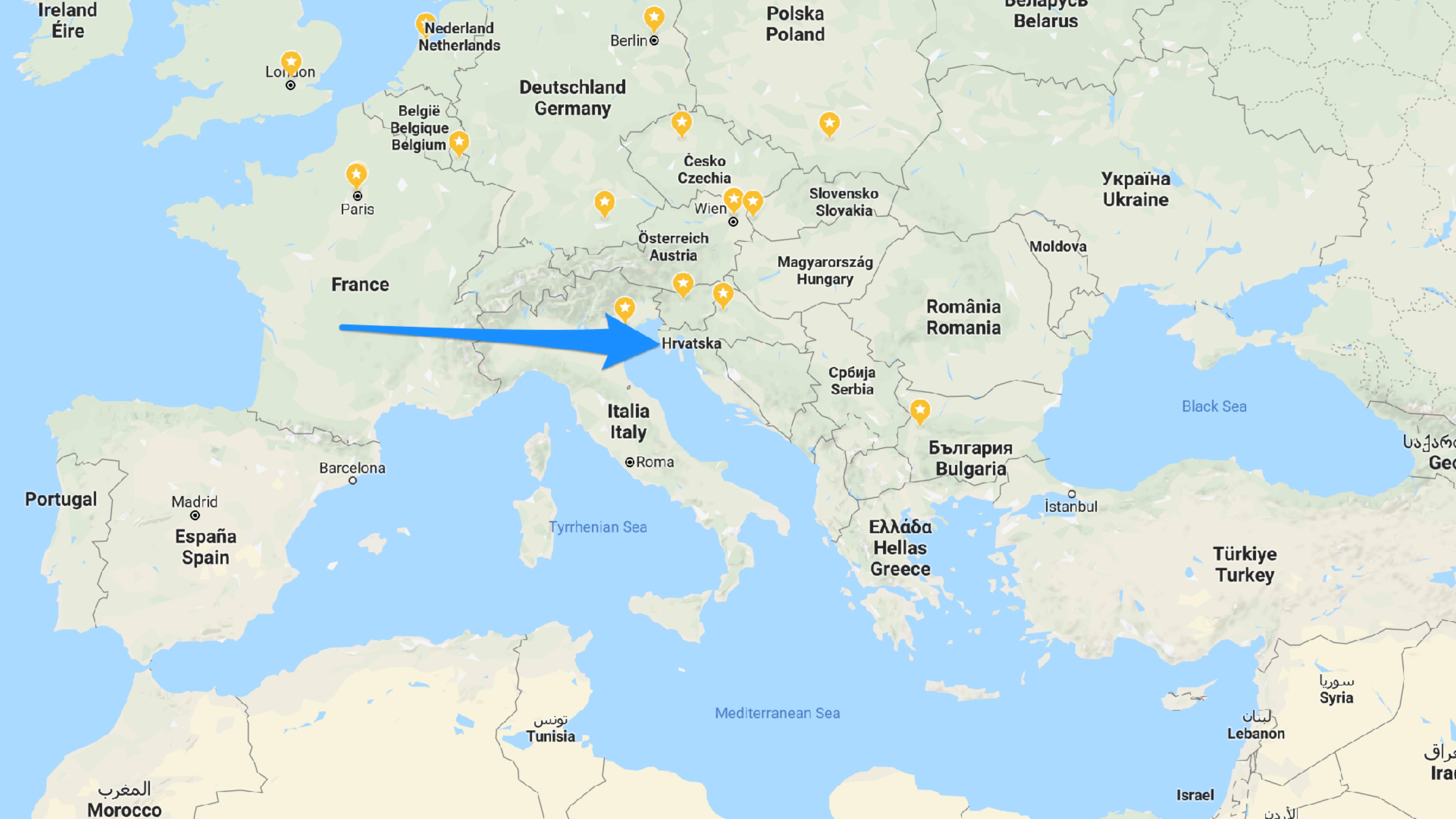




ANA BAOTIĆ

---

**SIGN HERE, PLEASE!**





# ASSECO at a glance

Present in  
**54**  
countries

Over  
**22,300**  
highly committed  
employees

- Founded in 1991
- **6<sup>th</sup>** largest software producer in Europe
- Traded on the WSE, included in the WIG20 blue chip index
- International presence

**6<sup>th</sup>**  
largest software  
& services  
vendor in Europe

**1,813**  
m EUR  
sales revenues in  
2016

**176**  
m EUR  
operating profit  
in 2016

**1bln**  
EUR  
market  
capitalization

# CERTIFICATE

---



CERTIFICATE

---

# CERTIFICATE



*Electronic document that  
proves ownership of a public key*

CERTIFICATE

---

# CERTIFICATE



*Electronic document that*

*proves ownership of a public key*

X.509 - structure

CERTIFICATE

---

# CERTIFICATE



*Electronic document that*

*proves ownership of a public key*

*X.509 - structure*

*TLS/SSL, electronic signatures*

CERTIFICATE

---

# IDENTITY



CERTIFICATE = IDENTITY +

---

## PUBLIC KEY



CERTIFICATE = IDENTITY + PUBLIC KEY

---

SIGNED BY CA



# ASYMMETRIC CRYPTOGRAPHY



## KEYS - ASYMMETRIC CRYPTOGRAPHY

---



► Private (owner)



► Public (shared)

## KEYS

---



- ▶ Private (owner)
- ▶ Decrypts messages
- ▶ Public (shared)
- ▶ Encrypts messages

## KEYS

---



- ▶ Private (owner)
- ▶ Decrypts messages
- ▶ Calculates signatures
- ▶ Public (shared)
- ▶ Encrypts messages
- ▶ Confirms signatures

# DIGITAL SIGNATURE

- ▶ Calculate a digest/hash of the message
- ▶ Sender encrypts with **private** key
- ▶ Recipient decrypts with **public** key



