Introduction

This guide explains, step by step, how to set up and use code signing on Windows 10/11. The process begins with installing and configuring OpenSSL, making sure it is added to the system's PATH so it can be used directly from the command line. Once OpenSSL is ready, you generate a private key, create a Certificate Signing Request (CSR), and issue a self-signed certificate. To make the certificate trusted across the system, it can also be added to the Trusted Root Certification Authorities store.

After the certificate setup, the guide shows how to use signtool.exe to sign executables, installers, or scripts. It also covers attaching a trusted timestamp and verifying the signed code to confirm its authenticity. By following these steps, participants not only secure their software against tampering but also gain practical experience with digital signatures, PKI concepts, and the best practices for safe software distribution.

Prerequisites

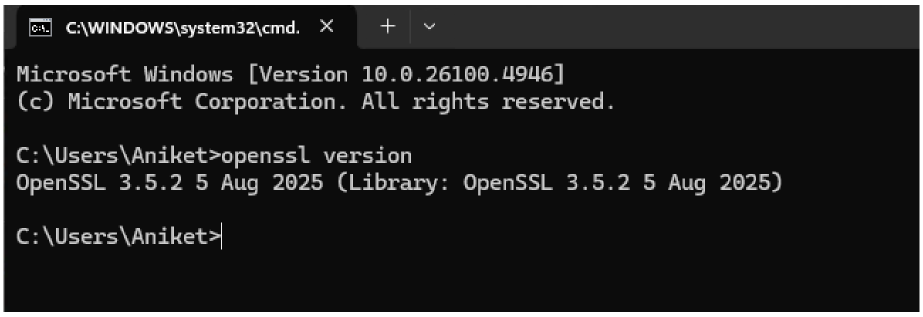
- Windows 10 or Windows 11 with permissions to install software and modify environment variables.
- Network access to obtain the OpenSSL installer from the official distribution page and the ability to run the installer despite SmartScreen warnings when appropriate.
- Windows Software Development Kit Installer

Step-1: Root Certificate Authority Server Setup (Using OpenSSL)

Install OpenSSL

- Download and run the full OpenSSL installer for Windows. Avoid the "Light" variant to ensure required components are available.
- When Windows shows "More info," choose "Run anyway," and allow the installer to make system changes to proceed with installation.

- Ensure the installer copies required OpenSSL DLLs into the Windows system directory, then complete the installation steps as prompted.
- Open the System Environment Variables dialog and edit the system “Path” variable to include the OpenSSL bin directory, then confirm by selecting New and OK to save changes.
- Verify the installation by opening Command Prompt and running the following command to confirm OpenSSL is accessible: ***“openssl version”***

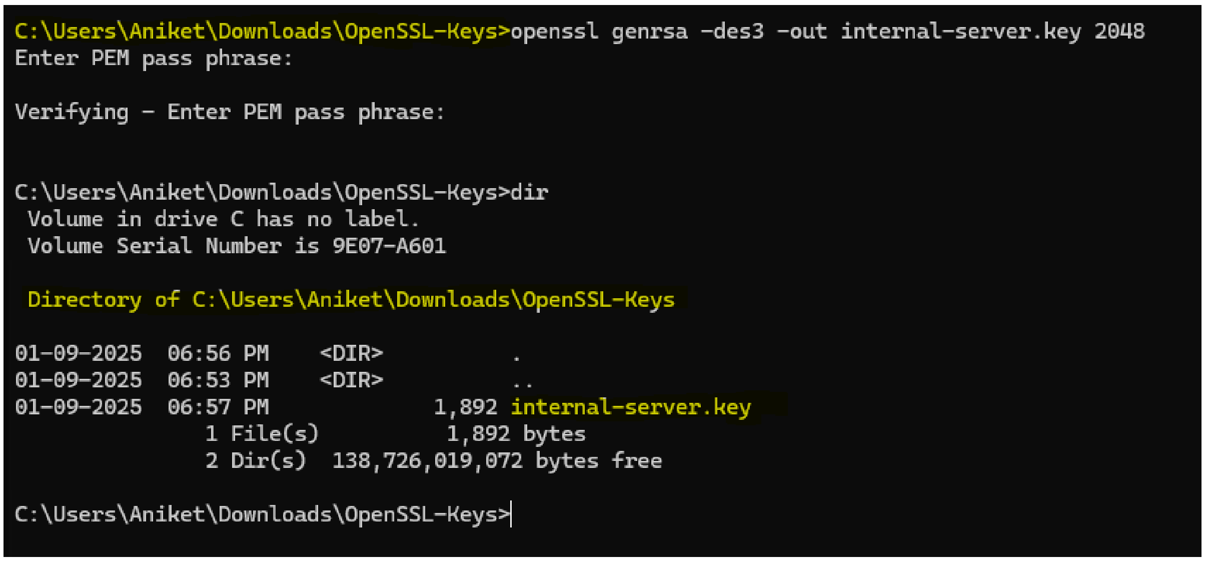


```
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aniket>openssl version
OpenSSL 3.5.2 5 Aug 2025 (Library: OpenSSL 3.5.2 5 Aug 2025)

C:\Users\Aniket>
```

Generate Private Key

In the working folder, generate a password-protected RSA private key with 2048-bit strength using the command below, then enter and confirm a strong passphrase when prompted.

