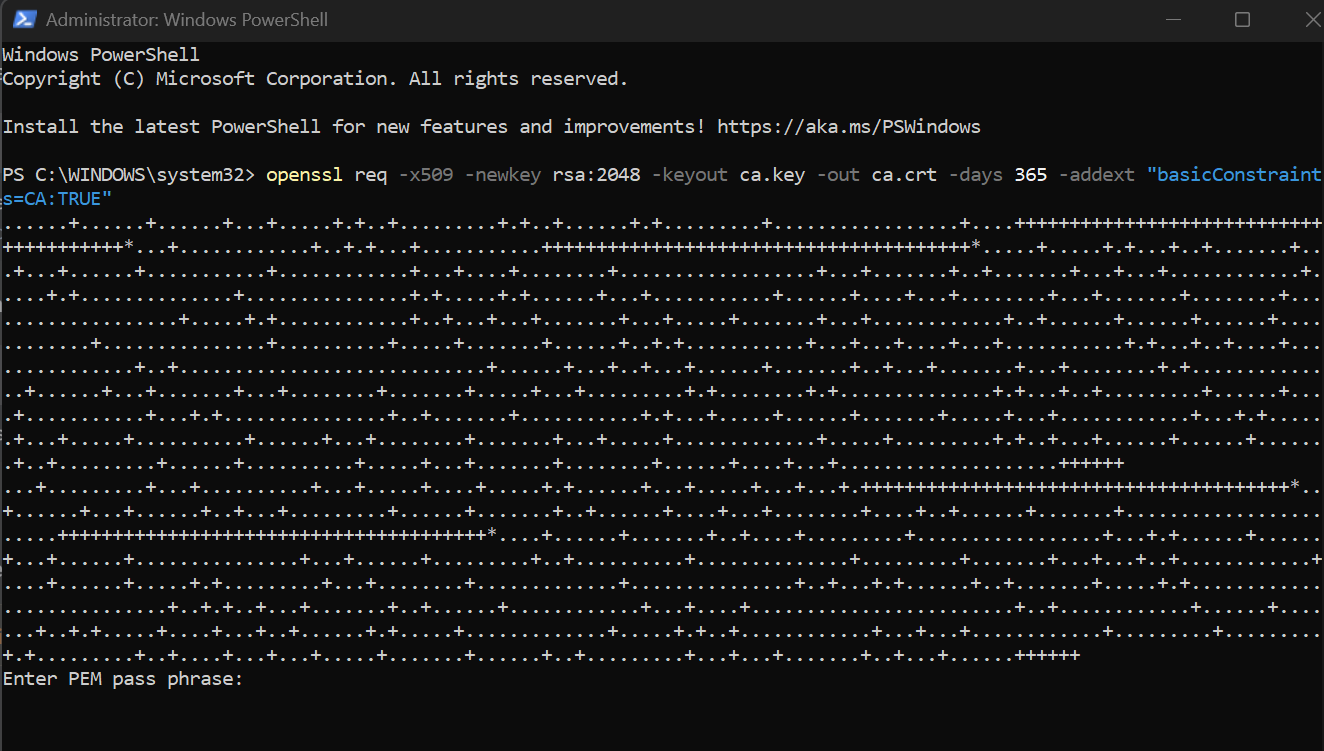
**Krok 1: Generowanie własnego Urzędu Certyfikacji (CA)**