CERTIFICATE = IDENTITY + PUBLIC KEY + SIGNED BY CA

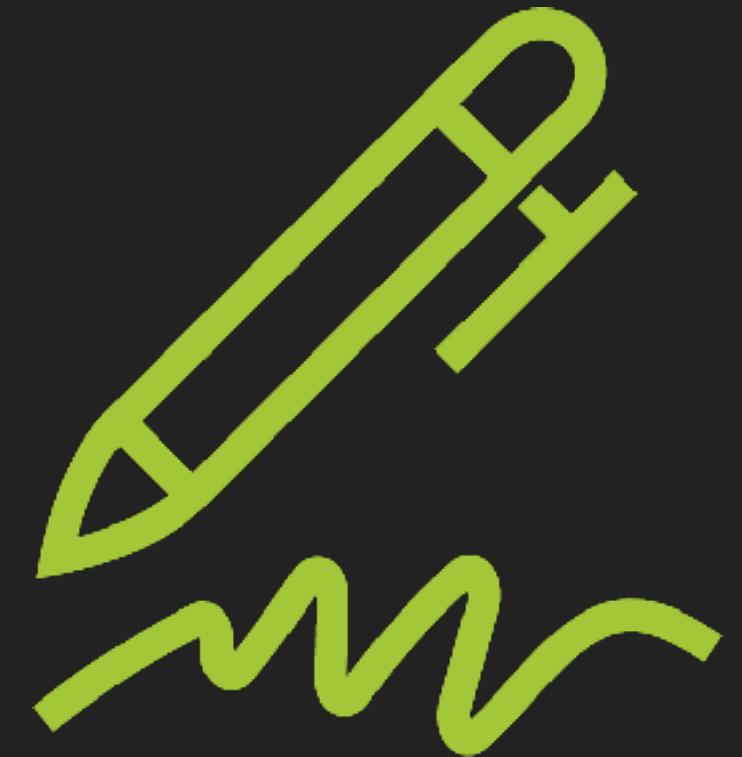
---



CA

---

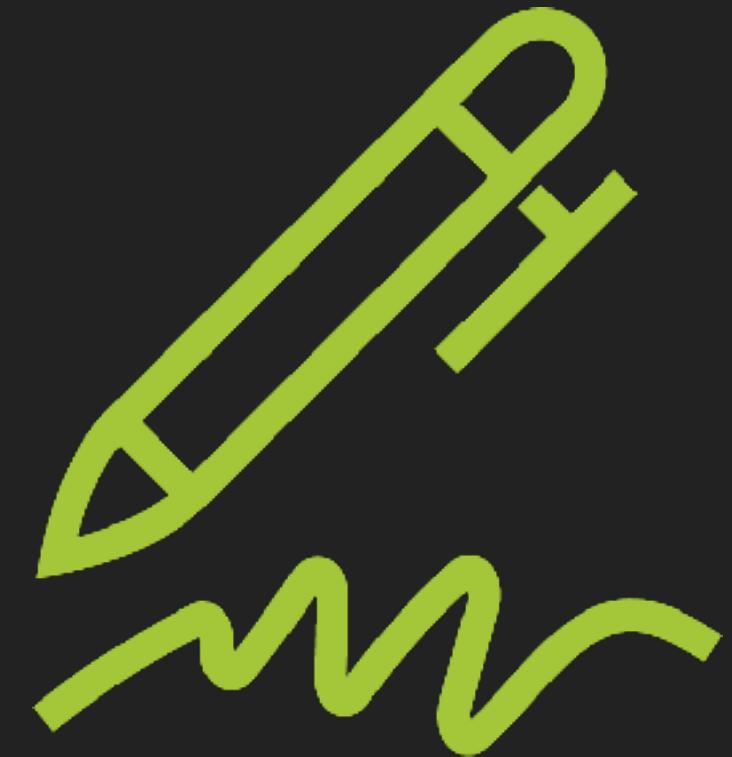
## CERTIFICATE AUTHORITY



*Trusted entity that issues digital certificates.*

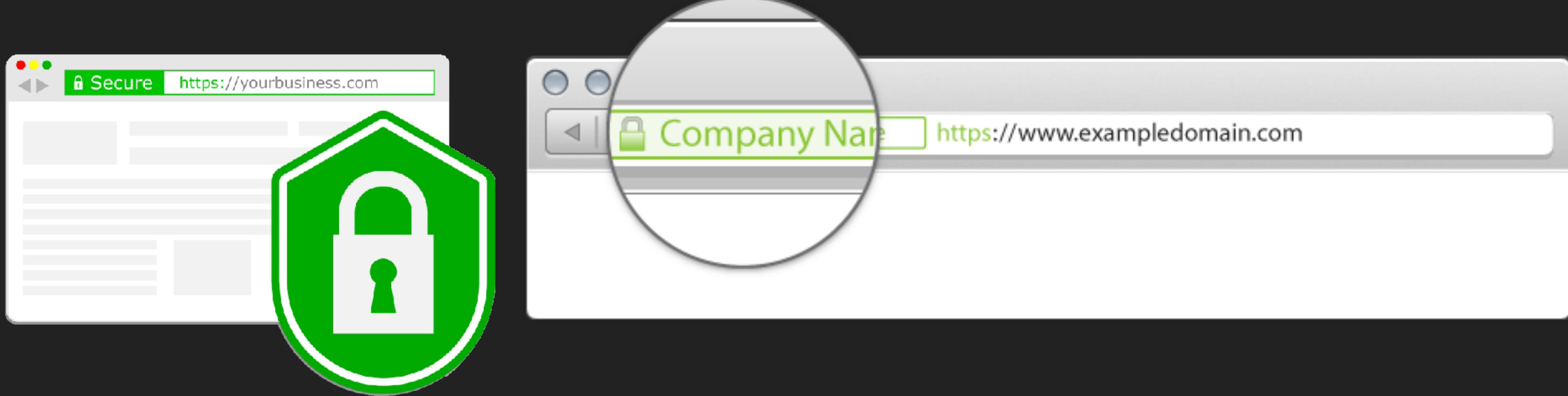
## CA CERTIFICATE

- ▶ Self-signed certificate
- ▶ Browsers, Android, OSX
- ▶ Comodo, IdenTrust, Symantec, GoDaddy, Let's Encrypt
- ▶ Impact of updates!



