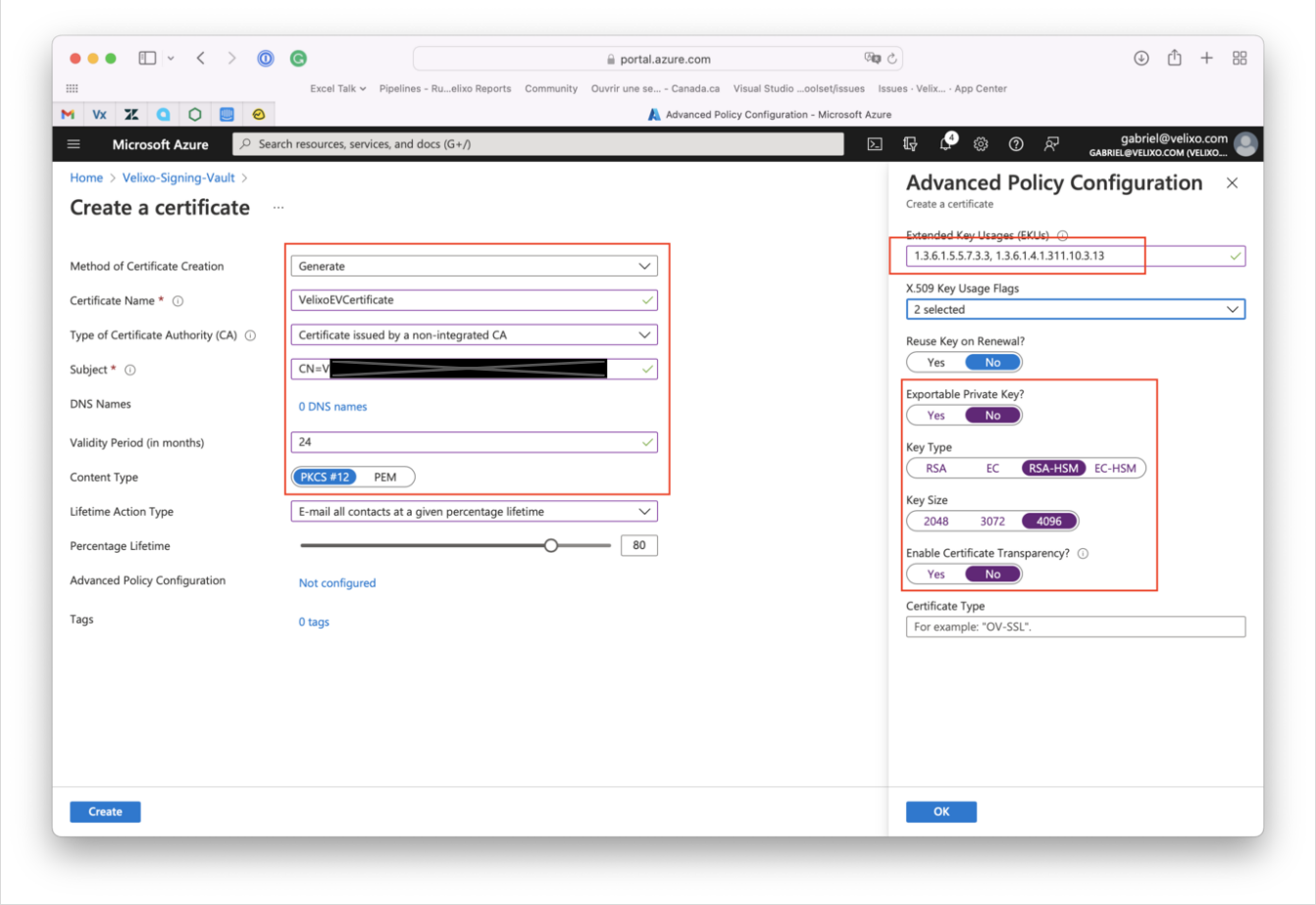
# EV Code Signing Certification Renewal Process

Gabriel Michaud · Last updated 10:07am

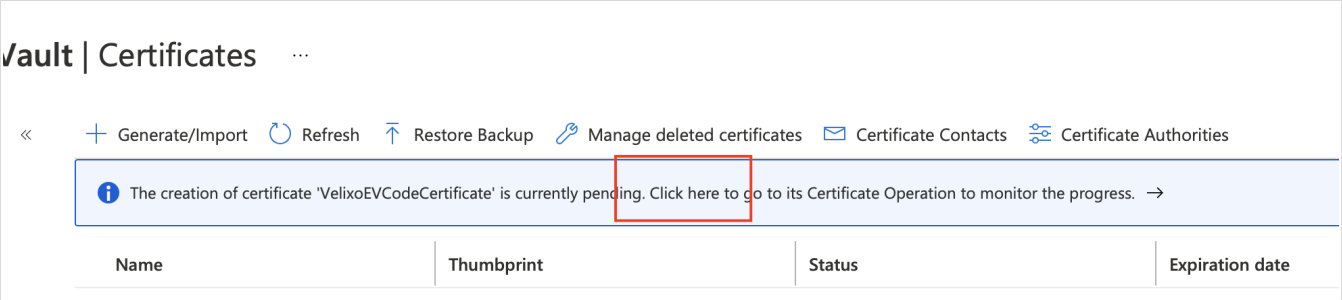
## Step 1 - Create the certificate in Azure Key Vault