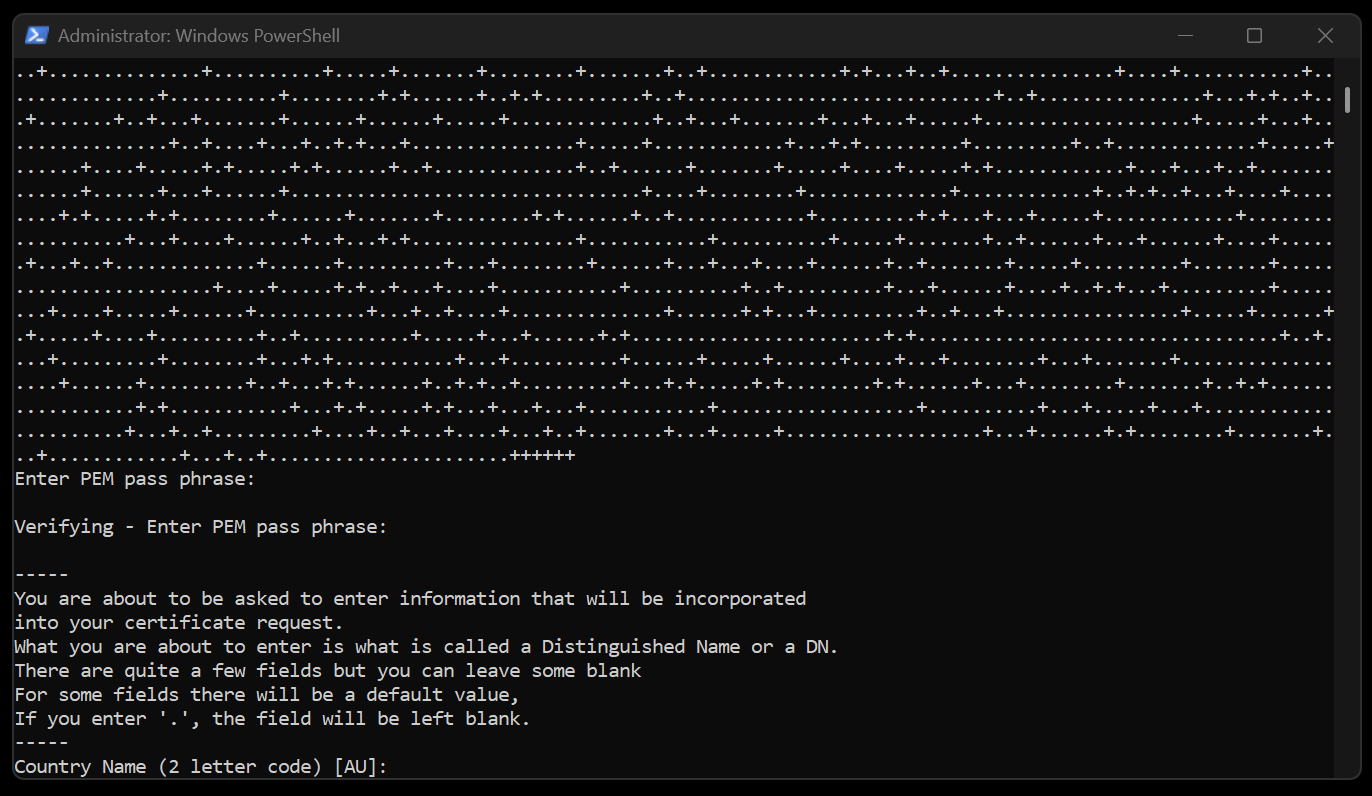
1. Przygotowanie

* openSSL działający w PowerShell
* PowerShell otwarty jako administrator

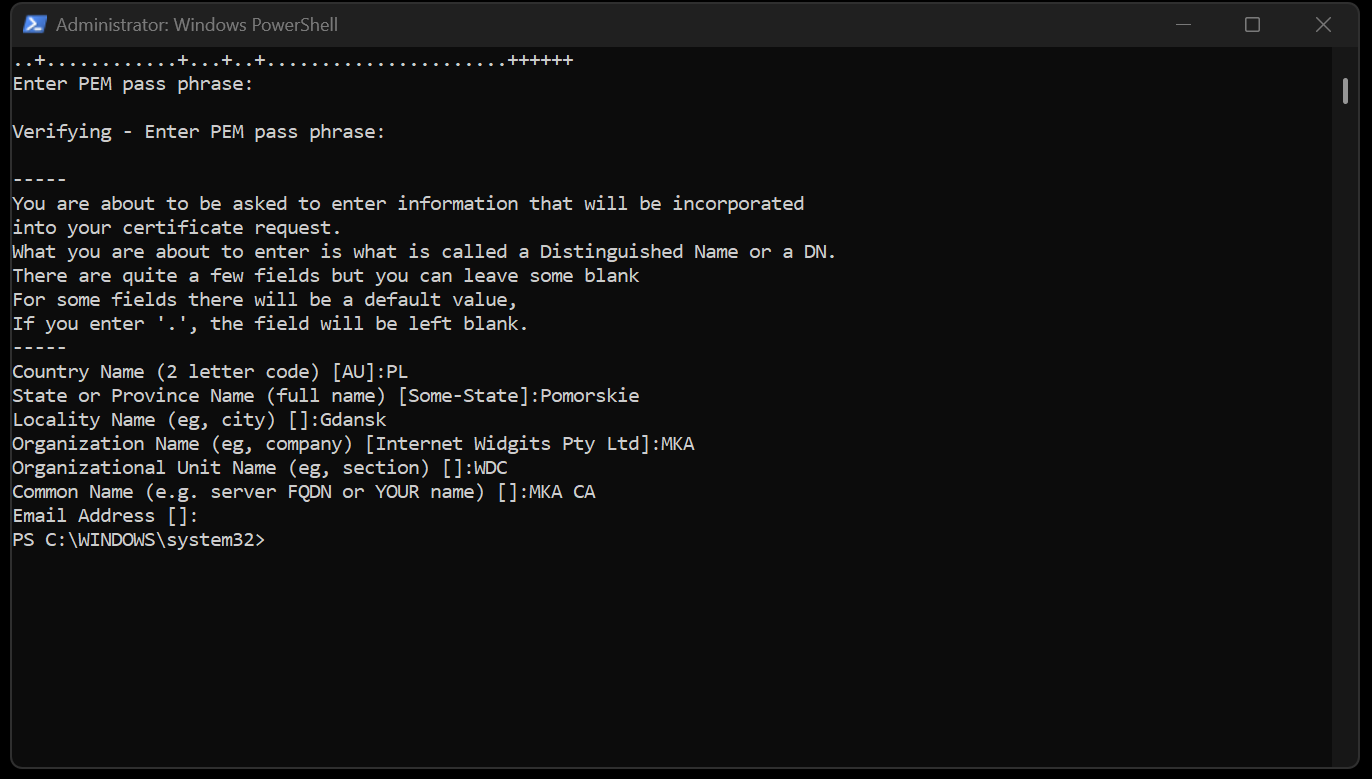
2. Generowanie klucza i certyfikatu CA

2.1 **PS:** openssl req -x509 -newkey rsa:2048 -keyout ca.key -out ca.crt -days 365 -addext "basicConstraints=CA:TRUE"

