Microsoft Authenticode

Integration Guide

CryptoServer HSM

SecurityServer 4.45.3

utimaco®

Imprint

Copyright 2026	Utimaco IS GmbH Krefelder Straße 220 52070 Aachen Germany
Phone	AMERICAS +1-844-UTIMACO (+1 844-884-6226) EMEA +49 800-627-3081 APAC +81 800-919-1301
Internet	https://support.hsm.utimaco.com/
e-mail	support@utimaco.com
Document Version	1.0.0
Date	2026-03-03
Status	PUBLISHED
Document No.	IG-2026-0015

Table of Contents

1 Introduction

This guide is part of the information and support provided by Utimaco. Additional documentation produced to support your Utimaco SecurityServer product can be found in the document directory of the Utimaco SecurityServer product bundle. All Utimaco SecurityServer product documentation is available from Utimaco's website at: https://utimaco.com/.

1.1 About This Guide

This guide describes how to enable HSM integration with Microsoft Authenticode.

1.1.1 Target Audience for This Guide

This guide is intended for Microsoft Authenticode and HSM administrators.

1.1.2 Document Conventions

The following conventions are used in this guide:

Convention	Use	Example
Bold	Items of the Graphical User Interface (GUI), e.g., menu options	Press **OK**
`Monospaced`	Code that is given for explanation or as an example, file paths	`chsm-create`
Italic	References and important terms	See *Sample Chapter* in the *CryptoServer - Sample Manual*

Table 1: Document conventions

We use special icons to highlight the most important notes and information.

Here you will find important safety information that should be followed.

Here you will find additional notes or supplementary information.

This message indicates the expected result after the successful execution of an instruction.

1.1.3 Abbreviations

The following abbreviations are used in this guide:

Abbreviation	***Meaning***
CA	Certificate Authority
CLI	Command Line Interface
CNG	Cryptography API Next Generation
CSAR	Cloud Service Architecture
CSP	Cryptographic Service Provider
CXI	Cryptographic eXtended Services Interface
FIPS	Federal Information Processing Standards
GUI	Graphical User Interface

Abbreviation	*Meaning*
HMAC	Hash-based message authentication code
HSM	Hardware Security Module
LAN	Local Area Network
MBK	Master Backup Key
PKI	Public Key Infrastructure
PS	PowerShell
RSA	Rivest-Shamir-Adleman
SDK	Software Development Kit
TSS	Time Stamp Server
URL	Uniform Resource Locator

Table 2: List of abbreviations

2 Overview

2.1 Microsoft Authenticode

Microsoft Authenticode is a code-signing technology that identifies the publisher of Authenticode-signed software. It also verifies that the software has not been tampered with since it was signed and published. Authenticode uses cryptographic techniques to verify publisher identity and code integrity.

Authenticode relies on proven cryptographic techniques from Microsoft and the use of one or more private keys to sign and timestamp published software. From a security point of view, it is important to maintain the confidentiality of these code signing keys. The CryptoServer Hardware Security Module (HSM) integrates with Microsoft Authenticode to provide a trusted system for protecting the organizational credentials of a software publisher. The

CryptoServer HSM secures the code signing keys on a certified industry standard FIPS 140-2.

This integration guide covers all the necessary information to install, configure and integrate Microsoft Authenticode with Utimaco Hardware Security Modules (HSM).

The benefits of using an HSM with Microsoft Authenticode include:

- Private key will be securely stored on HSM
- Hardware is FIPS 140-2 level 3 validated
- Trusted timestamp (TSS) for Authenticode.

Refer to the Microsoft documentation, for more information about installing Microsoft Authenticode.

2.2 Utimaco CryptoServer HSM

CryptoServer is a hardware security module developed by Utimaco IS GmbH. CryptoServer is a physically protected specialized computer unit designed to perform sensitive cryptographic tasks and to securely manage as well as store cryptographic keys and data. It can be used as a universal, independent security component for heterogeneous computer systems.

3 Integration Requirements and Prerequisites

Ensure that the system environment you will be using, meets the following hardware and software requirements.

This guide assumes that the user has already installed and configured required software.

3.1 Tested Versions

The integrations that have been successfully tested with the Utimaco HSM with Microsoft Authenticode.

Operating System	Microsoft .NET Framework	Windows SDK	Utimaco Security Server Version	Utimaco HSM
Windows Server 2019 Windows Server 2016 Windows 11	4.8	10.1	SecurityServer 4.45.3	CryptoServer CSeSeries/Se-Series

Table 3: List of tested versions

3.2 Software Requirements

Software	Software Requirements
HSM Interfaces	CryptoServer CSP/CNG Provider
Microsoft .NET Framework	4.8
Windows SDK	10.1

Table 4: List of software requirements

3.3 Hardware Requirements

Hardware	Hardware Requirements
Utimaco LAN HSM	CryptoServer CSe-Series/Se-Series LAN with firmware SecurityServer 4.45 or higher
Utimaco PCI-e HSM	CryptoServer CSe-Series/Se-Series PCI-e with firmware SecurityServer 4.45 or higher

Table 5: List of hardware requirements

Setup an account on the Utimaco support portal and request download access at the following URL: https://support.hsm.utimaco.com/.

3.4 Prerequisites

Before you begin, please ensure that you have:

- Installed/set up the operating system listed in Tested Versions.
- Installed/set up the SecurityServer version listed in Tested Versions.
- Replaced the CryptoServer Default Admin with a new admin user.
- Created and stored the MBK onto each HSM. Refer to the CryptoServer documentation to set up the MBK.
- Set up and configured CryptoServer. Refer to the CryptoServer documentation to setup the HSM.
- Installed and configured Microsoft SDK as listed in Tested Versions.

4 Configuring the CSP-CNG Provider

4.1 Introduction and Prerequisites

A CSP (Cryptographic Service Provider) is a general-purpose cryptography standard, developed by Microsoft. On one side it defines a cryptographic interface to be used by applications (CryptoAPI). On the other side it defines an interface to be used by manufacturers to integrate their cryptographic hardware.

A CNG (Cryptography API Next Generation) is the second-generation cryptographic interface, developed by Microsoft. It offers updated cryptographic algorithms and is intended for a long-term replacement of CSP.

When installing the CryptoServer Setup make sure to select the CPS/CNG - Cryptographic Service Provider (Microsoft) interface. A Cryptographic User should be created as well as an MBK should be generated.

Generating the MBK is necessary for the HSM to become operational. Without the MBK, one cannot run any cryptographic operations.

4.2 Creating HSM Users

Start the CryptoServer Administration Tool and login a user with the permission level of at least 02000000.

4.2.1 Creating a Key Manager User

If the Key manager and Crypto user roles are separated, a Key Manager user might need to be created.

More users with the permission level 00000010 might be needed (Group 1) to enforce "m of n" security policy for the key management and smart card authentication might need to be used.

For this guide only one Key Manager User will be created.

