

PKCS#11 Wrapper – APK Signing Integration Document

For APK signing, we use APKSigner, which is a command-line utility included in the Android SDK Build Tools that allows you to sign Android application package (APK) files and verify their signatures. It is an essential tool for Android developers to ensure the integrity and authenticity of their apps.

Encryption Consulting's PKCS#11 Wrapper exposes the keys held in your HSM through the standard PKCS#11 interface, which means APKSigner can use them through Java's built-in SunPKCS11 provider without any change to the tool itself. The keystore is given as type PKCS11 rather than a local file, so the private key never leaves the HSM and every signing operation is recorded centrally.

Section 1 covers the steps that are common to every platform. After that, follow only the track for your operating system — Sections 2 and 3 for Linux (Ubuntu), Sections 4 and 5 for macOS, or Sections 6 and 7 for Windows. Each track ends with signing and verification.

Prerequisites: Access to the CodeSign Secure portal, administrative rights on the client machine, and a sample APK file to sign.

1. Common Setup

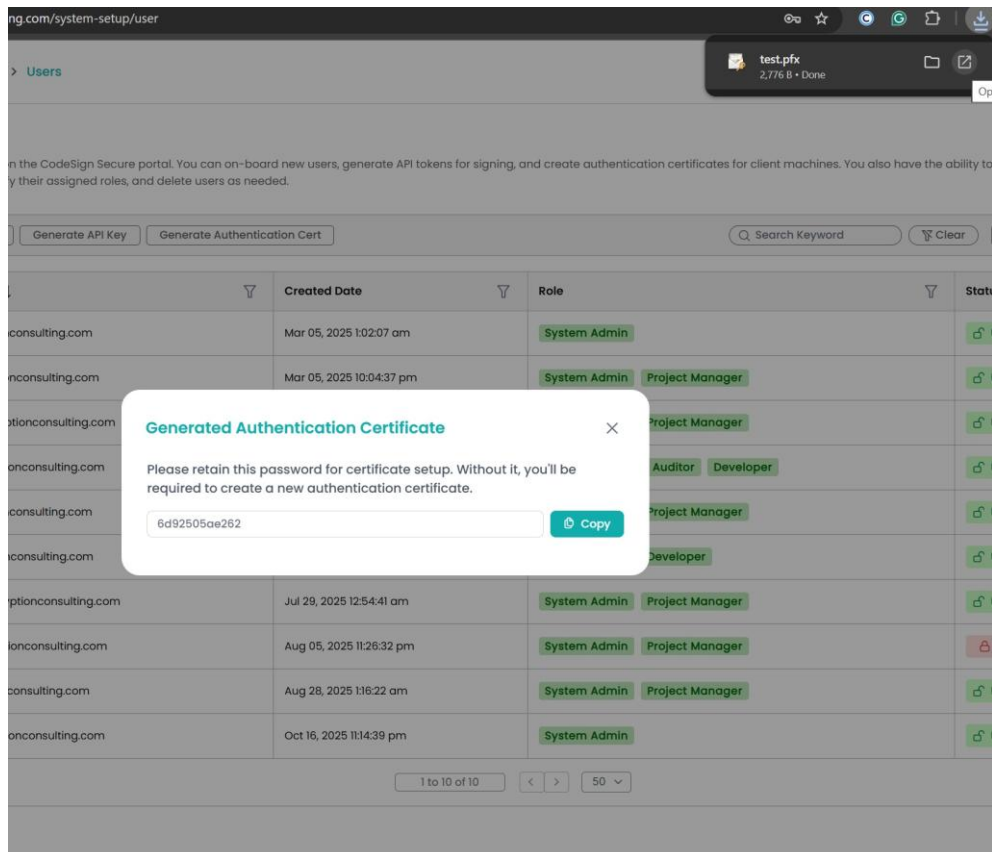
Two things are needed regardless of platform: a P12 authentication certificate that identifies your machine to CodeSign Secure, and the two configuration files that ship inside the PKCS#11 Wrapper download.

Steps:

Step 1: Generate a P12 Authentication Certificate

- Generate a P12 Authentication certificate from the System Setup > User > Generate Authentication Certificate dropdown in the CodeSign Secure portal.
- Enter the certificate name, user name, and expiry date. A .pfx certificate is generated and downloaded, and its password is displayed.

NOTE: The password is displayed only once, so copy and store it safely. Both the certificate path and its password are referenced by the wrapper configuration in your platform track.



Step 2: Understand the Two Configuration Files

- The PKCS#11 Wrapper download contains two files that must be edited before use:
 - **ec_pkcs11client.ini** – Points the wrapper at your CodeSign Secure server and at the P12 authentication certificate generated in Step 1.
 - **pkcs11properties.cfg** – The SunPKCS11 provider configuration that Java reads. Its path is supplied to APKSigner through the --provider-arg option.

NOTE: When using a Luna HSM, the slot number must also be added to the pkcs11properties.cfg file. Each platform track below shows this variation.

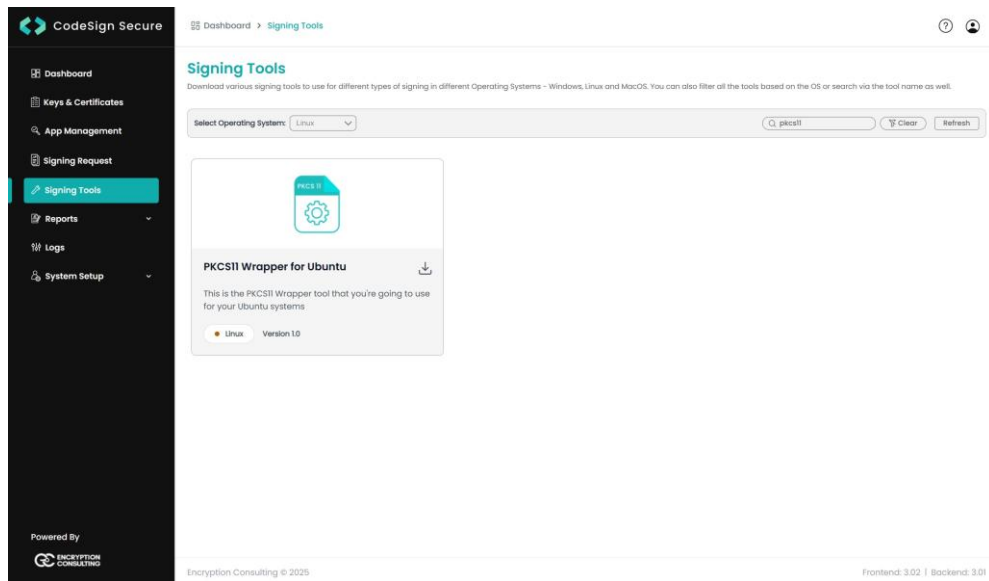
2. Linux (Ubuntu): Set Up the Wrapper

To sign your APK using APKSigner and our PKCS#11 wrapper in Ubuntu, please follow the below steps.

Steps:

Step 1: Download the PKCS#11 Wrapper for Ubuntu

- Go to EC CodeSign Secure v3.02's Signing Tools section and download the PKCS11 Wrapper for Ubuntu.



Step 2: Edit the Configuration Files

- Go to your Ubuntu client system and edit the configuration files (ec_pkcs11client.ini and pkcs11properties.cfg) downloaded in the PKCS11 Wrapper.

```

aryan@test:~/PKCS_Wrapper_Ubuntu$ cat ec_pkcs11client.ini
[logconfig_file]
name=/home/aryan/PKCS_Wrapper_Ubuntu/EC_PKCS11_CLIENT-LogConfig.xml

[server_url]
url=https://devcodesignsecure.encryptionconsulting.com

[pfxfile_path]
path=/home/aryan/PKCS_Wrapper_Ubuntu/Pkcs11Test.pfx

[pfxfile_passwd]
passwd=de48b6924cb8

[login_token]
username=Aryan@encryptionconsulting.com
passcode=8@!$U3QXS5Xtu2vf
aryan@test:~/PKCS_Wrapper_Ubuntu$ cat pkcs11properties.cfg
name=signingmanager
library=/home/aryan/PKCS_Wrapper_Ubuntu/ec_pkcs11client.so
attributes=compatibility
attributes(*, *, *) = {
CKA_TOKEN=true
}
aryan@test:~/PKCS_Wrapper_Ubuntu$

```

NOTE: When using Luna HSM, add the slot number in the pkcs11properties.cfg file as shown below.

```

aryan@test:~/PKCS_Wrapper_Ubuntu$ cat pkcs11properties.cfg
name=signingmanager
library=/home/aryan/PKCS_Wrapper_Ubuntu/ec_pkcs11client.so
slot=3
attributes=compatibility
attributes(*, *, *) = {
CKA_TOKEN=true
}

```

Step 3: Set the EC_INI_FILE_PATH Environment Variable

- Now we need to add the environment variable for the pkcs11 client library.
- Run the below command to open the bashrc file:

```
nano ~/.bashrc
```

```

administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ nano ~/.bashrc
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$

