

App signing
Manage and protect your app signing key. [Show more](#)

Looking for app integrity?
These features have moved:
• Play Integrity API
• Store listing visibility based on integrity checks
[Go to App integrity](#) [Dismiss](#)

App signing key certificate
This is the public certificate for the app signing key that Google uses to sign each of your releases. Use it to register your key with API providers. The app signing key itself is not accessible, and is kept on a secure Google server.

Certificate fingerprint	Fingerprint
MD5 certificate fingerprint	55 79 12 18 86 79 54 F8 86 37 25 24 24 46 27 35
SHA-1 certificate fingerprint	40 54 38 45 12 4F 20 1A 81 43 98 83 F8 68 13 64 64 24 48 54
SHA-256 certificate fingerprint	58 50 85 54 48 22 76 F5 F1 88 89 27 5E F1 52 23 58 85 83 18 98 13 63 86 84 84 88 63 84 34 34 A2

Upgrade your app signing key
Your current app signing key's encryption strength meets or exceeds Google Play's recommended minimum standard.
You should upgrade your app signing key to move your users to a new key. [Learn more](#)
Request key upgrade

Upload key certificate

Upgrade app signing key

This upgrade will apply your new key on Android N (API level 24) and above. New and existing installs on older Android releases will ignore the new key until they update to Android N or above. [Learn more](#)

How would you like to upgrade your app signing key?

- ☐ Let Google Play generate a new app signing key (recommended)
Google will automatically generate an app signing key with strong encryption. The private key will be held on Google's secure servers. You will be able to download the public certificate from the app signing page in order to register the key with API providers.
- ☐ Use the same app signing key as another app in this developer account
- ☒ Upload a new app signing key from Java KeyStore
when downloading the encryption_public_key.pem it will change every single time this entire window is closed and reopened if this process has to be done a second time to reset the signing key...
 - [Download encryption public key](#)
 - [Download PEPK tool](#)
Download the Play Encrypt Private Key (PEPK) tool. [Download source code](#)
 - Run the tool using the command below, which will export and encrypt your private key and its public key and sign the encrypted private key with your current upload key. Ensure that you replace the arguments highlighted in bold. Then enter your store and key passwords when prompted.


```
$ java -jar pepk.jar --keystore=foo.keystore --alias=foo --output=output.zip
--signing-keystore=uploadkey.keystore --signing-key-alias=upload-key-alias
--rsa-aes-encryption --encryption-key-path=/path/to/encryption_public_key.pem
```
 - [Upload generated ZIP](#) file should be called output.zip
- ☐ Upload a new app signing key from another repository

Reason for requesting an app signing key upgrade: Select a reason

Upgrade your key everywhere you distribute your app if you want to maximise app update compatibility

[Cancel](#) [Upgrade](#)

Suggest using “randomstring”

yum install curl -y

or

apt-get update && apt install curl -y

curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash

source ~/.bashrc

```
nvm install 21
nvm use 21
npm -g install randomstring
```

```
yum install java-17-openjdk-headless java-17-openjdk java-17-openjdk-devel
```

```
randomstring > google-key.txt
```

```
cp -rf ~/Downloads/pepk.jar
cp -rf ~/Downloads/encryption_public_key.pem
```

this string should be saved its your random password for the signing key
and my-release-key.jks should be kept safe

```
cat google-key.txt
```

```
keytool -genkey -v -keystore my-release-key.jks -alias alias_name -keyalg RSA -keysize 2048 -validity 10000
```

```
java -jar pepk.jar --keystore=my-release-key.jks --alias=alias_name --output=output.zip --include-cert --rsa-aes-encryption --encryption-key-path=encryption_public_key.pem
```

output.zip will be created here use this to upload to the request key update

after ./gradlew assembleRelease to create the .apk

creating appBundle .aab usually have to take place manually with the project folder as ./gradlew bundleRelease

acutally signing the appbundle .aab (using the exact same key that matches from my-release-key.jks google-key.txt)

```
jarsigner -verbose -sigalg SHA256withRSA -digestalg SHA-256 -keystore my-release-key.jks APP_NAME__arm64-v8a-release.aab alias_name
```

see also cordova for actually building html/js applications completely

```
npm -g install cordova serve
```