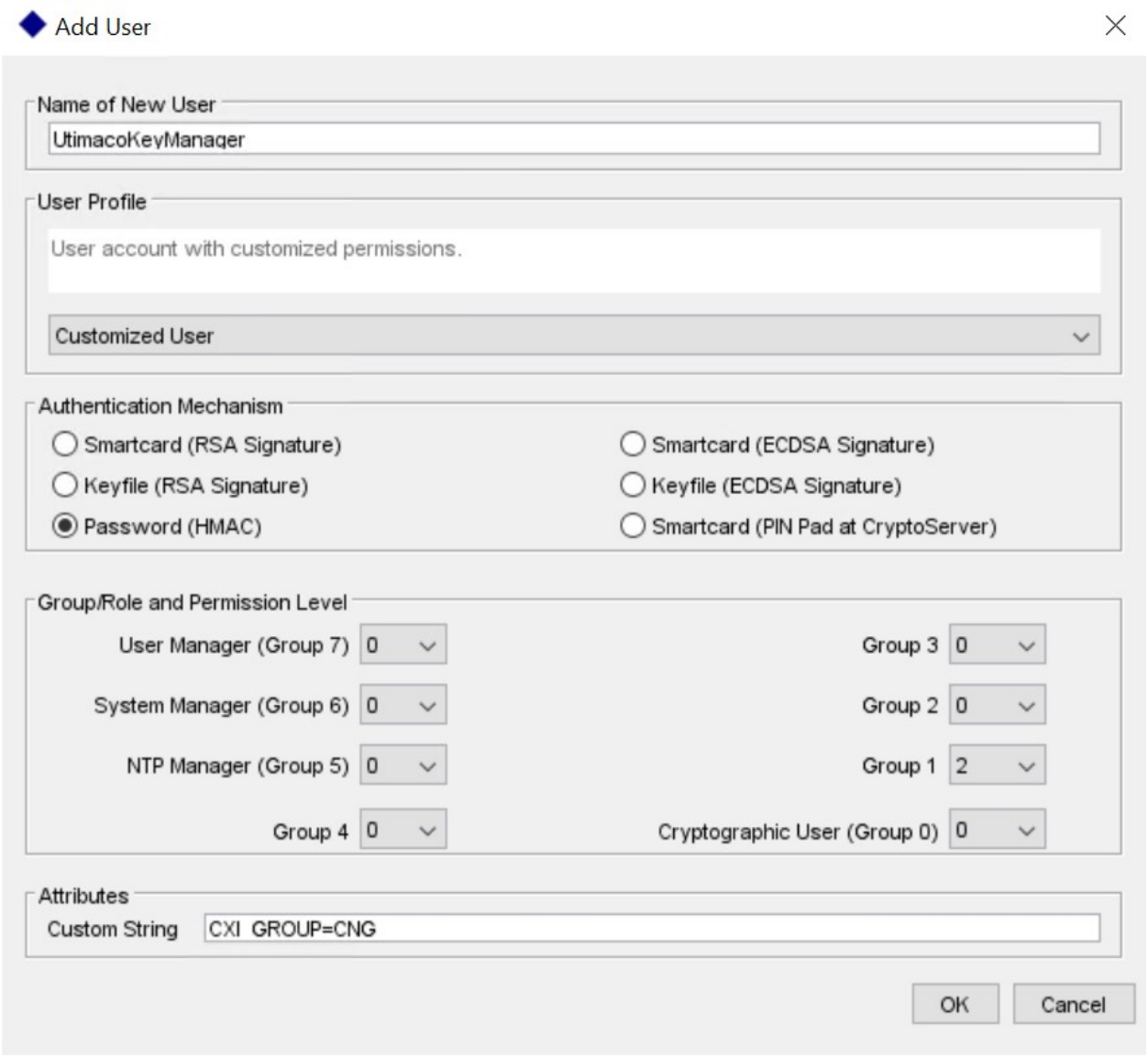

Figure 1 : Creating Key Manager user

4.2.2 Creating a Crypto User

Crypto Users with permission level of 00000002 will have to be created. Use encrypted passwords. For this guide, a user with permission level of 00000002, CXI Group "CNG" and HMAC password will be created.