```

- Now add the EC_INI_FILE_PATH variable with the path of the .ini at the end of the bashrc file, like:

```

export EC_INI_FILE_PATH=\
/home/administrator/PKCS11_Wrapper-Ubuntu/ec_pkcs11client.ini

```

```

fi
export EC_INI_FILE_PATH=/home/administrator/PKCS11_Wrapper-Ubuntu/ec_pkcs11client.ini

```

^{^G} Help ^{^O} Write Out ^{^W} Where Is ^{^K} Cut ^{^T} Execute ^{^C} Locati
^{^X} Exit ^{^R} Read File ^{^Y} Replace ^{^U} Paste ^{^J} Justify ^{^_} Go To

- Press Ctrl+X, then enter Y, and then press Enter to save.
- Reload the environment variables:

```
source ~/.bashrc
```

```

administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ source ~/.bashrc
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$

```

- Check whether the variable has been set correctly:

```
echo $EC_INI_FILE_PATH
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996: /PKCS11_Wrapper-Ubuntu$ echo SEC_INI_FILE_PATH
/home/administrator/PKCS11_Wrapper-Ubuntu/ec_pkcs11client.ini
administrator@administrator-Standard-PC-i440FX-PIIX-1996: /PKCS11_Wrapper-Ubuntu$
```

NOTE: If you cannot see the file location from the echo command, open a new terminal and try again.

Step 4: Install the Prerequisites

- Now, let us install some prerequisites in your client system to run the PKCS11 Wrapper.
 - Install Java 8:

```
sudo apt install openjdk-8-jdk
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996: /PKCS11_Wrapper-Ubuntu$ sudo apt install openjdk-8-jdk
[sudo] password for administrator:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  fonts-dejavu-extra libatk-wrapper-java libatk-wrapper-java-jni libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libat-dev
  openjdk-8-jdk-headless openjdk-8-jre openjdk-8-jre-headless x11proto-dev xorg-sgml-doctools xtrans-dev
Suggested packages:
  libice-doc libsm-doc libxcb-doc libat-dev openjdk-8-demo openjdk-8-source visualvm fonts-nanum fonts-ipafont-gothic fonts-ipafont-mincho fonts-wqy-microhei
  fonts-wqy-zenhei fonts-indic
The following NEW packages will be installed:
  fonts-dejavu-extra libatk-wrapper-java libatk-wrapper-java-jni libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libxau-dev libxcb1-dev libxdmcp-dev libat-dev openjdk-8-jdk
  openjdk-8-jdk-headless openjdk-8-jre openjdk-8-jre-headless x11proto-dev xorg-sgml-doctools xtrans-dev
0 upgraded, 18 newly installed, 0 to remove and 54 not upgraded.
Need to get 47.7 MB of archives.
After this operation, 163 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu noble/main amd64 fonts-dejavu-extra all 2.37-0 [1,947 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libatk-wrapper-java all 0.40.0-1build2 [54.3 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libatk-wrapper-java-jni amd64 0.40.0-1build2 [46.4 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu noble/main amd64 xorg-sgml-doctools all 1:1.11-1 [10.9 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu noble/main amd64 x11proto-dev all 2023.2-1 [602 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libice-dev amd64 2:1.0.10-1build3 [51.0 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libpthread-stubs0-dev amd64 0.4-1build3 [4,746 B]
Get:8 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libsm-dev amd64 2:1.2.3-1build3 [17.0 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libxau-dev amd64 1:1.0.9-1build6 [9,570 B]
Get:10 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libxdmcp-dev amd64 1:1.1.3-0ubuntu6 [26.5 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu noble/main amd64 xtrans-dev all 1.4.0-1 [68.9 kB]
Get:12 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libxcb1-dev amd64 1:1.15-1ubuntu2 [35.0 kB]
Get:13 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libx11-dev amd64 2:1.8.7-1build1 [732 kB]
Get:14 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libat-dev amd64 1:1.2.1-1.2build1 [194 kB]
Get:15 http://us.archive.ubuntu.com/ubuntu noble-updates/universe amd64 openjdk-8-jre-headless amd64 8u432-ga-us1-0ubuntu2-24.04 [30.7 MB]
Get:16 http://us.archive.ubuntu.com/ubuntu noble-updates/universe amd64 openjdk-8-jre amd64 8u432-ga-us1-0ubuntu2-24.04 [74.0 kB]
Get:17 http://us.archive.ubuntu.com/ubuntu noble-updates/universe amd64 openjdk-8-jdk-headless amd64 8u432-ga-us1-0ubuntu2-24.04 [6,833 kB]
Get:18 http://us.archive.ubuntu.com/ubuntu noble-updates/universe amd64 openjdk-8-jdk amd64 8u432-ga-us1-0ubuntu2-24.04 [4,005 kB]
Fetched 47.7 MB in 2s (25.8 MB/s)
Selecting previously unselected package fonts-dejavu-extra.
(Reading database ... 188938 files and directories currently installed.)
Preparing to unpack .../00-fonts-dejavu-extra-2.37-0-all.deb ...
```

- Set Java 8 as the active version:

```
sudo update-alternatives --config java
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996: /PKCS11_Wrapper-Ubuntu$ sudo update-alternatives --config java
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                          Priority  Status
  -----
* 0            /usr/lib/jvm/java-21-openjdk-amd64/bin/java  2111     auto mode
  1            /usr/lib/jvm/java-21-openjdk-amd64/bin/java  2111     manual mode
  2            /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java 1081     manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java to provide /usr/bin/java (java) in manual mode
```

- Install the Android SDK command-line tools:

```
sudo apt install google-android-cmdline-tools-13.0-installer
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~$ sudo apt install google-android-cmdline-tools-13.0-installer
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java default-jre-headless gettext google-android-build-tools-34.0.0-installer google-android-licenses intltool-debian java-common libmail-sendmail-perl
  libsys-hostname-long-perl make openjdk-21-jre-headless po-debconf
Suggested packages:
  default-jre autopoint gettext-doc libasprintf-dev libgettextpo-dev make-doc fonts-dejavu-extra fonts-ipafont-gothic fonts-ipafont-nincho fonts-wqy-microhei | fonts-wqy-zenhei
  fonts-indic libmail-box-perl
The following NEW packages will be installed:
  ca-certificates-java default-jre-headless gettext google-android-build-tools-34.0.0-installer google-android-cmdline-tools-13.0-installer google-android-licenses intltool-debian
  java-common libmail-sendmail-perl libsys-hostname-long-perl make openjdk-21-jre-headless po-debconf
0 upgraded, 13 newly installed, 0 to remove and 84 not upgraded.
Need to get 47.8 MB of archives.
After this operation, 521 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu noble/main amd64 ca-certificates-java all 20240118 [11.6 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu noble/main amd64 java-common all 0.75+exp1 [6,798 B]
Get:3 http://us.archive.ubuntu.com/ubuntu noble-updates/main amd64 openjdk-21-jre-headless amd64 21.0.5+11-1ubuntu1-24.04 [46.4 MB]
Get:4 http://us.archive.ubuntu.com/ubuntu noble/main amd64 default-jre-headless amd64 2:1.21-75+exp1 [3,094 B]
Get:5 http://us.archive.ubuntu.com/ubuntu noble/main amd64 gettext amd64 0.21-14ubuntu2 [864 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu noble/main amd64 make amd64 4.3-4.1build2 [100 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu noble/main amd64 intltool-debian all 0.35.0+20060710.6 [23.2 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu noble/main amd64 po-debconf all 1.0.21+nmu1 [233 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu noble/multiverse amd64 google-android-licenses all 1710437545-3build2 [6,160 B]
Get:10 http://us.archive.ubuntu.com/ubuntu noble/multiverse amd64 google-android-build-tools-34.0.0-installer amd64 34.0.0+1710437545-3build2 [16.4 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu noble/multiverse amd64 google-android-cmdline-tools-13.0-installer amd64 13.0+1710437545-3build2 [16.4 kB]
Get:12 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libsys-hostname-long-perl all 1.5-3 [10.6 kB]
Get:13 http://us.archive.ubuntu.com/ubuntu noble/main amd64 libmail-sendmail-perl all 0.80-3 [21.7 kB]
Fetched 47.8 MB in 2s (29.6 MB/s)
Preconfiguring packages ...
Selecting previously unselected package ca-certificates-java.
(Reading database ... 187856 files and directories currently installed.)
Preparing to unpack .../00-ca-certificates-java_20240118_all.deb ...
Unpacking ca-certificates-java (20240118) ...
Selecting previously unselected package java-common.
Preparing to unpack .../01-java-common_0.75+exp1_all.deb ...
Unpacking java-common (0.75+exp1) ...
Selecting previously unselected package openjdk-21-jre-headless:amd64.
Preparing to unpack .../02-openjdk-21-jre-headless_21.0.5+11-1ubuntu1-24.04_amd64.deb ...
Unpacking openjdk-21-jre-headless:amd64 (21.0.5+11-1ubuntu1-24.04) ...
Selecting previously unselected package default-jre-headless.
Preparing to unpack .../03-default-jre-headless_2:1.21-75+exp1_amd64.deb ...
Unpacking default-jre-headless (2:1.21-75+exp1) ...
```

