorithm, $cngKeyName, $cngKeyParams)
```

2. Enter the credentials associated with the protection method:

   - OCS Card Protection:

     The **Create new key** window appears.

     a. Select **Next**.

        The **Select a method to protect new key** window appears.

     b. Select **Operator Card Set protection**.
     c. Insert the OCS card in the HSM and select **Next**.

        The **Select token to protect key with** window appears.

        Notice that the OCS name appears in the **Current Operator Card Sets** box.

     d. Select **Next**.

        The **Choose modules you wish to load the key onto** window appears.

     e. Add the HSM and select **Finish**.

        A credentials window appears.

     f. Enter the OCS passphrase and select **Next**.

        The **Card reading complete** window appears.

     g. Select **Finish**.
   - Softcard protection:

     A softcard credentials window appears.

        a. Enter the Softcard passphrase.
        b. Select **Finish**.
    - Module protection:

      No window appears for module protection.

   A 2048-bit RSA key pair, called `Authenticode_MasterKey`, is generated. The key is encrypted in the HSM, and then pushed to the requesting On-Premise Client server, where it is stored as an Application Key Token in the `%NFAST_KMDATA%\local` folder (`:\ProgramData\nCipher\Key Management Data\local`).

3. Verify the new key:

```
> nfkminfo -k

Key list - 1 keys
 AppName caping              Ident s-1-5-21-3303772969-3979158951-288552622-500--
006507ceb938e7b01e42954ffd4e2238483710eb
```

4. Display the information about the key by copy-pasting the key name. The protection method is displayed according to the type of protection you are using:

For OCS	Cardset
For Softcard	Passphrase
For module protection	Module

```
> nfkminfo -k caping s-1-5-21-3303772969-3979158951-288552622-500--006507ceb938e7b01e42954ffd4e2238483710eb
Key AppName caping Ident s-1-5-21-3303772969-3979158951-288552622-500--
006507ceb938e7b01e42954ffd4e2238483710eb
 BlobKA length        1128
 BlobPubKA length     484
 BlobRecoveryKA length 1496
 name                 "Authenticode_MasterKey"
 hash                 3e7a6ad6f44bbbd4b01b798d1846427f6f7cf7a7
 recovery             Enabled
 protection           CardSet
 other flags          PublicKey !SEEAppKey !NVMemBlob +0x0
 cardset              1e7c555a529da8dd3a1d41604e83ce2b2b9032f8
 gentime              2021-04-19 20:18:15
 SEE integrity key    NONE

BlobKA
 format               6 Token
 other flags          0x0
 hkm                  2a2e6b22ad6a72673473511d91304efd2f76e197
 hkt                  1e7c555a529da8dd3a1d41604e83ce2b2b9032f8
 hkr                  none

BlobRecoveryKA
 format               9 UserKey
 other flags          0x0
 hkm                    none
 hkt                    none
 hkr                    fc4cbd1a6e88c08dd35912d0aecabf47ff1e0c2a

BlobPubKA
 format                 5 Module
 other flags            0x0
 hkm                    c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
 hkt                    none
 hkr                    none

Extra entry #1
 typecode               0x10000 65536
 length                 96
Not a blob
```

2.5. Generate the Authenticode signing certificate

A self-signed code signing certificate will be generated for the purpose of this *Integration Guide*.

1. Launch PowerShell as Administrator, and run `Generate_Authenticode_SelfCert.ps1`:

```
> .\Generate_Authenticode_SelfCert.ps1
```

Script:

```
$cngProviderName = "nCipher Security World Key Storage Provider"

$subjectName = "Authenticode Code Signing Certificate"

$friendlyName = "Authenticode_SelfCert"

# locationName = "Cert:\LocalMachine\My"
$locationName = "Cert:\CurrentUser\My"

$containerName = "Authenticode_MasterKey"