Link

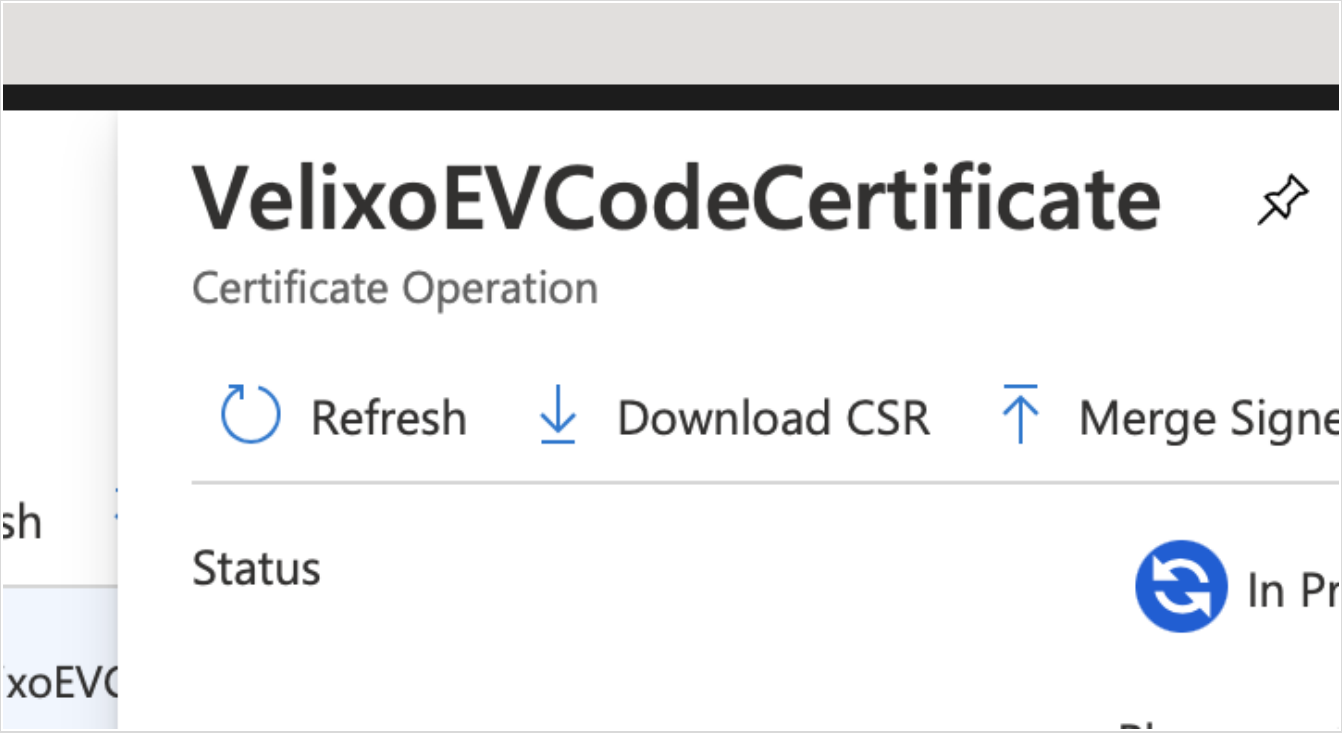



Screen Shot 2021-12-25 at 12.41.44.png

## Step 2 - Download CSR (Certificate Request File)




Screen Shot 2021-12-25 at 12.43.49.png



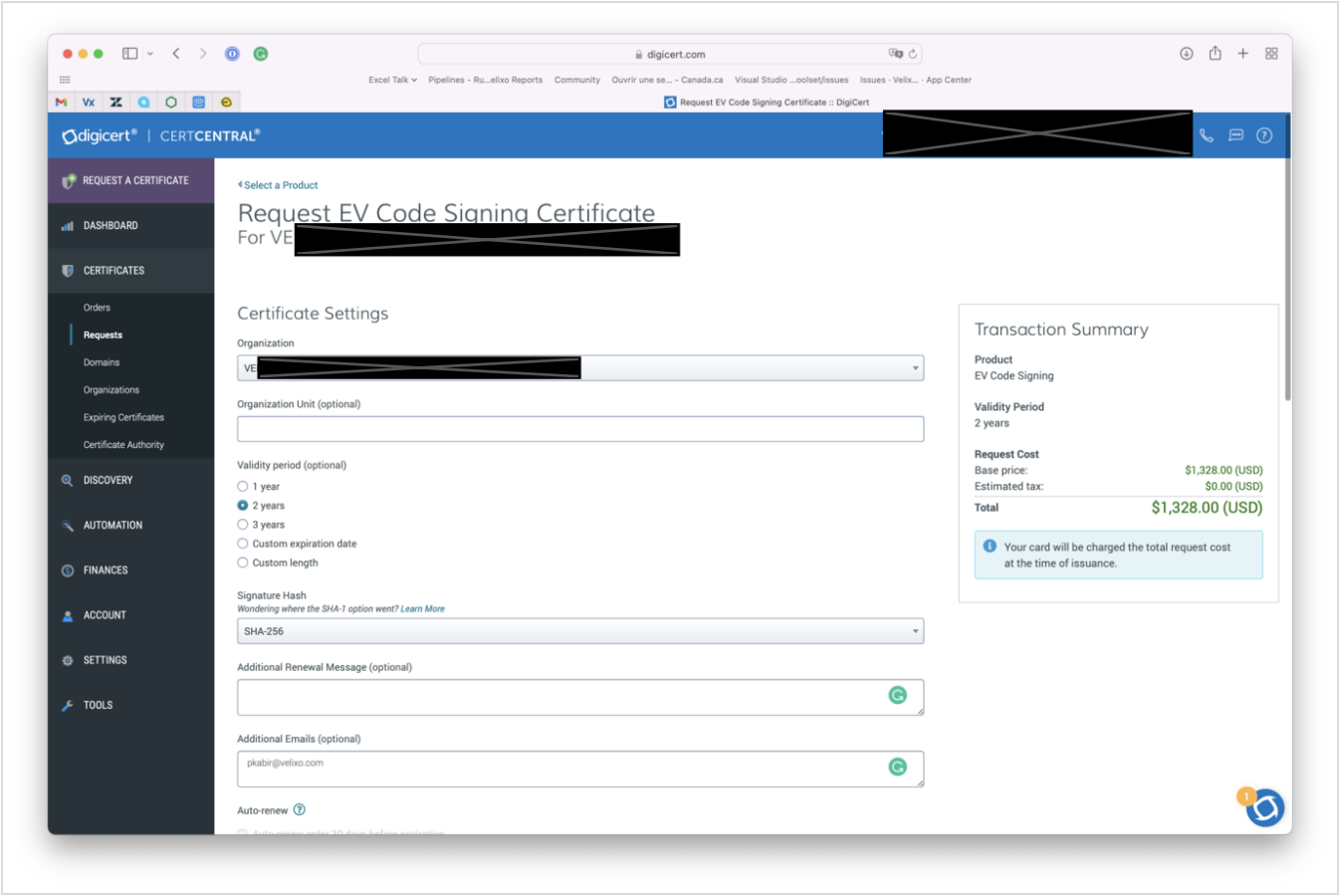

Screen Shot 2021-12-25 at 12.44.15.png

## Step 3 - Order certificate from DigiCert

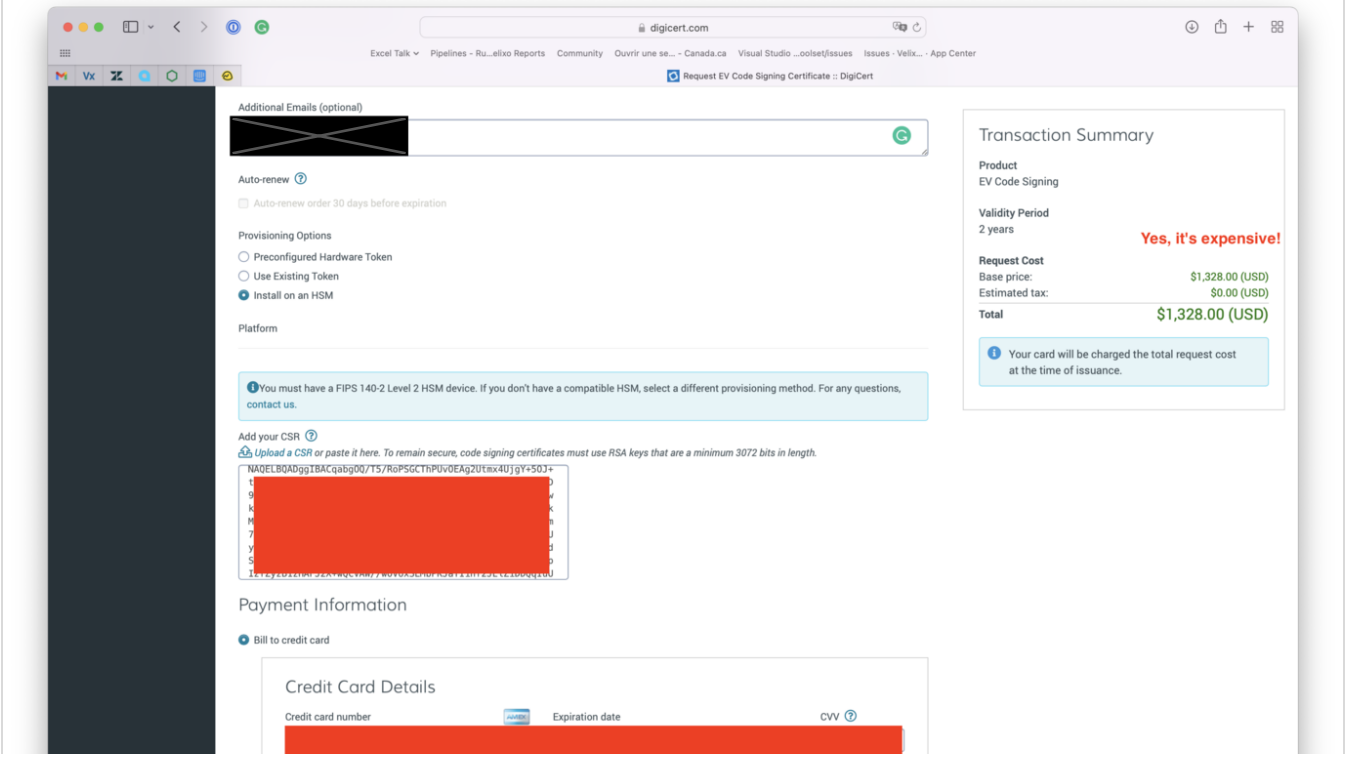
Link

Important things to do:

- Select length (this one will be 24 months, but we can do up to 36 months)
- Provisioning Options: "Install on an HSM"
- Additional E-mails: mine is here, Azure KeyVault also has some notification built-in, and I have added [redacted] as security precaution
- Include the CSR
- Billing information (in 1Password, I used the Amex)