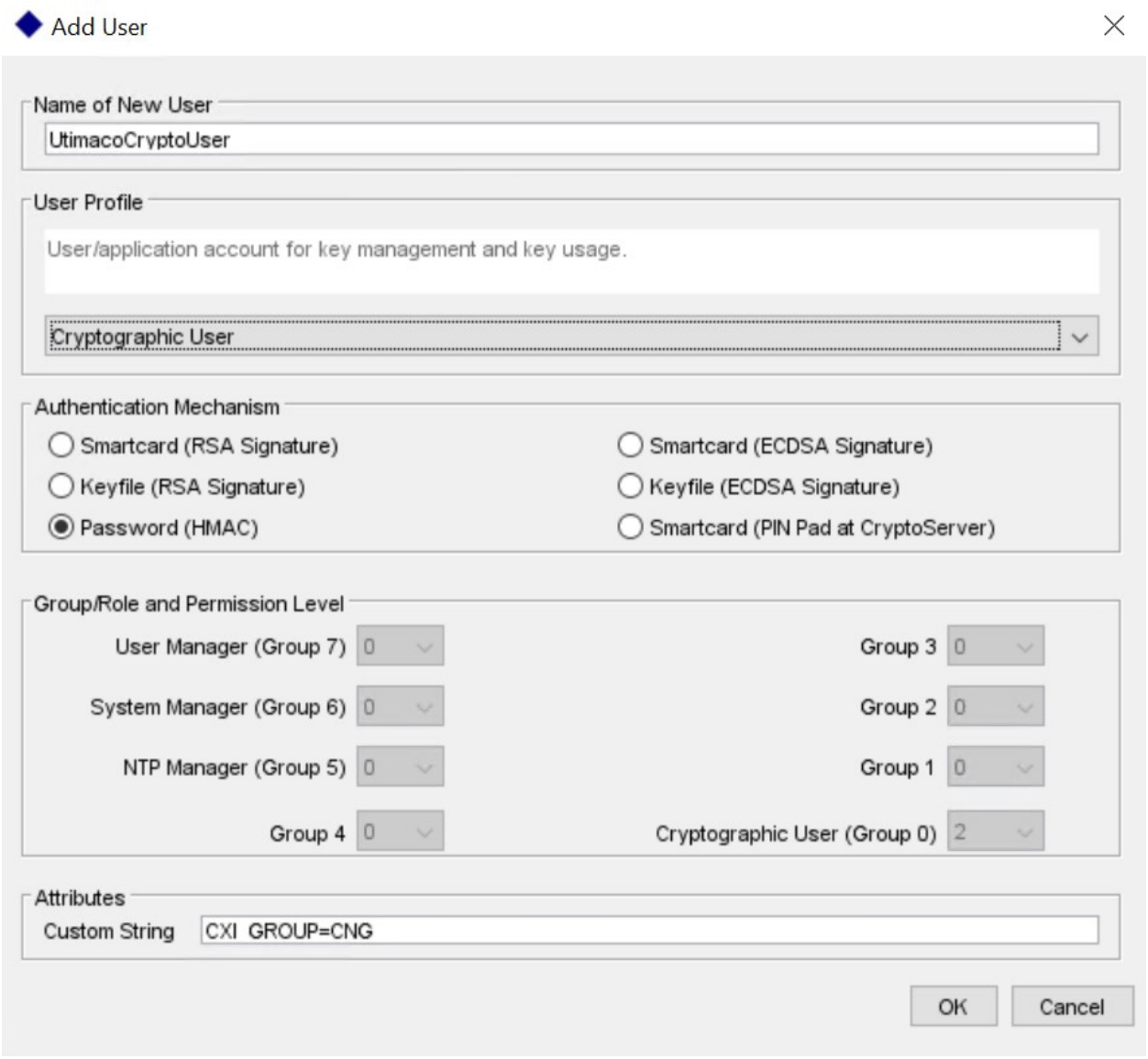

Figure 2 : Creating a Crypto user

Based on your requirement, the user can use Password (HMAC), Smart Card or KeyFile protection type. If you are using Smartcard Authentication, the prompt will go on the PIN Pad device to insert Smartcard and enter the pin. Then press OK button on the PIN Pad.

4.3 Setting up the CSP/CNG Provider

The `CS_CNG_CFG` environment variable contains the path and name of the configuration file.

By default, it is located at C:\ProgramData\Utimaco\CNG\cs_cng.cfg.

For more advanced configuration, refer to [CspCng];

1. Open the `cs_cng.cfg` file with an appropriate text editor

`›_ Console`

```
> notepad %CS_CNG_CFG%
```

2. For this installation set the path to the log file and set the log level to "ERROR".

`cs_cng.cfg`

```
# Path to the logfile (name of logfile is attached by the API)

Logpath = C:\ProgramData\Utimaco\CNG\log

# Loglevel (0 = NONE; 1 = ERROR; 2 = WARNING; 3 = INFO; 4 = TRACE)  Logging
= 1
```

To make your testing easier, it would be good to enable the CNG log file. That can be enabled by editing the Logging Loglevel. Set the LogPath and Logging Loglevel to 1. For testing you may want to increase it to 4.

The added LogPath points to a writable directory, not to a file.

If you encounter problems, check the log file named cs_cng.log in the LogPath defined directory. When you are done testing, you should change Logging to 1 or 2. This will limit the logging to only critical and important messages.

3. Set the Login. In this case, the name of the Cryptographic User is "UtimacoCryptoUser" with an HMAC password "Utimaco19".

cs_cng.cfg

```
Login = UtimacoCryptoUser,HMACPwd=Utimaco19
```

If using Smartcard or KeyFile protection make the appropriate change in the Login Section as shown below:

Login = username,RSASign=filename#password

Login = "SmartCardUser,RSASign=:cs2:auto:USB0@<HSM-IP>"

For additional information refer CryptoServer_csadm_Manual_Systemadministrators.pdf document, found on the product CD in the Documentation directory.

4. Set the IP address of the HSM

cs_cng.cfg

```
# default device and fallback devices

Device = 10.44.223.141
```

For more information regarding the commands and command parameters please check the Utimaco documentation. The device may be a CryptoServer (PCIe or LAN) device. The device line will follow one of these patterns, based on the HSM form-factor:

Device = 288@<HSM IP address> Hardware (LAN) HSM

OR

Device = /dev/cs2.0 Hardware (PCIe) HSM

4.3.1 Testing Connection

To enumerate providers, use the following command:

```
›_ Console
```

```
> cngtool EnumProvider
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
Utimaco CryptoServer Key Storage Provider
```

To get the provider information, use the following command:

›_ Console

```
>cngtool ProviderInfo

Provider : Utimaco CryptoServer Key Storage Provider

Device : 10.44.223.141

Group : CNG

Mode : Internal Key Storage

------------------------------------------------------------

Name : Utimaco CryptoServer Key Storage Provider

Name : Utimaco CryptoServer Key Storage Provider

Version : 0x02010000

Impl. -Type : 0x00000011

MaxNameLength : 0x00000104  Device : 10.44.223.141

Group : CNG

Mode : Internal Key Storage
```

5 Generate the Authenticode Key using PowerShell

The following procedure demonstrates the key and certificate generation for the Authenticode and signing executables using the PowerShell script.

5.1 Install the PowerShell PKI Module

PKI Module is intended to simplify various PKI management tasks, by using automation with Windows PowerShell. It is intended for Certification Authority (CA) management.

Using the below command in PowerShell the user will get all the PKI modules installed.

```
›_ PowerShell
```

```
PS C:\> Install-Module -Name PSPKI
```

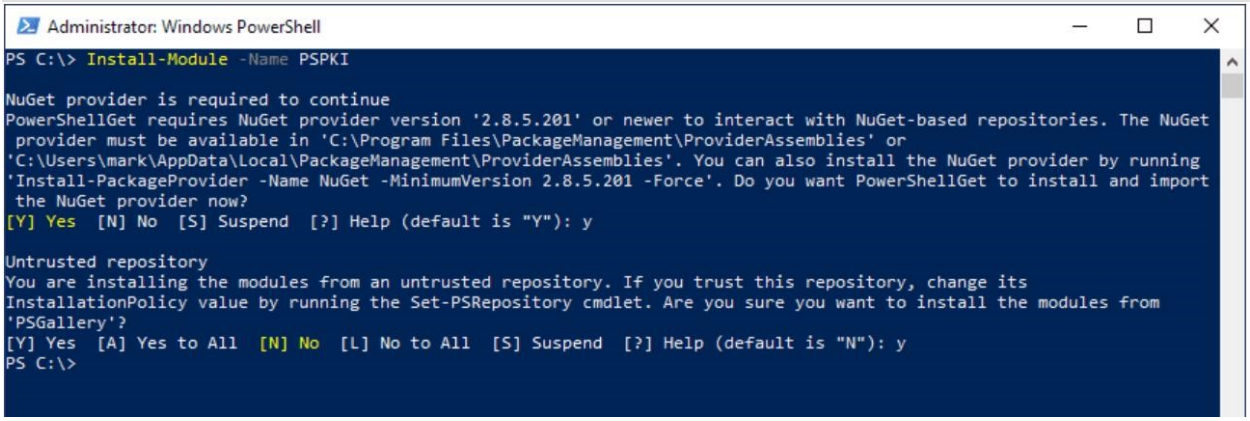

Figure 3 : PKI module installation window

Verify the following PKI modules are installed correctly by running following command.