CMD: “openssl genrsa -des3 -out internal-server.key 2048”

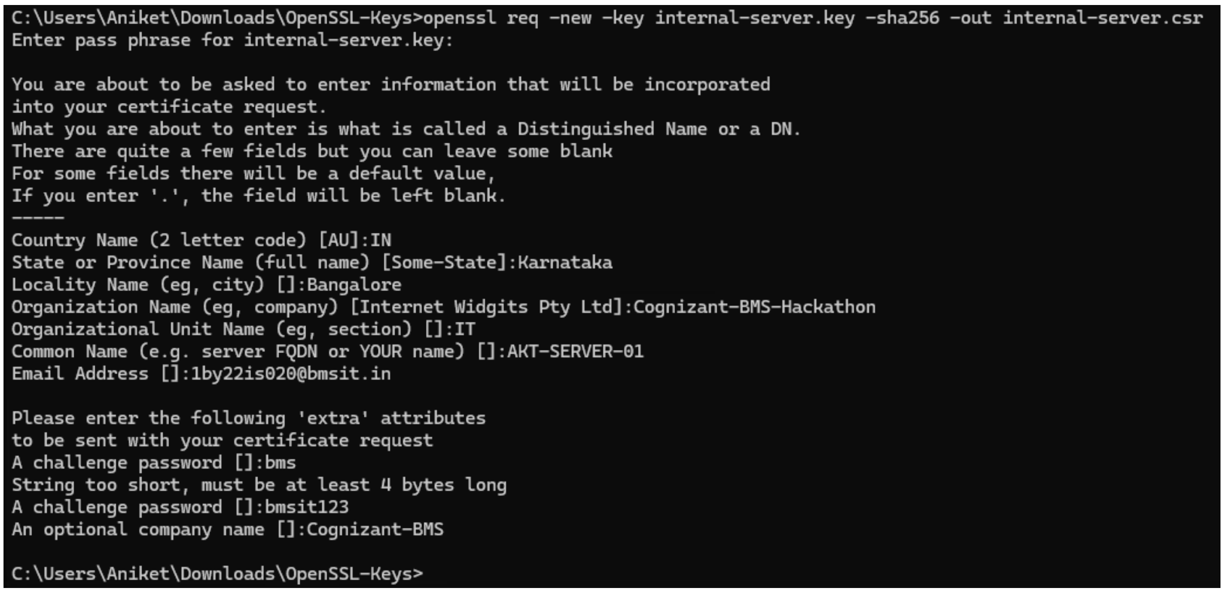


```
C:\Users\Aniket\Downloads\OpenSSL-Keys>openssl genrsa -des3 -out internal-server.key 2048
Enter PEM pass phrase:

Verifying - Enter PEM pass phrase:


C:\Users\Aniket\Downloads\OpenSSL-Keys>dir
 Volume in drive C has no label.
 Volume Serial Number is 9E07-A601

 Directory of C:\Users\Aniket\Downloads\OpenSSL-Keys

01-09-2025  06:56 PM    <DIR>          .
01-09-2025  06:53 PM    <DIR>          ..
01-09-2025  06:57 PM             1,892 internal-server.key
               1 File(s)          1,892 bytes
               2 Dir(s)  138,726,019,072 bytes free

C:\Users\Aniket\Downloads\OpenSSL-Keys>
```

Generate Certificate Signing Request (CSR) File

- Generate a CSR using the existing private key, entering the same passphrase used during key creation when prompted by OpenSSL.
- Determine the machine's hostname to use as the Common Name by running "hostname" in Command Prompt, and provide the requested subject details (e.g., email and CN) when prompted by the CSR command.

CMD: "openssl req -new -key internal-server.key -sha256 -out internal-server.csr"

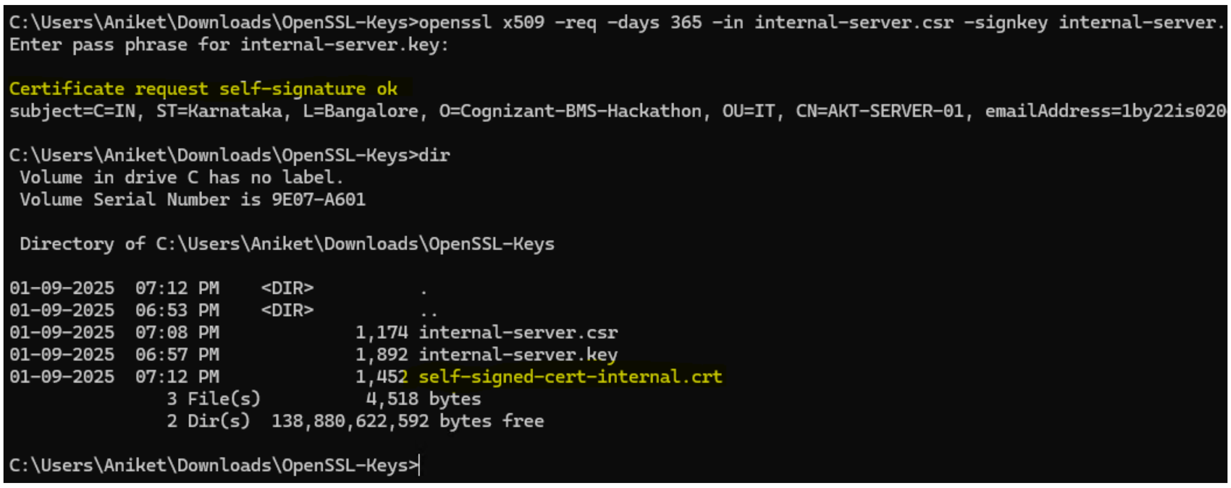


```
C:\Users\Aniket\Downloads\OpenSSL-Keys>openssl req -new -key internal-server.key -sha256 -out internal-server.csr
Enter pass phrase for internal-server.key:

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:Karnataka
Locality Name (eg, city) []:Bangalore
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Cognizant-BMS-Hackathon
Organizational Unit Name (eg, section) []:IT
Common Name (e.g. server FQDN or YOUR name) []:AKT-SERVER-01
Email Address []:1by22is020@bmsit.in

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:bms
String too short, must be at least 4 bytes long
A challenge password []:bmsit123
An optional company name []:Cognizant-BMS

C:\Users\Aniket\Downloads\OpenSSL-Keys>
```

Generate Self-Signed Certificate

- Create a self-signed certificate valid for 365 days from the CSR and private key using the command below, then supply the private key passphrase when requested.
- Upon completion, the certificate file "self-signed-cert-internal.crt" is saved in the current directory.