## CA - DOMAIN VS EXTENDED VALIDATION

---



# END CERTIFICATE

GeoTrust Global CA  
↳ Google Internet Authority G2  
↳ \*.google.com

 **\*.google.com**  
Issued by: Google Internet Authority G2  
Expires: Wednesday, 30 May 2018 at 14:50:00 Eastern Daylight Time  
✓ This certificate is valid

**▼ Details**

Subject Name \_\_\_\_\_  
Country US  
State/Province California  
Locality Mountain View  
Organization Google Inc  
Common Name \*.google.com

Issuer Name \_\_\_\_\_  
Country US  
Organization Google Inc  
Common Name Google Internet Authority G2

Serial Number 4266488754389871764  
Version 3  
Signature Algorithm SHA-256 with RSA Encryption ( 1.2.840.113549.1.1.11 )  
Parameters None

Not Valid Before Wednesday, 7 March 2018 at 13:50:00 Eastern Standard Time  
Not Valid After Wednesday, 30 May 2018 at 14:50:00 Eastern Daylight Time

Public Key Info \_\_\_\_\_  
Algorithm Elliptic Curve Public Key ( 1.2.840.10045.2.1 )  
Parameters Elliptic Curve secp256r1 ( 1.2.840.10045.3.1.7 )  
Public Key 65 bytes : 04 EA 3E F7 11 F1 FC 15 ...  
Key Size 256 bits  
Key Usage Encrypt, Verify, Derive

Signature 256 bytes : 70 DE 7F 51 26 71 B2 2B ...

OK

# INTERMEDIATE CERTIFICATE

GeoTrust Global CA  
↳ Google Internet Authority G2  
↳ \*.google.com

 **Google Internet Authority G2**  
Intermediate certificate authority  
Expires: Monday, 31 December 2018 at 18:59:59 Eastern Standard Time  
✓ This certificate is valid

▼ Details

Subject Name  
Country US  
Organization Google Inc  
Common Name Google Internet Authority G2

Issuer Name  
Country US  
Organization GeoTrust Inc.  
Common Name GeoTrust Global CA

Serial Number 01 00 21 25 88 B0 FA 59 A7 77 EF 05 7B 66 27 DF  
Version 3  
Signature Algorithm SHA-256 with RSA Encryption ( 1.2.840.113549.1.1.11 )  
Parameters None

Not Valid Before Monday, 22 May 2017 at 07:32:37 Eastern Daylight Time  
Not Valid After Monday, 31 December 2018 at 18:59:59 Eastern Standard Time

Public Key Info  
Algorithm RSA Encryption ( 1.2.840.113549.1.1.1 )  
Parameters None  
Public Key 256 bytes : 9C 2A 04 77 5C D8 50 91 ...  
Exponent 65537  
Key Size 2.048 bits  
Key Usage Encrypt, Verify, Derive  
Signature 256 bytes : CA 49 E5 AC D7 64 64 77 ...

OK

# ROOT CERTIFICATE

GeoTrust Global CA

- ↳ Google Internet Authority G2
- ↳ \*.google.com

**GeoTrust Global CA**  
Root certificate authority  
Expires: Saturday, 21 May 2022 at 00:00:00 Eastern Daylight Time  
✓ This certificate is valid

**Details**

Subject Name \_\_\_\_\_

Country US  
Organization GeoTrust Inc.  
Common Name GeoTrust Global CA

Issuer Name \_\_\_\_\_

Country US  
Organization GeoTrust Inc.  
Common Name GeoTrust Global CA

Serial Number 144470  
Version 3  
Signature Algorithm SHA-1 with RSA Encryption ( 1.2.840.113549.1.1.5 )  
Parameters None

Not Valid Before Tuesday, 21 May 2002 at 00:00:00 Eastern Daylight Time  
Not Valid After Saturday, 21 May 2022 at 00:00:00 Eastern Daylight Time

Public Key Info

Algorithm RSA Encryption ( 1.2.840.113549.1.1.1 )  
Parameters None  
Public Key 256 bytes : DA CC 18 63 30 FD F4 17 ...  
Exponent 65537  
Key Size 2.048 bits  
Key Usage Any

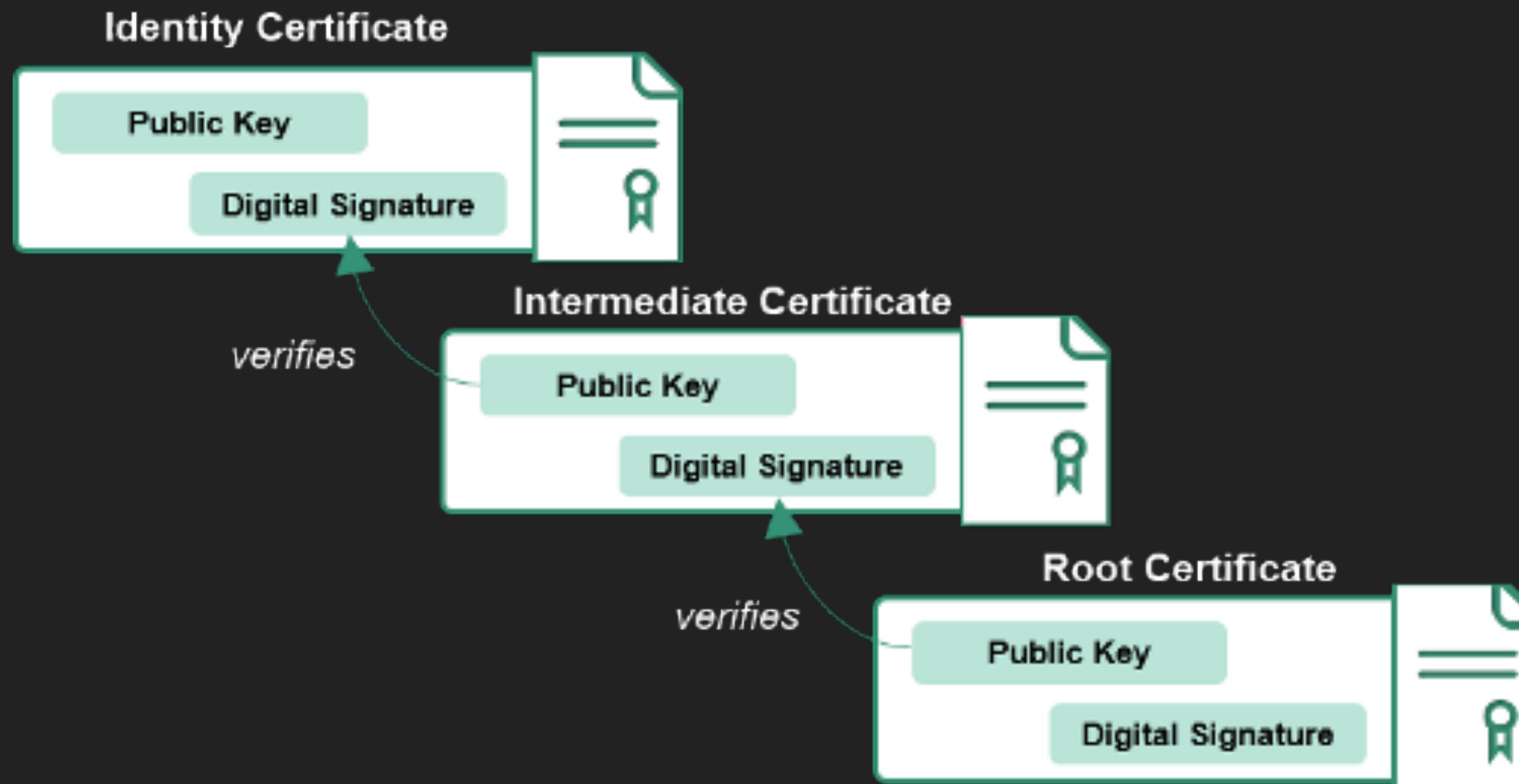
Signature 256 bytes : 35 E3 29 6A E5 2F 5D 54 ...