^ Ustawiam Haslo: silnehaslo



^ zaakceptowal haslo

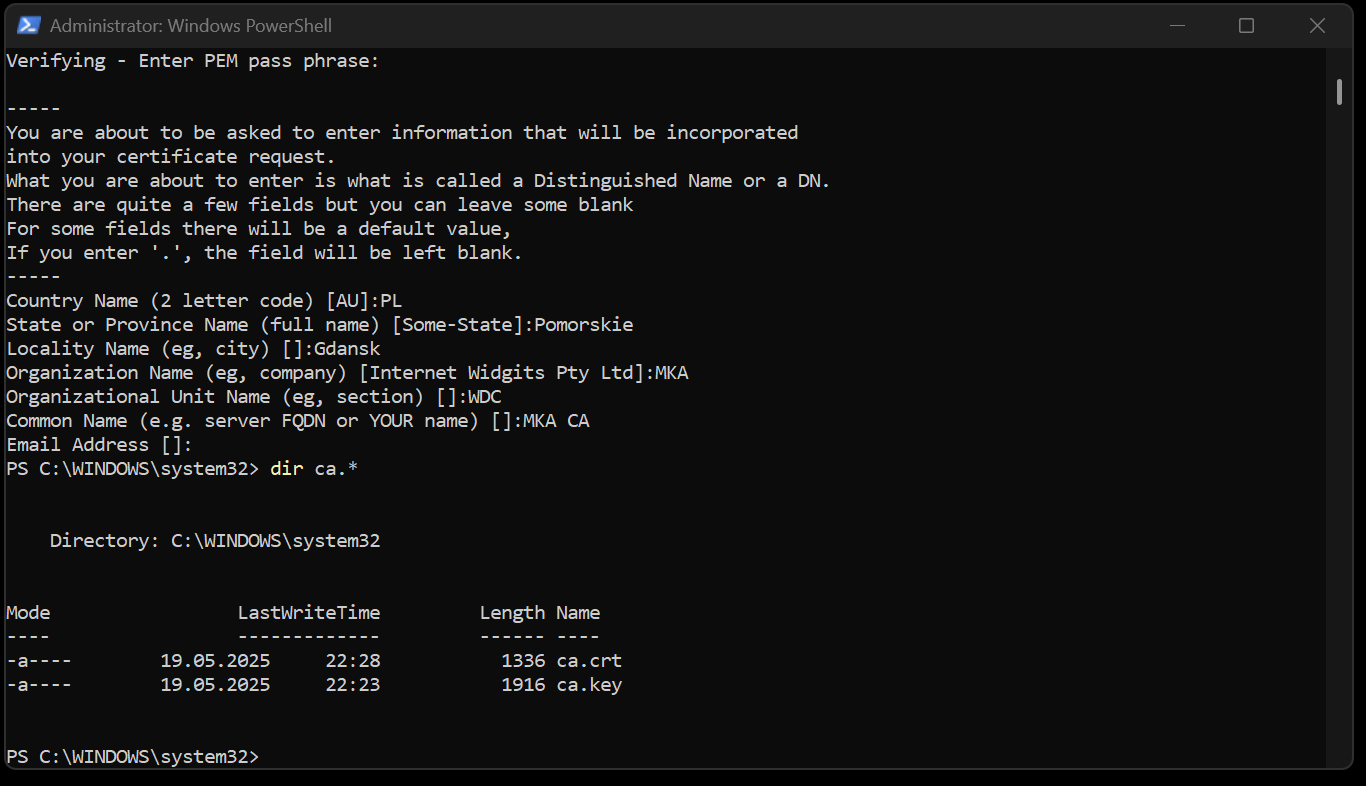


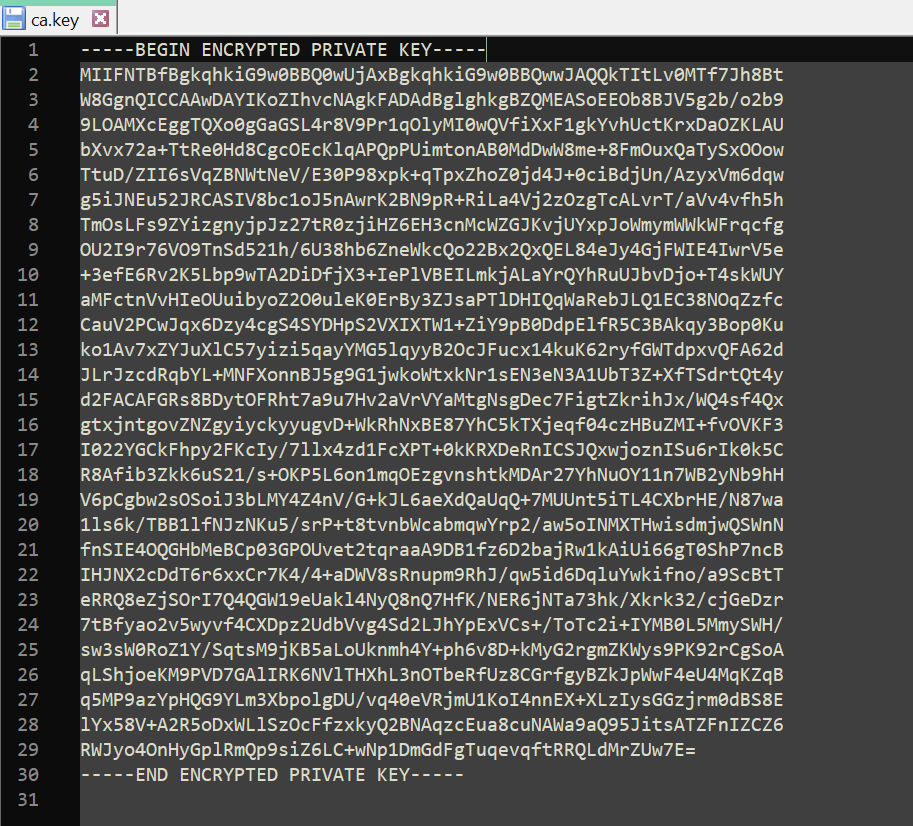
^ wypelnione inforamacje

2.2 **PS:** dir ca.\*

Powstaly:

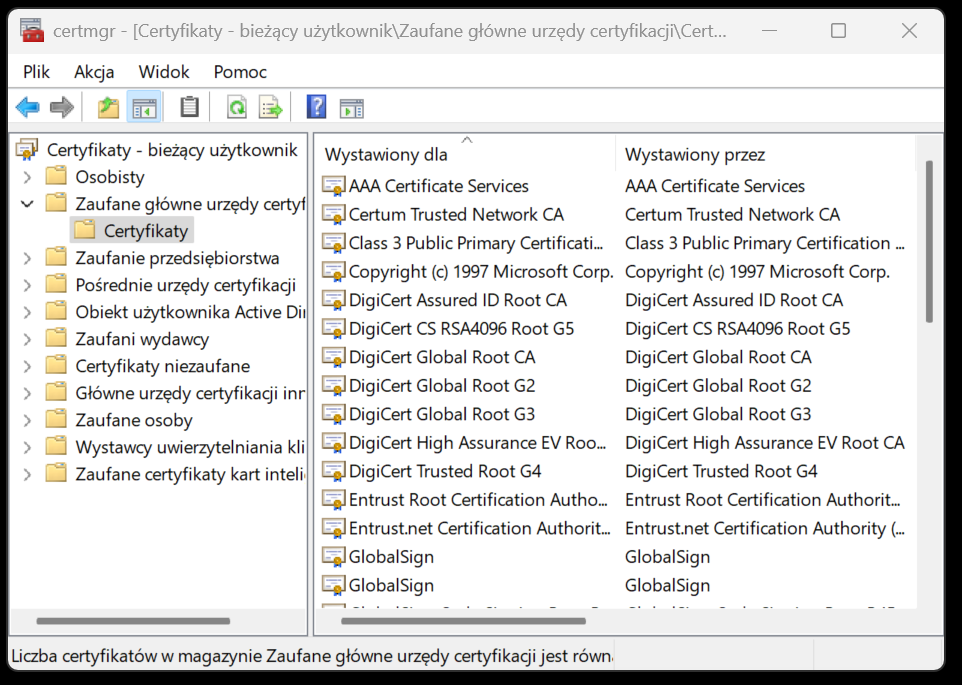
* ca.key - klucz CA (chronić jak hasło!)
* ca.crt - certyfikat CA do dystrybucji





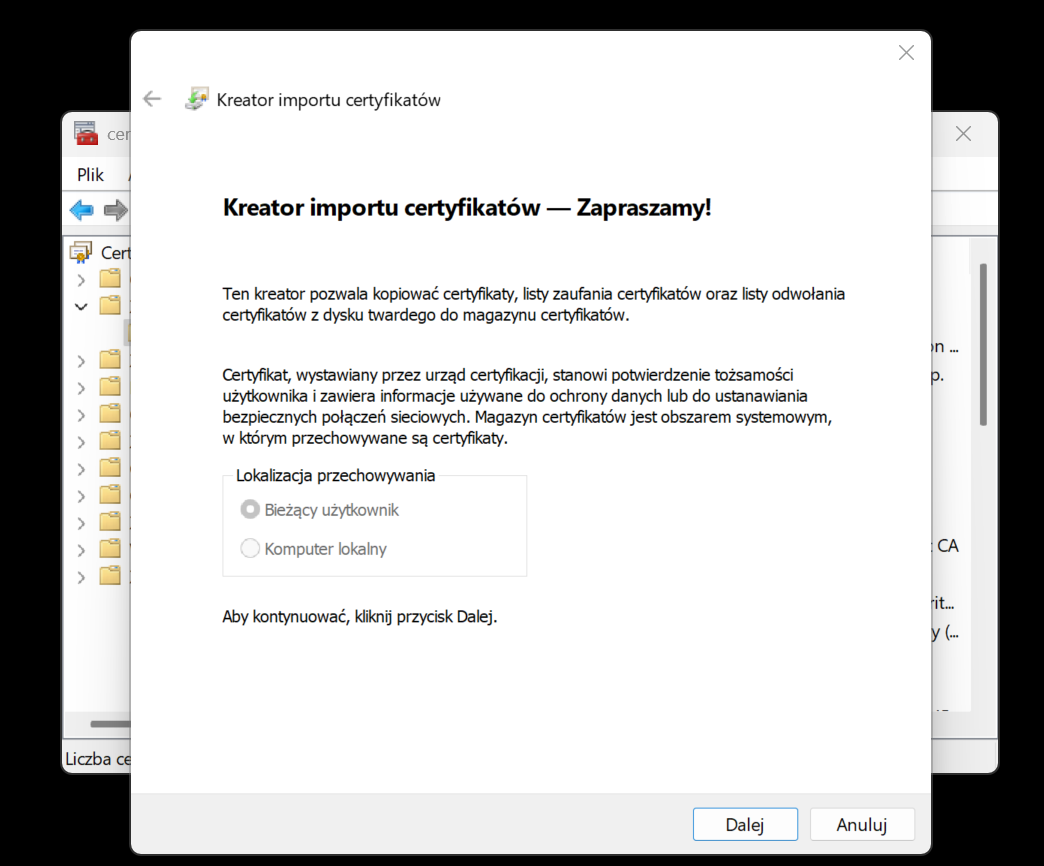
^ plik ca.key w Notatniku:

* Jeśli -----BEGIN ENCRYPTED PRIVATE KEY----- klucz jest zaszyfrowany
* Jeśli -----BEGIN PRIVATE KEY----- nie ma chrony hasłem

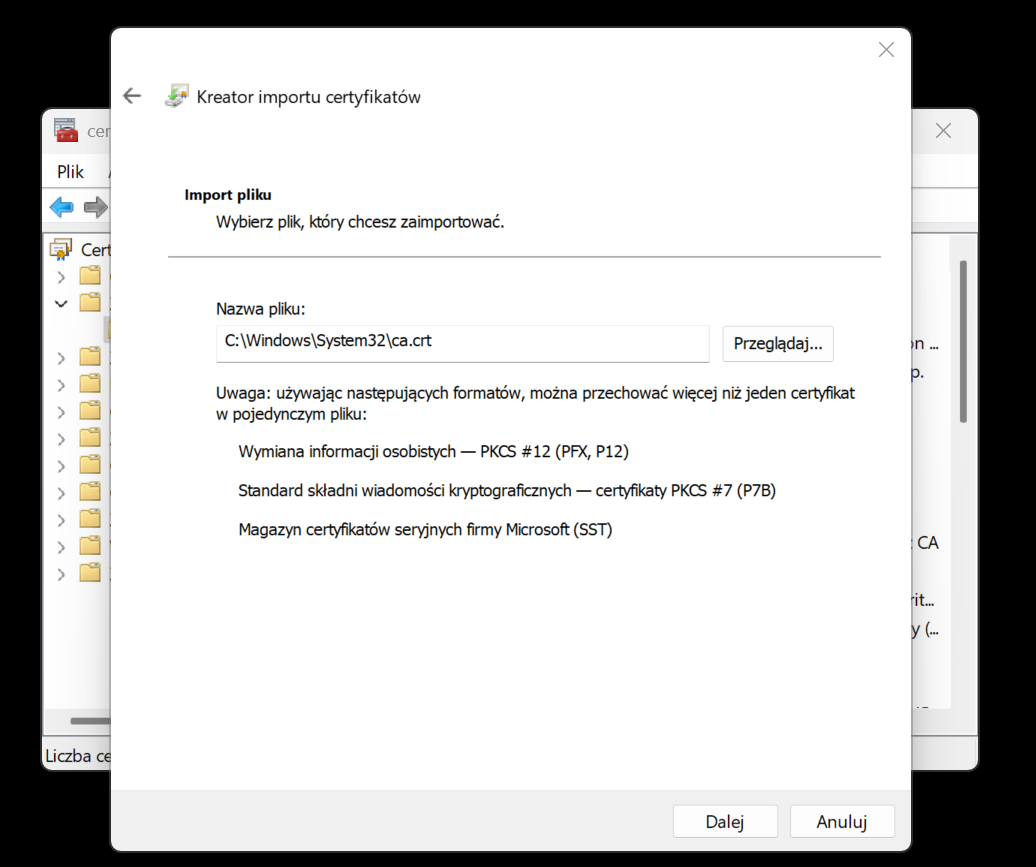
**Krok 2: Dodanie CA do zaufanych certyfikatów w Windows** 

^ otworzyc certmgr.msc

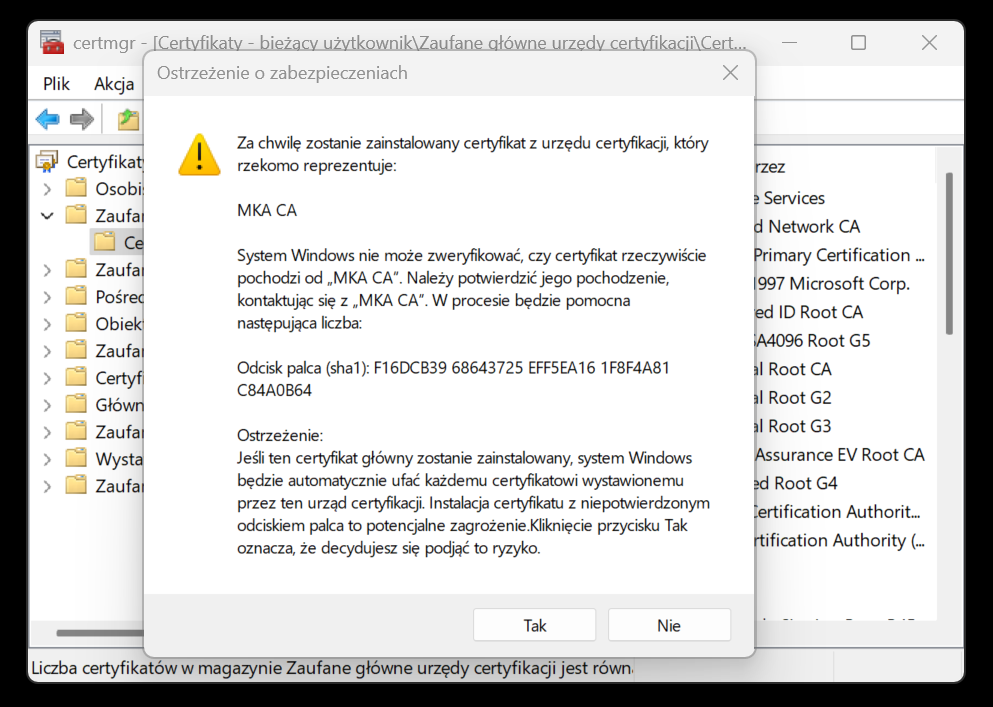
^ certyfikaty -> wszystkie zadania -> importuj...