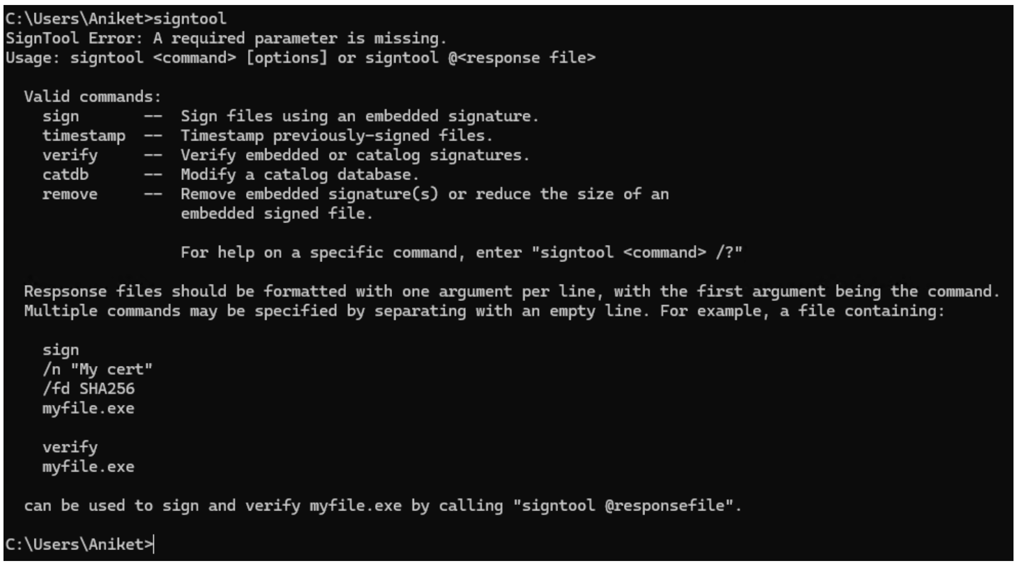
CMD: "openssl x509 -req -days 365 -in internal-server.csr -signkey internal-server.key -sha256 -out self-signed-cert-internal.crt"


```
C:\Users\Aniket\Downloads\OpenSSL-Keys>openssl x509 -req -days 365 -in internal-server.csr -signkey internal-server.
Enter pass phrase for internal-server.key:

Certificate request self-signature ok
subject=C=IN, ST=Karnataka, L=Bangalore, O=Cognizant-BMS-Hackathon, OU=IT, CN=AKT-SERVER-01, emailAddress=1by22is020

C:\Users\Aniket\Downloads\OpenSSL-Keys>dir
 Volume in drive C has no label.
 Volume Serial Number is 9E07-A601

 Directory of C:\Users\Aniket\Downloads\OpenSSL-Keys

01-09-2025  07:12 PM    <DIR>          .
01-09-2025  06:53 PM    <DIR>          ..
01-09-2025  07:08 PM             1,174 internal-server.csr
01-09-2025  06:57 PM             1,892 internal-server.key
01-09-2025  07:12 PM             1,452 self-signed-cert-internal.crt
               3 File(s)          4,518 bytes
               2 Dir(s)  138,880,622,592 bytes free

C:\Users\Aniket\Downloads\OpenSSL-Keys>
```

Step-2: Code Signing Using Self-Signed Certificate (Using SignTool.exe)

Install SignTool

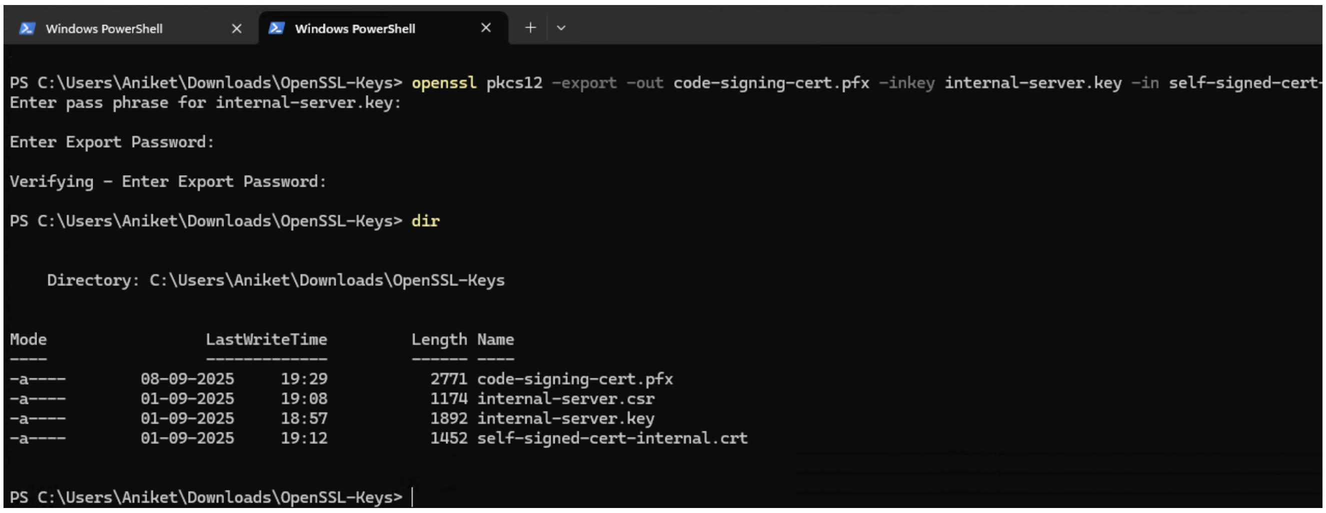
- Download and Install Windows SDK on the System.
- Add this PATH in the System Environment Variables: **“C:\Program Files (x86)\Windows Kits\10\bin\10.0.26100.0\x64\bin”**
- Run the Command: **“signtool”** to verify Installation


```
C:\Users\Aniket>signtool
SignTool Error: A required parameter is missing.
Usage: signtool <command> [options] or signtool @<response file>

  Valid commands:
    sign       --  Sign files using an embedded signature.
    timestamp  --  Timestamp previously-signed files.
    verify     --  Verify embedded or catalog signatures.
    catdb      --  Modify a catalog database.
    remove     --  Remove embedded signature(s) or reduce the size of an
                   embedded signed file.

                   For help on a specific command, enter "signtool <command> /?"

  Respsonse files should be formatted with one argument per line, with the first argument being the command.
  Multiple commands may be specified by separating with an empty line. For example, a file containing:

    sign
    /n "My cert"
    /fd SHA256
    myfile.exe

    verify
    myfile.exe

  can be used to sign and verify myfile.exe by calling "signtool @responsefile".

C:\Users\Aniket>
```

Setup a .PFX File (Private Key + Certificate)

- Make sure the Private Key, Certificate Signing Request File and the Certificate File is in the same folder.
- Run the command below, to generate the PFX File in the current working directory.

