Firma del código y generación de versiones

Parte I

Contexto

- Todas las apps que hacemos deben estar firmadas para poder ejecutarse en un dispositivo.
- Da igual cómo hayamos desarrollado la app, al final estos requisitos deben cumplirse igual.

Firma del código

- Para firmar el código es necesario un CERTIFICADO
- La misión del certificado es identificar el origen de la app de forma precisa e impedir falsificaciones.

¿Qué contiene el certificado?

- Información que identifica a dicho desarrollador
- La clave pública de un par de claves
- Tiene fechas de inicio y fin de validez

clave pública...?

- Este tipo de certificados se basan en sistemas de pares de claves: pública y privada. Se denomina criptografía asimétrica y se basa en
  - La clave pública encripta la información
  - La clave privada desencripta la información

Cómo se obtiene el certificado?

- Mediante un CSR o Code Signing Request

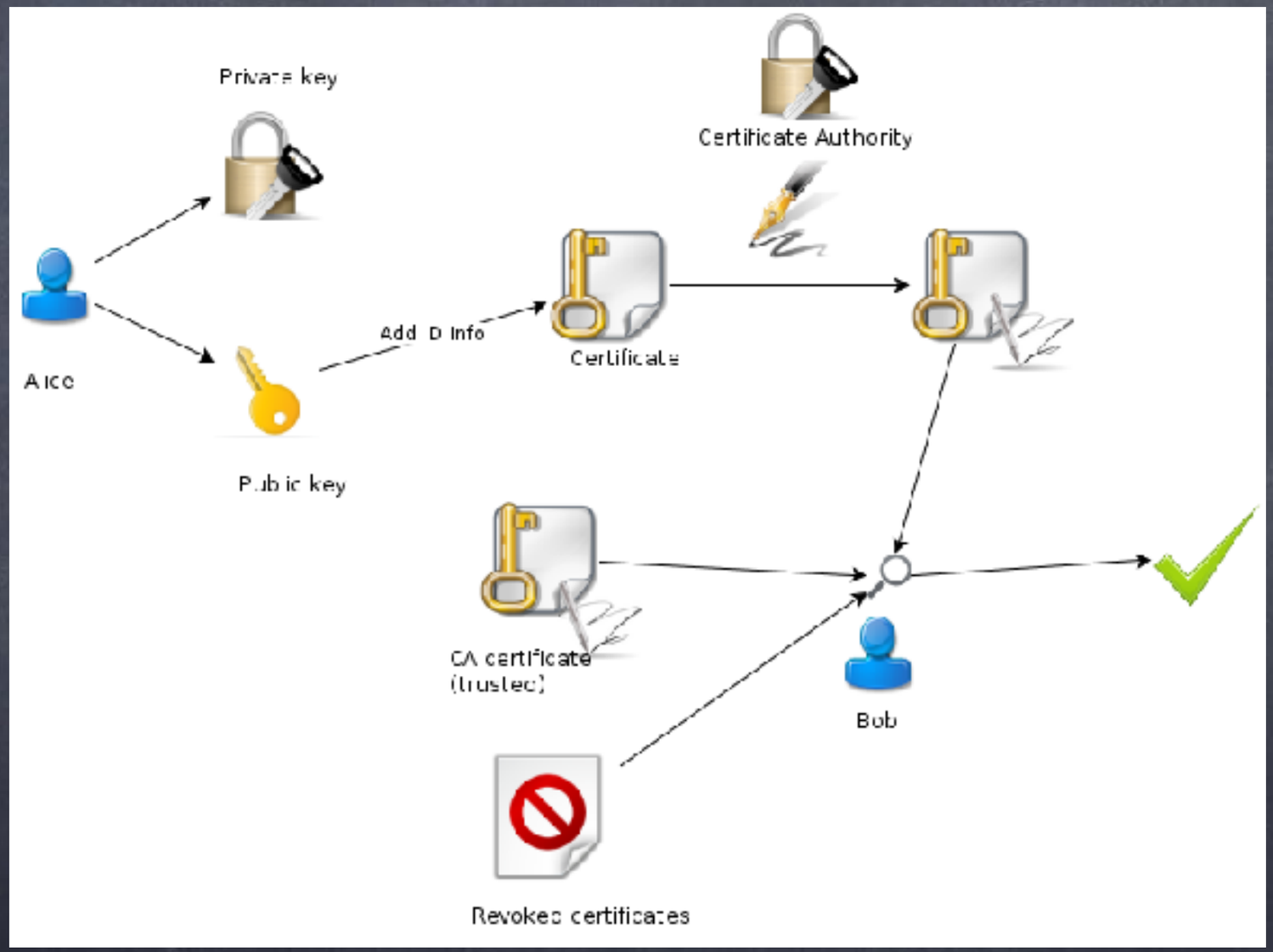
CODE SIGNING REQUEST

- Un CSR es un fichero que contiene un bloque de texto codificado
- El CSR se genera en el ordenador donde se instalará el certificado que se pretende obtener con él y contiene información que se incluirá en el certificado: nombre de la organización, dominio, etc y lo más importante, la clave pública que contendrá el certificado.
- Lo habitual es que las claves se generen a la vez que se genera el CSR
- Una vez generado el CSR, hay que enviarlo a una Autoridad de Certificación

AUTORIDAD DE CERTIFICACIÓN

- Una autoridad de certificación utilizará el fichero CSR para crear el certificado SSL aunque
- Hay que tener en cuenta que el certificado SOLO funcionará si se usa con la clave privada con la que se generó el CSR.

Esquema del certificado



iOS - LICENCIAS

- Para poder obtener un certificado es necesario inscribirse en el programa de desarrollo de Apple (y pagar).

	Individual	Organization	Enterprise
$$	$99/Yr	$99/Yr	$299/Yr
App Store	✓	✓	✗
No App Store	hasta 100 disp.	hasta 100 disp.	Ilimitado

iOS

- La necesidad de la firma se da para ejecutar las apps en dispositivos físicos, no en el simulador.

iOS

- Hay al menos dos maneras de gestionar la obtención del certificado y provisioning profile:
  - Automática a través de xCode
  - Manual (https://developer.apple.com) Es la que vamos a ver aquí.

Apple Developer Portal

- Es el sitio donde se gestiona no solo las cuentas de desarrollador sino todo lo relacionado con las apps, excepto la publicación en el AppStore que se gestiona en iTunesConnect.