- Ensure that the SDK Manager for Android Studio has been properly installed:

```
sdkmanager --version
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~$ sdkmanager --version
13.0
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~$
```

- Install Build tools using SDKManager, which contains the APKSigner:

```
sdkmanager "build-tools;34.0.0"
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~$ sdkmanager "build-tools;34.0.0"
Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.
[=====] 100% Computing updates...
```

- Ensure that APKSigner is present:

```
apksigner --version
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~$ apksigner --version
0.9
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~$
```

- Two packages are required to run the PKCS11 Wrapper on your system. First, install liblog4cxx-dev:

```
sudo apt-get install liblog4cxx-dev
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ sudo apt-get install liblog4cxx-dev
[sudo] password for administrator:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
liblog4cxx-dev is already the newest version (1.1.0-1build3).
0 upgraded, 0 newly installed, 0 to remove and 81 not upgraded.
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$
```

- The last prerequisite is to install the curl package:

```
sudo apt-get install curl
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ sudo apt-get install curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 81 not upgraded.
Need to get 227 kB of archives.
After this operation, 534 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu noble-updates/main amd64 curl amd64 8.5.0-2ubuntu10.5 [227 kB]
Fetched 227 kB in 0s (694 kB/s)
Selecting previously unselected package curl.
(Reading database ... 191536 files and directories currently installed.)
Preparing to unpack .../curl_8.5.0-2ubuntu10.5_amd64.deb ...
Unpacking curl (8.5.0-2ubuntu10.5) ...
Setting up curl (8.5.0-2ubuntu10.5) ...
Processing triggers for man-db (2.12.0-4build2) ...
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$
```

3. Linux (Ubuntu): Sign and Verify

Now that all the configurations and prerequisites have been installed, let us perform the signing operation.

Steps:

Step 1: Run the Signing Command

- The signing command will look something like this. Ensure you run this command only inside the folder where your PKCS11 Wrapper is installed.

```
apksigner sign \
  --provider-class sun.security.pkcs11.SunPKCS11 \
  --provider-arg <path of the pkcs11properties.cfg file> \
  --ks NONE \
  --ks-type PKCS11 \
  --ks-pass pass:abcd1234 \
```

```
--ks-key-alias <private key alias> \  
--in <path of the APK file you want to sign> \  
--out <path of the Signed APK file>
```

- A sample command is provided below:

```
apksigner sign \  
  --provider-class sun.security.pkcs11.SunPKCS11 \  
  --provider-arg \  
    /home/administrator/PKCS11_Wrapper-Ubuntu/pkcs11properties.cfg \  
  --ks NONE --ks-type PKCS11 --ks-pass pass:abcd1234 \  
  --ks-key-alias gpg2 --in Sample.apk --out signed.apk
```

```
Processing triggered for non-OS (LUNA) security...  
administrator@Administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ apksigner sign --provider-class sun.security.pkcs11.SunPKCS11 --provider-arg /home/administrator/PKCS11_Wrapper-Ubuntu/pkcs11properties.cfg --ks NONE --ks-type PKCS11 --ks-pass pass:abcd1234 --ks-key-alias gpg2 --in Sample.apk --out signed.apk  
administrator@Administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ ls -l  
total 5544  
-rw-rw-r-- 1 administrator administrator 2880 Nov 19 09:31 AuthCert-nCipher.pfx  
-rw-rw-rw- 1 administrator administrator 286 Nov 19 09:32 ec_pkcs11client.ini  
-rw-rw-r-- 1 administrator administrator 146517 Nov 21 15:33 ec_pkcs11_client.log  
-rw-rw-rw- 1 administrator administrator 525 Oct 31 09:21 EC_PKCS11_CLIENT-LogConfig.xml  
-rw-rw-rw- 1 administrator administrator 839360 Nov 6 06:57 ec_pkcs11client.so  
-rw-rw-rw- 1 administrator administrator 155 Nov 19 09:34 pkcs11properties.cfg  
-rw-rw-r-- 1 administrator administrator 2317600 Oct 31 09:21 Sample.apk  
-rw-rw-r-- 1 administrator administrator 2320097 Nov 21 15:33 signed.apk  
-rw-rw-r-- 1 administrator administrator 26256 Nov 21 15:33 signed.apk.idsig
```

NOTE: When working with Luna HSM, append `--min-sdk-version 28` to the signing command shown above.

Step 2: Option Reference

- The following are the options and their meaning in the command:

Option	Description
<code>--provider-class</code>	The Java security provider to use. This is always <code>sun.security.pkcs11.SunPKCS11</code> , the JDK's built-in PKCS#11 provider.
<code>--provider-arg</code>	The path to the <code>pkcs11properties.cfg</code> file in your system, which tells the provider how to reach the PKCS#11 Wrapper.
<code>--ks</code>	The keystore. This is <code>NONE</code> because the keys live in the HSM rather than in a keystore file.
<code>--ks-type</code>	The keystore type, which must be <code>PKCS11</code> .
<code>--ks-pass</code>	The keystore password, supplied in the form <code>pass:<password></code> .
<code>--ks-key-alias</code>	The alias of the private key to sign with.
<code>--in</code>	The path of the APK file you want to sign.
<code>--out</code>	The path of the signed APK file to produce.

Option	Description
<code>--min-sdk-version</code>	The lowest Android API level the signature must be valid for. Required when working with a Luna HSM.

Step 3: Verify the Signature

- After successfully signing the APK, let us verify it using this command:

```
apksigner verify -verbose <path of the signed APK file>
```

- A sample command is provided below:

```
apksigner verify -verbose signed.apk
```

```
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$ apksigner verify -verbose signed.apk
Verifies
Verified using v1 scheme (JAR signing): true
Verified using v2 scheme (APK Signature Scheme v2): true
Verified using v3 scheme (APK Signature Scheme v3): true
Verified using v3.1 scheme (APK Signature Scheme v3.1): false
Verified using v4 scheme (APK Signature Scheme v4): false
Verified for SourceStamp: false
Number of signers: 1
administrator@administrator-Standard-PC-i440FX-PIIX-1996:~/PKCS11_Wrapper-Ubuntu$
```

NOTE: When working with Luna HSM, use: `apksigner verify -verbose --min-sdk-version 28 <path of the signed APK file>`

Step 4: Cross-Check in CodeSign Secure

- Open the CodeSign Secure portal and navigate to Reports > Signing Request Report, where every signing request performed through the wrapper is recorded for audit purposes.

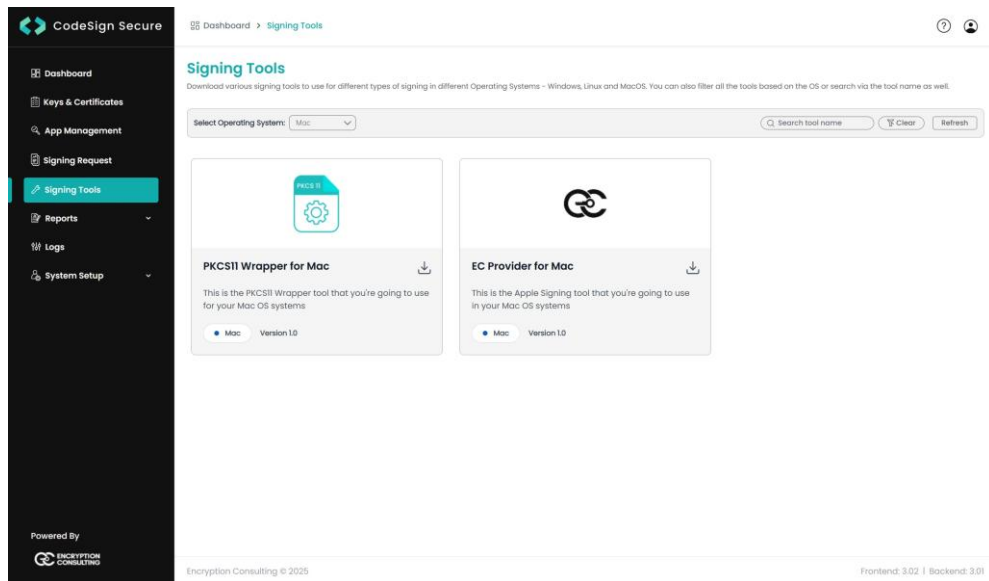
4. macOS: Set Up the Wrapper

To perform APK Signing using the APKSigner tool and our PKCS11 Wrapper on a macOS machine, you would need to follow the below steps.

Steps:

Step 1: Download the PKCS#11 Wrapper for macOS

- Go to EC CodeSign Secure v3.02's Signing Tools section and download the PKCS11 Wrapper for MacOS.



