**Signing Code in Android and iOS Applications**

**How to Sign Code in Android and iOS Apps?**

**Icon

Description automatically generated**

**Table of Contents**

[Introduction 4](#_Toc79504824)

[What is Code Signing? 4](#_Toc79504825)

[The Operating Procedure of Code Sign 4](#_Toc79504826)

[What is Code Signing Certificate? 4](#_Toc79504827)

[To Increase the Data Security 4](#_Toc79504828)

[Change of Technology in Fast Paced Manner 4](#_Toc79504829)

[Who Uses Code Signing? 4](#_Toc79504830)

[Why Code Sign for Android Apps? 5](#_Toc79504831)

[How to Sign Mobile App Code for Google Play store? 5](#_Toc79504832)

[1. To Generate an Upload Key 5](#_Toc79504833)

[2. To Sign the App with Key 6](#_Toc79504834)

[3. To Configure New App in Play App Signing 8](#_Toc79504835)

[4. To Upload the Signed App in Google Play 10](#_Toc79504836)

[Why Code Sign for iOS Apps? 10](#_Toc79504837)

[How to Sign Mobile App Code for Apple Store? 10](#_Toc79504838)

[1. To Create a CSR from your Local macOS Machine 10](#_Toc79504839)

[2. To Generate a Distribution Certificate 12](#_Toc79504840)

[3. To Create App Id 13](#_Toc79504841)

[4. Provisioning the Profiles 14](#_Toc79504842)

[5. To configure XCode to use Provisioning Profile 17](#_Toc79504843)

[6. To Create an App Profile 17](#_Toc79504844)

[7. To Upload the App using XCode 19](#_Toc79504845)

[8. To Publish the App 22](#_Toc79504846)

[Conclusion 25](#_Toc79504847)

**List of Figures**

[Figure 1: Code Signing for Google Play Store 5](#_Toc79504892)

[Figure 2: Create a New Upload Key 6](#_Toc79504893)

[Figure 3: Signing the App with the Product Key 7](#_Toc79504894)

[Figure 4: Signing the App with the Product Key 8](#_Toc79504895)

[Figure 5: Pop-up Menu 8](#_Toc79504896)

[Figure 6: Configure Play App Signing 9](#_Toc79504897)

[Figure 7: Certificate Signing Request 11](#_Toc79504898)

[Figure 8: Certificate Information 11](#_Toc79504899)

[Figure 9: Certificate Conclusion 12](#_Toc79504900)

[Figure 10: Generate Distribution Certificate 13](#_Toc79504901)

[Figure 11: App ID Creation 14](#_Toc79504902)

[Figure 12: Provisioning the Profiles 16](#_Toc79504903)

[Figure 13: XCode Signing 17](#_Toc79504904)

[Figure 14: App profile Creation Menu 19](#_Toc79504905)

[Figure 15: Upload App Using XCode 21](#_Toc79504906)

[Figure 16: Publishing the App 24](#_Toc79504907)

**Introduction**

Mobile mobiles, desktops, and other devices have become a part of daily life in this digital era. People use immense numbers of applications on their devices for professional and personal needs. Therefore, it is important to secure such applications from malware to safeguard the data and gain people's trust in the applications. The use of code signing can do this. This document will provide you with code signing, reasons to code sign apps, and the procedure to code sign mobile apps in Google Play Store and Apple IOS platform.

**What is Code Signing?**

Code signing is a genre of digital certificate that ensures the credibility of the code of the program or software downloaded by the users. When a user tries to download a software or application on his/her device, the user must check the authenticity of the source. Code signing allows the user to see whether the code contains a valid security certificate. Thus, code signing is proof that the software or application has not been meddled with or altered from the code source. To understand the importance of code signing, it is inevitable to know how the code signing works.

**The Operating Procedure of Code Sign**

Code signing works on the asymmetric encryption method, which involves a Public-Private key pair. Two different keys are mathematically connected to encrypt and decrypt the messages. It uses the Public Key Infrastructure (PKI) used in HTTPS. While one key is widely available for all users, the other key is kept secret to ensure safe messages. The code signing is done using a digital certificate, known as a code signing certificate.