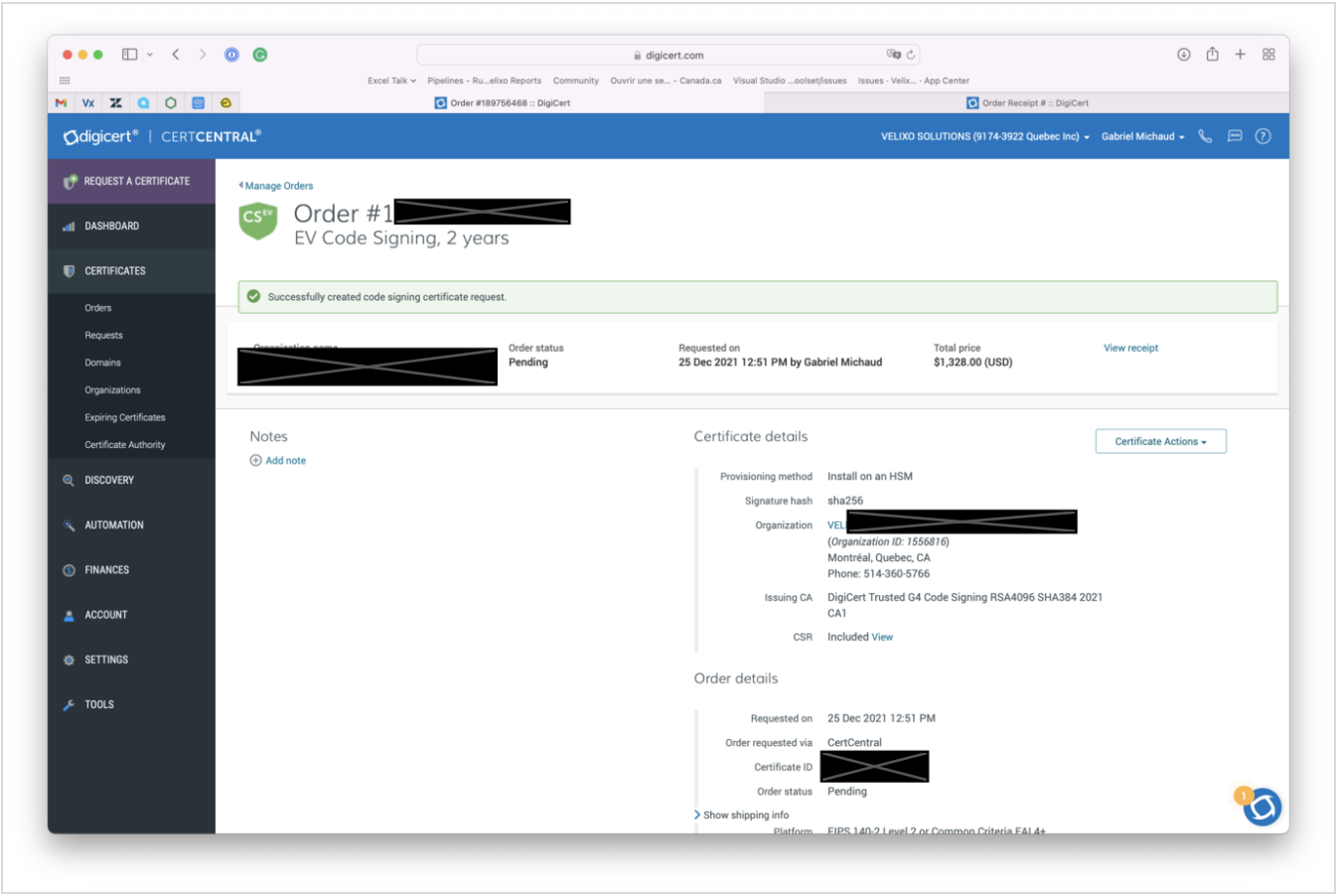



Screen Shot 2021-12-25 at 12.49.29.png




Screen Shot 2021-12-25 at 12.49.40.png

Order confirmation




Screen Shot 2021-12-25 at 12.51.24.png

## Step 4 - Validation Process

EV (Extended Validation) keys have a more stringent process to ensure that we are indeed Velixo, a real organization, and that I am authorized to request a signing key. A few hours after the purchase, they sent me this email:


image.png

My response:

[redacted]

Attached is the information available in the Quebec Business Registry showing the '[redacted]

[redacted]

The company name, address and phone number can be found online in the Yellow Pages directory: https://www.yellowpages.ca/bus/Quebec/Montreal/Solutions-

[redacted]

This was accepted and they responded with next step, phone validation:

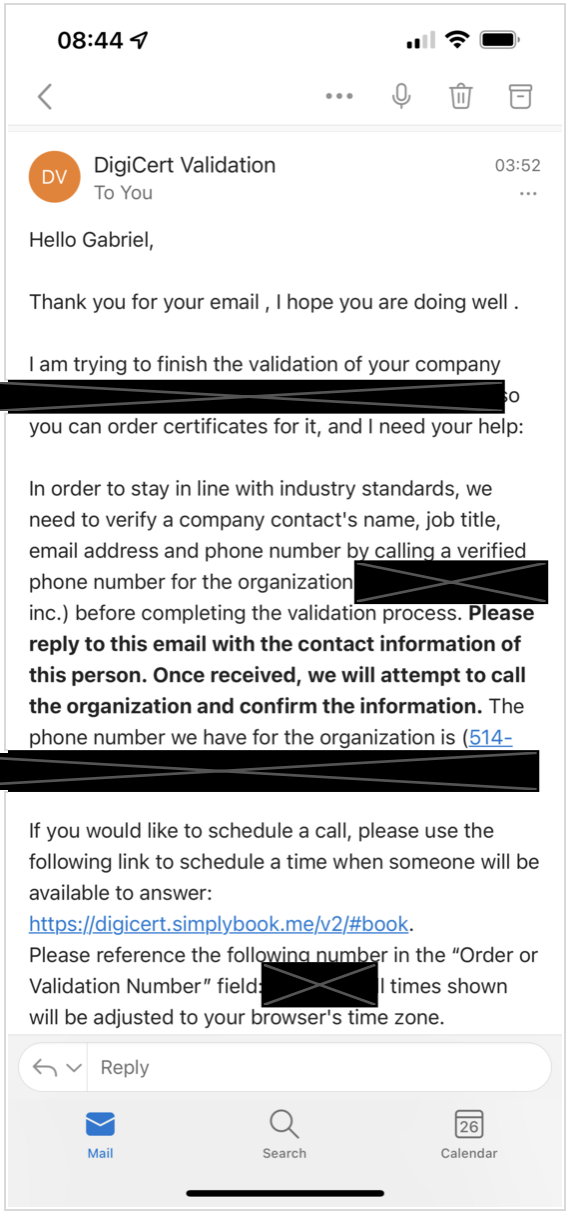



image.png

During the validation call, they asked me to confirm I was aware of the EV Certificate request and that I was authorized to sign it.