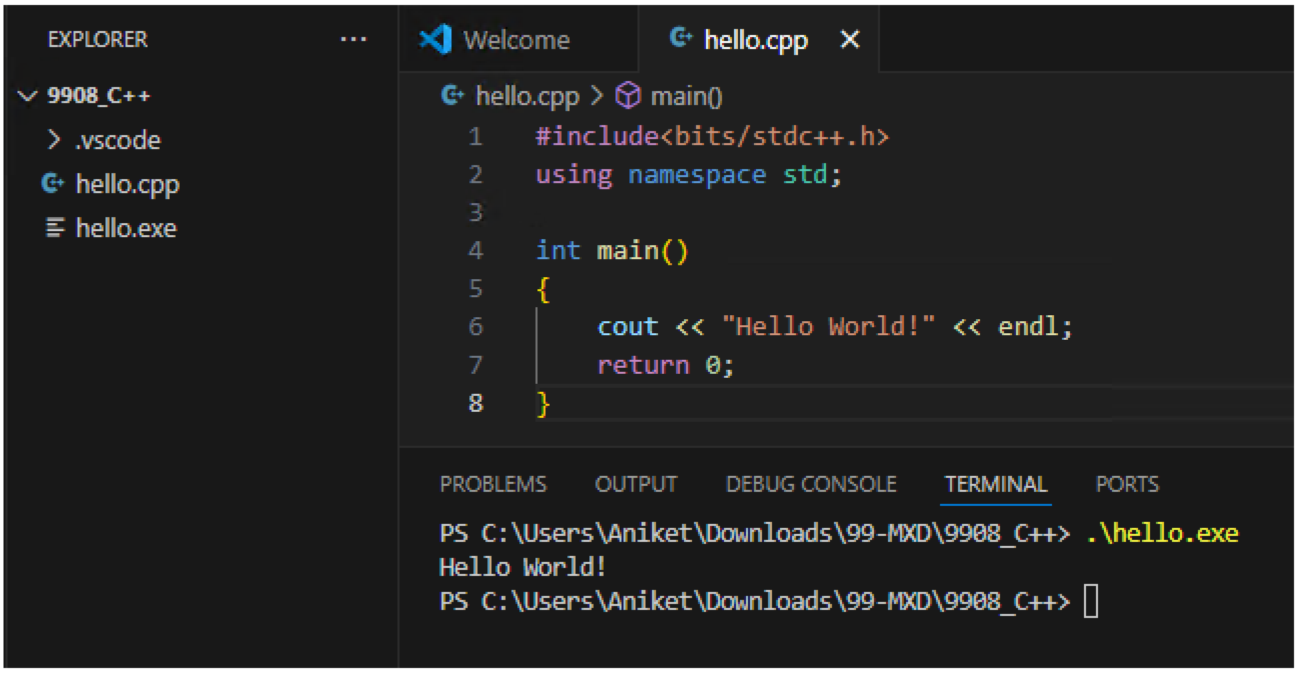
CMD: "openssl pkcs12 -export -out code-signing-cert.pfx -inkey internal-server.key -in self-signed-cert-internal.crt"



```
PS C:\Users\Aniket\Downloads\OpenSSL-Keys> openssl pkcs12 -export -out code-signing-cert.pfx -inkey internal-server.key -in self-signed-cert-
Enter pass phrase for internal-server.key:

Enter Export Password:

Verifying - Enter Export Password:

PS C:\Users\Aniket\Downloads\OpenSSL-Keys> dir


    Directory: C:\Users\Aniket\Downloads\OpenSSL-Keys


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        08-09-2025     19:29           2771 code-signing-cert.pfx
-a----        01-09-2025     19:08           1174 internal-server.csr
-a----        01-09-2025     18:57           1892 internal-server.key
-a----        01-09-2025     19:12           1452 self-signed-cert-internal.crt


PS C:\Users\Aniket\Downloads\OpenSSL-Keys> |
```

Sign with SignTool

- Creation of a Test Binary by writing a "Hello World!" Program in C++, resulting to a .EXE which is then signed using the "SignTool.exe".



```cpp
#include<bits/stdc++.h>
using namespace std;

int main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