**What is Code Signing Certificate?**

It is a digital certificate containing information about the entity or the developer who created the code or the software program. Certificate Authority issues this digital certificate. A code signing certificate binds the identity of the developer or the organization with a Public key and provides a Private related key to them. The developer or the organization signs its code with a Private key, and the users are provided with the private key to verify the identity of the developer or the organization. The digital certificate is based on X.509 standard.

The validity of such digital certificates is generally for a year or two based on the requirement. However, the validity is kept short mainly for two reasons.

* To increase the data security.
* Change of technology is fast-paced manner.

## **To Increase the Data Security**

If the key is renewed within short periods, the older keys will become invalid. This reduces the possibility of key theft by the hacker because the older keys would be invalid with the latest update.

## **Change of Technology in Fast Paced Manner**

The technology used for the past couple of years is not the same today. Considering this factor, the keys are updated frequently to cope with the developing technology.

# **Who Uses Code Signing?**

Code signing is generally used by all application engineers, who use it to develop commercial software applications. It is also used by trusted application stores such as the IOS App Store and Google Play Store. These stores made it mandatory to sign code for the software distributed on their platform. This code signed applications are used in all digital electronics such as Personal computers, laptops, Mobile Phones, Tablets, Televisions, and other devices, which allows the externally sourced applications to perform in their system.

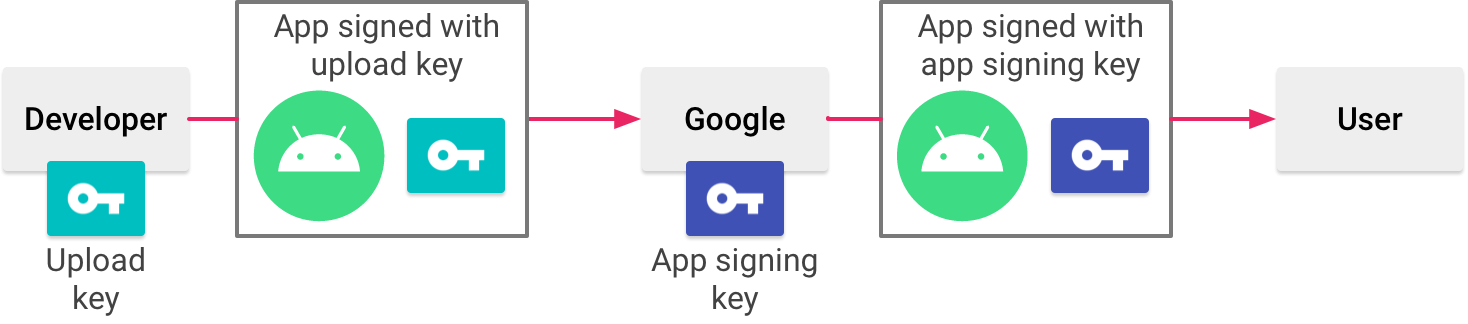
# **Why Code Sign for Android Apps?**

Android is the most used operating system in this world of technology. Because of this reason, most cybercriminals are fixing their targets through malware applications. Therefore, it's highly recommended for android customers to install only trusted applications on their devices to avoid this kind of cybercrimes. Hence, to get the trust towards your application from customers, it is recommended to do codesign for your android apps.

# **How to Sign Mobile App Code for Google Play store?**

It is mandatory to do codesign for the app, to publish them on Google Play Store's platform. If you are publishing an application for the first time, it is mandatory to configure your Play App Signing before uploading your first app in Google Play.

Note: Play App Signing is optional for mobile applications created before August 2021.



**Figure 1: Code Signing for Google Play Store**

Image Reference:<https://developer.android.com/studio/images/publish/appsigning_googleplayappsigningdiagram_2x.png>

The procedure to sign the mobile app and release it in Google Play Store includes a bunch of steps which are as follows.