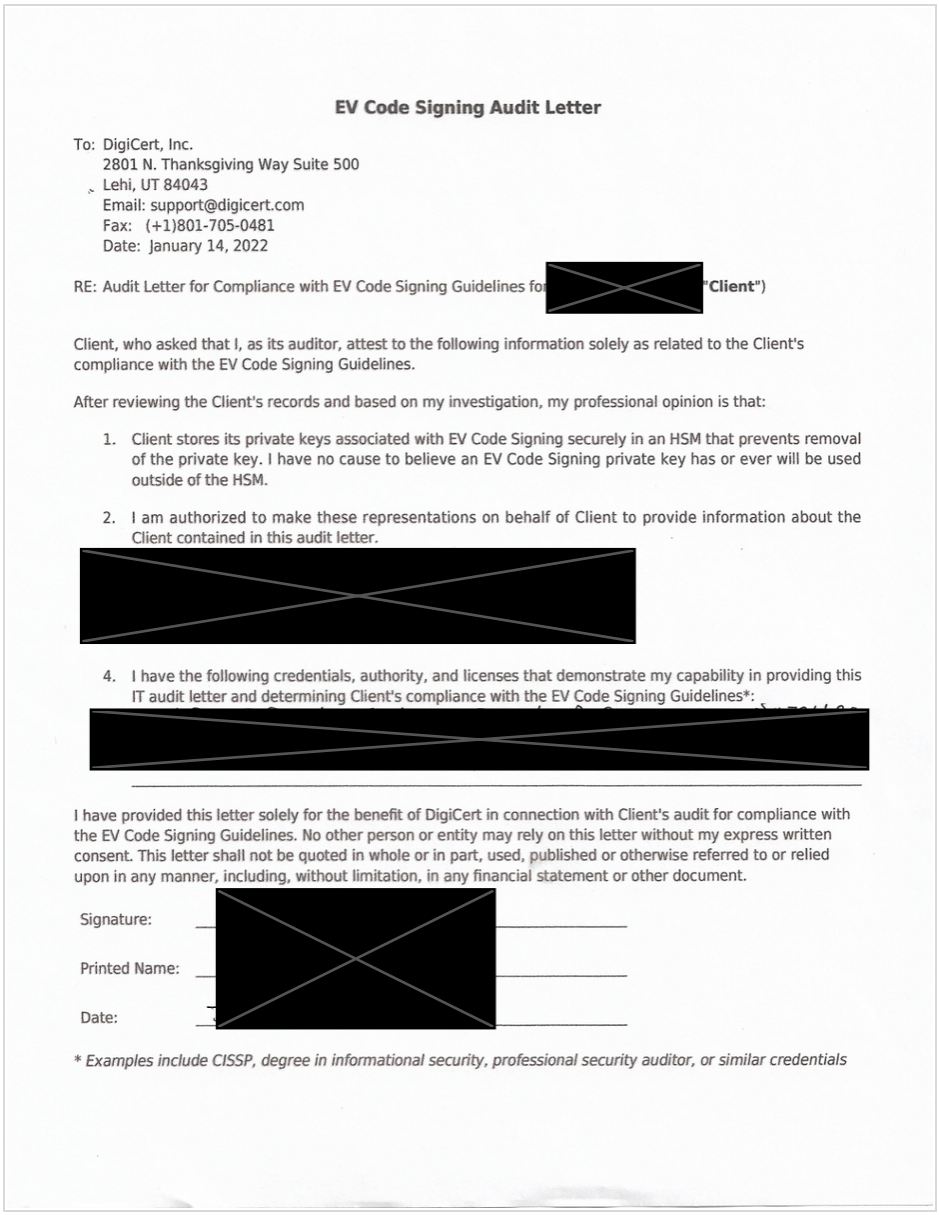
## Step 4b - Audit Letter

As part of the validation process DigiCert is asking us to obtain a signed audit letter from someone with a CISSP, degree in informational security, professional security auditor, or similar credentials, confirming that we store the private keys associated with EV Code Signing securely in an HSM that prevents removal of the private key.

A job was posted to Upwork to get the necessary consultant today:

https://www.upwork.com/jobs/~01c855a74d154e78fe

Letter signed by consultant



**EV Code Signing Audit Letter**

To: DigiCert, Inc.
2801 N. Thanksgiving Way Suite 500
Lehi, UT 84043
Email: support@digicert.com
Fax: (+1)801-705-0481
Date: January 14, 2022

RE: Audit Letter for Compliance with EV Code Signing Guidelines for [redacted] ("**Client**")

Client, who asked that I, as its auditor, attest to the following information solely as related to the Client's compliance with the EV Code Signing Guidelines.

After reviewing the Client's records and based on my investigation, my professional opinion is that:

1. Client stores its private keys associated with EV Code Signing securely in an HSM that prevents removal of the private key. I have no cause to believe an EV Code Signing private key has or ever will be used outside of the HSM.

2. I am authorized to make these representations on behalf of Client to provide information about the Client contained in this audit letter.

[redacted]

4. I have the following credentials, authority, and licenses that demonstrate my capability in providing this IT audit letter and determining Client's compliance with the EV Code Signing Guidelines*:

[redacted]

I have provided this letter solely for the benefit of DigiCert in connection with Client's audit for compliance with the EV Code Signing Guidelines. No other person or entity may rely on this letter without my express written consent. This letter shall not be quoted in whole or in part, used, published or otherwise referred to or relied upon in any manner, including, without limitation, in any financial statement or other document.

Signature: [redacted]

Printed Name:

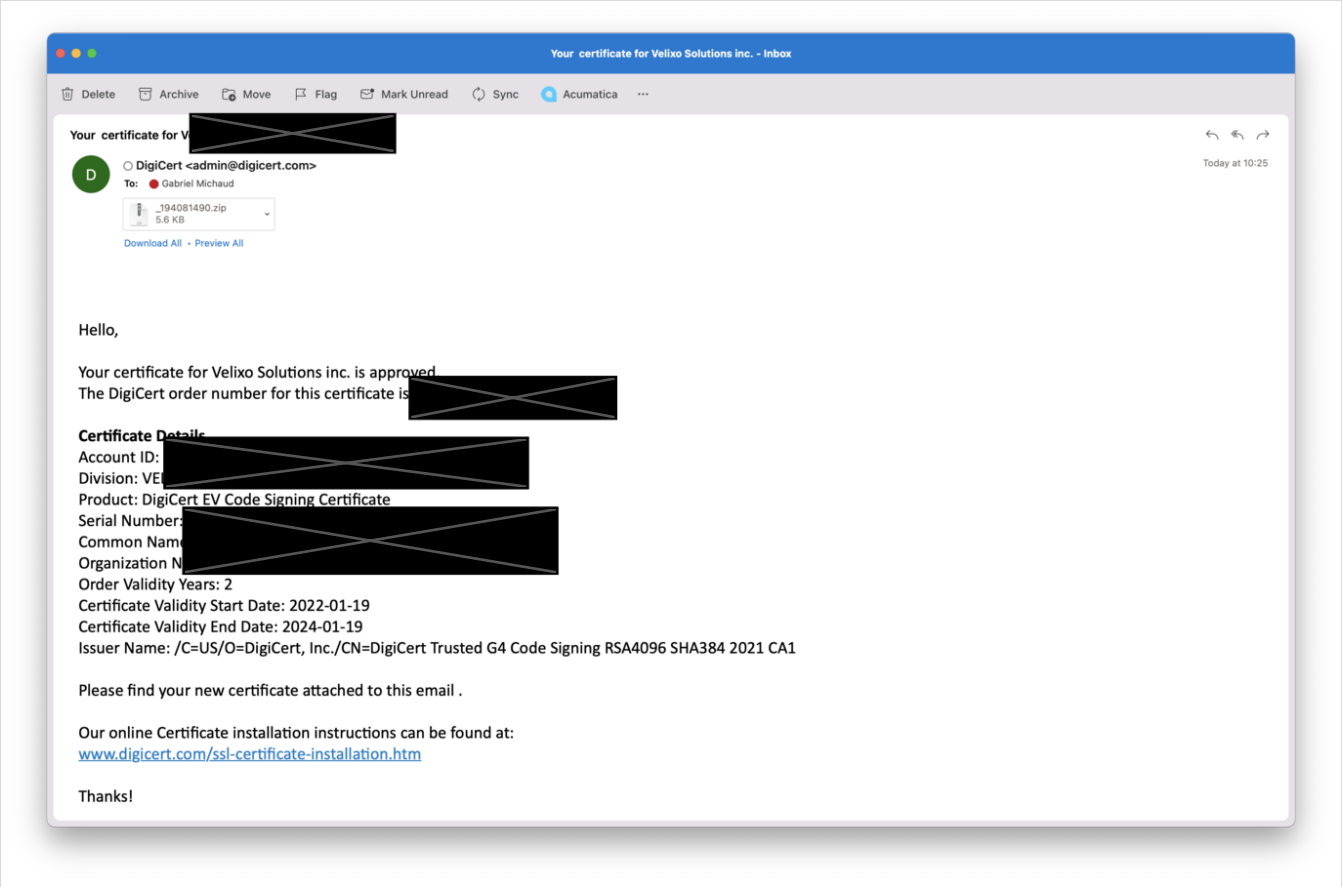
Date:

** Examples include CISSP, degree in informational security, professional security auditor, or similar credentials*

HSM_VelixoSolutionsinc.pdf

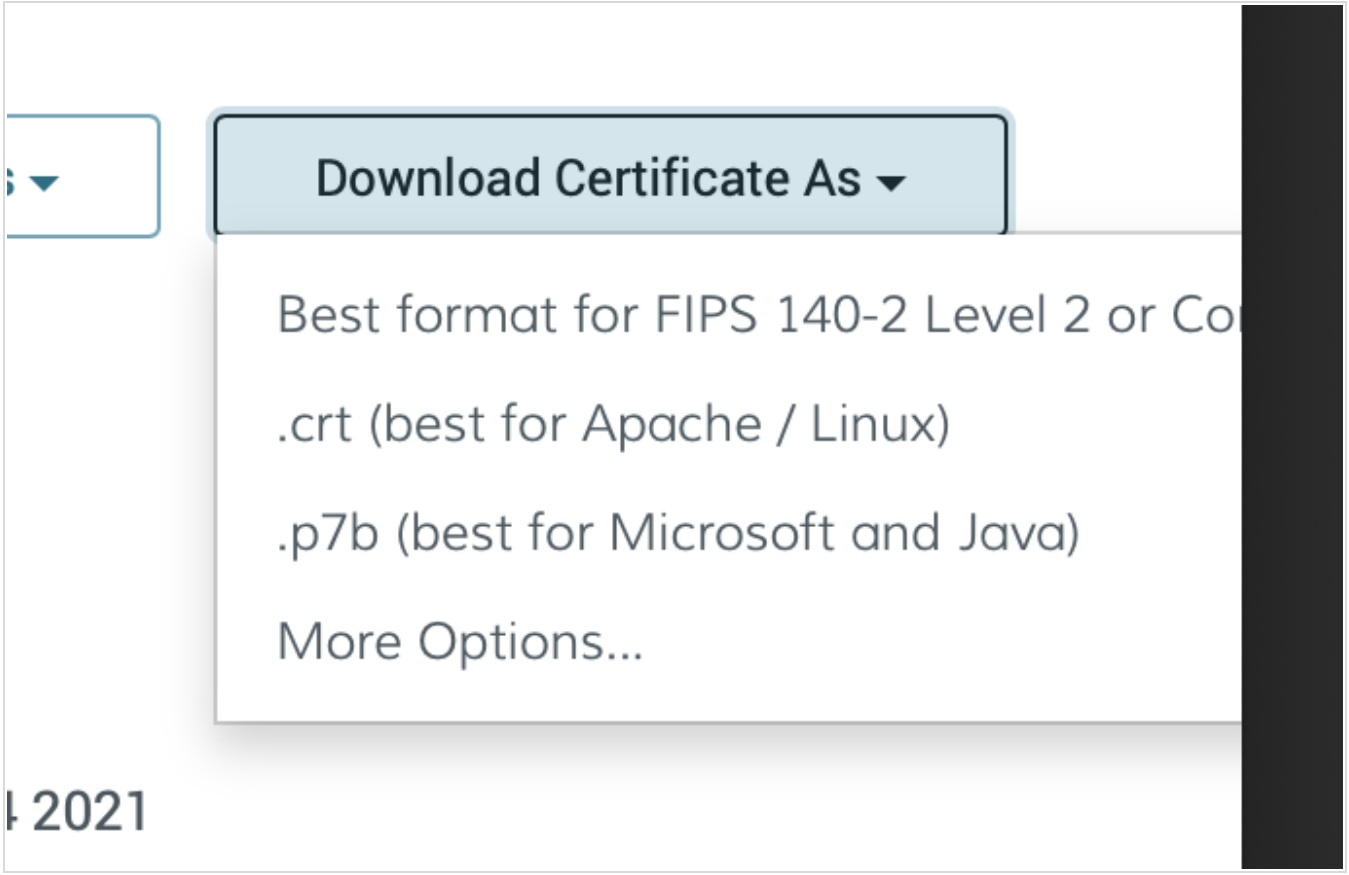
## Step 5 - Importing the key into Azure Key Vault