```
PS C:\Users\Aniket\Downloads\99-MXD\9908_C++> .\hello.exe
Hello World!
PS C:\Users\Aniket\Downloads\99-MXD\9908_C++>
```

- Use SignTool with the PFX bundle and password to sign a valid executable

CMD: "signtool sign /f C:\Users\Aniket\Downloads\OpenSSL-Keys\code-signing-cert.pfx /p bmsit /fd sha256 /tr http://timestamp.digicert.com /td sha256 hello.exe"

- **Parameters:**

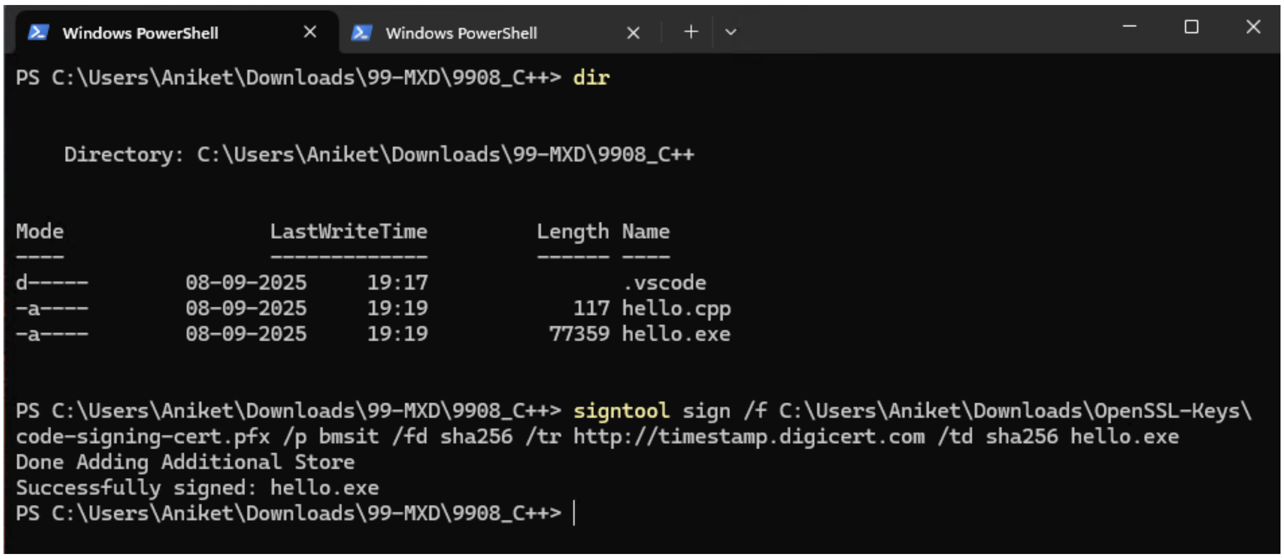
  /f : code-signing-cert.pfx [PFX File]

  /p : Password

  /tr : Timestamp server

  /td sha256 : Hash algorithm for timestamp

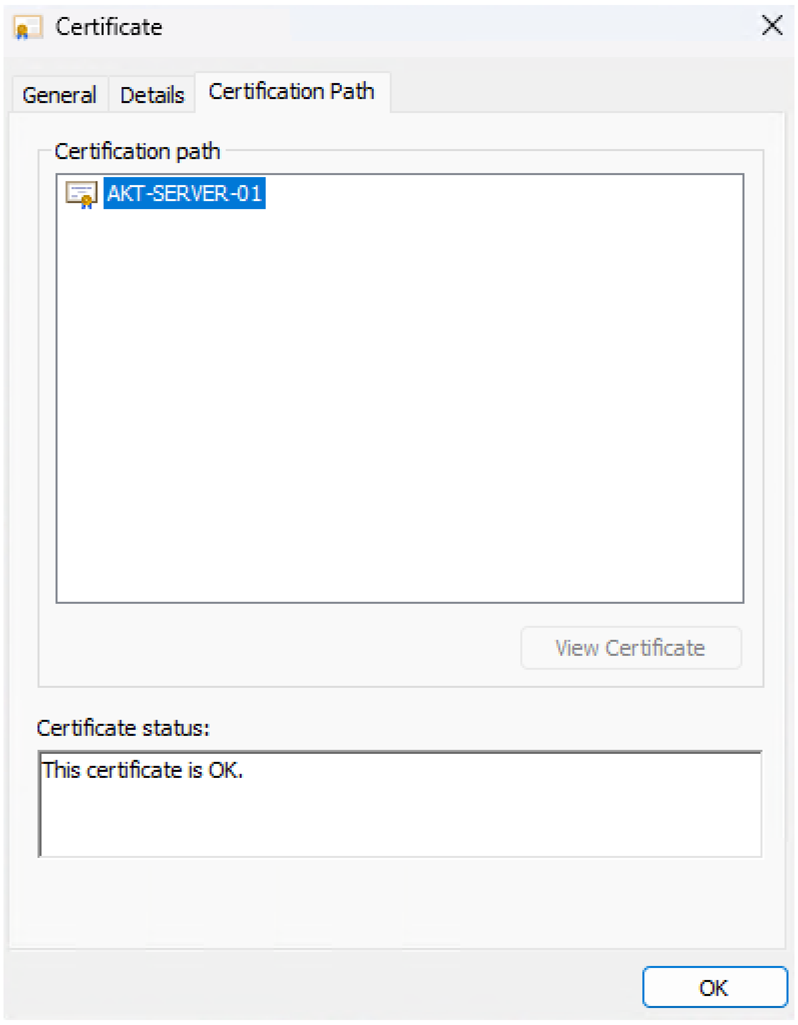
  /fd sha256 : File digest algorithm



```
PS C:\Users\Aniket\Downloads\99-MXD\9908_C++> dir


    Directory: C:\Users\Aniket\Downloads\99-MXD\9908_C++


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        08-09-2025     19:17                .vscode
-a----        08-09-2025     19:19            117 hello.cpp
-a----        08-09-2025     19:19          77359 hello.exe