1. Generating an upload key.
2. Sign the app with the upload key.
3. Configure Play App Signing.
4. Upload the app to Google Play.
5. **To Generate an Upload Key**
   1. In the menu bar, click on **Build** > **Generate Signed Bundle/APK**.
   2. Select **Android App Bundle** or APK and click on **Next**.
   3. Click on **Create new**.

***Note:*** *It will be available below the field of Key store path.*

* 1. Provide the following information in the **New Key Store** window. Refer to Figure 2.

1.4.1. Select the **Key Store Path**.

***Note:*** *It is the location at where your keystore will be created.*

1.4.2. Create and confirm your keystore password.

***Note:*** *This password will secure your keystore.*

1.4.3. Enter the **Alias**.

***Note:*** *It is an identification name for your Key.*

1.4.4. Create and confirm your key password.

***Note:***

* *This password will secure your key.*
* *This should be different from your keystore password.*

1.4.5. Enter **Validity (years)** for your key.

***Note:***

* *Your key should have a minimum validity of 25 years.*
* *You can sign application updates throughout the lifespan.*

1.4.6. Enter your **First and Last Name, Organizational Unit, Organization, City or Locality, State or Province, and Country Code (XXX)** details.

***Note:***

* *It will not be displayed in your application.*
* *It will be included in your certificate, as a part of the APK.*

1.4.7. Make sure that all the details filled in are correct.

1.4.8. Click **OK**.

***Note:*** *Now your key will be generated and stored in the given path.*

Graphical user interface, application

Description automatically generated

1.4.8

1.4.6

1.4.5

1.4.4

1.4.2

1.4.3

1.4.1

**Figure 2: Create a New Upload Key**

Image Reference: <https://developer.android.com/studio/images/publish/keystore-wizard_2x.png>

1. **To Sign the App with Key**
   1. Click on **Build** > **Generate Signed Bundle/APK**.

***Note:*** *If you have the* ***Generate Single Bundle or*** *APK dialog open, you can directly proceed from the next step.*

* 1. Select **Android App Bundle** or **APK** from the dialog box.
  2. Click on **Next**.
  3. Select your module from the drop-down list box. Refer to Figure 3.
  4. Enter the path of the key store and alias name for the key.
  5. Enter the passwords.

***Note:*** *You should enter both passwords of the key store and alias.*

* 1. Click on **Next**.

***Note:*** *Next window will be opened.*

* 1. Select a destination folder for the signed app and select build type. Refer to Figure 4.

***Note:*** *You can also choose product flavour(s), if required.*

* 1. Select the **Signature Versions**.

***Note:*** *Your app will be supported only in the selected version of the signature.*

* 1. Click on **Finish** button.

***Note:***

* *Once after android studio builds your signed app, you can locate or analyse your app by clicking on the notifications in the pop-up menu. Refer to Figure 5.*
* *Now your application is ready to sign in to Google Play and for release upload.*

A screenshot of a computer

Description automatically generated with medium confidence

2.4

2.7

2.6

2.6

2.5

2.5

**Figure 3: Signing the App with the Product Key**

Image Reference: <https://developer.android.com/studio/images/publish/generate-signed-apk-wizard_2x.png>

Graphical user interface, application, Teams

Description automatically generated

2.8

2.10

**Figure 4: Signing the App with the Product Key**

Image reference: <https://developer.android.com/studio/images/publish/signed-apk-destination_2x.png>

Graphical user interface, text

Description automatically generated with medium confidence

**Figure 5: Pop-up Menu**

Image Reference: <https://developer.android.com/studio/images/publish/generate_signed_bundle_popup-2x.png>

1. **To Configure New App in Play App Signing**

Follow the below procedure to configure the new app in Play App Signing to distribute it in Google Play.

* 1. Sign into the **Google Play Console**.