OK

## CERTIFICATES

---

# CERTIFICATE CHAINS



# SUBVERSION

- ▶ Primary risk key theft
- ▶ Scheme flawed
- ▶ HSM for key storage
- ▶ Offline

## ANDROID

- ▶ Unknown CA
- ▶ Self-signed
- ▶ No intermediate cert on server

## SIGNING ANDROID APPLICATIONS

---



## SIGNING ANDROID APPLICATIONS

---



- ▶ Generate a key and keystore
- ▶ Sign your application

# SIGNING ANDROID APPLICATIONS - NEW KEY STORE

New Key Store

Key store path: `/Users/abaotic/workspaces/demo/DemoApplication/app/Untitled`

Password:  Confirm:

Key

Alias: `Demo application key`

Password:  Confirm:

Validity (years): `50`

Certificate

First and Last Name: `Ana Baotić`

Organizational Unit: `My Organizational Unit`

Organization: `My Organization`

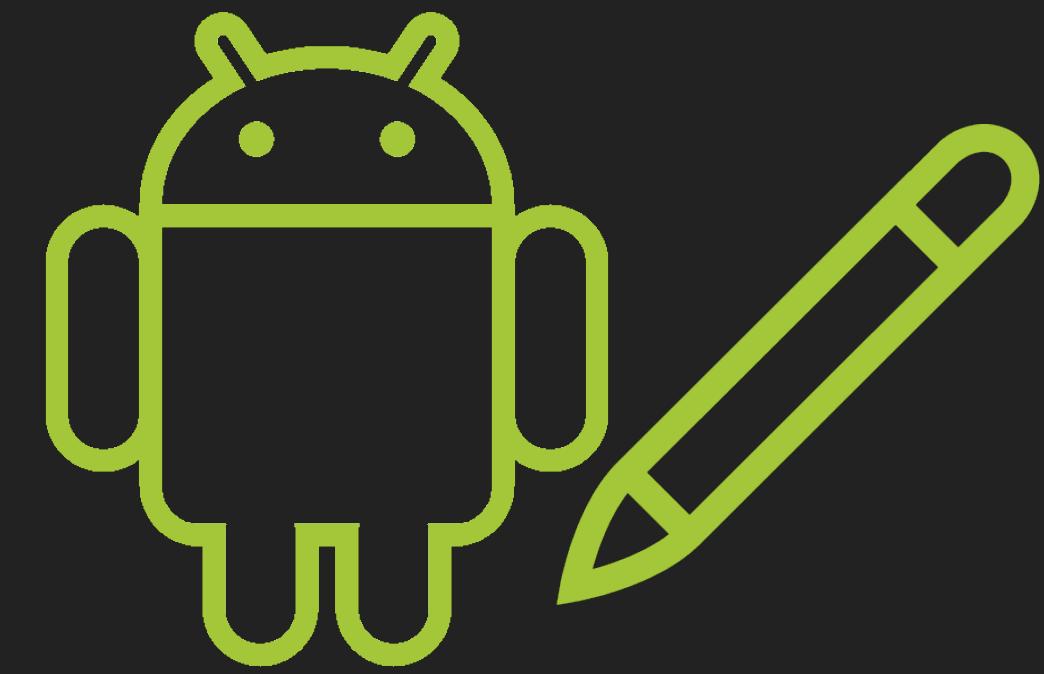
City or Locality: `Zagreb`

State or Province: `Croatia`

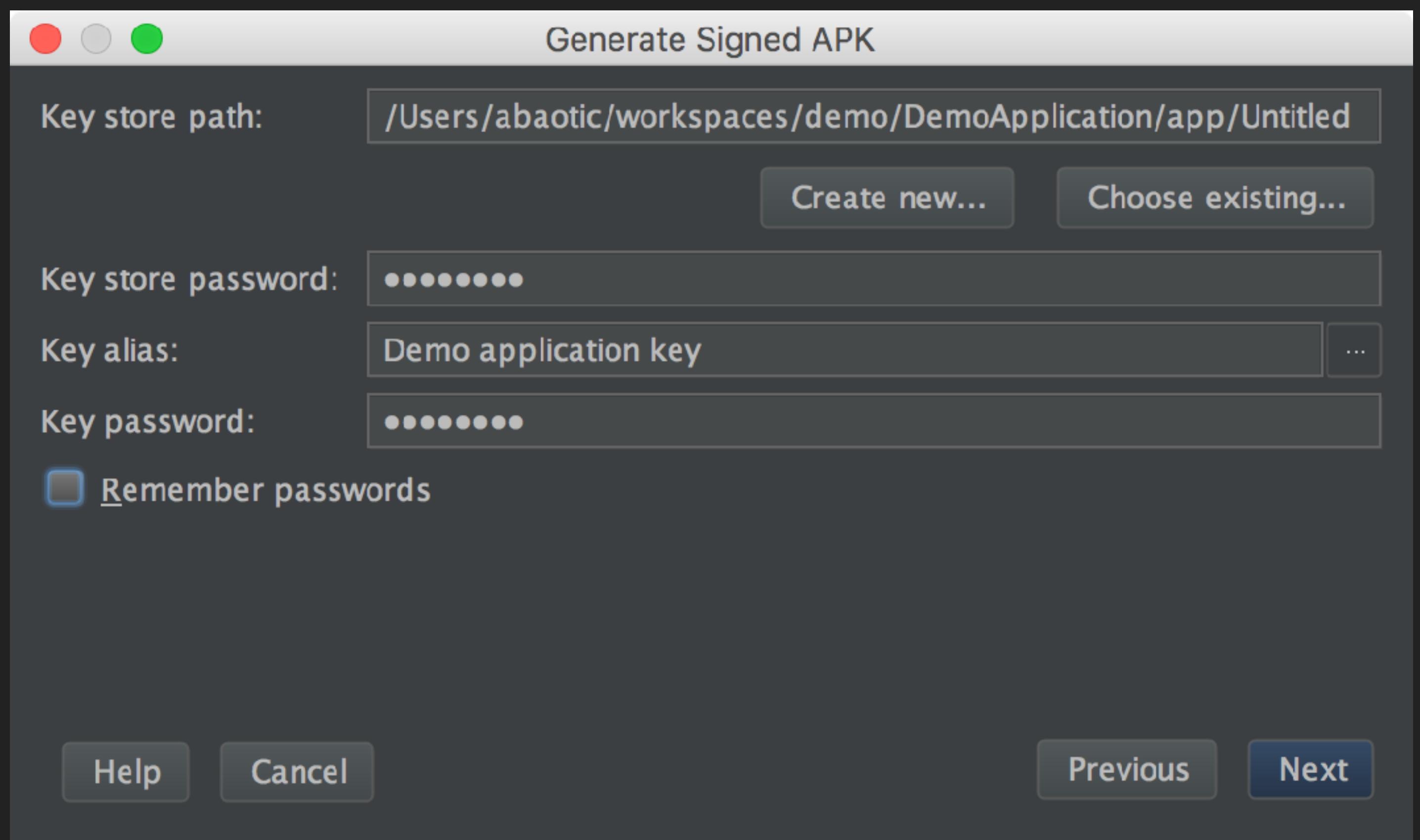
Country Code (XX): `385`

?

Cancel OK



## SIGNING ANDROID APPLICATIONS - SIGN THE APP



### ALTERNATIVE

- ▶ `keytool` to generate a key
- ▶ `zipalign` to align unsigned apk
- ▶ `jarsigner` to sign aligned apk

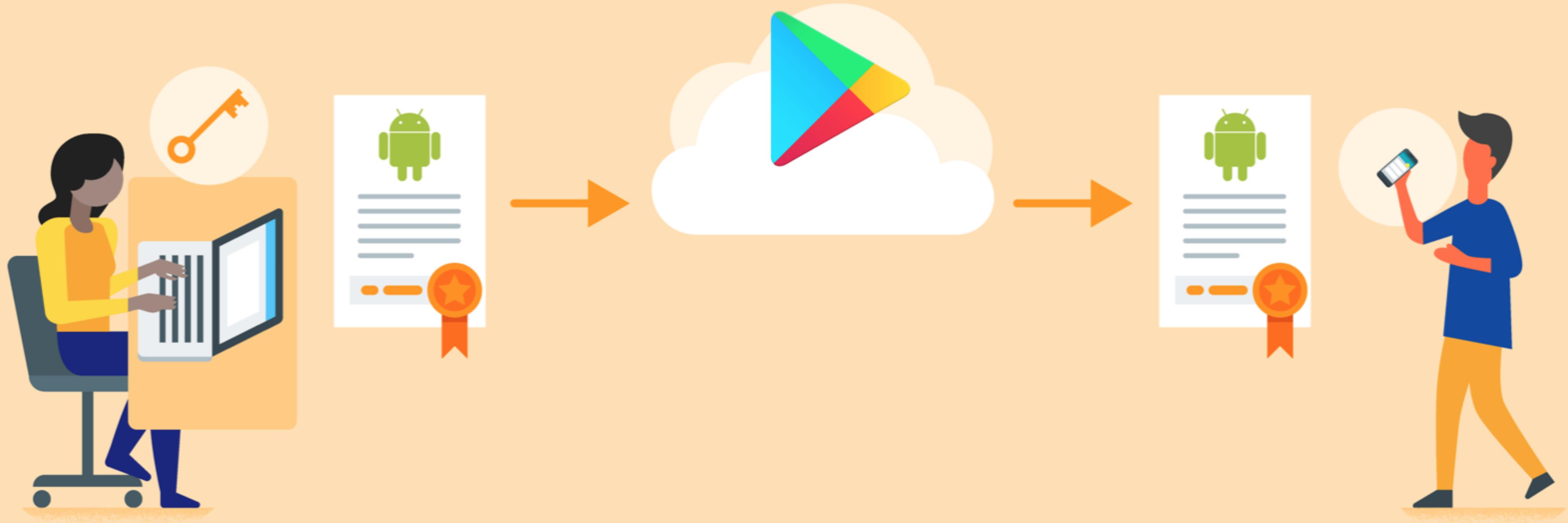
## DECISIONS TO BE MADE

- ▶ Who will be responsible for the signing key?