Step 2: Edit the Configuration Files

- Go to your macOS client system and edit the configuration files (ec_pkcs11client.ini and pkcs11properties.cfg) downloaded in the PKCS11 Wrapper.

```
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % cat ec_pkcs11client.ini
[logconfig_file]
name=/Users/subhayuroy/PKCS11_Wrapper-Mac/EC_PKCS11_CLIENT-LogConfig.xml

[server_url]
;url=https://lunacodesignsecure.encryptionconsulting.com
url=https://devcodesignsecure.encryptionconsulting.com
;url=http://127.0.0.1:6666

[pfxfile_path]
path=/Users/subhayuroy/PKCS11_Wrapper-Mac/AuthCert-nCipher.pfx

[pfxfile_passwd]
passwd=f4550251e262

[login_token]
username=Sam@encryptionconsulting.com
passcode=8@!$U3QXS5Xtu2vf
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % cat pkcs11properties.cfg
name=signingmanager
library=/Users/subhayuroy/PKCS11_Wrapper-Mac/ec_pkcs11client.dylib
attributes=compatibility
attributes(*, *, *) = {
CKA_TOKEN=true
}
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac %
```

NOTE: When using Luna HSM, add the slot number in the pkcs11properties.cfg file as shown below.

```
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % cat pkcs11properties.cfg
name=signingmanager
library=/Users/subhayuroy/PKCS11_Wrapper-Mac/ec_pkcs11client.dylib
slot=3
attributes=compatibility
attributes(*, *, *) = {
  CKA_TOKEN=true
}
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac %
```

Step 3: Install and Activate Java 17

- Install Java 17:

```
brew install openjdk@17
```

```
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % brew install openjdk@17
-- Downloading https://formulae.brew.sh/api/formula_jms.json 100.0%
-- Downloading https://formulae.brew.sh/api/cask_jms.json 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/openssl@1.1?name=feats/2.8.13 100.0%
-- Fetching dependencies for openssl@1.1: libpng, freetype, giflib, fontconfig, pcre2, python-packaging, openssl, sqlite, xz, python@3.11, libxml2, gettext, glib, xorgproto, libman, libjpeg, libwebp, libtiff, libltdl, libxrender, lz4, psmisc, xz, graphite2, harfbuzz, pangocairo, cairo, libffi and little-cms2 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/libpng?name=feats/1.6.44 100.0%
-- Fetching libpng 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/freetype?name=feats/2.13.3 100.0%
-- Fetching freetype 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/giflib?name=feats/5.2.2 100.0%
-- Fetching giflib 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/fontconfig?name=feats/2.15.0 100.0%
-- Fetching fontconfig 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/pcre2?name=feats/10.44 100.0%
-- Fetching pcre2 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/python-packaging?name=feats/24.2 100.0%
-- Fetching python-packaging 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/openssl@1.1?name=feats/4.0.0 100.0%
-- Fetching openssl@1.1 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/sqlite?name=feats/3.47.1 100.0%
-- Fetching sqlite 100.0%
-- Downloading https://ghcr.io/v2/homebrew/core/xz?name=feats/5.6.3 100.0%
Already downloaded: /Users/subhayuroy/Library/Caches/Homebrew/downloads/c6d799185cc173418bb280bdc3d9587c9f48375647d85775762a297299--xz-5.6.3.bottle.manifest.json
```

- Set Java 17 as the active version. For Zsh:

```
nano ~/.zshrc
```

```
UW PICO 5.09 File: /Users/subhayuroy/.zshrc

export PATH=/usr/local/bin:$PATH
export PATH=/Users/subhayuroy/Flutter/bin:$PATH
export LANG=en_US.UTF-8
export LANGUAGE=en_US.UTF-8
export LC_ALL=en_US.UTF-8
export PATH="$PATH":"$HOME/.pub-cache/bin"
export PATH=/Users/subhayuroy/PKCS11_Wrapper-Mac/cmdline-tools/bin:$PATH
export JAVA_HOME=$(/usr/libexec/java_home -v 17)
export PATH=$JAVA_HOME/bin:$PATH
```

- For Bash:

```
nano ~/.bash_profile
```

```
[subhayuroy@Subhayus-MacBook-Pro bin % /usr/libexec/java_home -v 17
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home
[subhayuroy@Subhayus-MacBook-Pro bin % nano ~/.zshrc
[subhayuroy@Subhayus-MacBook-Pro bin % source ~/.zshrc
subhayuroy@Subhayus-MacBook-Pro bin %
```

- Add these lines:

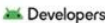
```
export JAVA_HOME=$(/usr/libexec/java_home -v 17)
export PATH=$JAVA_HOME/bin:$PATH
```

- And then run one of the following, matching your shell:

```
source ~/.zshrc
source ~/.bash_profile
```

Step 4: Install the Android SDK Command-Line Tools

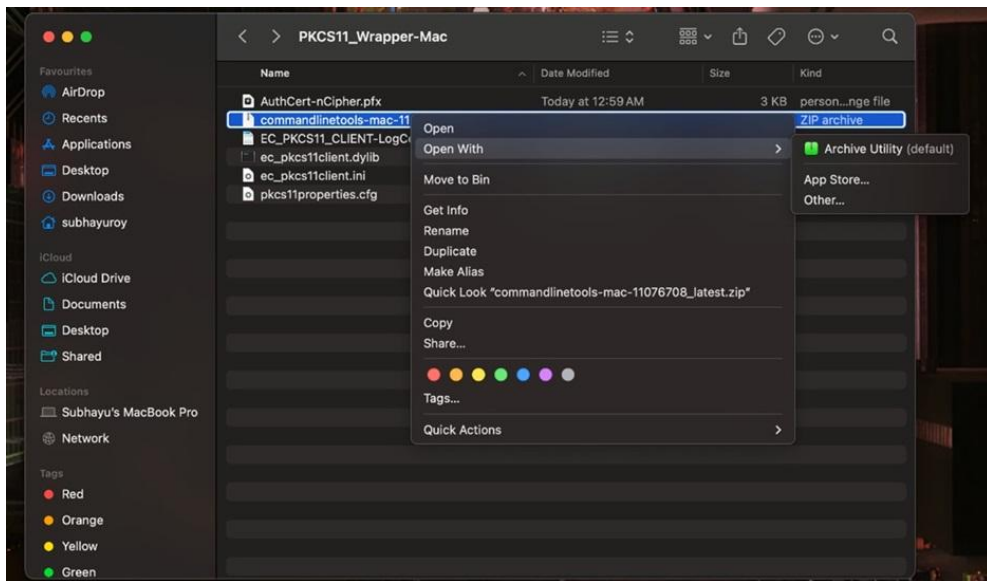
- Install the Android SDK command-line tools from this [site](#).

 Developers Essentials ▾ Develop ▾ More ▾   Language ▾ Android Studio 

Command line tools only

Platform	SDK tools package	Size	SHA-256 checksum
Windows	commandlinetools-win-11076708_latest.zip	153.6 MB	4d6931209eebb1bfb7c7e8b240a6a3cb3ab24479ea294f3539429574b1eec862
Mac	commandlinetools-mac-11076708_latest.zip	153.6 MB	7bc5c72ba0275c80a8f19684fb92793b83a6b5c94d4d179fc5988930282d7e64
Linux	commandlinetools-linux-11076708_latest.zip	153.6 MB	2d2d50857e4eb553af5a6dc3ad507a17adf43d115264b1afc116f95c92e5e258

Command-line tools are included in Android Studio. If you do not need Android Studio, you can download the basic Android command-line tools above. You can use the included [sdkmanager](#) to download other SDK packages.



- Ensure that the SDK Manager for Android Studio has been properly installed:

```
sdksmanager --sdk_root=/Users/subhayuroy/PKCS11_Wrapper-Mac \
--version
```

```
subhayuroy@Subhayus-MacBook-Pro bin % sdksmanager --sdk_root=/Users/subhayuroy/PKCS11_Wrapper-Mac --version
12.0
subhayuroy@Subhayus-MacBook-Pro bin %
```

- Install Build tools using SDKManager, which contains the APKSigner:

```
sdksmanager --sdk_root=/Users/subhayuroy/PKCS11_Wrapper-Mac \
"build-tools;34.0.0"
```

```
subhayuroy@Subhayus-MacBook-Pro bin % sdksmanager --sdk_root=/Users/subhayuroy/PKCS11_Wrapper-Mac "build-tools;34.0.0"
Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.
License android-sdk-license:
[ 100% Computing updates...
Terms and Conditions
This is the Android Software Development Kit License Agreement
1. Introduction
1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.
1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).
1.4 "Google" means Google Inc., a Delaware corporation with principal place of business at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.
2. Accepting the License Agreement
2.1 In order to use the SDK, you must first agree to the License Agreement. You may not use the SDK if you do not accept the License Agreement.
2.2 By clicking to accept, you hereby agree to the terms of the License Agreement.
2.3 You may not use the SDK and may not accept the License Agreement if you are a person barred from receiving the SDK under the laws of the United States or other countries, including the country in which you are resident or from which you use the SDK.
2.4 If you are agreeing to be bound by the License Agreement on behalf of your employer or other entity, you represent and warrant that you have full legal authority to bind your employer or such entity to the License Agreement. If you do not have the requisite authority, you may not accept the license Agreement or use the SDK on behalf of your employer or other entity.
```