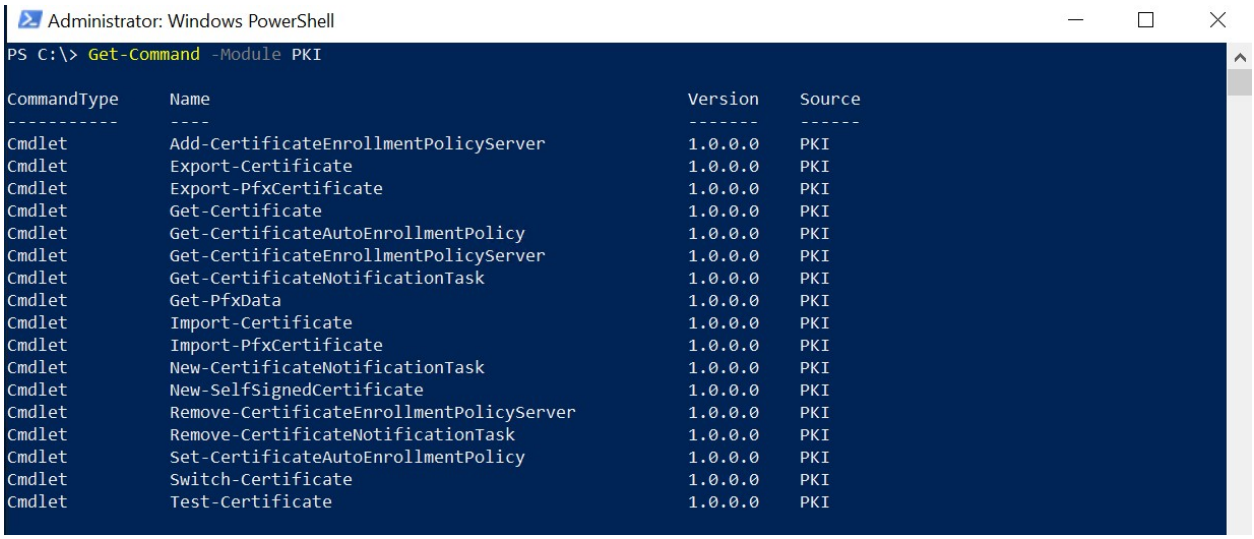

Figure 4 : PKI module list

5.2 Generate the Authenticode Key

To generate the Authenticode Key, follow the below steps:

1. Create a PowerShell script file with name `Generate_AuthenticodeKey.ps1` at appropriate location and add the following content into the script file.

Generate_AuthenticodeKey.ps1

```
#Define Utimaco Provider

$UtimacoProviderName = "Utimaco CryptoServer Key Storage Provider"

#Define Algorithm

$AlgorithmName = "RSA"

#Define Key Size

$KeySize = 2048

# Provide the Key Name

$KeyName = "Authenticode_TestKey"

$KeyParams = New-Object

System.Security.Cryptography.CngKeyCreationParameters

$KeyParams.provider = New-Object

System.Security.Cryptography.CngProvider($UtimacoProviderName)

$KeyParams.KeyCreationOptions =

[System.Security.Cryptography.CngKeyCreationOptions]::OverwriteExistingKe y


$keySizeProperty = New-Object

System.Security.Cryptography.CngProperty("Length",[System.BitConverter]::

GetBytes($KeySize),

[System.Security.Cryptography.CngPropertyOptions]::None);

$KeyParams.Parameters.Add($keySizeProperty)

$Algorithm = New-Object

System.Security.Cryptography.CngAlgorithm($AlgorithmName)
```

```
Generate_AuthenticodeKey.ps1
```

```
$Key = [System.Security.Cryptography.CngKey]::Create($Algorithm, $KeyName, $KeyParams)
```

2. Launch PowerShell as Administrator, and run `Generate_AuthenticodeKey.ps1`.

```
›_ PowerShell
```

```
> .\Generate_AuthenticodeKey.ps1
```

If you are using Smartcard Authentication, the prompt will go on the PIN Pad device to insert Smartcard and enter the pin. Then press OK button on the PIN Pad.

3. The generated keys will be protected by the HSM and can be verified using below command

```
›_ Console
```

```
>cngtool ListKeys
------------------------------------------------------------
Provider : Utimaco CryptoServer Key Storage Provider
Device : 10.44.223.141
Group : CNG
Mode : Internal Key Storage
------------------------------------------------------------
Index AlgId Size Group Name Spec
------------------------------------------------------------
1 RSA 2048 CNG Authenticode_TestKey 0
```

If you are using Smartcard Authentication, the prompt will go on the PIN Pad device to insert Smartcard and enter the pin. Then press OK button on the PIN Pad.

5.3 Generate the Authenticode Signing Certificate

To generate a self-signed code signing certificate, follow the below steps:

1. Create a PowerShell script file with name `Generate_Authenticode_SelfCert.ps1` at appropriate location and add the following content into the script file.

Generate_Authenticode_SelfCert.ps1

```
#Define Utimaco Provider

$UtimacoProviderName = "Utimaco CryptoServer Key Storage Provider"

#Define Subject Name of the Self Signed Certificate

$SubjectName = "Authenticode Certificate"

#Define Friendly Name

$FriendlyName = "Authenticode_SelfCert"

#Based on the Key Store (Local Machine or Current user) make appropriate
changes in location name parameter #LocationName = "Cert:\LocalMachine\My"

$LocationName = "Cert:\CurrentUser\My"

#Container name should match with Keyname parameter in 5.2 section

$ContainerName = "Authenticode_TestKey"

#Generate Self Signed Certificate

New-SelfSignedCertificate -Subject $SubjectName -FriendlyName

$FriendlyName -Type CodeSigningCert -CertStoreLocation `

$LocationName -Provider $UtimacoProviderName -ExistingKey -Container
$ContainerName
```

2. Launch PowerShell as Administrator, and run `Generate_Authenticode_SelfCert.ps1`.

```
›_ PowerShell

> .\Generate_Authenticode_SelfCert.ps1

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

Thumbprint Subject

---------- -------

1974F986D9B8BF32F47FC2AF33D6271DD905C44F CN=Authenticode Certificate
```

If you are using Smartcard Authentication, the prompt will go on the PIN Pad device to insert Smartcard and enter the pin. Then press OK button on the PIN Pad.

3. The self-signed certificate can be viewed in a PowerShell window, as seen below.

```
›_ PowerShell

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

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

Thumbprint                                Subject

----------                                -------

1974F986D9B8BF32F47FC2AF33D6271DD905C44F  CN=Authenticode Certificate
```

5.4 Sign and Time-stamp the Code with Microsoft SDK

Sign Tool is a command-line tool, and it is used for verifying signatures in files, digitally signing files, and to time-stamp files. To time stamp an `.exe file`, the user needs to use the on-line Time Stamp Server (TSS).