# WE THE DEVELOPERS

---



## EXPECTATIONS

- ▶ Application can be updated throughout its lifetime

## PREREQUISITES

- ▶ Signing key must be safe
- ▶ Signing key must be accessible

# REALITY

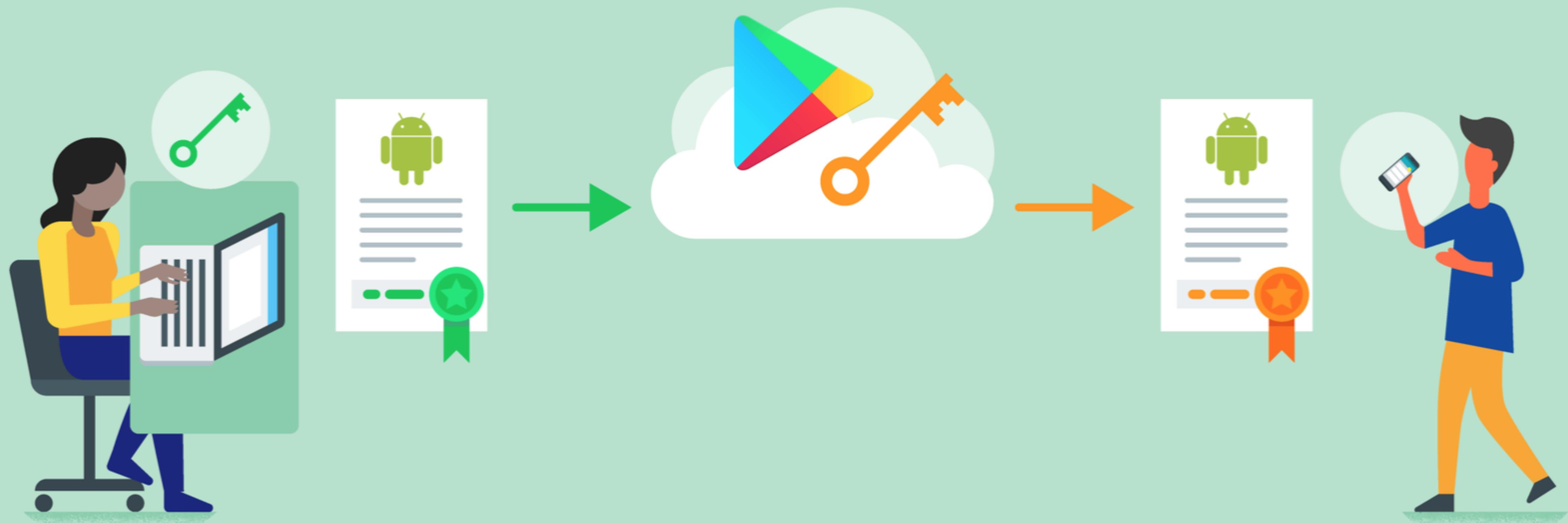
- ▶ People spill coffee
- ▶ People switch companies

# REALITY NIGHTMARE

```
signingConfigs {  
    release {  
        storeFile file("/not_where_you_think_it_is/ks.jks")  
        storePassword "password"  
        keyAlias "my-alias"  
        keyPassword "password"  
    }  
}
```

## GOOGLE PLAY APP SIGNING

---



<https://goo.gl/3aCBeC>

**DECISION IS PERMANENT**

### NEW APPS

- ▶ Create upload key and sign apk
- ▶ Google Play App Signing ->**Accept**
- ▶ Upload signed apk
- ▶ **Register app signing key!**

## EXISTING APPS

- ▶ Opt-in
- ▶ Submit signing key to Google and
- ▶ Create upload key
- ▶ Update keystores
- ▶ Continue signing with upload key

## MITIGATION

- ▶ Upload key lost/compromised?

## DECISIONS TO BE MADE

- ▶ Who will be responsible for the signing key?