- Ensure that APKSigner is present:

```
ls /Users/subhayuroy/PKCS11_Wrapper-Mac/build-tools/34.0.0/\
apksigner
```

```
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % ls /Users/subhayuroy/PKCS11_Wrapper-Mac/build-tools/34.0.0/apksigner

/Users/subhayuroy/PKCS11_Wrapper-Mac/build-tools/34.0.0/apksigner
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac %
```

Step 5: Install the Remaining Packages

- Two packages are required to run the PKCS11 Wrapper on your system. First, install the log4cxx library:

```
brew install log4cxx
```

[illegible]

- The last prerequisite is to install the curl package:

```
brew install curl
```

[illegible]

Step 6: Add the Relative Paths to Your PATH

- You need to ensure all the relative paths are added to your PATH variable in the ~/.zshrc file:

```
export PATH=/Users/subhayuroy/PKCS11_Wrapper-Mac/cmdline-tools/\
bin:$PATH
export JAVA_HOME=$(/usr/libexec/java_home -v 17)
export PATH=$JAVA_HOME/bin:$PATH
export ANDROID_SDK_ROOT=/Users/subhayuroy/PKCS11_Wrapper-Mac
export PATH=$PATH:/Users/subhayuroy/PKCS11_Wrapper-Mac/\
build-tools/34.0.0
```

UW PICO 5.09

```
export PATH=/usr/local/bin:$PATH
export PATH=/Users/subhayuroy/flutter/bin:$PATH
export LANG=en_US.UTF-8
export LANGUAGE=en_US.UTF-8
export LC_ALL=en_US.UTF-8
export PATH="$PATH":"$HOME/.pub-cache/bin"
export PATH=/Users/subhayuroy/PKCS11_Wrapper-Mac/cmdline-tools/bin:$PATH
export JAVA_HOME=$(/usr/libexec/java_home -v 17)
export PATH=$JAVA_HOME/bin:$PATH
export ANDROID_SDK_ROOT=/Users/subhayuroy/PKCS11_Wrapper-Mac
export PATH=$PATH:/Users/subhayuroy/PKCS11_Wrapper-Mac/build-tools/34.0.0
```

5. macOS: Sign and Verify

Now that all the configurations and prerequisites have been installed, let us perform the signing operation. On macOS the signing command invokes the APKSigner jar through java directly, so that the PKCS#11 module can be exported to the tool.

Steps:

Step 1: Run the Signing Command

- The signing command will look something like this. Ensure you run this command only inside the folder where your PKCS11 Wrapper is installed.

```
java \  
  --add-exports=jdk.crypto.cryptoki/sun.security.pkcs11=ALL-UNNAMED \  
  -jar <path of the apksigner.jar in your system> sign \  
  --provider-class sun.security.pkcs11.SunPKCS11 \  
  --provider-arg <path of the pkcs11properties.cfg file> \  
  --ks NONE \  
  --ks-type PKCS11 \  
  --ks-pass pass:abcd1234 \  
  --ks-key-alias <private key alias> \  
  --in <path of the APK file you want to sign> \  
  --out <path of the Signed APK file>
```

- A sample command is provided below:

```
java \  
  --add-exports=jdk.crypto.cryptoki/sun.security.pkcs11=ALL-UNNAMED \  
  -jar /Users/subhayuroy/PKCS11_Wrapper-Mac/build-tools/34.0.0/  
lib/apksigner.jar \  
  sign \  
  --provider-class sun.security.pkcs11.SunPKCS11 \  
  --provider-arg \  
    /Users/subhayuroy/PKCS11_Wrapper-Mac/pkcs11properties.cfg \  
  --ks NONE \  
  --ks-type PKCS11 \  
  --ks-pass pass:abcd1234 \  
  --ks-key-alias gpg2 \  
  --in Sample.apk \  
  --out signed.apk
```

```

subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % sudo java --add-exports=jdk.crypto.cryptoki/sun.security.pkcs11-ALL-UNNAMED \
-jar /Users/subhayuroy/PKCS11_Wrapper-Mac/build-tools/34.0.0/lib/apksigner.jar \
sign \
--provider-class sun.security.pkcs11.SunPKCS11 \
--provider-arg /Users/subhayuroy/PKCS11_Wrapper-Mac/pkcs11properties.cfg \
--ks NONE \
--ks-type PKCS11 \
--ks-pass pass:abcd1234 \
--ks-key-alias gpg2 \
--in Sample.apk \
--out signed.apk
Password:
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % ls -al
total 12832
drwxr-xr-x@ 16 subhayuroy  staff   512 Dec  5 02:08 .
drwxr-xr-x@  71 subhayuroy  staff  2272 Dec  4 17:54 ..
-rw-r--r--@  1 subhayuroy  staff   6148 Dec  4 18:20 .DS_Store
drwxr-xr-x@  2 subhayuroy  staff    64 Dec  4 18:22 .temp
-rw-r--r--@  1 subhayuroy  staff   2880 Dec  4 00:59 AuthCert-nCipher.pfx
-rw-r--r--@  1 subhayuroy  staff    525 Oct 31 09:21 EC_PKCS11_CLIENT-LogConfig.xml
-rw-r--r--@  1 subhayuroy  staff 2317600 Oct 31 19:51 Sample.apk
drwxr-xr-x@  3 subhayuroy  staff    96 Dec  4 18:22 build-tools
drwxr-xr-x@  7 subhayuroy  staff   224 Dec  4 18:19 cmdline-tools
-rw-r--r--@  1 root        staff  295540 Dec  5 02:08 ec_pkcs11_client.log
-rw-r--r--@  1 subhayuroy  staff  700568 Nov  6 09:00 ec_pkcs11client.dylib
-rw-r--r--@  1 subhayuroy  staff    360 Dec  4 01:00 ec_pkcs11client.ini
drwxr-xr-x@  3 subhayuroy  staff    96 Dec  4 18:21 licenses
-rw-r--r--@  1 subhayuroy  staff   153 Dec  4 00:57 pkcs11properties.cfg
-rw-r--r--@  1 root        staff 2320897 Dec  5 02:08 signed.apk
-rw-r--r--@  1 root        staff   26256 Dec  5 02:08 signed.apk.idsig
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac %

```

NOTE: The `--add-exports` option is required because the JDK used here keeps the SunPKCS11 classes in an encapsulated module. The Linux track does not need it because it runs on Java 8, which predates the module system.

Step 2: Verify the Signature

- After successfully signing the APK, let us verify it using this command:

```
apksigner verify -verbose <path of the signed APK file>
```

- A sample command is provided below:

```
apksigner verify -verbose signed.apk
```

```

subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac % apksigner verify -verbose signed.apk
Verifies
Verified using v1 scheme (JAR signing): true
Verified using v2 scheme (APK Signature Scheme v2): true
Verified using v3 scheme (APK Signature Scheme v3): true
Verified using v3.1 scheme (APK Signature Scheme v3.1): false
Verified using v4 scheme (APK Signature Scheme v4): false
Verified for SourceStamp: false
Number of signers: 1
subhayuroy@Subhayus-MacBook-Pro PKCS11_Wrapper-Mac %

```

Step 3: Cross-Check in CodeSign Secure

- Open the CodeSign Secure portal and navigate to Reports > Signing Request Report, where every signing request performed through the wrapper is recorded for audit purposes.

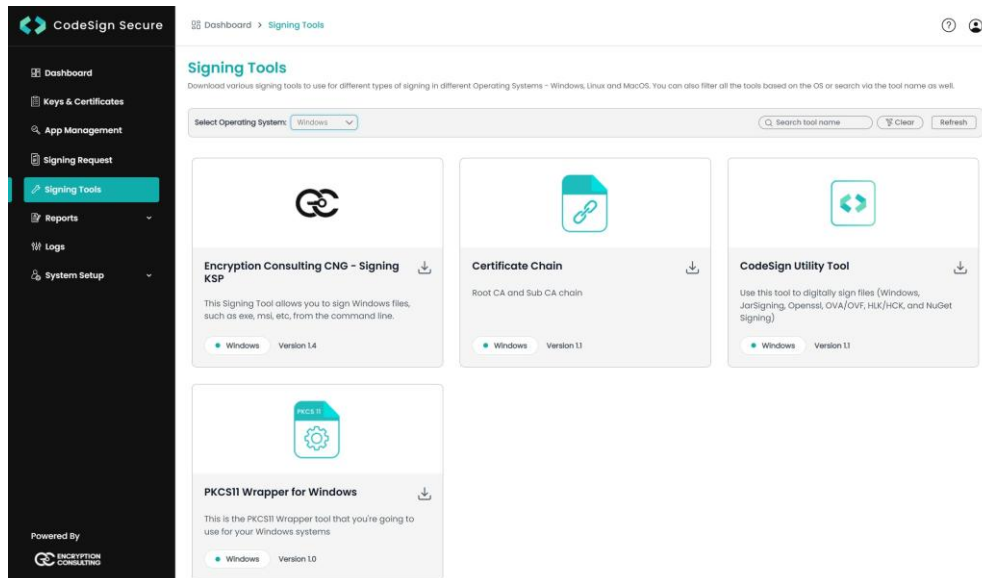
6. Windows: Set Up the Wrapper

To perform APK Signing using the APKSigner tool and our PKCS11 Wrapper on a Windows machine, you would need to follow the below steps.