- Vemos qué elementos intervienen de más general a más particular:
  - Certificados
  - AppID
  - Provisioning Profile

iOS CERTIFICADOS

- En el portal de Apple se pueden generar certificados de varios tipos:
  - Certificados de la cuenta
    - Pueden valer para TODAS las apps de una misma cuenta
      - Desarrollo
      - Distribución
  - Certificados para Push Notifications
    - Para UNA SOLA app y hay tres tipos:
      - Desarrollo
      - Distribución
      - Mixto

iOS AppID

- En el portal de desarrollo de Apple hay que definir un identificador* de la app y otros datos, así como las características que utilizará (Push Notifications, In-App Purchases, Game Center, etc.)
- *el identificador se lo que se conoce como bundle id y se admiten dos formatos:
  - Explícitos, válidos solo para una app. Son lo más habitual
  - Wildcards, identificadores genéricos para varias apps que no declaran características específicas de una app. Más info sobre cuándo usar wildcards aquí.

iOS Provisioning profile

- El provisioning profile es un fichero binario que contiene:
  - Identificador de la app (bundle id)
  - Certificado(s) válido(s) para la app
  - Dispositivo(s) que puede(n) ejecutarla

iOS: Provisioning Profiles

- La distinción entre desarrollo y distribución está en qué puede hacerse con unos y con otros. Los primeros permiten depurar el código y no distribuir la app y viceversa.

iOS Desarrollo

Certificado(s)
iOS Developer

dispositivo(s)
UUID

BUNDLE ID

PROVISIONING PROFILE

iOS Desarrollo

- El provisioning profile es un fichero binario que contiene:
  - Certificados válidos para la app
  - Dispositivos que pueden ejecutarla
  - Identificador de la app

iOS Distribución

- Básicamente hay 3 formas de distribuir las apps de iOS:
  - App Store, Ad-hoc, Enterprise
- Generar las versiones admite al menos dos opciones
  - Visualmente en xCode
  - Manualmente mediante CLI

iOS Distribución

- Para la distribución, siguen siendo necesarias la creación del certificado y provisioning profile de distribución, aunque éste último cambia su contenido.

iOS Distribución

Certificado
iOS Distribution

dispositivo(s)
UUID
(solo ad-hoc)

BUNDLE ID

PROVISIONING PROFILE

Android

- El acceso a la cuenta de Desarrollador es más sencillo y económico: $25 una vez y para siempre. Básicamente se compra el derecho a publicar apps en el Google Play Store.