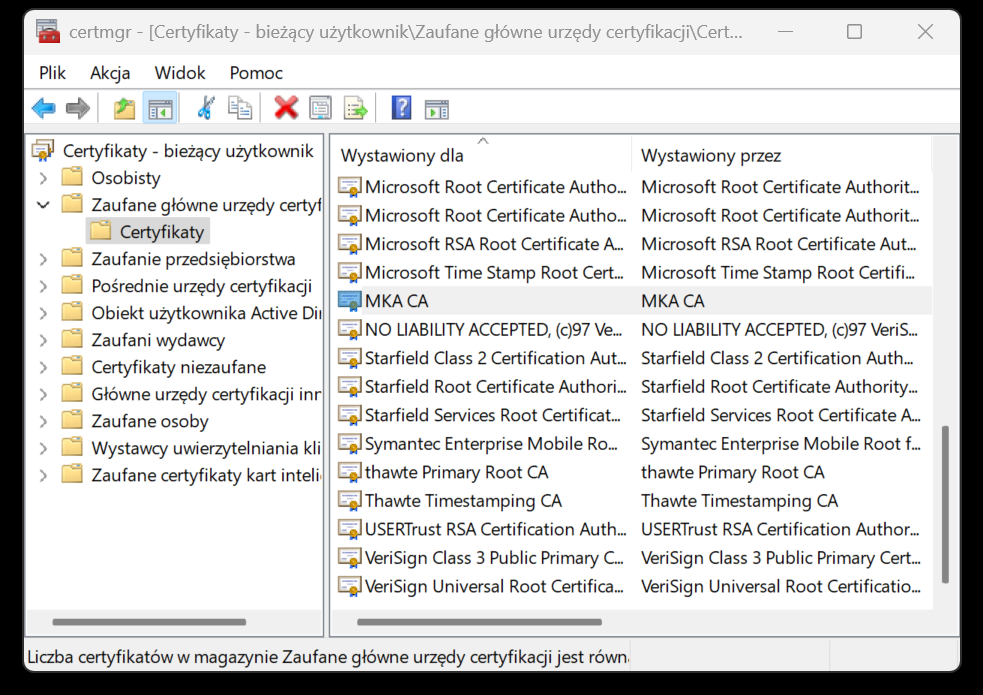
^ dalej...



^ wybrac ca.crt



^ tak..

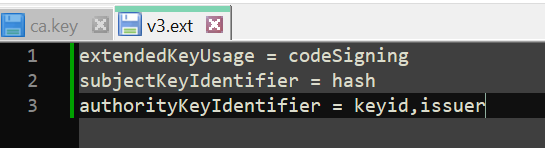


^ jest na liscie certyfikatow

Teraz komputer będzie ufał WSZYSTKIM certyfikatom podpisanym przez MKA CA

**Krok 3: Generowanie certyfikatu dla aplikacji**

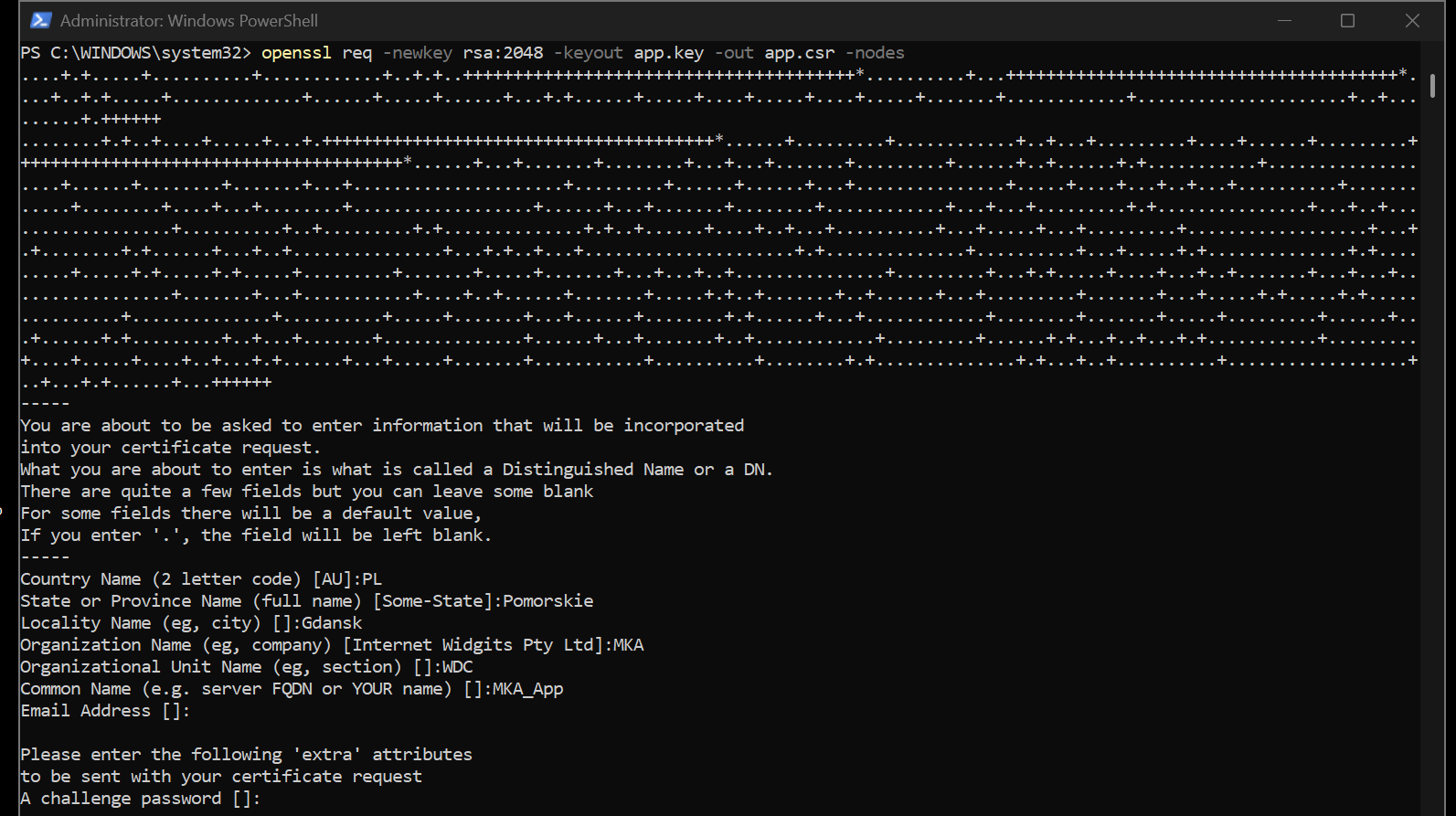
1. v3



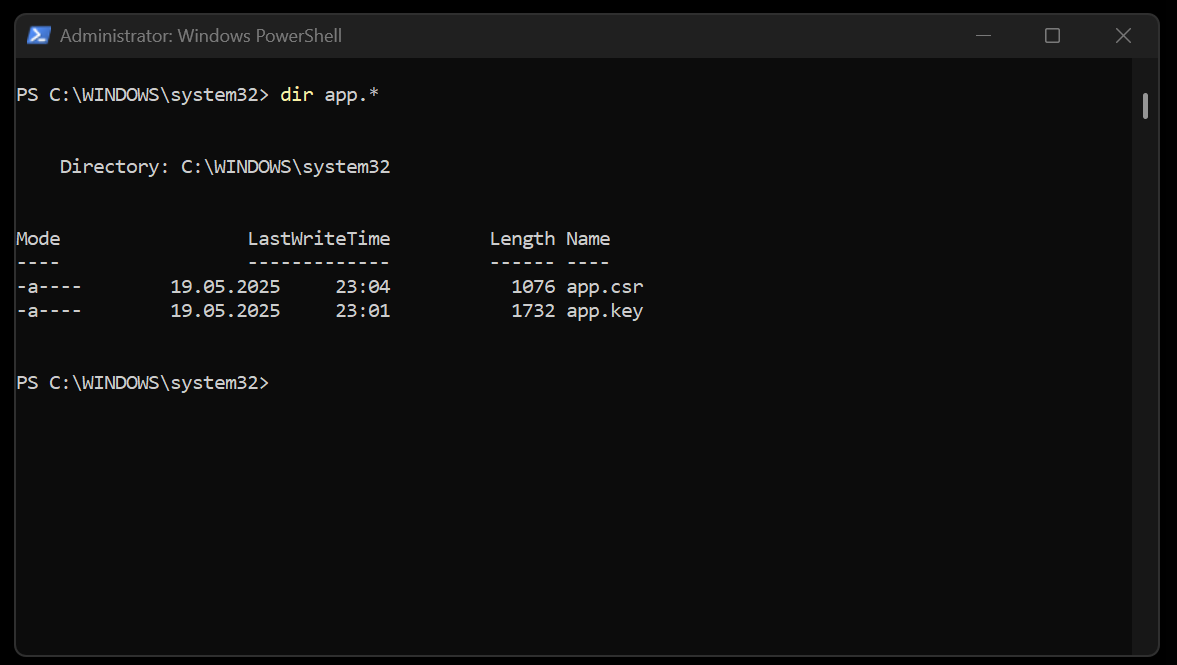
^ zapisax w “C:\Windows\System32\v3.ext"