Steps:

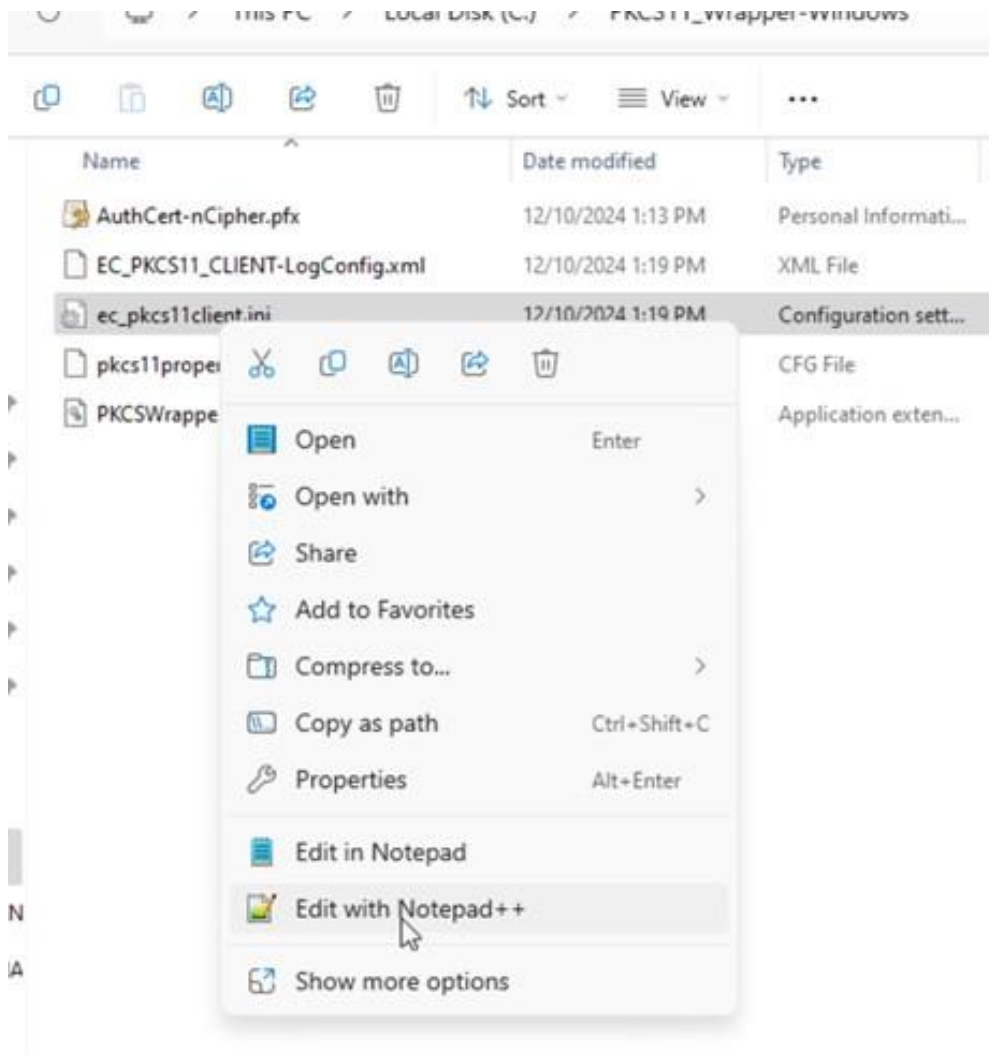
Step 1: Download the PKCS#11 Wrapper for Windows

- Go to EC CodeSign Secure v3.02's Signing Tools section and download the PKCS11 Wrapper for Windows.

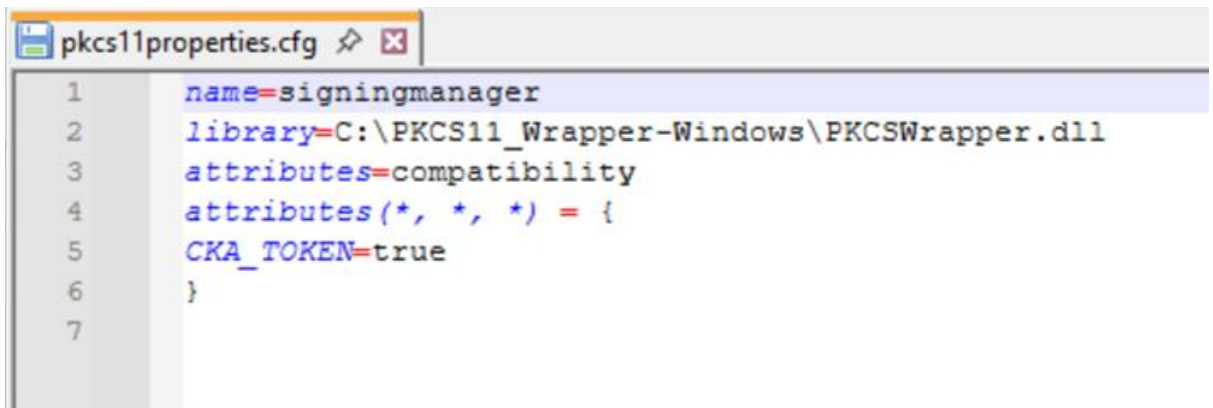


Step 2: Edit the Configuration Files

- Go to your Windows client system and edit the configuration files (ec_pkcs11client.ini and pkcs11properties.cfg) downloaded in the PKCS11 Wrapper.

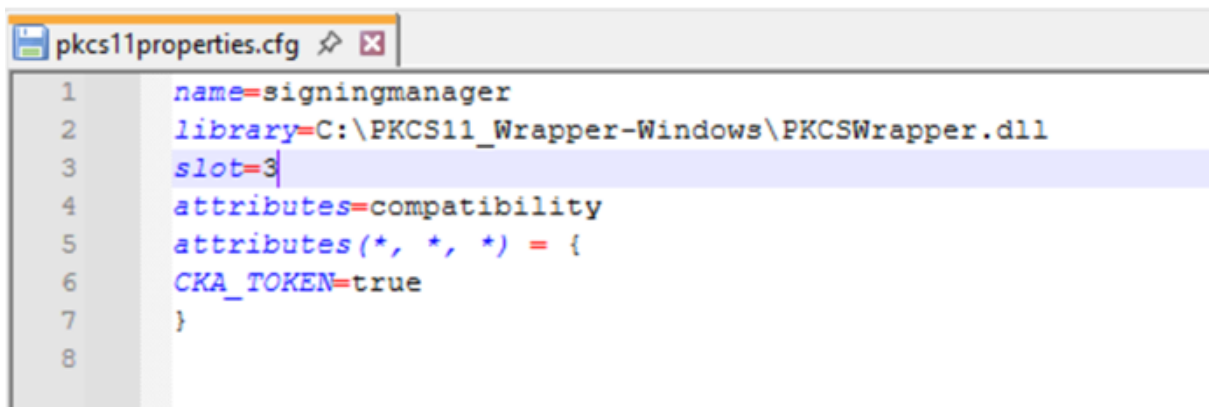


```
ec_pkcs11client.ini
1  [logconfig_file]
2  name=C:\PKCS11_Wrapper-Windows\EC_PKCS11_CLIENT-LogConfig.xml
3
4  [server_url]
5  url=https://devcodesignsecure.encryptionconsulting.com
6
7  [pfxfile_path]
8  path=C:\PKCS11_Wrapper-Windows\Pkcs11Test.pfx
9
10 [pfxfile_passwd]
11 passwd=bb881220628e
12
13 [login_token]
14 username=Aryan@encryptionconsulting.com
15 passcode=8@!$U3QXS5Xtu2vf
16
17
```



```
1  name=signingmanager
2  library=C:\PKCS11_Wrapper-Windows\PKCSWrapper.dll
3  attributes=compatibility
4  attributes(*, *, *) = {
5  CKA_TOKEN=true
6  }
7
```