Android

- En Android los requisitos para ejecutar el código son parecidos a Apple.
- El sistema de las firmas también se apoya en certificados creados a partir de la clave pública de un par de claves pero hay algunas diferencias en cuanto a cómo se gestionan los procesos de la firma en depuración y distribución.

Android

- Keystore
  - Almacén de claves (la clave privada del par de claves)
    - Puede contener una o varias claves privadas.
    - Es análogo en cuanto almacén seguro de datos al Keychain de iOS pero difiere en que el keychain es para todo el S.O. mientras que el keystore se implementa a nivel de cada app.

Android DEBUG

- Cuando se ejecuta desde el IDE, Android Studio automáticamente firma el APK con un certificado de depuración generado por las herramientas de Android SDK.
- La primera vez que se ejecuta o depura el proyecto en Android Studio, el IDE automáticamente crea el keystore y el certificado de depuración en $HOME/.android/debug.keystore, y configura el keystore y las contraseñas de claves.
- Este certificado no es válido para la distribución y no sería admitido en markets como Google Play Store.

Android Distribución

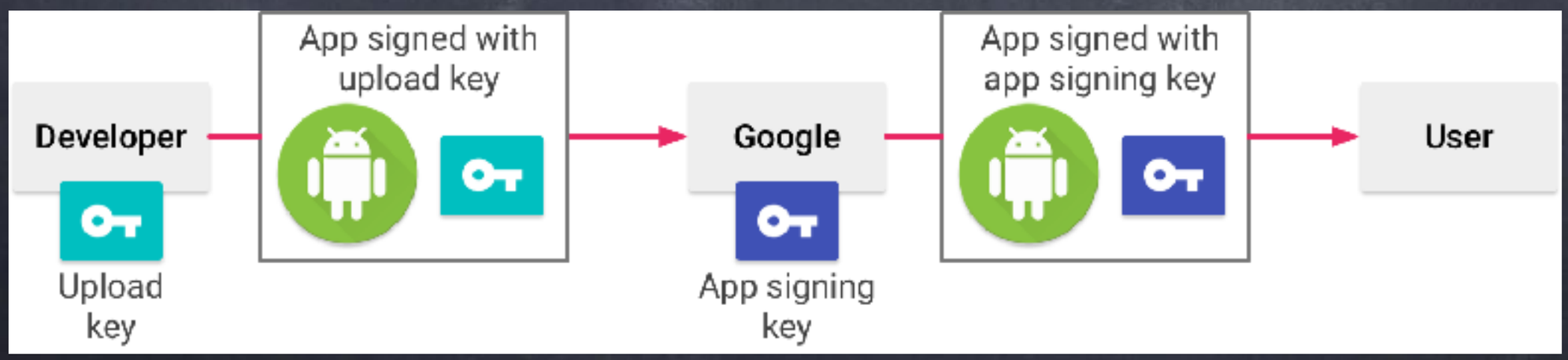
- Las alternativas para distribuir las apps son básicamente estas:
  - En Markets: Además de Google Play Store hay otros alternativos p.ej: Amazon etc
  - Distribuyendo las apk libremente

Android Distribución

- En el caso de distribución mediante Google Play Store hay dos formas de establecer la firma de la app
  - Usar el sistema de firma de Google Play (clave de subida y clave de firma)
  - Clave propia (se firma con la clave de subida)

Firma con Google Play

- Se apoya en dos claves:
  - clave de firma (administrada por Google)
  - clave de subida (administrada por el usuario)