5.4.1 Exporting a Certificate

1. Create a PowerShell script file with name `Export_Certificate.ps1` at appropriate location and add the following content into the script file.

```
Export_Certificate.ps1
```

```
$SelfCertificate = Get-ChildItem -Path Cert:\CurrentUser\My\ | WhereObject
{$_.Subject -eq "CN=Authenticode Certificate"}

#Exporting the Self Signed Certificate at the appropriate location

Export-Certificate -FilePath "C:\Authenticode\Authenticode

Certificate.cer" -Cert $SelfCertificate
```

2. Launch PowerShell as Administrator, and run `Export_Certificate.ps1`.

```
›_ PowerShell
```

```
> . \Export_Certificate.ps1

PS C:\> C:\Authenticode\Export_Certificate.ps1


Directory: C:\Authenticode


Mode LastWriteTime Length Name

---- ------------- ------ ----

-a---- 3/24/2022 6:35 AM 1329 Authenticode Certificate.cer
```

3. To view the Properties, double-click on the exported certificate.

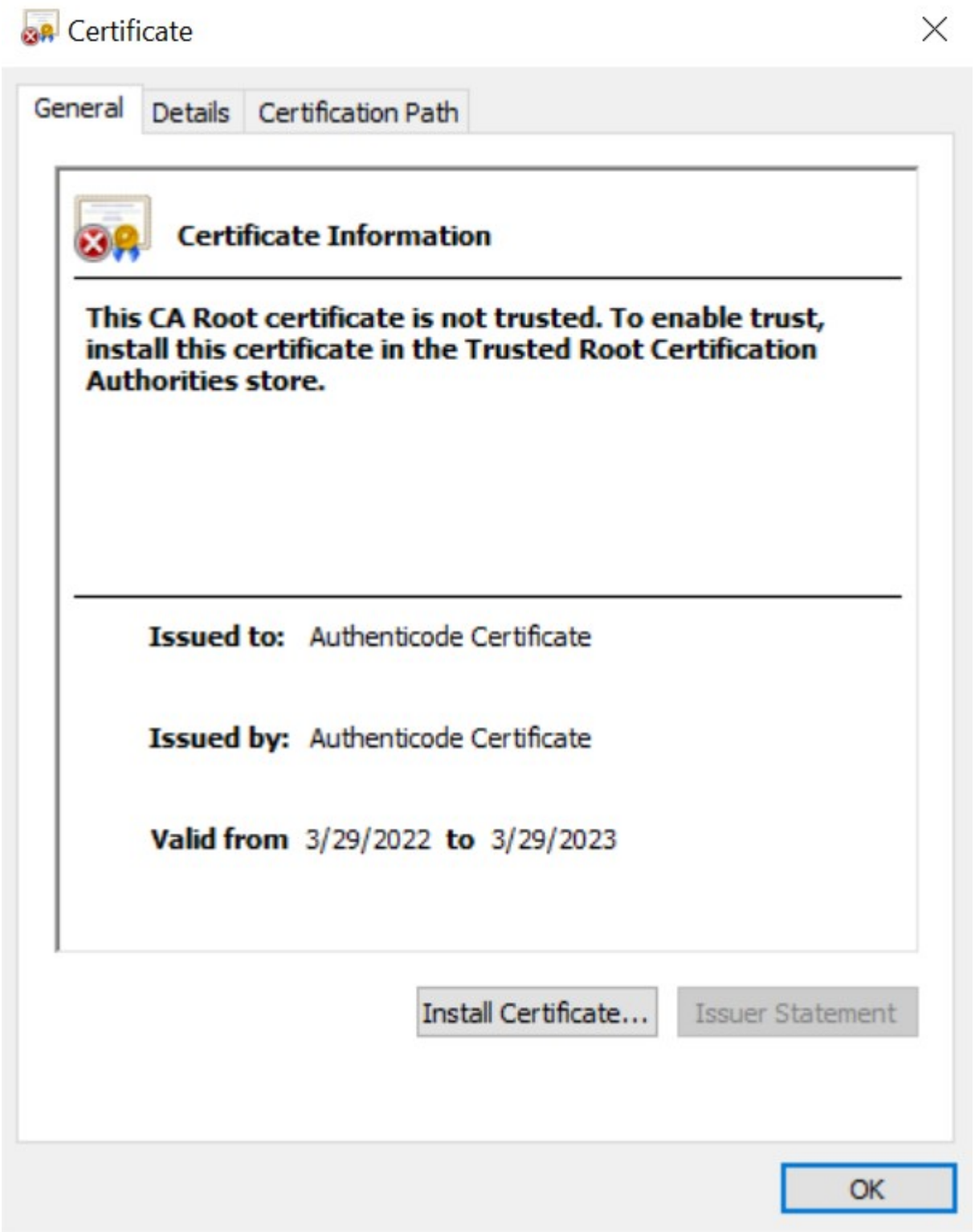

Figure 5 : Exported certificate details

5.4.2 Sign and Time-stamp the Executable

There are different ways to access the code-signing certificate using Signtool.

- The preferred method to do this is directly from the certificate stored with the thumbprint.
- The second method is by using the key name
- The third method is by pointing to a .cer file that was created by exporting the certificate.

In this guide an executable file called `MyApplication.exe` was created, signed, and timestamped.

1. Create a PowerShell script file with name `Sign_Timestamp_Executable.ps1` and add the following content into the script file.

`Sign_Timestamp_Executable.ps1`

```
#The first method using the certificate hash value
#Get the certHash Value from Step 3 in section 5.3
$certHash = "1974f986d9b8bf32f47fc2af33d6271dd905c44f"

#The second method is name of the key
#Key name generated in section 5.2
#$ContainerName = "Authenticode_TestKey"

#The third method is by using an exported certificate path
#Self Signed Certificate name generated in section 5.3
#$SelfCertName = "Authenticode Certificate"
#$SelfSignedCertificatePath = "C:\Authenticode\Authenticode
Certificate.cer"

# Certificate Services Time Stamp Server
$timestampServer = "http://timestamp.digicert.com”

# File to be Signed
$fileName = "C:\Authenticode\MyApplication.exe" signtool sign /debug /tr
$timestampServer /td sha256 /fd sha256 /a $fileName
```

2. Launch PowerShell as Administrator and run `Sign_Timestamp_Executable.ps1`. Enter the passphrase when prompted.

```
›_ PowerShell
```

```
> .\Sign_Timestamp_Executable.ps1

The following certificates were considered:
    Issued to: Authenticode Certificate

    Issued by: Authenticode Certificate

    Expires:   Fri Mar 17 09:57:01 2023

    SHA1 hash: 1974f986d9b8bf32f47fc2af33d6271dd905c44f
After EKU filter, 1 certs were left.
After expiry filter, 1 certs were left.
After Subject Name filter, 1 certs were left. After Private Key filter, 1 certs were left. The following certificate was selected:
    Issued to: Authenticode Certificate

    Issued by: Authenticode Certificate

    Expires:   Fri Mar 17 09:57:01 2023

    SHA1 hash: 1974f986d9b8bf32f47fc2af33d6271dd905c44f
Done Adding Additional Store
```

```
›_ PowerShell
```

```
Successfully signed: C:\Authenticode\MyApplication.exe

Number of files successfully Signed: 1

Number of warnings: 0


Number of errors: 0
```

If you are using Smartcard Authentication, the prompt will go on the PIN Pad device to insert Smartcard and enter the pin. Then press OK button on the PIN Pad.

3. You can verify that your application is now signed by right clicking on it and selecting Properties. On the Digital Signatures tab (if it exists), you can view the signing certificate and timestamp.