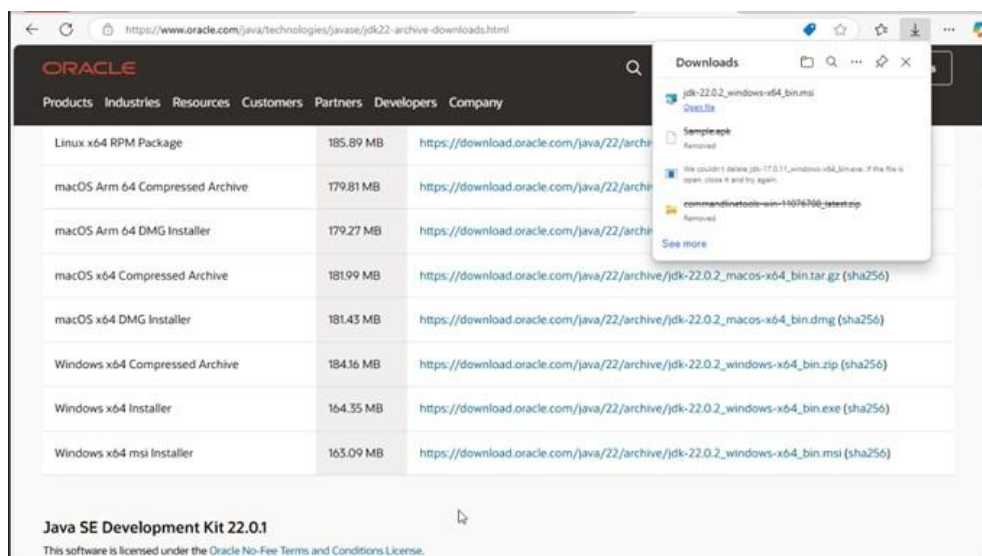
NOTE: When using Luna HSM, add the slot number in the pkcs11properties.cfg file as shown below.



```
1  name=signingmanager
2  library=C:\PKCS11_Wrapper-Windows\PKCSWrapper.dll
3  slot=3
4  attributes=compatibility
5  attributes(*, *, *) = {
6  CKA_TOKEN=true
7  }
8
```

Step 3: Install and Activate Java 22

- Install Java 22 from [Oracle's official site](https://www.oracle.com/java/technologies/javase/jdk-22-archive-downloads.html), and follow the instructions in the msi file.

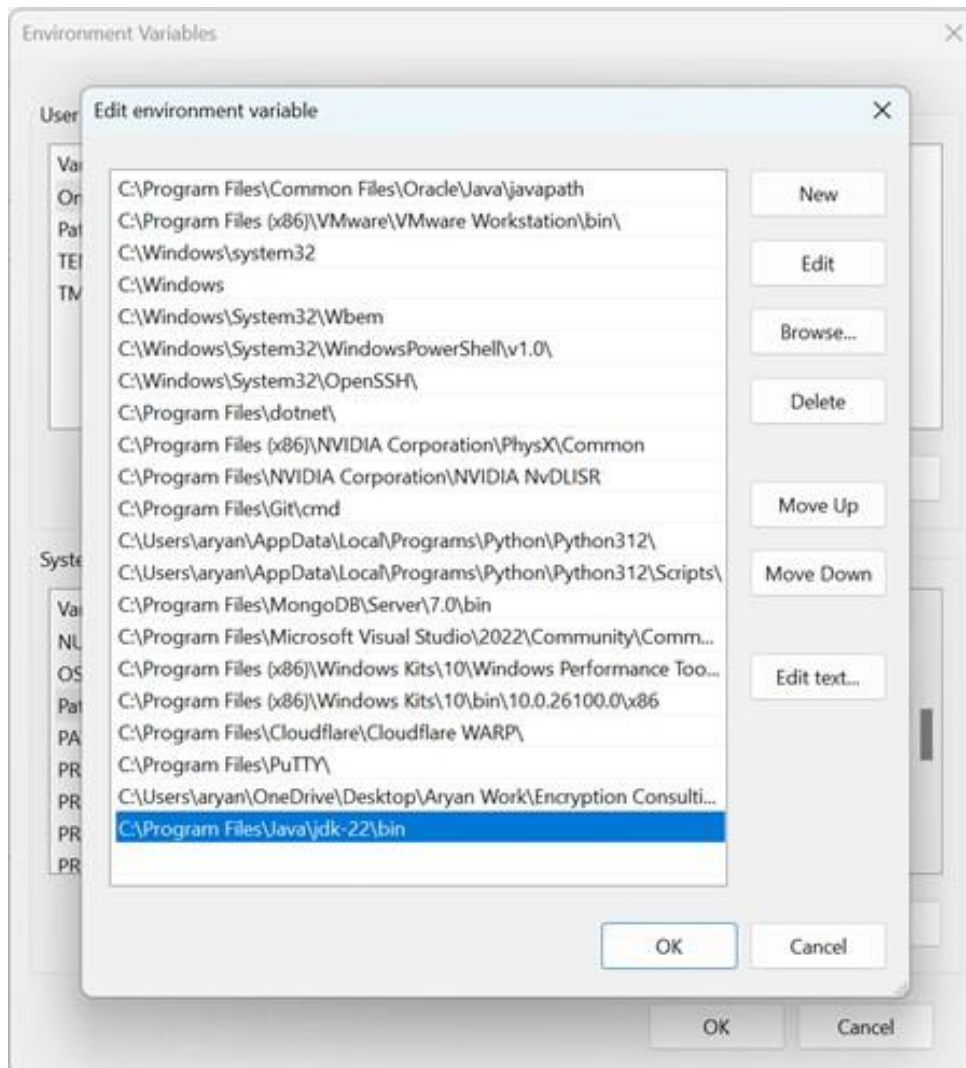


Operating System / Architecture	File Name	Size	Download Link
Linux x64 RPM Package	jdk-22.0.2_linux-x64_bin.rpm	185.89 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_linux-x64_bin.rpm
macOS Arm 64 Compressed Archive	jdk-22.0.2_macos-arm64_bin.tar.gz	179.81 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_macos-arm64_bin.tar.gz
macOS Arm 64 DMG Installer	jdk-22.0.2_macos-arm64_bin.dmg	179.27 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_macos-arm64_bin.dmg
macOS x64 Compressed Archive	jdk-22.0.2_macos-x64_bin.tar.gz	181.99 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_macos-x64_bin.tar.gz
macOS x64 DMG Installer	jdk-22.0.2_macos-x64_bin.dmg	181.43 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_macos-x64_bin.dmg
Windows x64 Compressed Archive	jdk-22.0.2_windows-x64_bin.zip	184.16 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_windows-x64_bin.zip
Windows x64 Installer	jdk-22.0.2_windows-x64_bin.exe	164.35 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_windows-x64_bin.exe
Windows x64 MSI Installer	jdk-22.0.2_windows-x64_bin.msi	165.09 MB	https://download.oracle.com/java/22/archive/jdk-22.0.2_windows-x64_bin.msi

Java SE Development Kit 22.0.1
This software is licensed under the Oracle No-Fee Terms and Conditions License.



- Set Java 22 as the active version by storing the bin path in the PATH variable.



Step 4: Install the Android SDK Command-Line Tools

- Install the Android SDK command-line tools from this link [here](#).