Firma con Google Play

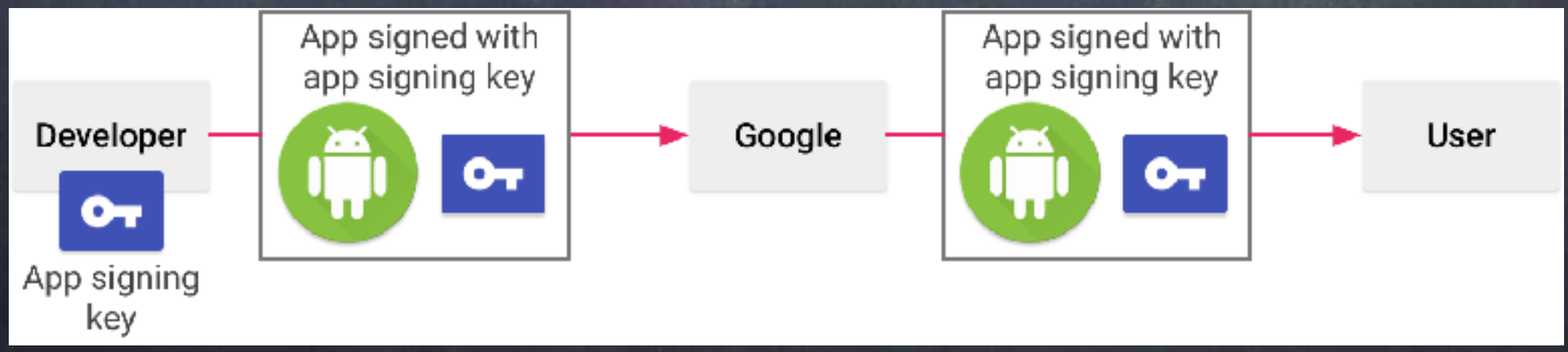
- Si se pierde la clave de subida, o si se ve comprometida, Google puede revocarla y generar una nueva.
- Esto permite seguir subiendo nuevas versiones de la aplicación como actualizaciones de la aplicación original, aunque cambien las claves de subida.

Firma con claves propias

- Impone más responsabilidad, puesto que hay que custodiar el keystore como las claves de firma, ya que si se pierde la clave de firma ya NO se podrán subir más versiones como actualizaciones de la app.

Firma con claves propias

- En este caso la firma se hace de manera local usando la clave de firma de aplicaciones y se sube el APK directamente a Google Play Store para su distribución.



Métodos de firma

- Manual
  - Build -> Generate Signed APK...
- Permite firmar varias compilaciones (build variants) a la vez.

Métodos de firma

- Automático
  - El IDE permite establecer configuraciones de firma y asignarlas a cada tipo de compilación de lanzamiento
  - La configuración de firma se compone de una ubicación y contraseña de keystore, y un alias y una contraseña de clave.
  - La información de firma se incluye en texto sin formato en los archivos de compilación de Gradle.
  - Si se comparte el código públicamente, conviene mantener protegida la información de firma quitándola de los archivos de compilación y almacenándola por separado.

ANDROID

- Further reading: Android tiene bastante trabajada la documentación, cuenta con traducciones al castellano de la mayoría de sus documentos.

PLATAFORMAS HÍBRIDAS

- Cuando se desarrollan apps mediante tecnologías híbridas, lo más habitual es que el IDE de desarrollo propio de la plataforma establezca un "puente" hacia las herramientas de las plataformas nativas para las que se crean las apps.

- Este es el caso de Xamarin

Antes de Publicar la versión

- Tanto en iOS como en Android hay herramientas para validar y probar las apps antes de su publicación.

Apple TestFLight

- Permite desplegar una app para un entorno reducido de usuarios y poder probarla de forma previa a su distribución "oficial".
- Admite testers internos hasta 25 (10 dispositivos max) y externos hasta 10000. Las pruebas pueden durar hasta 3 meses.
- Permite enviar a los desarrolladores así como logs, informes de fallos o feedback en general de la app
- Se gestiona a través de iTunesConnect (funciona mediante invitación)
- Where to Go From Here? RW

Android
Alfa y Beta Testing

- Es menos restrictivo que Apple. Solo es necesario tener una cuenta Google
- Alfa Testing (Pocos usuarios, versiones más experimentales)
- Beta Testing (Más usuarios, versiones más estables)
    - Ámbito cerrado
        - El desarrollador puede seleccionar a los testers (con su email), es lo más parecido a TestFlight
    - Ámbito abierto
        - Se escoge un número de testers que pueden acceder a probar la app, no se tiene información que los identifique
    - Ámbito mixto (Comunidades de Google+). Solo pueden acceder a probar la app los miembros de una comunidad establecida por el desarrollador.

- Further Reading: Aquí

Herramientas de automatización

- FASTLANE iOS y Android
  - Despliegue de betas
  - Despliegue a Stores
  - Firma del código y re-firma de apps
  - Toma de capturas
  - Funciona mediante scripts de Ruby y es customizable
  - Permite integrar herramientas de CI; Jenkins, Travis CI, etc.

Where to Go From Here?

- Proceso de subida a las Stores
  - Requisitos
    - Revisión, etc.
- MDM's
- Otras herramientas