1. **PS** : openssl req -newkey rsa:2048 -keyout app.key -out app.csr -nodes

(Generuj klucz i wniosek certyfikacyjny (CSR))



^ tak samo tylko inne common name (nazwa aplikacji)

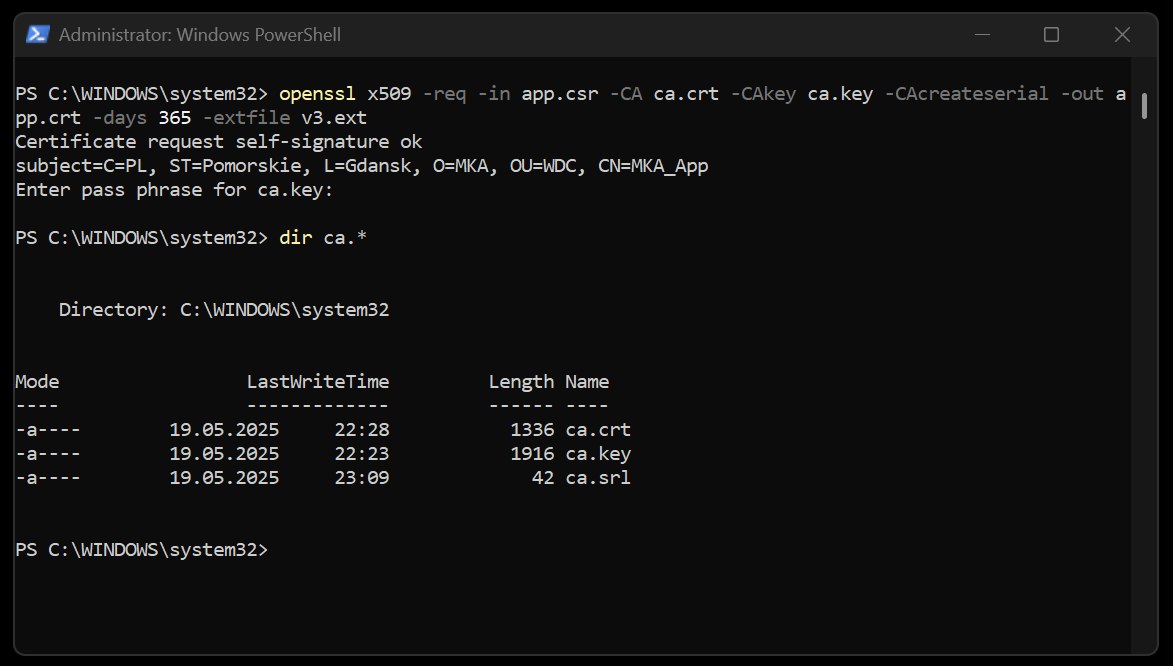


^ Powstaną pliki

* app.key (klucz prywatny)
* app.csr (wniosek)

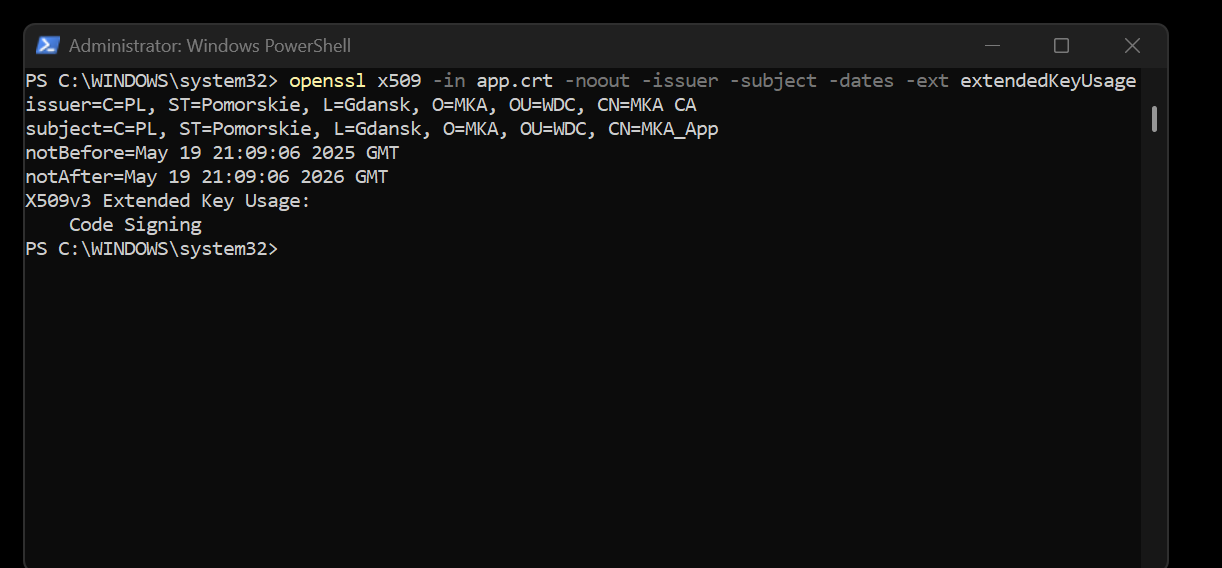
1. **PS:** openssl x509 -req -in app.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out app.crt -days 365 -extfile v3.ext

(Podpisz CSR certyfikatem CA)



^ poprosi o haslo i wygeneruje ca.srl (Plik śledzący numery seryjne certyfikatów)