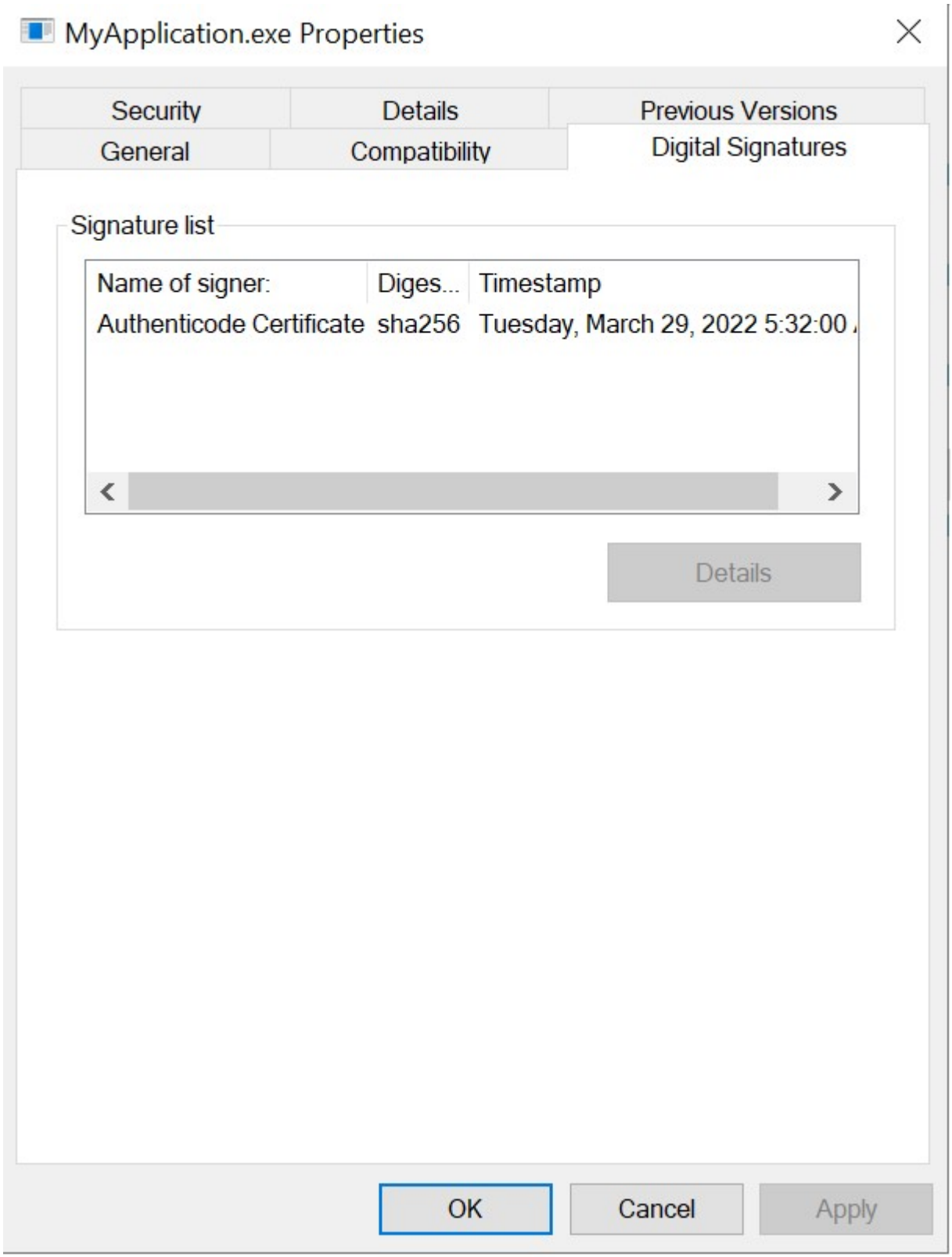

Figure 6 : Digital signature details

6 Generate the Authenticode Key using CLI

6.1 Create Certificate Request

It is necessary to create a specific certificate for the code signing purposes. This certificate is installed in the local Windows certificate store (e.g., personal store). To retrieve an official code signing certificate issued by a certification authority you have to create a certificate request (CSR) first. Normally an official certificate authority (e.g., VeriSign, Thawte, DigiCert) will create and sign a certificate based on your certificate request. If you don't need an officially signed certificate, you can also use an in-house certificate authority (e.g., Microsoft Windows Server Certification Authority).

To create a code signing certificate request you first need to create a template file .inf. You will then issue the certificate request based on this template file using Microsoft's utility certreq.exe.

Create a file called request.inf, which should include with amongst others the following information:

- The subject details must include a 2-letter country code "C" and a common name "CN" which may be your company name.
- Key algorithm and key length as required (e.g., RSA, 2048 bit key).
- Name of the Cryptographic Service Provider. For use with CryptoServer this needs to be Utimaco CryptoServer Key Storage Provider.
- You can add a KeyContainer parameter to set the key name in the CryptoServer. This helps to distinguish several code signing keys from each other. If no KeyContainer is specified a random string is generated, starting with CertReq.

`request.inf`

[Version]

```
Signature = "$Windows NT$"

[NewRequest]

Subject = "CN=YourCompany Code Signing,O=YourCompany,L=Aachen,C=DE"

KeyAlgorithm = RSA

KeyLength = 2048

Exportable = FALSE

MachineKeySet = FALSE

ProviderName = "Utimaco CryptoServer Key Storage Provider"

KeyUsage = CERT_DIGITAL_SIGNATURE_KEY_USAGE

KeyUsageProperty = NCRYPT_ALLOW_SIGNING_FLAG

HashAlgorithm = SHA256

[EnhancedKeyUsageExtension]

OID = 1.3.6.1.5.5.7.3.3 ; Code signing
```

It is important, that the ProviderName is given as Utimaco CryptoServer Key Storage Provider. This links the code signing certificate with the private key which is stored in the CryptoServer.

1. Open a command prompt.
2. Change to the directory where you have saved the request.inf file.
3. Execute the following command. If you like to review the actions of the command on the CryptoServer for debug purposes enable logging in the CNG configuration file.

```
›_ Console
```

```
C:\>certreq -new request.inf request.req

CertReq: Request Created
```

certreq creates a certificate request file request.req that can either be sent to an official certificate authority or be signed with your in-house certificate authority.

If you are using Smartcard Authentication, the prompt will go on the PIN Pad device to insert Smartcard and enter the pin. Then press OK button on the PIN Pad.

6.2 Install Code Signing Certificate

After creating a certificate request, you obtain the certificate from a certificate authority or by your own certificate authority. In the following we name that file codesign.crt. To use your code signing certificate you need to install this in your local Windows certificate store. This can be achieved by GUI or command line.