***Note:***

* *If you don’t have a Google play Console account, you must create it.*
* *Charges applicable for the creation of this account.*



**Figure 6: Configure Play App Signing**

**Image reference:**

<https://www.google.com/imgres?imgurl=https%3A%2F%2Fi.ytimg.com%2Fvi%2Fzt_cNQse05s%2Fmaxresdefault.jpg&imgrefurl=https%3A%2F%2Fdeveloper.android.com%2Fdistribute%2Fconsole&tbnid=ZcqR9DoHW2yUVM&vet=12ahUKEwjIkv7EhqTyAhVUSSsKHUikBwgQMygAegUIARCvAQ..i&docid=1k2NHA0uWlwNIM&w=1280&h=720&q=play%20console%20android&ved=2ahUKEwjIkv7EhqTyAhVUSSsKHUikBwgQMygAegUIARCvAQ>

3.2 Create a new release.

***Note:*** *It is a combination of various versions of the applications prepared for the launch.*

3.2.1. Follow the below steps to create new release.

* Release can be created on three different testing tracks or to production. Those testing tracks are **open testing**, **closed testing**, and **internal testing**.
* Open **Play Console** > select the required **release** method > Click on **Create new release**.
* To prepare the app’s release, name the release and enter the release notes.
* Click onthe **Save** button and select **Review release,** after finishing the preparation of the release.

***Note:***

* + *Before rolling out the release, it is necessary to ensure the set-up of app’s store listing, prepare an app for the review in the App content page and set up the app’s price.*
  + *If required, make changes in the release details by using* ***Review release****.*
  + Select **Start rollout.**

3.3 After creating a new release, configure app signing under the **App Integrity** section.

***Note:*** *If you want to use the key generated by Google Play, then you need not to configure it again. The key which is used for the app's first release, will become a default upload key, and should be used for future releases.*

* To use the available key from your developer account, select Change app signing key > Use my own key> Use the same key as another app in this account. Then select the app and click Continue.
* To create an own signing key, select Change app signing key > Use my own key to upload a private key and its public certificate.

Note: You need to review and accept the terms and conditions to continue*.*

3.4 In the App Bundles section > click on Browse files to locate and upload the code signed app in Google Play store using the upload key.

Note: Google Play generates and manages the app's signing key when the app is configured using Play App Signing. These keys are stored on the same infrastructure that Google uses to store its own keys, where Google's Key Management Service protects them.

1. **To Upload the Signed App in Google Play**

Once the process of building and signing the release version of the app is completed, the app should be uploaded to Google Play to inspect, test, and publish the app. After verification by the Play Store, the app can be uploaded to the Google Play Console.

# **Why Code Sign for iOS Apps?**

Every developer or organization who needs to upload their applications on the iOS platform needs a certificate from Apple. The code signing makes sure that the software has not been modified since the developer code signed it. As the apps are developed and code signed by trusted developers, users feel safe using the applications.

# **How to Sign Mobile App Code for Apple Store?**

The Prerequisite for Code signing in the Apple store consist of 4 steps that follow:

* + Create a Certificate Signing Request
  + Generate a distribution certificate
  + Create App ID and
  + Provisioning profiles.

The first step of code signing is Certificate Signing Request (CSR). First, the developer needs to create a Certificate Signing Request (CSR) from their local machine, with some basic details which confirm the developer's identity. Next, this CSR needs to be sent to the Certificate Authority (CA), Apple, for the iOS platform. Apple then confirms the developer's identity and issues a certificate to the developer.

1. **To Create a CSR from your Local macOS Machine**
   1. Open **Keychain Access**.

***Note:*** *This can be opened using Spotlight Search on Mac.*

* 1. From the menu bar, select **Keychain Access** -> **Certificate Assistant** -> **Request a Certificate from a Certificate Authority**. Refer to Figure 7.
  2. Fill in the **Certificate Information** form. Refer to Figure 8.

***Note:*** *Leave the****CA Email Address****field empty.*

* 1. Select the option **Saved to disk**.
  2. Click on **Continue**.

***Note:*** *Now, your CSR will be created in your local machine; you can locate it using the Show in Finder option. Refer to Figure 9.*

* 1. Click on **Done**.