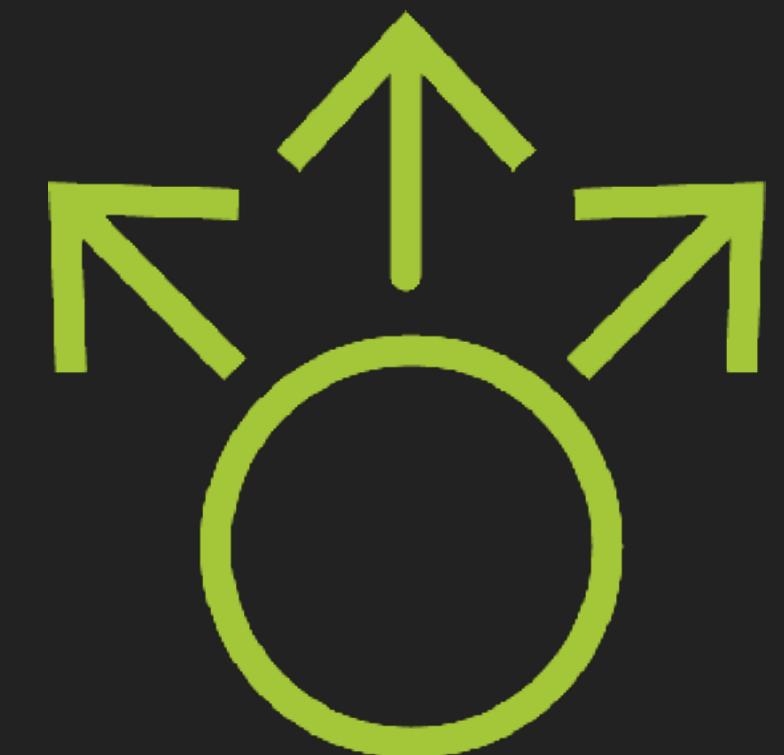
## DECISIONS TO BE MADE

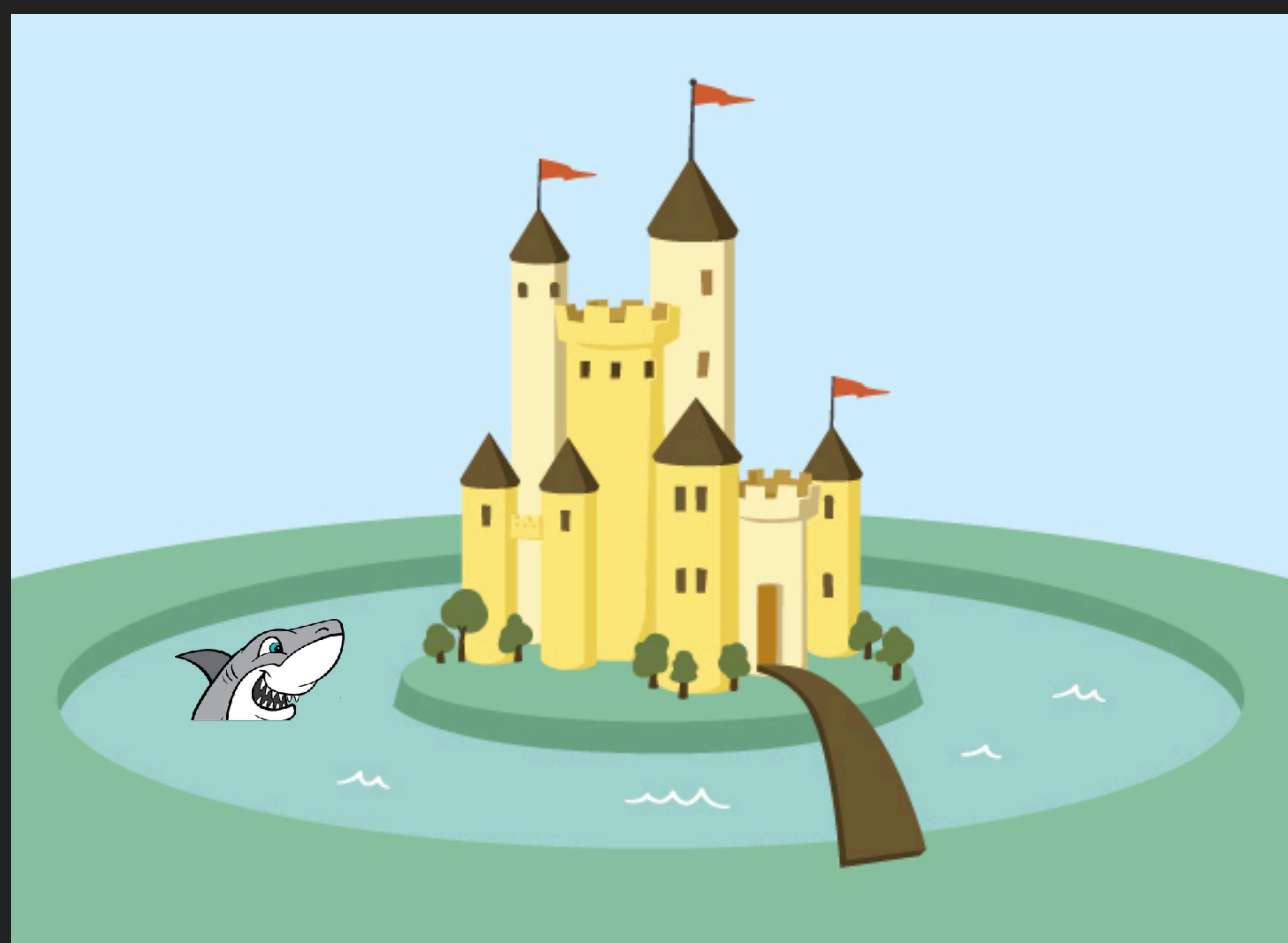
- ▶ Who will be responsible for the signing key?
- ▶ Applicable to all modules/applications/flavours?



## PROS AND CONS

- ▶ Modularity
- ▶ Permission based sharing
- ▶ update-able
- ▶ Single point of failure