New-SelfSignedCertificate -Subject $subjectName -FriendlyName $friendlyName -Type CodeSigningCert
-CertStoreLocation $locationName -Provider $cngProviderName -ExistingKey -Container $containerName
```

2. Enter the credentials associated with the protection method:
   - OCS card protection:

     The **Load key** window appears.

     a. Select **Next**.

        The **Choose modules you wish to load the key onto** window appears.

     b. Add the HSM and select **Finish**.

A credentials window appears.

c. Enter the OCS passphrase and select **Next**.

The **Card reading complete** window appears.

d. Select **Finish**.

- Softcard protection:

  A softcard credentials window appears.

  a. Enter the Softcard passphrase.

  b. Select **Finish**.

- Module protection:

  No pop-up window appears for module protection.

3. When the script completes, the PowerShell command line displays the following information:

```
> PowerShell -ExecutionPolicy Bypass -File Generate_Authenticode_SelfCert.ps1


   PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My

Thumbprint                                Subject
----------                                -------
1974F986D9B8BF32F47FC2AF33D6271DD905C44F  CN=Authenticode Code Signing Certificate
```

4. The self-signed certificate can be viewed at any time in a PowerShell window:

```
> Get-ChildItem -Path Cert:\CurrentUser\My -CodeSigningCert -Recurse


   PSParentPath: Microsoft.PowerShell.Security\Certificate::CurrentUser\My

Thumbprint                                Subject
----------                                -------
1974F986D9B8BF32F47FC2AF33D6271DD905C44F  CN=Authenticode Code Signing Certificate
```

The certificate can also be viewed with the Windows Certificate Manager.

2.6. Sign and time-stamp the code with Microsoft SDK

The `signtool` utility is used for signing. The Entrust on-line Time Stamp Server (TSS) is used to time stamp.

`signtool` can access the code-signing certificate the following ways:

- Directly from the certificate store with the thumbprint. This is the preferred method.
- By name, for example `Authenticode Code Signing Certificate`.
- By pointing to a `.cer` file that was created by exporting the certificate.

If you plan to access the certificate by pointing to a `.cer` file:

1. Export the certificate to your preferred location in the file system.
2. Launch PowerShell as Administrator, and run `Export_Certificate.ps1`:

   ```
   > .\Export_Certificate.ps1
   ```

   Script:

   ```
    $certificate = Get-ChildItem -Path Cert:\CurrentUser\My\ | Where-Object {$_.Subject -eq "CN=Authenticode
   Code Signing Certificate"}

    Export-Certificate -FilePath "C:\Users\Administrator\Documents\Authenticode Code Signing Certificate.cer"
   -Cert $certificate
   ```

3. View the exported certificate by right-click on it and selecting **Properties**.
4. An executable called `MyTestApplication.exe` was created for the purpose of this guide. Sign and time stamp it by running one of the signing scripts provided. There is a script for each protection method. Modify the script as needed.

For all protection methods:

1. Run the required script:

   - OCS card protection:

     a. Run the following script:

        ```
        > .\Sign_with_MS_SDK_OCS.ps1
        ```

        Script:

        ```
        # CNG Provider
        #$cngProviderName = "nCipher Security World Key Storage Provider"


        # Certificate Key
        #$containerName = "Authenticode_MasterKey"


        # Certificate Name
        ```

```
#$certName = "Authenticode Code Signing Certificate"
#$certificatePath = "C:\Users\Administrator\Documents\Authenticode Code Signing
Certificate.cer"
$certHash = "1974f986d9b8bf32f47fc2af33d6271dd905c44f"