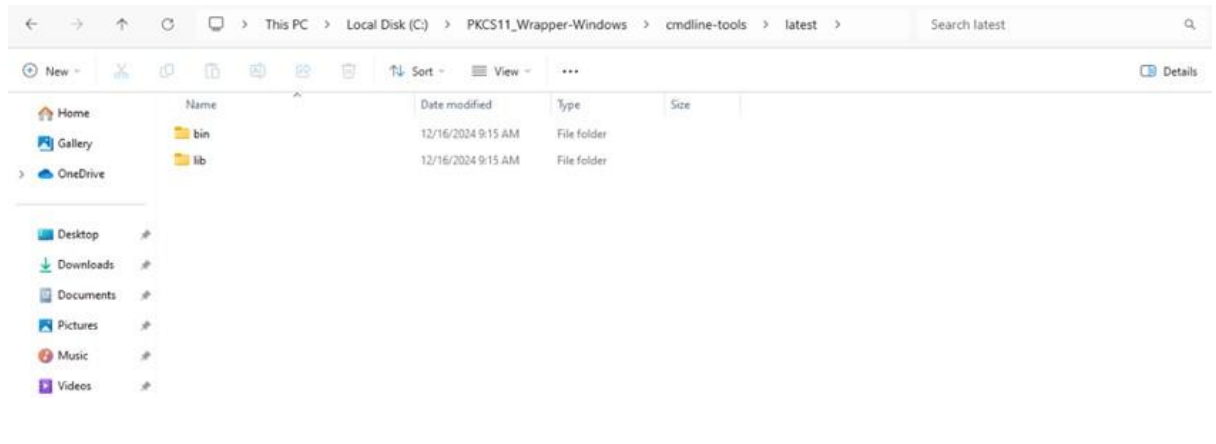


Command line tools only

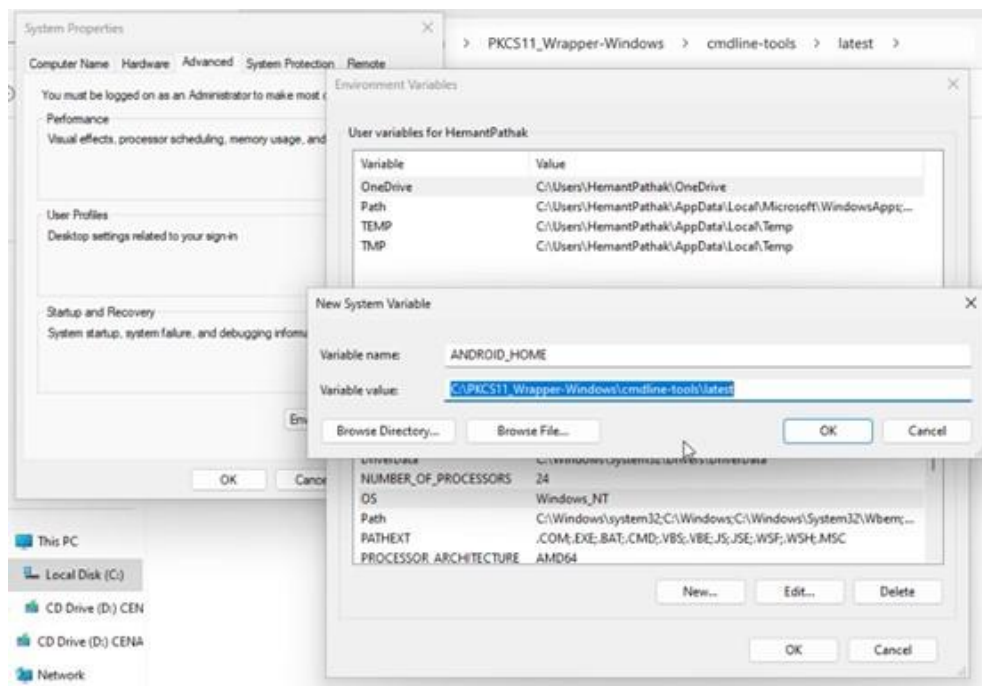
Platform	SDK tools package	Size	SHA-256 checksum
Windows	commandlinetools-win-11076708_latest.zip	153.6 MB	4d6931209eebb1bf7c7e8b240a6a3cb3ab24479ea294f3539429574b1eec862
Mac	commandlinetools-mac-11076708_latest.zip	153.6 MB	7bc5c72ba0275c80a8f19684fb92793b83a6b5c94d4d179fc5988930282d7e64
Linux	commandlinetools-linux-11076708_latest.zip	153.6 MB	2d2d50857e4eb553af5a6dc3ad507a17adf43d115264b1afc116f95c92e5e258

Command-line tools are included in Android Studio. If you do not need Android Studio, you can download the basic Android command-line tools above. You can use the included [sdkmanager](#) to download other SDK packages.

- Extract the files into a “cmdline-tools” folder and create a subfolder named latest. Now, move the bin and lib folders into the latest folder.



- Set an environment variable called ANDROID_HOME and set it to the path where you extracted the command line tools.



- Install Build tools using SDKManager, which contains the APKSigner:

```
.\bin\sdkmanager --channel=0 --install "build-tools;34.0.0"
```

```
C:\PKCS11_Wrapper-Windows\cmdline-tools\latest>.\bin\sdkmanager.bat --channel=0 --install "build-tools;34.0.0"
Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.
Warning: Errors during XML parse:
Warning: Additionally, the fallback loader failed to parse the XML.
[=====] 52% Unzipping... android-14/renderscri
```

- Ensure that APKSigner is present:

```
Apksigner.bat --version
```

```
C:\PKCS11_Wrapper-Windows\build-tools\31.0.0>apksigner.bat --version
0.9
C:\PKCS11_Wrapper-Windows\build-tools\31.0.0>
```

Step 5: Patch Apksigner.bat for the PKCS#11 Module

- Open the Apksigner.bat file and note its current last line.

```
if "%a:~0,2%" NEQ "-J" goto notJ
    set javaOpts=%javaOpts% -%a:~2%
    shift /1
    goto firstArg

:notJ
set params=%params% %1
shift /1
goto firstArg

:endArgs

set javaOpts=%javaOpts% %defaultXmx% %defaultXss%
call "%java_exe%" %javaOpts% -jar "%jarpath%" %params%
```

- Update the last line to the below command:

```
call "%java_exe%" --add-exports ^
jdk.crypto.cryptoki/sun.security.pkcs11=ALL-UNNAMED ^
%javaOpts% -jar "%jarpath%" %params%
```

```

if "%a:~0,5%" NEQ "-JXss" goto notXss
set defaultXss=
:notXss

if "%a:~0,2%" NEQ "-J" goto notJ
set javaOpts=%javaOpts% -%a:~2%
shift /1
goto firstArg

:notJ
set params=%params% %1
shift /1
goto firstArg

:endArgs

set javaOpts=%javaOpts% %defaultXmx% %defaultXss%
call "%java_exe%" --add-exports jdk.crypto.cryptoki/sun.security.pkcs11=ALL-UNNAMED %javaOpts% -jar "%jarpath%" %params%

```

NOTE: This is a single logical line in the batch file. The caret characters shown above are batch line continuations, so you may either keep them or join the three lines into one.

NOTE: This edit performs the same job as the --add-exports option used in the macOS command, so the Windows signing command itself stays short.

7. Windows: Sign and Verify

Now that all the configurations and prerequisites have been installed, let us perform the signing operation.

Steps:

Step 1: Run the Signing Command

- The signing command will look something like this. Ensure you run this command only inside the folder where your PKCS11 Wrapper is installed.

```

apksigner sign ^
--provider-class sun.security.pkcs11.SunPKCS11 ^
--provider-arg <path of the pkcs11properties.cfg file> ^
--ks NONE ^
--ks-type PKCS11 ^
--ks-pass pass:abcd1234 ^
--ks-key-alias <private key alias> ^
--in <path of the APK file you want to sign> ^
--out <path of the Signed APK file>

```

- A sample command is provided below:

```

apksigner sign ^
--provider-class sun.security.pkcs11.SunPKCS11 ^
--provider-arg ^
C:\Users\riley\Downloads\PKCS11_Wrapper-Windows\pkcs11properties.cfg ^
--ks NONE --ks-type PKCS11 --ks-pass pass:secretpassword ^
--ks-key-alias gpg2 --in Sample.apk --out signed.apk

```

NOTE: The caret characters are Command Prompt line continuations. You may keep them or join the command onto a single line.

```
C:\Users\HemantPathak\Desktop\PKCS11_Wrapper-Windows>apksigner sign --provider-class sun.security.pkcs11.SunPKCS11 --provider-arg pkcs11pr
operties.cfg --ks NONE --ks-type PKCS11 --ks-pass pass:secretpassword --ks-key-alias gpg2 --in Sample.apk --out signed.apk
nCipher Corp. Ltd
---Initializing curl-----slot description---
---api name ---C_OpenSession
---Session Handle----- 2260
--- utc time ---2260
Mechanism: CKM_SHA3_224_RSA_PKCS_PSS not found in mapping.
Mechanism: CKM_SHA3_256_RSA_PKCS_PSS not found in mapping.
Mechanism: CKM_SHA3_384_RSA_PKCS_PSS not found in mapping.
Mechanism: CKM_SHA3_512_RSA_PKCS_PSS not found in mapping.
Mechanism: CKM_SHA3_224_RSA_PKCS not found in mapping.
Mechanism: CKM_SHA3_256_RSA_PKCS not found in mapping.
Mechanism: CKM_SHA3_384_RSA_PKCS not found in mapping.
Mechanism: CKM_SHA3_512_RSA_PKCS not found in mapping.
Mechanism: CKM_EC_MONTGOMERY_KEY_PAIR_GEN not found in mapping.
Mechanism: CKM_EC_EDWARDS_KEY_PAIR_GEN not found in mapping.
```

NOTE: The --ks-pass value must match the password configured for your wrapper. The sample above uses a different value from the Linux and macOS samples, so use whichever applies to your own configuration.

Step 2: Verify the Signature

- After successfully signing the APK, let us verify it using this command:

```
apksigner verify -verbose <path of the signed APK file>
```

- A sample command is provided below:

```
apksigner verify -verbose signed.apk
```

```
C:\Users\HemantPathak\Desktop\PKCS11_Wrapper-Windows>apksigner verify -verbose signed.apk
Verifies
Verified using v1 scheme (JAR signing): true
Verified using v2 scheme (APK Signature Scheme v2): true
Verified using v3 scheme (APK Signature Scheme v3): true
Verified using v3.1 scheme (APK Signature Scheme v3.1): false
Verified using v4 scheme (APK Signature Scheme v4): false
Verified for SourceStamp: false
Number of signers: 1
C:\Users\HemantPathak\Desktop\PKCS11_Wrapper-Windows>
```

Step 3: Cross-Check in CodeSign Secure

- Open the CodeSign Secure portal and navigate to Reports > Signing Request Report, where every signing request performed through the wrapper is recorded for audit purposes.