# ANDROID KEYSTORE SYSTEM

---

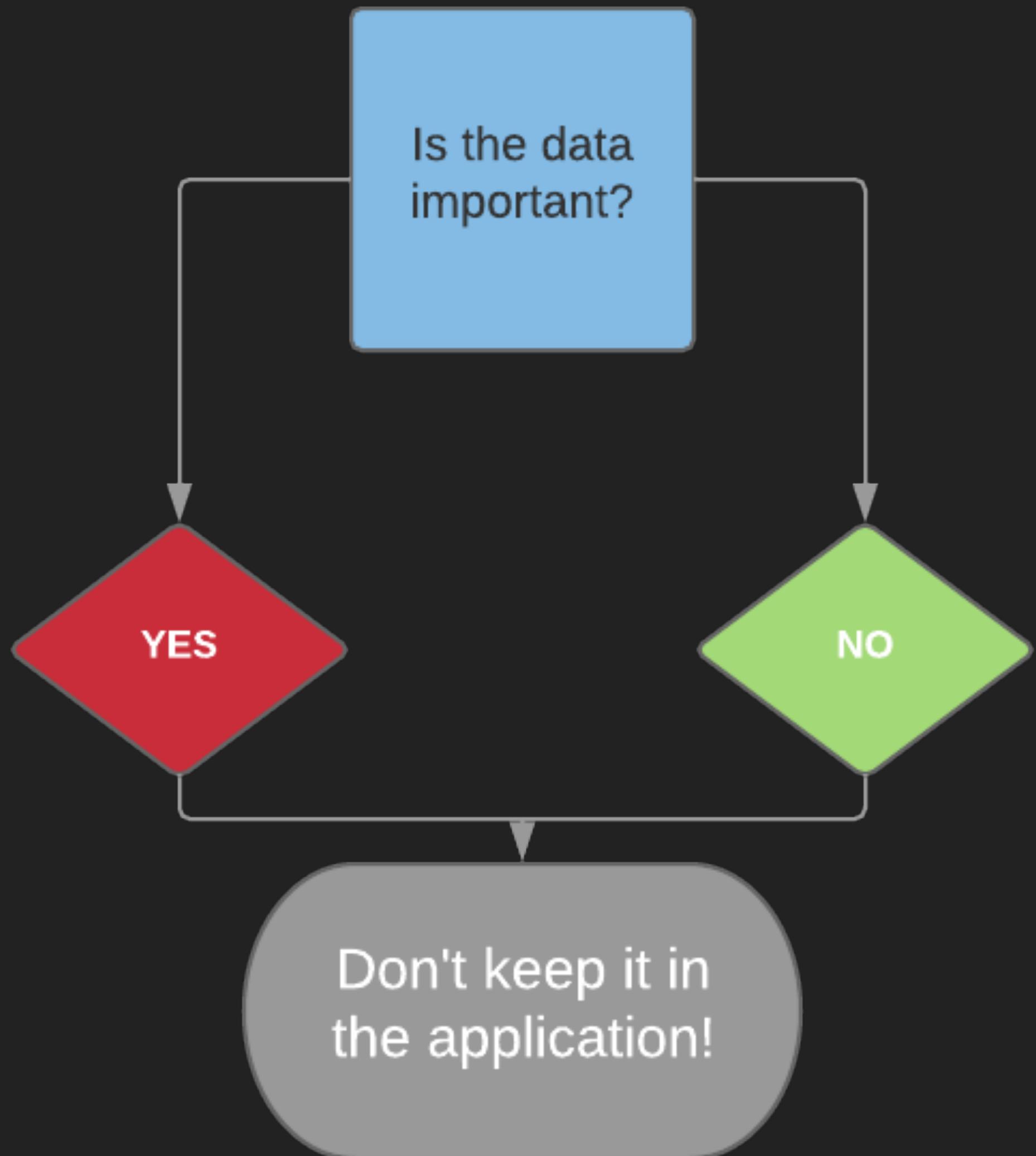


## ANDROID KEYSTORE SYSTEM



*The Android Keystore system safeguards your cryptographic keys from extraction.*

# SHOULD I KEEP SENSITIVE DATA IN MY APP?



### EXTRACTION PREVENTION

- ▶ System process in charge of cryptographic operations
- ▶ Key material bound to TEE, SE



# KEY USE AUTHORIZATIONS

- ▶ Once defined, immutable
- ▶ Cryptography
- ▶ Temporal validity interval
- ▶ User authentication



## UTILITY METHODS

- ▶ [KeyInfo.isInsideSecurityHardware\(\)](#)
- ▶ [KeyInfo.isUserAuthenticationRequirementEnforcedBySecureHardware\(\)](#)



# KEYCHAIN VS ANDROID KEYSTORE PROVIDER

- ▶ System-wide credentials
- ▶ App-specific credentials
- ▶ System provided UI
- ▶ No interaction
- ▶ AndroidKeystore API 18



## KEY PAIR ENTRY IN KEYSTORE

---

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance(  
    KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore");  
kpg.initialize(new KeyGenParameterSpec.Builder(  
    alias,  
    KeyProperties.PURPOSE_SIGN | KeyProperties.PURPOSE_VERIFY)  
    .setDigests(KeyProperties.DIGEST_SHA256,  
        KeyProperties.DIGEST_SHA512)  
    .build());  
  
KeyPair kp = kpg.generateKeyPair();
```

## KEY PAIR ENTRY IN KEYSTORE

---

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance(  
    KeyProperties.KEY_ALGORITHM_EC, "AndroidKeyStore");  
kpg.initialize(new KeyGenParameterSpec.Builder(  
    alias,  
    KeyProperties.PURPOSE_SIGN | KeyProperties.PURPOSE_VERIFY)  
    .setDigests(KeyProperties.DIGEST_SHA256,  
        KeyProperties.DIGEST_SHA512)  
    .build());  
  
KeyPair kp = kpg.generateKeyPair();
```

## KEY PAIR ENTRY IN KEYSTORE

---

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance(  
    KeyProperties.KEY_ALGORITHM_EC, "AndroidKeystore");  
kpg.initialize(new KeyGenParameterSpec.Builder(  
    alias,  
    KeyProperties.PURPOSE_SIGN | KeyProperties.PURPOSE_VERIFY)  
    .setDigests(KeyProperties.DIGEST_SHA256,  
        KeyProperties.DIGEST_SHA512)  
    .build());  
  
KeyPair kp = kpg.generateKeyPair();
```

## LISTING ENTRIES

---

```
KeyStore ks = KeyStore.getInstance("AndroidKeyStore");  
ks.load(null);
```

```
Enumeration<String> aliases = ksAliases();
```

## LISTING ENTRIES

---

```
KeyStore ks = KeyStore.getInstance("AndroidKeyStore");  
ks.load(null);
```

```
Enumeration<String> aliases = ksAliases();
```

## SIGNING DATA

---

```
Signature s = Signature.getInstance("SHA256withECDSA");
s.initSign((PrivateKeyEntry) entry).getPrivateKey());
s.update(data);
byte[] signature = s.sign();
```

## SIGNING DATA

---

```
Signature s = Signature.getInstance("SHA256withECDSA");
s.initSign((PrivateKeyEntry) entry).getPrivateKey());
s.update(data);
byte[] signature = s.sign();
```

## VERIFYING SIGNATURES

---

```
Signature s = Signature.getInstance("SHA256withECDSA");
s.initVerify(((PrivateKeyEntry) entry).getCertificate());
s.update(data);
boolean valid = s.verify(signature);
```

## VERIFYING SIGNATURES

---

```
Signature s = Signature.getInstance("SHA256withECDSA");
s.initVerify(((PrivateKeyEntry) entry).getCertificate());
s.update(data);
boolean valid = s.verify(signature);
```

# KEY ATTESTATION

- ▶ Is a key stored in hardware-backed keystore
- ▶ Small number of devices API 24+



# ANDROID P(ANCAKE?)

- ▶ Key rotation
- ▶ StrongBox Keymaster (HSM)
- ▶ ...



## GOOGLE TRANSPARENCY REPORT

---

<https://transparencyreport.google.com/>

**CHECK YOUR DOMAIN (NOW!)**

**<https://www.ssllabs.com/ssltest/analyze.html>**

Q&A

---

**THANK YOU!**