# OCS Password
#$password = "xxxxxxxxxxx"


# Entrust Certificate Services Time Stamp Server
$timestampServer = "http://timestamp.entrust.net/TSS/RFC3161sha2TS"


# File to be Signed
$fileName = "C:\Users\Administrator\Documents\MyTestApplication.exe"


#signtool sign /debug /fd SHA256 /n $certName /td SHA256 /tr $timestampServer $fileName
#signtool sign /debug /fd SHA256 /csp $cngProviderName /kc $containerName /f $certificatePath
/p $password /td SHA256 /tr $timestampServer $fileName
signtool sign /debug /fd SHA256 /sha1 $certHash /td SHA256 /tr $timestampServer $fileName
```

b. Select the HSM and enter the passphrase when prompted.

- Softcard protection:

  a. Run the following script:

```
> .\Sign_with_MS_SDK_Softcard.ps1
```

Script:

```
# CNG Provider
$cngProviderName = "nCipher Security World Key Storage Provider"


# Certificate Key
$containerName = "Authenticode_MasterKey"


# Certificate Name
$certName = "Authenticode Code Signing Certificate"
$certificatePath = "C:\Users\Administrator\Documents\Authenticode Code Signing Certificate.cer"

# Softcard Password
$password = "1234"

# Entrust Certificate Services Time Stamp Server
$timestampServer = "http://timestamp.entrust.net/TSS/RFC3161sha2TS"

# File to be Signed
$fileName = "C:\Users\Administrator\Documents\MyTestApplication.exe"

signtool sign /debug /fd SHA256 /csp $cngProviderName /kc $containerName /f $certificatePath /p
$password /td SHA256 /tr $timestampServer $fileName
```

  b. Enter the passphrase when prompted.

- Module protection:

a. Run the following script:

```
> .\Sign_with_MS_SDK_Module.ps1
```

Script:

```
# CNG Provider
$cngProviderName = "nCipher Security World Key Storage Provider"


# Certificate Key
$containerName = "Authenticode_MasterKey"


# Certificate Name
$certName = "Authenticode Code Signing Certificate"
$certificatePath = "C:\Users\Administrator\Documents\Authenticode Code Signing Certificate.cer"
#$certHash = "1974f986d9b8bf32f47fc2af33d6271dd905c44f"


# Password
#$password = "123"
#$password = "xxxxxxxxxxx"


# Entrust Certificate Services Time Stamp Server
$timestampServer = "http://timestamp.entrust.net/TSS/RFC3161sha2TS"


# File to be Signed
$fileName = "C:\Users\Administrator\Documents\MyTestApplication.exe"


signtool sign /debug /fd SHA256 /n $certName /td SHA256 /tr $timestampServer $fileName
```

For all protection types, the PowerShell command line displays script output information similar to the following:

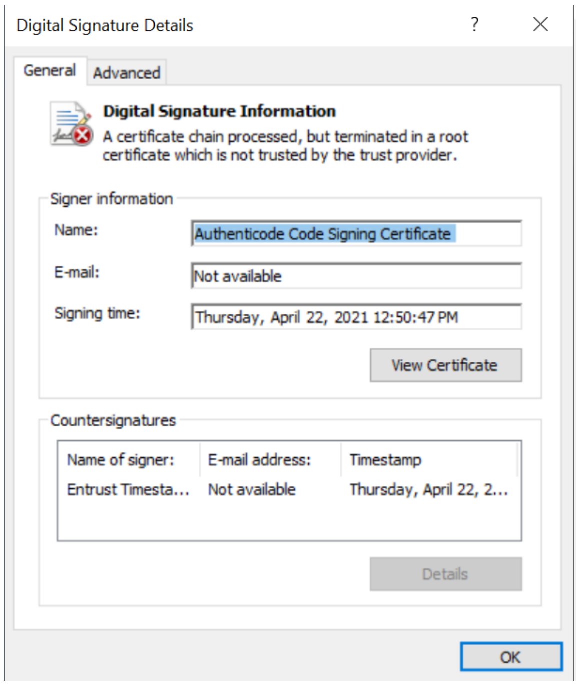
```
The following certificates were considered:
    Issued to: Authenticode Code Signing Certificate
    Issued by: Authenticode Code Signing Certificate
    Expires:   Tue Apr 19 17:48:34 2022
    SHA1 hash: 1974F986D9B8BF32F47FC2AF33D6271DD905C44F