Graphical user interface, text, application

Description automatically generated

**Figure 7: Certificate Signing Request**

Image Reference: <https://blog.codemagic.io/uploads/2020/11/CSR_1.png>

Graphical user interface

Description automatically generated

**Figure 8: Certificate Information**

Image Reference: <https://blog.codemagic.io/uploads/2020/11/CSR_2.png>

Graphical user interface

Description automatically generated

**Figure 9: Certificate Conclusion**

Image Reference: <https://blog.codemagic.io/uploads/2020/11/CSR_3.png>

1. **To Generate a Distribution Certificate**

***Note:*** *You need to have an Apple Developer Membership to generate certificates in Apple Developer Portal.*

* 1. Login to the Apple Developer Portal using your Apple ID and password.
  2. 2.2. Select the Certificates, IDs & Profiles option from the left menu of the Dashboard.
  3. 2.3. Under the Certificates option > click on the "+" button.
  4. 2.4. Select the iOS Distribution option and click on Continue. Refer to Error! Reference source not found.
  5. 2.5. Click on Choose File to upload the CSR file we had generated previously.
  6. 2.6. Click Continue.

***Note:*** *Now, you can Download the generated iOS Distribution Certificate to your local machine.*

* 1. Double click on the downloaded certificate to add it to your local device's Keychain.

***Note:***

* *You can export any certificate from your Keychain by right-clicking and selecting Export.*
* *You will have to supply a Certificate Password for securing the certificate.*

Graphical user interface, text, application, email

Description automatically generated

**Figure 10: Generate Distribution Certificate**

Image Reference: <https://blog.codemagic.io/uploads/2020/11/certificate_3.png>

1. **To Create App Id**

Note: It is mandatory to register an Apple ID to make your application uniquely identifiable when released in App Store. It is also mandatory to provision the profile.

Refer to Figure 11.

* 1. In the Identifiers option menu, click on the "+" button.
  2. 3.2. Select the App IDs.
  3. 3.3. Click on Continue.
  4. 3.4. Enter the Description and Bundle ID of the app.
  5. 3.5. Click on Continue.

***Note:*** *You will get the Bundle ID of your project under the Xcode General tab.*

* 1. Click on **Register**.

***Note:*** *Now Bundle ID is successfully registered for your iOS app.*

Graphical user interface, text, application

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text, letter

Description automatically generated

**Figure 11: App ID Creation**

Image Reference:

* <https://blog.codemagic.io/uploads/2020/11/id_1.png>
* <https://blog.codemagic.io/uploads/2020/11/id_2.png>
* <https://blog.codemagic.io/uploads/2020/11/id_3.png>
* <https://blog.codemagic.io/uploads/2020/11/id_4.png>

1. **Provisioning the Profiles**

***Note:*** *Provisioning will define the rule for running the application inside the device. It is a combination of Team ID, Bundle ID, APP ID, Device ID, and Entitlements.*

Refer to Figure 12.

* 1. n the Profiles option menu, click on the "+" button.
  2. 4.2. Select the App Store option from the list.
  3. Click on Continue.

***Note:*** *It will help generate a distribution provisioning profile to publish the app in the Apple Play Store.*

* 1. Select the pre-registered App ID.
  2. 4.5. Click on Continue.
  3. 4.6. Select a Distribution Certificate on which the app is signed.
  4. 4.7. Click on Continue.
  5. 4.8. Enter Provisioning Profile Name**.**

***Note:* *It is recommended to make this provisioning name-friendly for identifying the profile in the Apple Developer Portal****.*

* 1. Click on **Generate**.
  2. Click on **Download,** to **download** the distribution provisioning profile to your local machine.