6.2.1 Command Line Procedure

If you have created the certificate request on the same computer using certreq Windows has “remembered” this open request. To install the certificate now simply run:

```
›_ Console
```

```
C:\>certreq -accept -user codesign.crt
Installed Certificate:
  Serial Number: 2d0000002ba56ec00d8b611e4a00000000002b
  Subject: CN=YourCompany Code Signing, O=YourCompany, L=Aachen, C=DE
  NotBefore: 3/25/2022 8:31 AM
  NotAfter: 3/25/2023 8:31 AM
  Thumbprint: f61f71e40bcb5d14452d7edd2a034d22801fb547
```

Since Windows has also remembered that the key for this certificate was created with the Utimaco CryptoServer Key Storage Provider it has already associated the certificate with that key provider and container. Thus, you can directly move on to sign code.

However, if the request was created on another computer or if you need to reinstall the certificate an error will be shown:

```
›_ Console
```

```
C:\>certreq -accept -user codesign.crt
Certificate Request Processor: Cannot find object or property. 0x80092004
(-2146885628)
```

In that case you have to first import the certificate and then manually associate it with the key provider and container:

1. Run the following command to import the certificate:

```
›_ Console
```

```
C:\>certutil -addstore -user My codesign.crt
My "Personal"
Related Certificates:

Exact match:
Element 0:
Serial Number: 2d0000002ba56ec00d8b611e4a00000000002b
Issuer: CN=Utimaco-RootCA, DC=utimaco, DC=local
 NotBefore: 3/25/2022 8:31 AM
 NotAfter: 3/25/2023 8:31 AM
Subject: CN=YourCompany Code Signing, O=YourCompany, L=Aachen, C=DE
Non-root Certificate
Template:
1.3.6.1.4.1.311.21.8.16593323.14862581.6636168.15641503.12204691.200.1114
8576.10529166
Cert Hash(sha1): f61f71e40bcb5d14452d7edd2a034d22801fb547
CertUtil: -addstore command completed successfully.
```

2. Then run the following command to obtain the serial number. Replace “YourCompany Code Signing” with the common name (“CN” field) of your certificate.

```
›_ Console
```

```
C:\>certutil -store -user My "YourCompany Code Signing" | findstr Serial

Serial Number: f61f71e40bcb5d14452d7edd2a034d22801fb547
```

3. Use the certutil tool to link the private key on the CryptoServer with the code signing certificate:

```
›_ Console
```

```
C:\>certutil -repairstore -user My <SerialNumber>

My "Personal"

================ Certificate 0 ================

Serial Number: f61f71e40bcb5d14452d7edd2a034d22801fb547

Issuer: CN=Utimaco-RootCA, DC=utimaco, DC=local

 NotBefore: 3/25/2022 8:31 AM
```

```
›_ Console
```

```
 NotAfter: 3/25/2023 8:31 AM

Subject: CN=YourCompany Code Signing, O=YourCompany, L=Aachen, C=DE

Non-root Certificate

Template:

1.3.6.1.4.1.311.21.8.16593323.14862581.6636168.15641503.12204691.200.1114
8576.10529166

Cert Hash(sha1): f61f71e40bcb5d14452d7edd2a034d22801fb547

  Key Container = tq-e5742d68-a308-4768-969c-dc11f7c3ed63

  Unique container name: D5A46CE713A51CA294D36197C327E614   Provider =
Utimaco CryptoServer Key Storage Provider

Private key is NOT exportable

Signature test passed

CertUtil: -repairstore command completed successfully.
```

6.2.2 GUI Procedure

Using the GUI, the procedure is as follows:

1. Run certmgr.msc, either by pressing Windows-R or opening a console and entering this command.
2. Right Click on Certificates - Current User > Personal and then click on All Task > Import and follow the instruction to import the signed certificate. Verify the certificate is successfully imported in Certificates - Current User > Personal > Certificates.

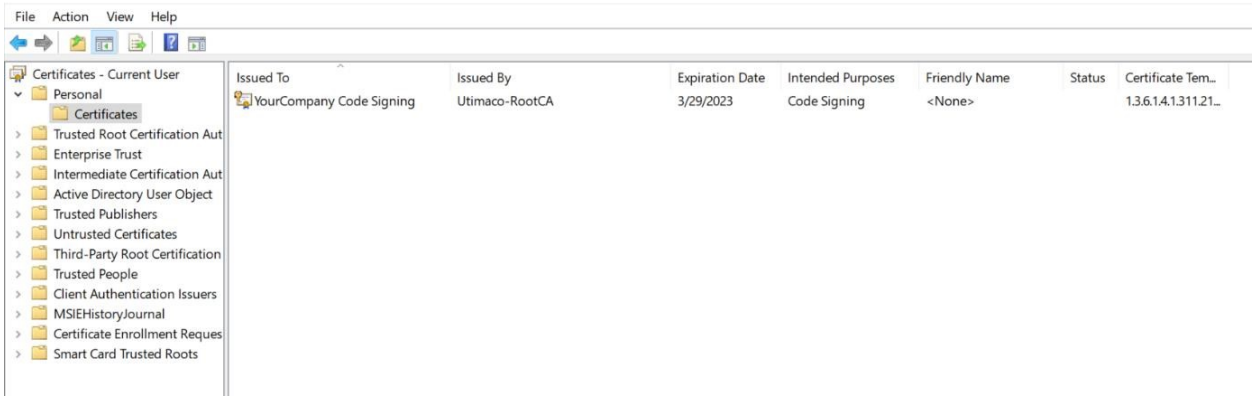

Figure 7 : Code signing certificate

3. Double click the certificate and confirm that there is a private key mapped with this certificate. Check the message at the bottom.

