[How do I create a self-signed certificate for code signing on Windows?](https://stackoverflow.com/questions/84847/how-do-i-create-a-self-signed-certificate-for-code-signing-on-windows)

<https://stackoverflow.com/questions/84847/how-do-i-create-a-self-signed-certificate-for-code-signing-on-windows>

<https://msdn.microsoft.com/en-us/library/windows/desktop/aa386968(v=vs.85).aspx>

<https://blogs.technet.microsoft.com/sbs/2008/05/08/installing-a-self-signed-certificate-as-a-trusted-root-ca-in-windows-vista/>

## Updated Answer

If you are using the following Windows versions or later: Windows Server 2012, Windows Server 2012 R2, or Windows 8.1 then [MakeCert is now deprecated](https://msdn.microsoft.com/en-us/library/windows/desktop/aa386968(v=vs.85).aspx), and Microsoft recommends using [the PowerShell Cmdlet **New-SelfSignedCertificate**](https://technet.microsoft.com/library/hh848633).

If you're using an older version such as Windows 7, you'll need to stick with MakeCert or another solution. Some people [suggest](https://www.reddit.com/r/PowerShell/comments/3190yr/powershell_40_but_no_newselfsignedcertificate/) the [Public Key Infrastructure Powershell (PSPKI) Module](https://pspki.codeplex.com/).

## Original Answer

While you can create a self-signed code-signing certificate (SPC - [Software Publisher Certificate](http://msdn.microsoft.com/en-us/library/8s9b9yaz.aspx)) in one go, I prefer to do the following:

### Creating a self-signed certificate authority (CA)

makecert -r -pe -n "CN=My CA" -ss CA -sr CurrentUser ^

-a sha256 -cy authority -sky signature -sv MyCA.pvk MyCA.cer

(^ = allow batch command-line to wrap line)

This creates a self-signed (-r) certificate, with an exportable private key (-pe). It's named "My CA", and should be put in the CA store for the current user. We're using the [SHA-256](http://en.wikipedia.org/wiki/SHA-2) algorithm. The key is meant for signing (-sky).

The private key should be stored in the MyCA.pvk file, and the certificate in the MyCA.cer file.

### Importing the CA certificate

Because there's no point in having a CA certificate if you don't trust it, you'll need to import it into the Windows certificate store. You can use the Certificates MMC snapin, but from the command line:

certutil -user -addstore Root MyCA.cer

### Creating a code-signing certificate (SPC)

makecert -pe -n "CN=My SPC" -a sha256 -cy end ^

-sky signature ^

-ic MyCA.cer -iv MyCA.pvk ^

-sv MySPC.pvk MySPC.cer

It is pretty much the same as above, but we're providing an issuer key and certificate (the -ic and -iv switches).

We'll also want to convert the certificate and key into a PFX file:

pvk2pfx -pvk MySPC.pvk -spc MySPC.cer -pfx MySPC.pfx

If you want to protect the PFX file, add the -po switch, otherwise PVK2PFX creates a PFX file with no passphrase.

### Using the certificate for signing code

signtool sign /v /f MySPC.pfx ^

/t http://timestamp.url MyExecutable.exe

([See why timestamps may matter](https://stackoverflow.com/a/4417466/57611))

If you import the PFX file into the certificate store (you can use PVKIMPRT or the MMC snapin), you can sign code as follows:

signtool sign /v /n "Me" /s SPC ^

/t http://timestamp.url MyExecutable.exe

Some possible timestamp URLs for signtool /t are:

* http://timestamp.verisign.com/scripts/timstamp.dll
* http://timestamp.globalsign.com/scripts/timstamp.dll
* http://timestamp.comodoca.com/authenticode

### Full Microsoft documentation

* [signtool](http://msdn.microsoft.com/en-us/library/8s9b9yaz.aspx)
* [makecert](http://msdn.microsoft.com/en-us/library/bfsktky3.aspx)
* [pvk2pfx](http://msdn.microsoft.com/en-us/library/windows/hardware/ff550672(v=vs.85).aspx)

### Downloads

For those who are not .NET developers, you will need a copy of the Windows SDK and .NET framework. A current link is available here: [SDK & .NET](http://msdn.microsoft.com/en-us/windowsserver/bb980924.aspx) (which installs makecert in C:\Program Files\Microsoft SDKs\Windows\v7.1). Your mileage may vary.

MakeCert is available from the Visual Studio Command Prompt. Visual Studio 2015 does have it, and it can be launched from the Start Menu in Windows 7 under "Developer Command Prompt for VS 2015" or "VS2015 x64 Native Tools Command Prompt" (probably all of them in the same folder).

EX:

makecert -r -pe -n "CN=Yash Tech" -ss CA -sr CurrentUser ^ -a sha256 -cy authority -sky signature -sv YashCA.pvk YashCA.cer

certutil -user -addstore Root "E:\Inno Compiler Output File\Yash Hyd.p7b"

makecert -pe -n "CN=Yash SPC" -a sha256 -cy end ^ -sky signature ^ -ic YashCA.cer -iv YashCA.pvk ^ -sv YashSPC.pvk YashSPC.cer

pvk2pfx -pvk YashSPC.pvk -spc YashSPC.cer -pfx YashSPC.pfx

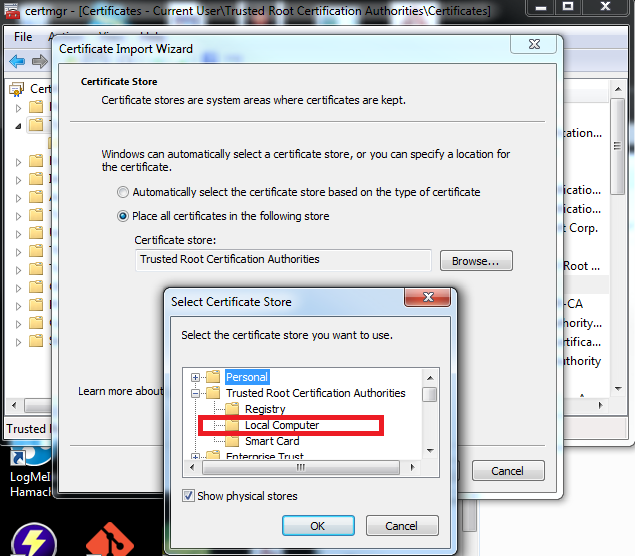
signtool sign /v /f YashSPC.pfx ^ /t <http://timestamp.verisign.com/scripts/timstamp.dll> "E:\Inno Compiler Output File\MPTA.exe"

signtool sign /v /n "Yash" /s SPC ^ /t <http://timestamp.url> MyExecutable.exe

Thanks to [the link posted by harrymc](https://superuser.com/a/145398/53265), I spent almost a day investigating this problem and figured out it was Windows Server 2008 default domain policy.

I suspect this problem only applies to Windows PC in a domain network environment. The default domain policy doesn't allow user to install additional certificate to Trusted Root Certification Authorities, but the worst thing is if you tried Windows 7 will still say "Import Successfull" anyway.

If you want to check whether your domain policy allow you to install certificate to Trusted Root Certification Authority, when importing the cert via certmgr.msc manually select the store and tick 'Show physical stores'. You should be able to place the cert into **Trusted Root Certification Authorities\Local Computer**



If you can't see above, then it has to be enabled via group policy editor on your Windows Server Domain Controller (client PC restart is required for it to take effect):