1. Test **PS:** openssl x509 -in app.crt -noout -issuer -subject -dates -ext extendedKeyUsage

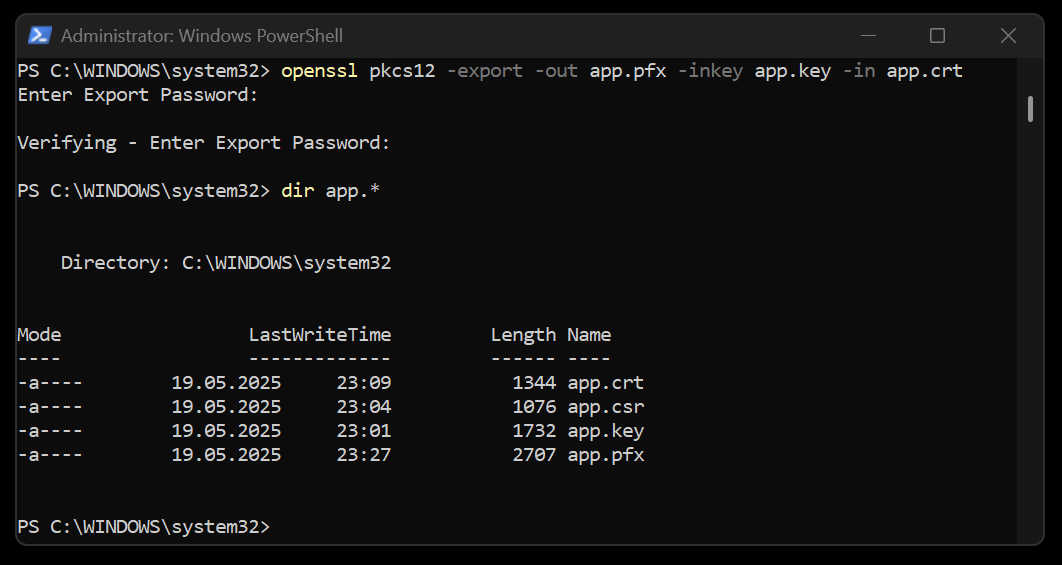


^ jest.

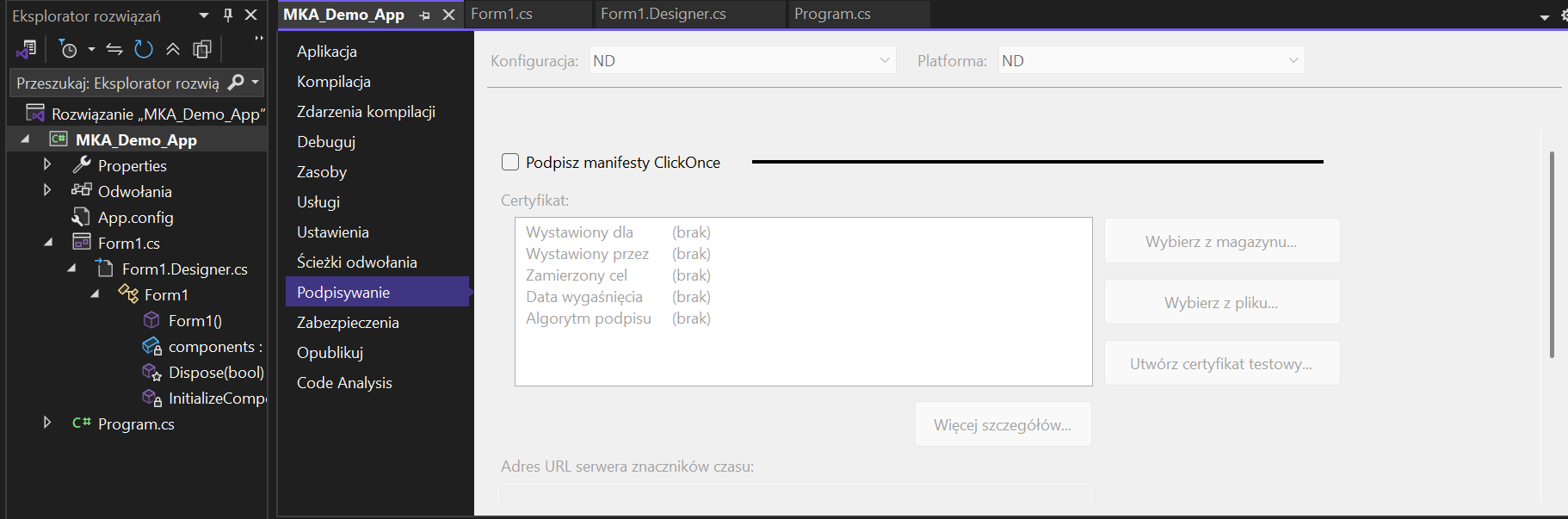
**Krok 4: Podpisywanie kodu w Visual Studio**

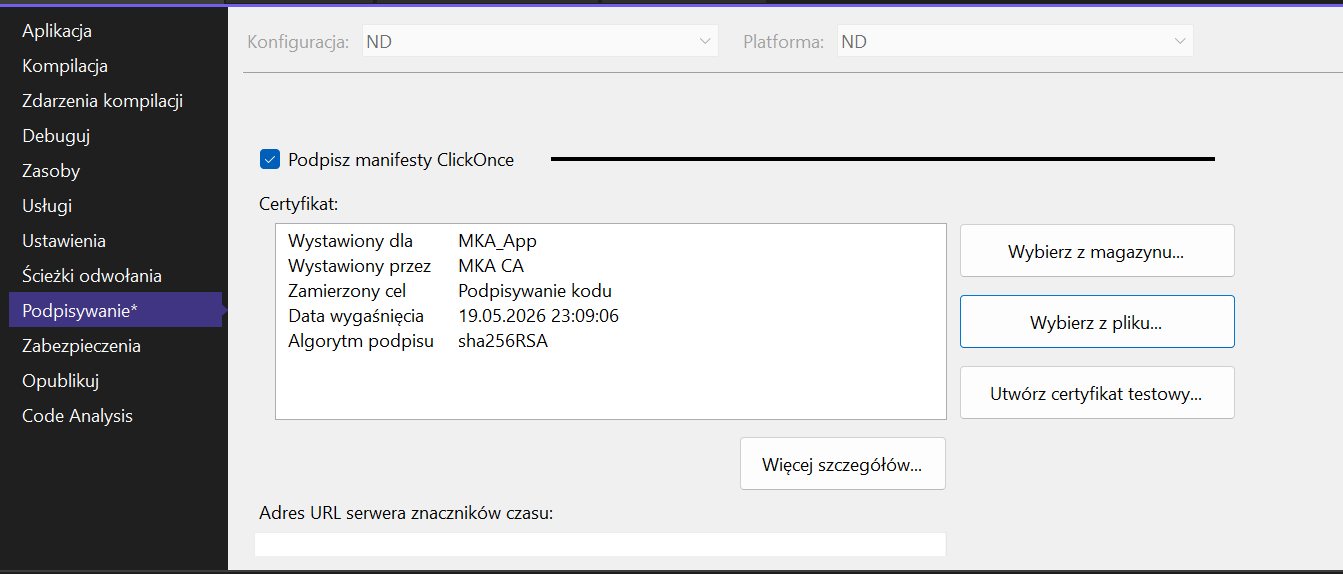
1. Przygotowanie certyfikatu pfx

PS: openssl pkcs12 -export -out app.pfx -inkey app.key -in app.crt



1. Wlasciwosci projektu w Visual Studio

^click one i wybierz plik

^ podpisany

1. Test :W visualu juz jest, ale nie ma w wlasciwosiach aplikacji w bin/debug, bo Podpis nie jest "widoczny" dla Windows

Eksplorator plików pokazuje podpisy tylko od zaufanych urzędów certyfikacji (np. DigiCert, Sectigo). certyfikat self-signed może być uznany za "nieznany", wiec tzrba reczne podpisywanie:

1. Probuje recznie podpisac komeda PS:

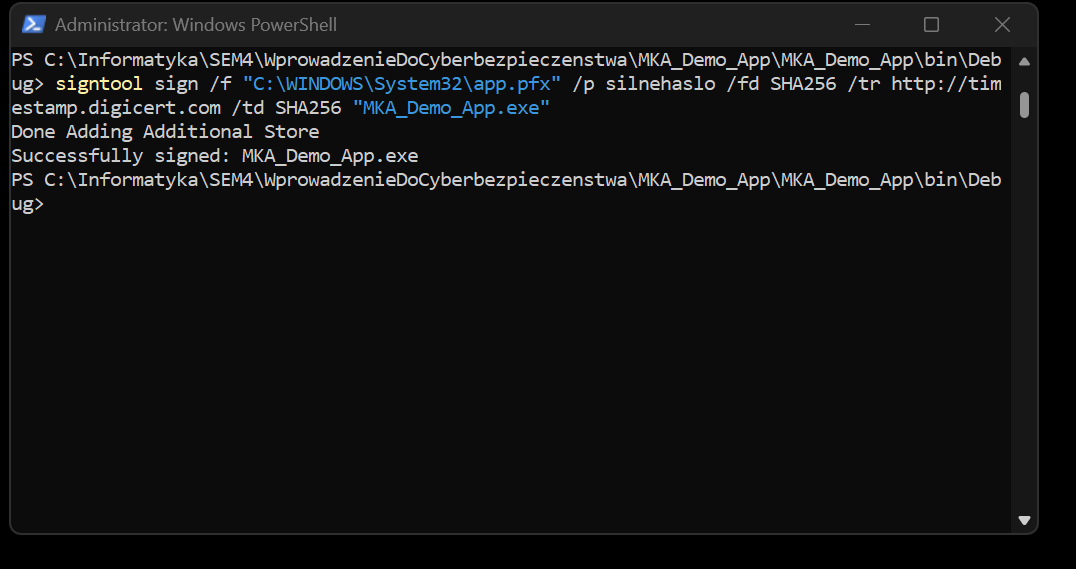
**PS:** signtool sign /f "C:\WINDOWS\System32\app.pfx" /p silnehaslo /fd SHA256 /tr http://timestamp.digicert.com "MKA\_Demo\_App.exe"

Jednak nie dziala , bo ponieważ nowsze wersje signtool wymagają jawnego określenia algorytmu timestampu

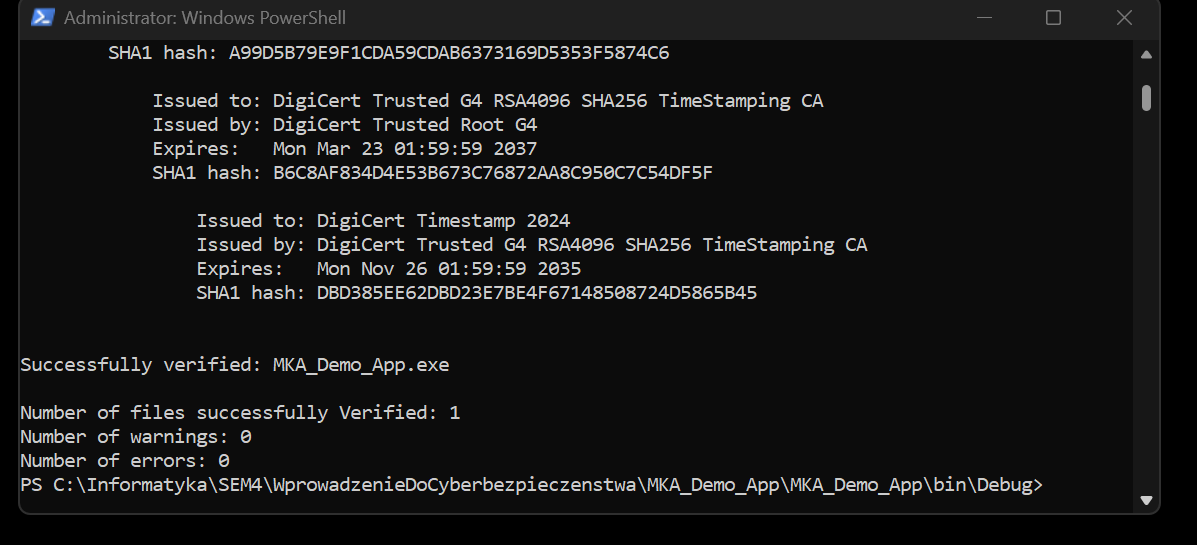
więc

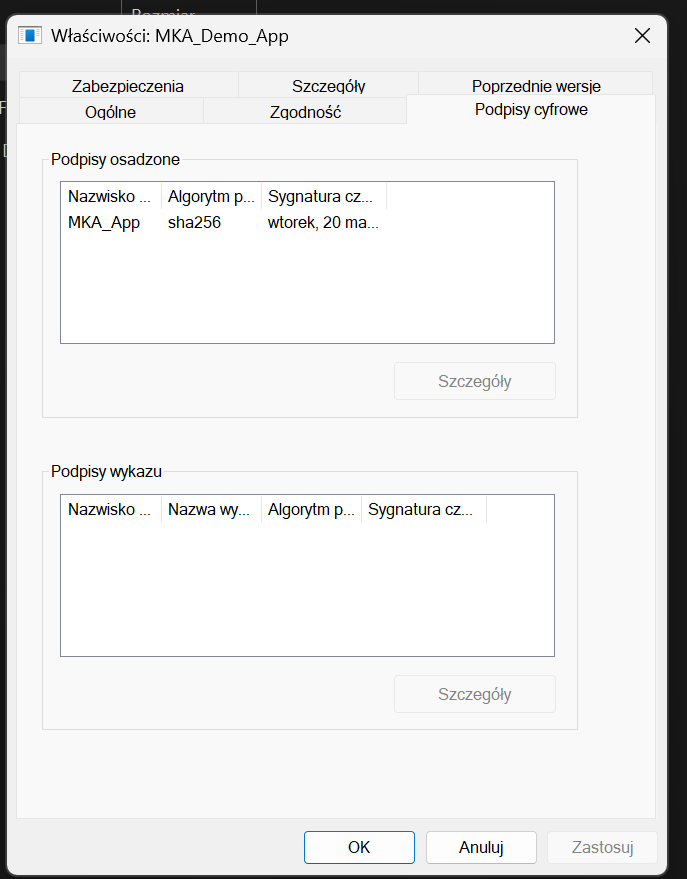
**PS:** signtool sign /f "C:\WINDOWS\System32\app.pfx" /p silnehaslo /fd SHA256 /tr [http://timestamp.digicert.com](http://timestamp.digicert.com/) /td SHA256 "MKA\_Demo\_App.exe"



^ i dziala

Test: PS: signtool verify /v /pa "MKA\_Demo\_App.exe"

^ jest i dziala



^ jest, a nie bylo wczesniej