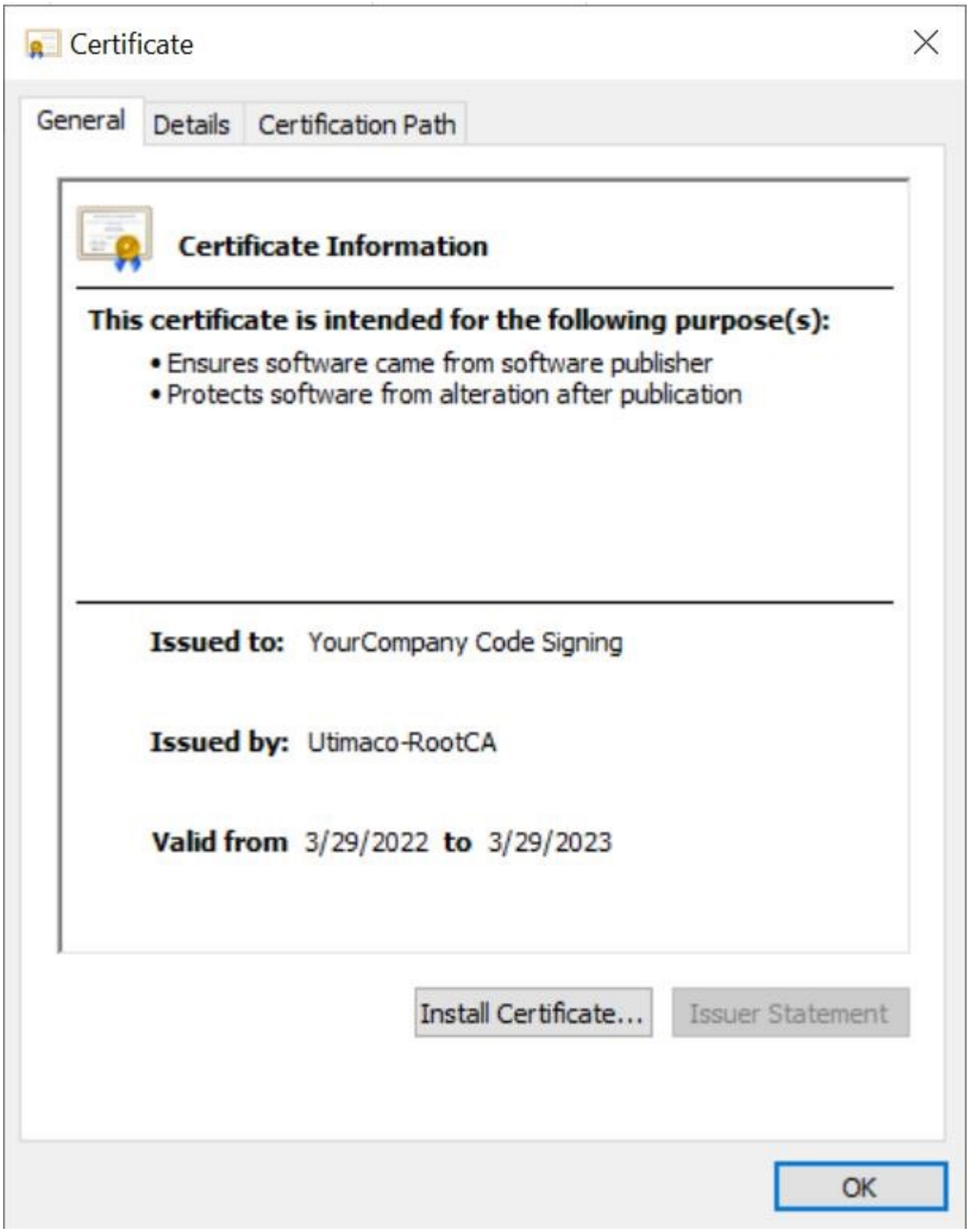

Figure 8 : Certificate properties

4. In case the private key is not mapped with private in the CryptoServer, repair the code signing certificate using the certutil repairstore utility.

- Browse to the details tab.
- Select the serial number or thumb print field.
- Copy the data.
- Use the certutil tool to link the private key on the CryptoServer with the code signing certificate. Don't forget the double quotes since the serial number or thumb print copied before usually contains spaces.

```
›_ Console
```

```
C:\>certutil -repairstore -user My <SerialNumber>

My "Personal"

================ Certificate 0 ================

Serial Number: f61f71e40bcb5d14452d7edd2a034d22801fb547

Issuer: CN=Utimaco-RootCA, DC=utimaco, DC=local

 NotBefore: 3/25/2022 8:31 AM

 NotAfter: 3/25/2023 8:31 AM

Subject: CN=YourCompany Code Signing, O=YourCompany, L=Aachen, C=DE

Non-root Certificate
```

```
›_ Console
```

```
Template:
1.3.6.1.4.1.311.21.8.16593323.14862581.6636168.15641503.12204691.200.11148576.10529166
Cert Hash(sha1): f61f71e40bcb5d14452d7edd2a034d22801fb547
  Key Container = tq-e5742d68-a308-4768-969c-dc11f7c3ed63
  Unique container name: D5A46CE713A51CA294D36197C327E614   Provider = Utimaco CryptoServer Key Storage Provider
Private key is NOT exportable
Signature test passed
CertUtil: -repairstore command completed successfully.
```

5. After the repairstore command has been successfully executed, refresh the certificate manager snap in, open the certificate and make sure you see the message "You have a private key that corresponds to this certificate".

7 Code Signing

Once the code signing certificate has been installed in the local personal Windows certificate store, it is possible to sign your executables, dynamic link libraries or cabinet files. Depending on how you have installed signtool you might have to open a developer console in order to include signtool in your local Windows search path.

Use the following basic command to sign your executable. Replace "YourCompany Code Signing" with the common name ("CN" field) of your certificate. You can also add the /fd sha256 parameter to use the more secure SHA256 digest algorithm.

›_ Console

```
C:\Autheticode>signtool sign /v /n "YourCompany Code Signing" sample.exe
The following certificate was selected:
    Issued to: YourCompany Code Signing     Issued by: Utimaco-RootCA
    Expires:   Sat Mar 25 08:31:30 2023
    SHA1 hash: F61F71E40BCB5D14452D7EDD2A034D22801FB547
Done Adding Additional Store
Successfully signed: sample.exe

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

If you are using Smartcard Authentication, the prompt will go on the PIN Pad device to insert Smartcard and enter the pin. Then press OK button on the PIN Pad.

It is advisable to also include a time stamp in your code signature. With a time stamp, the signature usually stays valid even after the expiry date of the code signing certificate. Add a timestamping authority like Verisign (http://timestamp.verisign.com/scripts/timstamp.dll) as extra parameter to signtool as shown next.

```
›_ Console
```

```
C:\Autheticode>signtool sign /v /n "YourCompany Code Signing"
/t http://timestamp.verisign.com/scripts/timstamp.dll sample.exe
The following certificate was selected:
    Issued to: YourCompany Code Signing     Issued by: Utimaco-RootCA
    Expires:   Sat Mar 25 08:31:30 2023
    SHA1 hash: F61F71E40BCB5D14452D7EDD2A034D22801FB547
Done Adding Additional Store
Successfully signed: sample.exe
Number of files successfully Signed: 1
Number of warnings: 0 Number of errors: 0
```

8 Troubleshooting

Error	Diagnosis
Error B91D0003 Pin Pad API no device found	Check the connectivity between pin pad device and remote server by using below command csadm GetCardInfo=:cs2:cjo:USB0@<ip-address> where pin pad device is connected
SmartCard LoginUser= failed:	Enter the correct pin on pin pad device
Error related to timestampServer while running the script	Check the entry for timestamp server in script
PPD installation error	Check the logs for error and reinstall it using below command: ppd – config=C:ProgramData\Utimaco\PPD\ppd.cfg – install
Cngtool command not working	Check the Environment Variable

Table 6: List of errors and their diagnoses

9 Further Information

This document forms a part of the information and support which is provided by the Utimaco IS GmbH. Additional documentation can be found on the product CD in the Documentation directory.

All CryptoServer product documentation is also available at the Utimaco IS GmbH website: https://utimaco.com/

10 References

Reference	Title/Company	Document No.
[CSADMIN]	CryptoServer – csadm Manual/Utimaco IS GmbH	2009-0003
[CSADMIN2]	CryptoServer_csadm_Manual_Systemadministrators.pdf	2009-0003
[CSLAN]	CryptoServerLAN_V5_Manual_Systemadministrators.pdf	2018-0010
[CSP-CNG]	CryptoServer_Manual_CSP_CNG.pdf	2008-0002