PS C:\Users\Aniket\Downloads\99-MXD\9908_C++> signtool sign /f C:\Users\Aniket\Downloads\OpenSSL-Keys\
code-signing-cert.pfx /p bmsit /fd sha256 /tr http://timestamp.digicert.com /td sha256 hello.exe
Done Adding Additional Store
Successfully signed: hello.exe
PS C:\Users\Aniket\Downloads\99-MXD\9908_C++> |
```

Add Self-Signed Certificate to the Trusted Root Certification Authority of Windows

- Trust the self-signed root certificate locally, import the .crt into the Local Machine “Trusted Root Certification Authorities” store using the Certificate Import Wizard and the “Place all certificates in the following store” option.



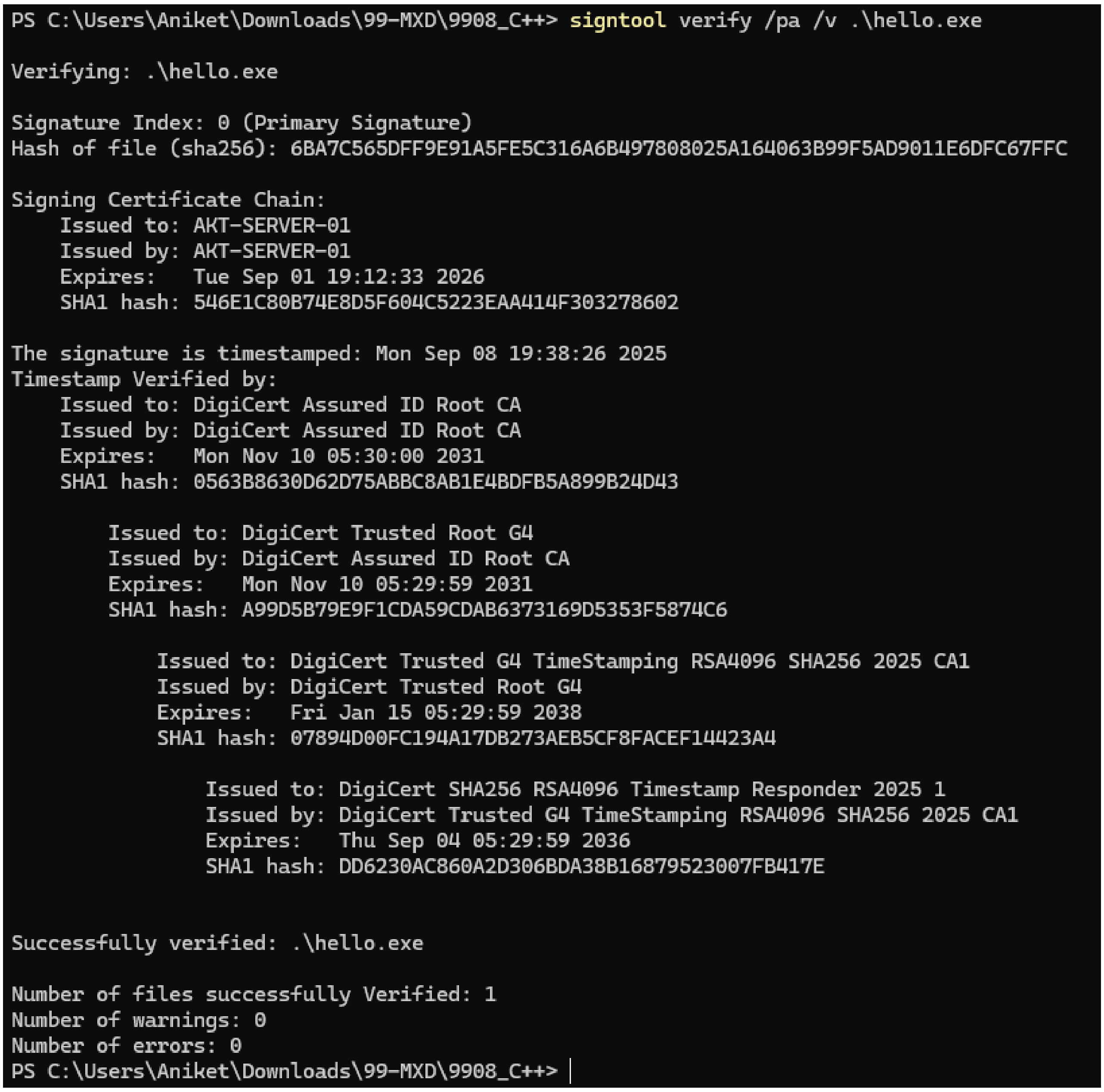
AKT-SERVER-01

Certificate status:

This certificate is OK.

Verification

- Verify the signed artifact using “**signtool verify /pa /v hello.exe**” to view the signature details, including the hash, signature index, and timestamping chain.



```
PS C:\Users\Aniket\Downloads\99-MXD\9908_C++> signtool verify /pa /v .\hello.exe

Verifying: .\hello.exe

Signature Index: 0 (Primary Signature)
Hash of file (sha256): 6BA7C565DFF9E91A5FE5C316A6B497808025A164063B99F5AD9011E6DFC67FFC

Signing Certificate Chain:
    Issued to: AKT-SERVER-01
    Issued by: AKT-SERVER-01
    Expires:   Tue Sep 01 19:12:33 2026
    SHA1 hash: 546E1C80B74E8D5F604C5223EAA414F303278602

The signature is timestamped: Mon Sep 08 19:38:26 2025
Timestamp Verified by:
    Issued to: DigiCert Assured ID Root CA
    Issued by: DigiCert Assured ID Root CA
    Expires:   Mon Nov 10 05:30:00 2031
    SHA1 hash: 0563B8630D62D75ABBC8AB1E4BDFB5A899B24D43

        Issued to: DigiCert Trusted Root G4
        Issued by: DigiCert Assured ID Root CA
        Expires:   Mon Nov 10 05:29:59 2031
        SHA1 hash: A99D5B79E9F1CDA59CDAB6373169D5353F5874C6

            Issued to: DigiCert Trusted G4 TimeStamping RSA4096 SHA256 2025 CA1
            Issued by: DigiCert Trusted Root G4
            Expires:   Fri Jan 15 05:29:59 2038
            SHA1 hash: 07894D00FC194A17DB273AEB5CF8FACEF14423A4

                Issued to: DigiCert SHA256 RSA4096 Timestamp Responder 2025 1
                Issued by: DigiCert Trusted G4 TimeStamping RSA4096 SHA256 2025 CA1
                Expires:   Thu Sep 04 05:29:59 2036
                SHA1 hash: DD6230AC860A2D306BDA38B16879523007FB417E