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Figure 12: Provisioning the Profiles**

Image References:

* <https://blog.codemagic.io/uploads/2020/11/profile_1.png>
* <https://blog.codemagic.io/uploads/2020/11/profile_2.png>
* <https://blog.codemagic.io/uploads/2020/11/profile_3.png>
* <https://blog.codemagic.io/uploads/2020/11/profile_4.png>
* <https://blog.codemagic.io/uploads/2020/11/profile_5.png>
* <https://blog.codemagic.io/uploads/2020/11/profile_6.png>

All our prerequisite steps are completed now that we can proceed with Xcode signing. Which Consist of 3 steps

* + - Configuring Xcode to use provisioning profile
    - Creating APP Profile
    - Uploading the app

1. **To configure XCode to use Provisioning Profile**
   1. Using **XCode,** open the project. Refer to ***Figure 13***.
   2. Select the root project directory > go to the **Signing and Capabilities** tab.

***Note:*** *Here, you can either check****Automatically manage signing****or do the signing manually.*

Graphical user interface, text, application

Description automatically generated

**Figure 13: XCode Signing**

Image Reference: <https://blog.codemagic.io/uploads/2020/11/certificate_3.png>

1. **To Create an App Profile**

Refer to Figure 14.

* 1. Log in to **App Store Connect**.
  2. Go to **My Apps**.
  3. Click on the “**+**” button > select **New App**.
  4. Fill in the form details > click on **Create**.

***Note:*** *The profile of your app is created now.*

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

**Figure 14: App profile Creation Menu**

Image Reference:

* <https://blog.codemagic.io/uploads/2020/11/store_1.png>
* <https://blog.codemagic.io/uploads/2020/11/store_2.png>
* <https://blog.codemagic.io/uploads/2020/11/store_3.png>

1. **To Upload the App using XCode**

Refer to Figure 15.

* 1. From XCode, select Generic iOS Device from the target devices list.
  2. 7.2. Go to Product > Archive.

Note: This will open a dialog box displaying all the archives you've created in the past.

* 1. 7.3. Select the Build, which you want to upload to App Store Connect.
  2. 7.4. Click on Distribute App.
  3. 7.5. Select App Store Connect > click on Next.

Note: Both the checkboxes will be checked as default. Need not change it; click on Next.

* 1. 7.6. Select the Distribution Certificate and the App name from the drop-down list.
  2. 7.7. Click on Next.
  3. 7.8. Make sure that the information is verified.
  4. 7.9. Click on Upload.

Note: Wait until the upload process gets completed.

* 1. 7.10. Click on Done.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

**Figure 15: Upload App Using XCode**

Image References:

* <https://blog.codemagic.io/uploads/2020/11/upload_1.png>
* <https://blog.codemagic.io/uploads/2020/11/upload_2.png>
* <https://blog.codemagic.io/uploads/2020/11/upload_3.png>
* <https://blog.codemagic.io/uploads/2020/11/upload_4.png>
* <https://blog.codemagic.io/uploads/2020/11/upload_5.png>
* <https://blog.codemagic.io/uploads/2020/11/upload_6.png>

1. **To Publish the App**

Refer to Figure 16.

* 1. Open **App Store Connect**.
  2. Navigate to your **App.**
  3. In App Information, fill in the following details.
* Privacy Policy URL
* Age Rating
* Category.
  1. In **Pricing and Availability**, select the cost of the app.

Click on Save.

8.6. Open Prepare for Submission.

8.7. Upload the app's Screenshots for various screen sizes and fill in the other required information.

8.8. Select the Build version.

**Note:** The Selected build version will be published along with the app.

8.9. Enter the Copyright information.

8.10. Click on Submit for Review.

8.11. Choose the appropriate options in the final form.

8.12. Click on Submit*.*

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

**Figure 16: Publishing the App**

Image References:

* <https://blog.codemagic.io/uploads/2020/11/store_5.png>
* <https://blog.codemagic.io/uploads/2020/11/store_6.png>
* <https://blog.codemagic.io/uploads/2020/11/store_7.png>
* <https://blog.codemagic.io/uploads/2020/11/store_8.png>
* <https://blog.codemagic.io/uploads/2020/11/store_9.png>

**Conclusion**

It is wise to use code signing for android and IOS applications to prevent them from the interference of a third party, which could lead to cyber-attack when such hacked mobile apps are installed in the user mobiles. Furthermore, the codesign ensures the user that the application is from a trusted developer.