Once the certificate has been approved, you will receive the following e-mail message:



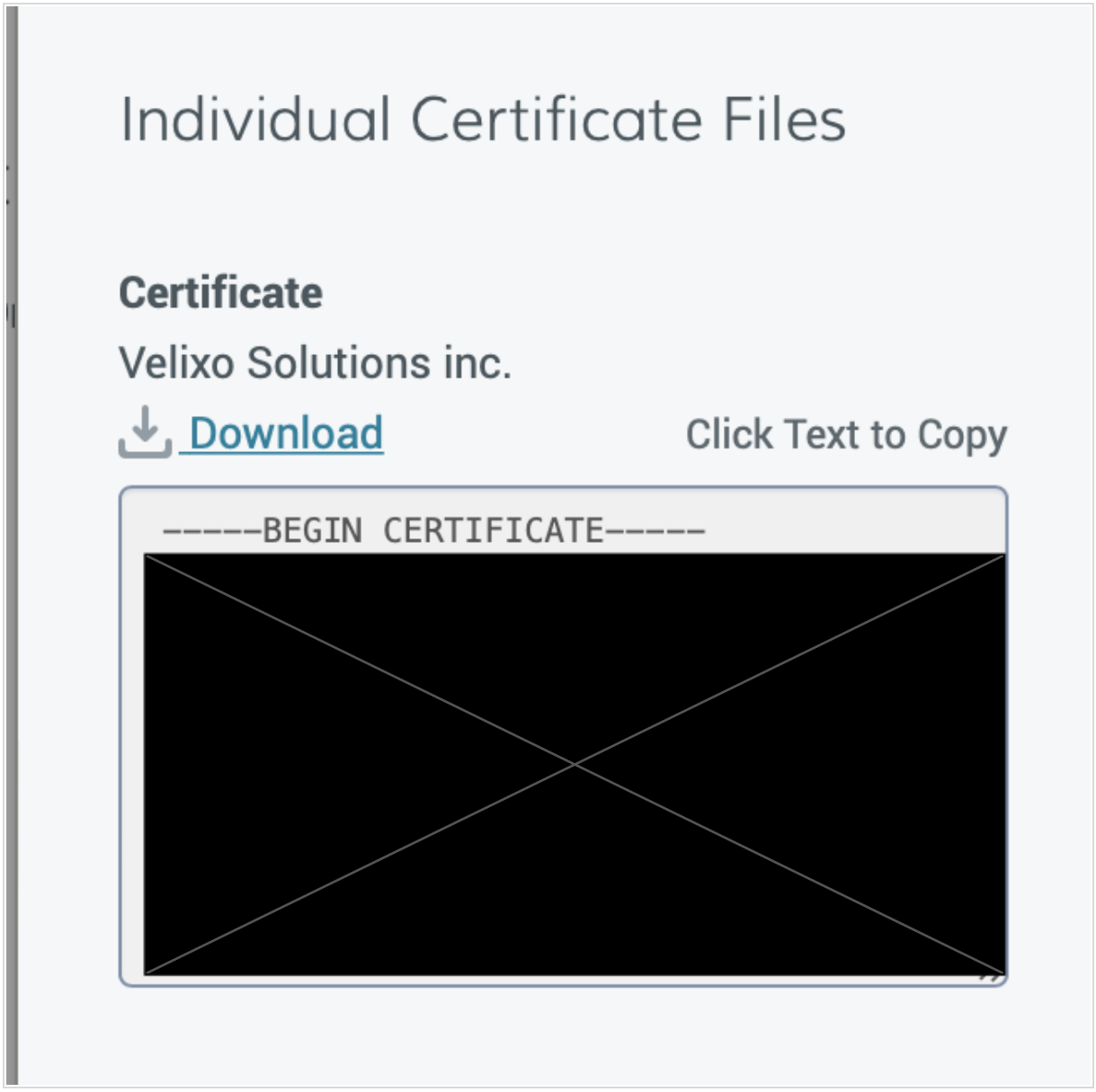

Screen Shot 2022-01-19 at 10.35.54.png

Note that the attached certificate does **NOT** work on Azure. You need to login to the DigiCert portal and find the order here. From the order page, click "Download Certificate As - More Options..."



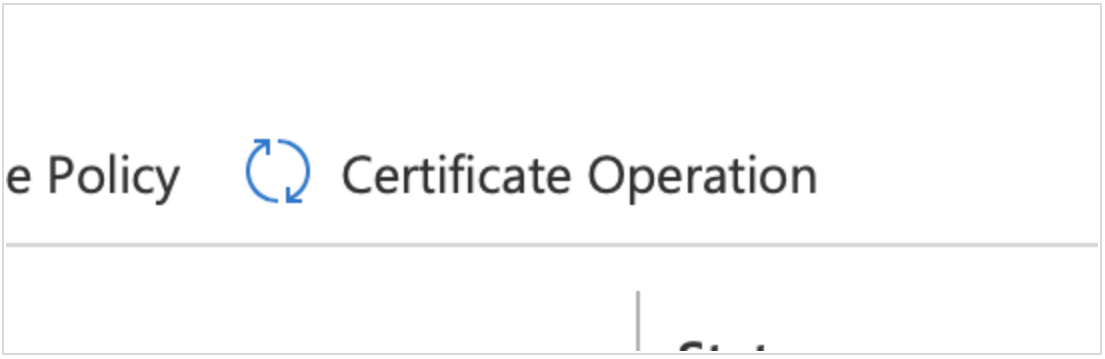

Screen Shot 2022-01-19 at 10.37.03.png

And then download the certificate file:

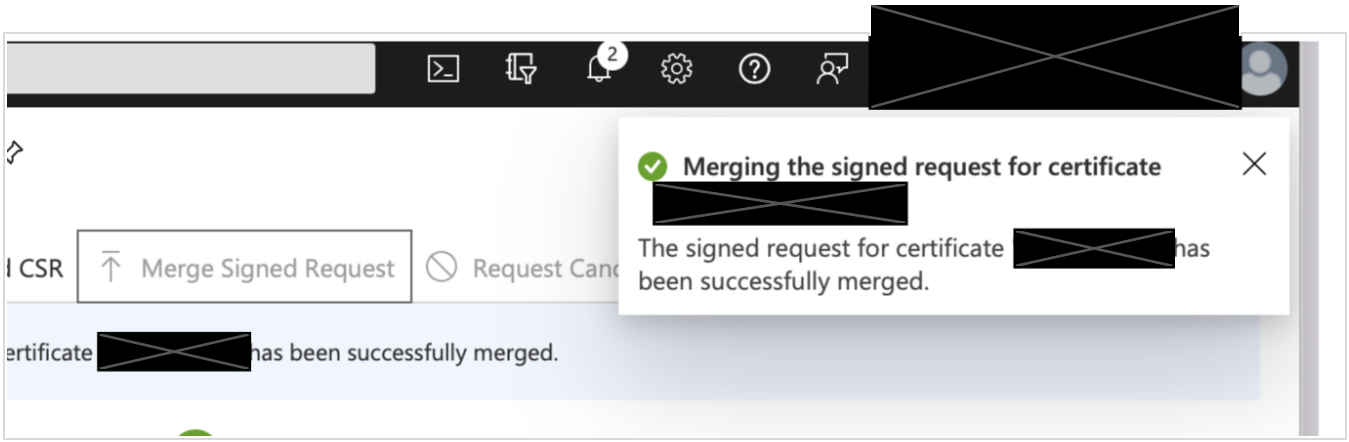



Screen Shot 2022-01-19 at 10.37.12.png

You can then go back to Azure KeyVault, under the Certificate operation button, and click "Merge Signed Request"




Screen Shot 2022-01-19 at 10.38.07.png




Screen Shot 2022-01-19 at 10.59.29.png

(note: it's greyed out in this screenshot because I already completed it)

## Step 6 - Modifying build script and pipeline to use the new key

(The following was adapte from https://stackoverflow.com/a/57934787/624448)

Once an EV certificate is in Azure Key Vault, it isn't coming out in any usual fashion. You must call it from Pipelines using the excellent Azure Sign Tool.

To get the permissions to access this certificate externally you can follow this page but **beware** it misses a step. So read that document first, then these summarized steps, to get the Key Vault set up:

1. Open the Azure portal, go to the Azure Active Directory area, and create an App registration: put in a memorable name, ignore the Redirect URI, and save it.
2. Go to your specific Key Vault, then Access control (IAM), then Add role assignment. Type the name of the app you just created into the select input box. Also choose a Role, I suggest Reader and then save.

3. **The Missing Part:** Still in the Key Vault, click the Access policies menu item. Click Add Access Policy and add your application. The Certificate Permissions need to have the Get ticked. And the Key Permissions, despite the fact that you may not have any keys at all in this vault, need to have Get and Sign. You would have thought these two would be in the certificate perms...
4. Go back to the application you just created. Select the Certificates & secrets, and either choose to upload a certificate (a new one purely for accessing the Key Vault remotely) or create a client secret. If the latter, keep a copy of the password, you won't see it again!
5. In the Overview section of the app will be the Application (client) ID. This, and the password or certificate, is what will be fed to the Azure Sign Tool later on in a Pipelines task.

Handling the actual code signing from Azure requires a number of steps. The following applies to Microsoft hosted agents, although similar issues will affect any private agents that you have.

1. The Azure Sign Tool needs the .NET Core SDK to be installed, but a version that's at least version 2.x, and since the **latest** .NET Core SDK is always used, this means as long as the version of Windows is current enough, you don't need to install it yourself. And you can see which version of the SDK is shipped with which Windows agent.
2. The current Hosted OS version in Azure Pipelines, also called Default Hosted, is, at the time of writing, Windows Server 2012 R2. Which isn't up to date enough. Installing a newer .NET Core SDK to overcome this is a time drag on every build, and although the installation works, calling the Azure Sign Tool may not work. It seems to be finding only older versions of the SDK, and throws this error: Unable to find an entry point named 'SignerSignEx3' in DLL 'mssign32'.
3. So the easiest thing to do is change your build to use a later OS image. Windows 2019 works like a charm. And there is no need to install any version of .NET Core.
4. Then create a command line task to install the Azure Sign Tool. You can use a .NET Core CLI task as well, but there is no need. In the task, type this:
5. ```
   set DOTNET_SKIP_FIRST_TIME_EXPERIENCE=true
   dotnet tool install --global AzureSignTool --version 2.0.17
   ```
6. Naturally using whichever version that you want.
7. The DOTNET_SKIP_FIRST_TIME_EXPERIENCE environment variable isn't strictly necessary, but setting it speeds things up quite a bit (see here for an explanation).
8. Finally, create another command line task and type in the Azure Sign Tool command that you wish to run with. On Windows this would be something like below, note with ^ not
   ```

/as a line continuation marker. Naturally, see here for more parameter information:

9.
```
AzureSignTool.exe sign -du "MY-URL" ^
  -kvu https://MY-VAULT-NAME.vault.azure.net ^
  -kvi CLIENT-ID-BIG-GUID ^
  -kvs CLIENT-PASSWORD ^
  -kvc MY-CERTIFICATE-NAME ^
  -tr http://timestamp.digicert.com ^
  -v ^
  $(System.DefaultWorkingDirectory)/Path/To/My/Setup/Exe
```

And in theory, you should have success! The output of the sign tool is rather good, and usually nails where the problem is.

---