Successfully verified: .\hello.exe

Number of files successfully Verified: 1
Number of warnings: 0
Number of errors: 0
PS C:\Users\Aniket\Downloads\99-MXD\9908_C++>
```

Step-3: Secure Software Distribution

A local HTTPS Flask server that issues 5-minute JSON Web Tokens (JWT) to authorize downloads of a signed executable, optionally expose an unsigned executable for contrast, and provide a SHA-256 checksum endpoint for integrity verification.

Install dependencies

Install Flask and PyJWT in the command prompt.

CMD: “pip install flask pyjwt”

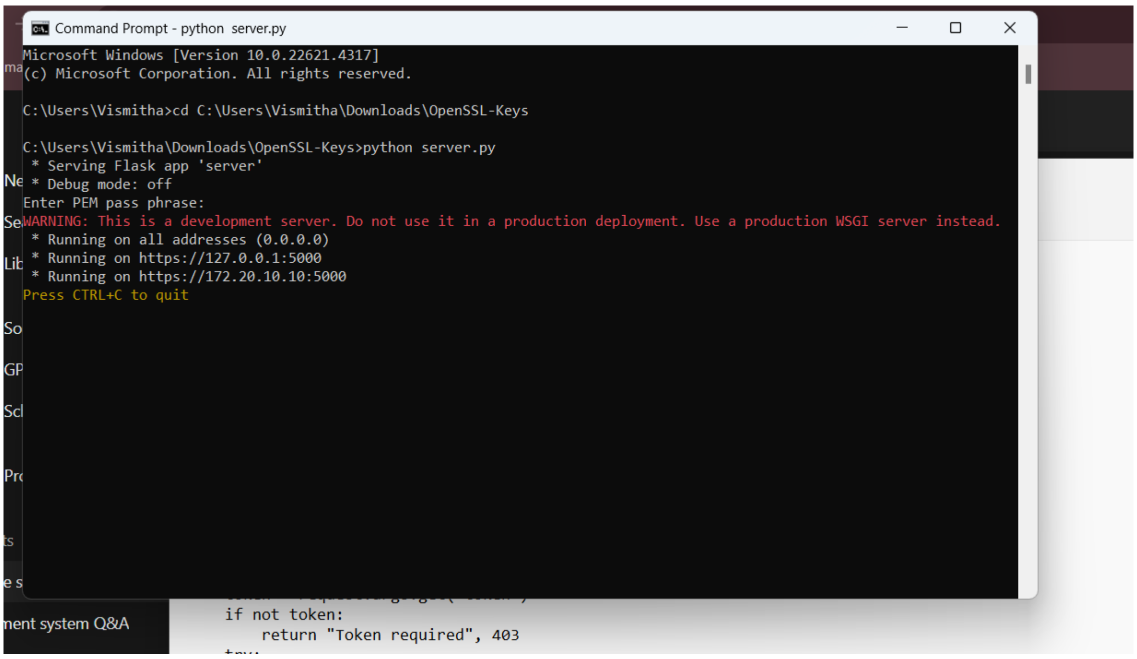
Server-Side Code:

```
from flask import Flask, request, send_file, jsonify

import jwt, datetime, hashlib


app = Flask(__name__)


SECRET_KEY = "super-secret-key"  # Change this for production


# --- Generate short-lived token (5 minutes) ---

@app.route("/get-token", methods=["GET"])

def get_token():

    expiry = datetime.datetime.utcnow() + datetime.timedelta(minutes=5)

    token = jwt.encode({"exp": expiry}, SECRET_KEY, algorithm="HS256")

    return jsonify({"token": token})


# --- Secure download for SIGNED exe ---

@app.route("/download/signed", methods=["GET"])

def download_signed():

    token = request.args.get("token")

    if not token:

        return "Token required", 403
    try:
        jwt.decode(token, SECRET_KEY, algorithms=["HS256"])
    except jwt.ExpiredSignatureError:
        return "Token expired", 403
    except jwt.InvalidTokenError:
        return "Invalid token", 403
    return send_file("hello.exe", as_attachment=True)

# --- Download UNSIGNED exe (for contrast) ---
@app.route("/download/unsigned", methods=["GET"])
def download_unsigned():
    return send_file("MyApp.exe", as_attachment=True)

# --- SHA-256 checksum for signed exe ---
@app.route("/checksum", methods=["GET"])
def checksum():
    sha256_hash = hashlib.sha256()
    with open("hello.exe", "rb") as f:
        for chunk in iter(lambda: f.read(4096), b""):
            sha256_hash.update(chunk)
    return jsonify({"file": "hello.exe", "sha256": sha256_hash.hexdigest()})

if __name__ == "__main__":
    app.run(
        host="0.0.0.0",
        port=5000,
        ssl_context=("self-signed-cert-internal.crt", "internal-server.key"),
    )
```

Run the Server

Open Command Prompt in the folder and start the server.

CMD: “python server.py”



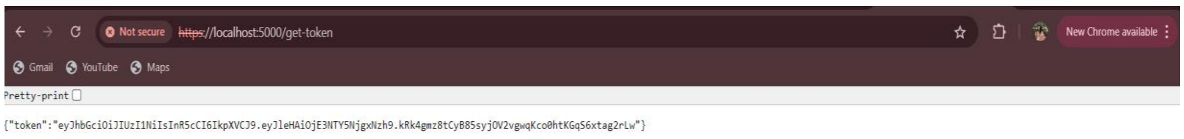
```
Microsoft Windows [Version 10.0.22621.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Vismitha>cd C:\Users\Vismitha\Downloads\OpenSSL-Keys

C:\Users\Vismitha\Downloads\OpenSSL-Keys>python server.py
 * Serving Flask app 'server'
 * Debug mode: off
Enter PEM pass phrase:
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on https://127.0.0.1:5000
 * Running on https://172.20.10.10:5000
Press CTRL+C to quit
```

Get JWT Token

In a browser, visit the token endpoint to receive a JWT valid for 5 minutes.

https://localhost:5000/get-token

The response will look like: {"token": "eyJhbGciOi..."} — copy the string value of token for the next step.