After EKU filter, 1 certs were left.
After expiry filter, 1 certs were left.
After Hash filter, 1 certs were left.
After Private Key filter, 1 certs were left.
The following certificate was selected:
    Issued to: Authenticode Code Signing Certificate
    Issued by: Authenticode Code Signing Certificate
    Expires:   Tue Apr 19 17:48:34 2022
    SHA1 hash: 1974F986D9B8BF32F47FC2AF33D6271DD905C44F

Done Adding Additional Store
Successfully signed: C:\Users\Administrator\Documents\MyTestApplication.exe

Number of files successfully Signed: 1
Number of warnings: 0
Number of errors: 0
```

2. Select the **Properties** of the test executable that was signed. Select the **Digital Signatures** tab, and **Details**. Notice the time stamp and other signing parameters just created.


Digital Signature Details

General | Advanced

Digital Signature Information
A certificate chain processed, but terminated in a root certificate which is not trusted by the trust provider.

Signer information

Name: Authenticode Code Signing Certificate
E-mail: Not available
Signing time: Thursday, April 22, 2021 12:50:47 PM

Countersignatures

Name of signer:	E-mail address:	Timestamp
Entrust Timesta...	Not available	Thursday, April 22, 2...

2.7. nShield and Microsoft Strong Name integration

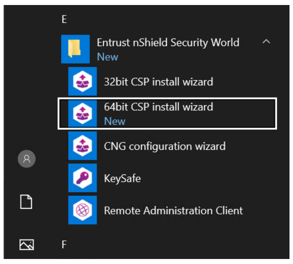
Strong names prevent spoofing of your code by a third party.

Strong name signing is provided with the Microsoft SDK to give .NET Framework assemblies unique identities. To get a valid strong name, an assembly is strong-name signed during the build process. This is done by using the private key that corresponds to the public key in the strong name.

The strong name signature can then be verified using the public key.

Make sure you use the 32-bit version of the `sn` utility. The 64-bit version is not supported.

1. Install and register the CAPI provider, which is required for strong name signing with nShield products. Run both the **32bit CSP install wizard** and the **64bit CSP install wizard** and follow the steps. The following shows an example of the 64bit installer.



Do not place the HSM in `Pre-initialization` mode as descried in Install and register the CNG provider. Name the OCS `MSAuthenticodeCAPI`.

2. Open a command window as **Administrator**.
3. Set the cryptography service provider.

   Notice that the output **nCipher Enhanced Cryptographic Provider** differs from the **nCipher Security World Key Storage Provider** of the CNG.

```
>sn -c "nCipher Enhanced Cryptographic Provider"

Microsoft (R) .NET Framework Strong Name Utility  Version 4.0.30319.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Default CSP set to 'nCipher Enhanced Cryptographic Provider'
```

4. Create the container:

```
>keytst -c StrongNamed_Container
CSP being used: nCipher Enhanced Cryptographic Provider

Container 'StrongNamed_Container' created
```

5. Create the key in the container:
   a. Run the following command:

```
>keytst -cs StrongNamed_Container
CSP being used: nCipher Enhanced Cryptographic Provider
Generating 2048-bit key(s) for the 'StrongNamed_Container' container.

New signature key generated OK.
```

      The key protection wizard starts. For example, for the **64-bit CSP install**:

   b. Select **Next**.

      The **Choose an Operator Card Set** window appears.

   c. Select the OCS and select **Next**.

The **Choose Modules** window appears.

d. Select the HSM and select **Next**.

   The **Load a Token** window appears.

e. Enter the passphrase and select **Next**.

   A completion window appears.

f. Select **Finish**.

6. List the container and the key that you created:

```
>csputils -U ALL

File ID    Container name       Container owner      DLL name   S X
=========  ===================  ===================  =========  = =
49942fc5f  StrongNamed_Cont...  WIN-G0JD49QMHEV\...  ncsp       *

1 container and 1 key found.
```

7. Show the public half of the key created above. If using an OCS, select the cardset and the HSM. Enter the passphrase as done above.

```
>keytst -p StrongNamed_Container
CSP being used: nCipher Enhanced Cryptographic Provider

Container 'StrongNamed_Container':

Key exchange key not present.
Public half of Signature key is:
06 02 00 00 00 24 00 00 52 53 41 31 00 08 00 00
01 00 01 00 17 F1 63 62 50 41 94 16 90 3F 25 CD
2B 6F FF C5 91 29 74 BA AE 67 EF C4 3D E9 E5 26
10 EE E4 D9 71 1C 0D 7F 2D 71 89 D1 E6 A9 77 A3
AE EB 79 0A 5F D6 E4 3B 41 5A 17 C3 30 B3 8F DF
A0 01 E2 93 8E 6F 6B 5B E2 E5 01 A7 19 25 48 EF
4C 65 29 8D 5E E1 0D 55 E8 93 43 9D 9A 41 EC 93
BD A1 83 11 CF C8 E3 A0 4E AC 78 DC 70 A2 46 56
3F 56 FA 65 C4 4C C1 95 57 CD 40 FE 3A 3F 7B 00
BD BA D5 AB EB 1B A4 06 7A 80 A5 41 27 24 B0 18
45 58 7E B5 7A 5C 80 5D 8C 7D 06 78 2B 3A 7F 36
96 42 D9 29 65 EE 1D 7D 1F E2 15 B6 8C C5 32 66
CB DC C5 54 27 4E D5 76 41 E6 3C FF 00 C0 E6 91
64 FD 4D 2E E8 FD 3D 7A D0 88 CF 0F 14 FA AE 38
70 6D 57 99 31 74 0E B1 29 16 A5 D0 8F FE 1B 08
09 51 AE 7F 76 FC 1F D0 84 7A 4E 76 0C DC E0 D3
2E C9 2F 8F 5E D6 62 7C 38 75 2D AA 47 EF 56 0F
6F 56 78 A7
```

8. Extract the public half of the key to a file:

```
sn -pc StrongNamed_Container C:\Users\Administrator\Documents\StrongNamed_Container.snk

Microsoft (R) .NET Framework Strong Name Utility Version 4.0.30319.33440
Copyright (c) Microsoft Corporation. All rights reserved.
```

```
Public key written to StrongNamed_Container.snk
```

9. The executable `MyTestProgram.cs` was created for the purpose of this guide.

   Compile `MyTestProgram.cs` with the public key part of `StrongNamed_Container`:

```
csc /delaysign+ /keyfile:"C:\Users\Administrator\Documents\StrongNamed_Container.snk"
/out:C:\Users\Administrator\Documents\MyTestProgram.exe C:\Users\Administrator\Documents\MyTestProgram.cs

Microsoft (R) Visual C# Compiler version 4.8.3761.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.
```

10. Re-sign the signed `MyTestProgram.exe` executable with `StrongNamed_Container`:

```
>sn -Rc C:\Users\Administrator\Documents\MyTestProgram.exe StrongNamed_Container

Microsoft (R) .NET Framework Strong Name Utility  Version 4.0.30319.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Assembly 'C:\Users\Administrator\Documents\MyTestProgram.exe' successfully re-signed
```

11. Verify the self-consistency of the strong name signature:

```
>sn -v C:\Users\Administrator\Documents\MyTestProgram.exe

Microsoft (R) .NET Framework Strong Name Utility  Version 4.0.30319.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Assembly 'C:\Users\Administrator\Documents\MyTestProgram.exe' is valid
```

12. Disassemble `MyTestProgram.exe`:

```
>ildasm C:\Users\Administrator\Documents\MyTestProgram.exe
```