```
{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3NTY5NjgxNzh9.kRk4gmz8tCyB85syjOV2vgwqKco0htKGqS6xtag2rLw"}
```

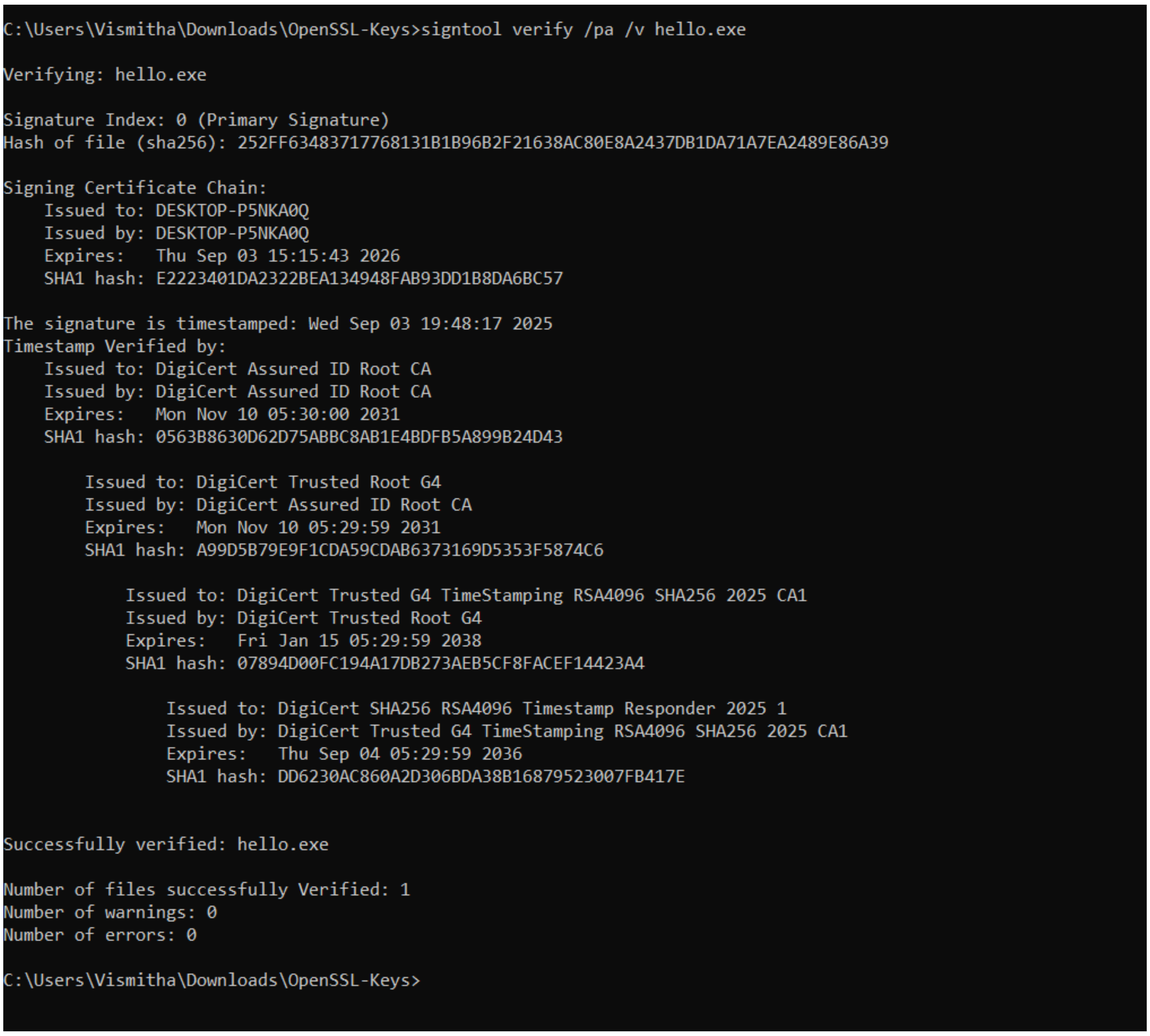
Download Signed App

Use the token to download the signed executable.

https://localhost:5000/download/signed?token=PASTE_TOKEN_HERE

Verify Signature.

CMD: “signtool verify /pa /v hello.exe”



```
C:\Users\Vismitha\Downloads\OpenSSL-Keys>signtool verify /pa /v hello.exe

Verifying: hello.exe

Signature Index: 0 (Primary Signature)
Hash of file (sha256): 252FF63483717768131B1B96B2F21638AC80E8A2437DB1DA71A7EA2489E86A39

Signing Certificate Chain:
    Issued to: DESKTOP-P5NKA0Q
    Issued by: DESKTOP-P5NKA0Q
    Expires:   Thu Sep 03 15:15:43 2026
    SHA1 hash: E2223401DA2322BEA134948FAB93DD1B8DA6BC57

The signature is timestamped: Wed Sep 03 19:48:17 2025
Timestamp Verified by:
    Issued to: DigiCert Assured ID Root CA
    Issued by: DigiCert Assured ID Root CA
    Expires:   Mon Nov 10 05:30:00 2031
    SHA1 hash: 0563B8630D62D75ABBC8AB1E4BDFB5A899B24D43

        Issued to: DigiCert Trusted Root G4
        Issued by: DigiCert Assured ID Root CA
        Expires:   Mon Nov 10 05:29:59 2031
        SHA1 hash: A99D5B79E9F1CDA59CDAB6373169D5353F5874C6

            Issued to: DigiCert Trusted G4 TimeStamping RSA4096 SHA256 2025 CA1
            Issued by: DigiCert Trusted Root G4
            Expires:   Fri Jan 15 05:29:59 2038
            SHA1 hash: 07894D00FC194A17DB273AEB5CF8FACEF14423A4

                Issued to: DigiCert SHA256 RSA4096 Timestamp Responder 2025 1
                Issued by: DigiCert Trusted G4 TimeStamping RSA4096 SHA256 2025 CA1
                Expires:   Thu Sep 04 05:29:59 2036
                SHA1 hash: DD6230AC860A2D306BDA38B16879523007FB417E


Successfully verified: hello.exe

Number of files successfully Verified: 1
Number of warnings: 0
Number of errors: 0

C:\Users\Vismitha\Downloads\OpenSSL-Keys>
```