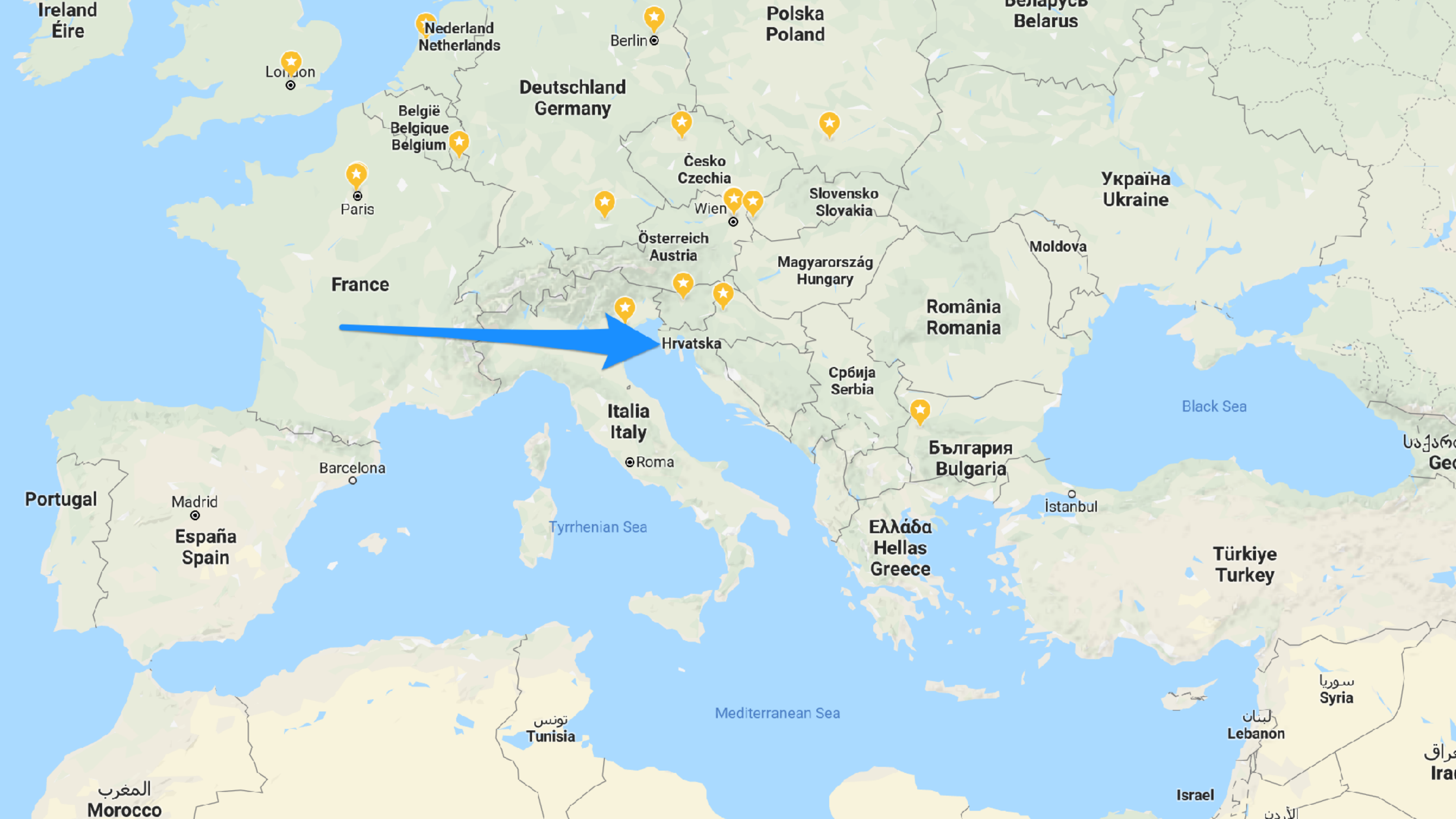
ANA BAOTIĆ

SIGN HERE, PLEASE!


Ireland
Éire
Nederland
Netherlands
Berlin
Polska
Poland
Беларусь
Belarus
London
Deutschland
Germany
België
Belgique
Belgium
Česko
Czechia
Slovensko
Slovakia
Україна
Ukraine
Paris
Wien
Österreich
Austria
Magyarország
Hungary
Moldova
France
România
Romania
Hrvatska
Србија
Serbia
Italia
Italy
Black Sea
България
Bulgaria
Barcelona
Roma
Portugal
Madrid
İstanbul
España
Spain
Tyrrhenian Sea
Ελλάδα
Hellas
Greece
Türkiye
Turkey
Mediterranean Sea
سوريا
Syria
تونس
Tunisia
لبنان
Lebanon
المغرب
Morocco
Israel

ASSECO at a glance

Present in
54
countries

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

Over
22,300
highly committed employees

6th
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

Electronic document that

proves ownership of a public key

CERTIFICATE

Electronic document that

proves ownership of a public key

X.509 - structure

CERTIFICATE

Electronic document that

proves ownership of a public key

X.509 - structure

TLS/SSL, electronic signatures

IDENTITY

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

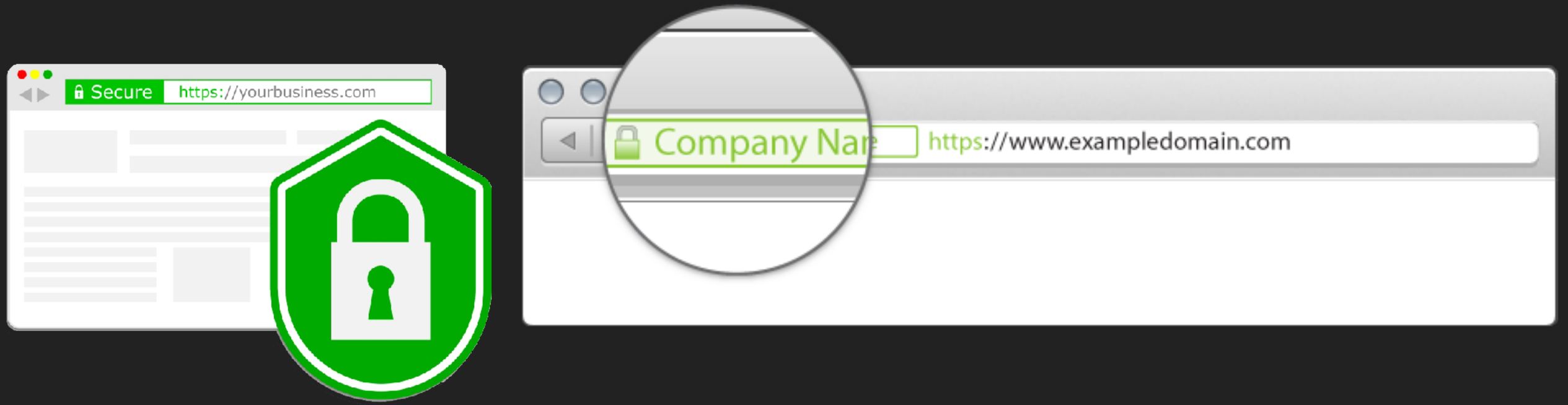
CERTIFICATE AUTHORITY

Trusted entity that issues digital certificates.

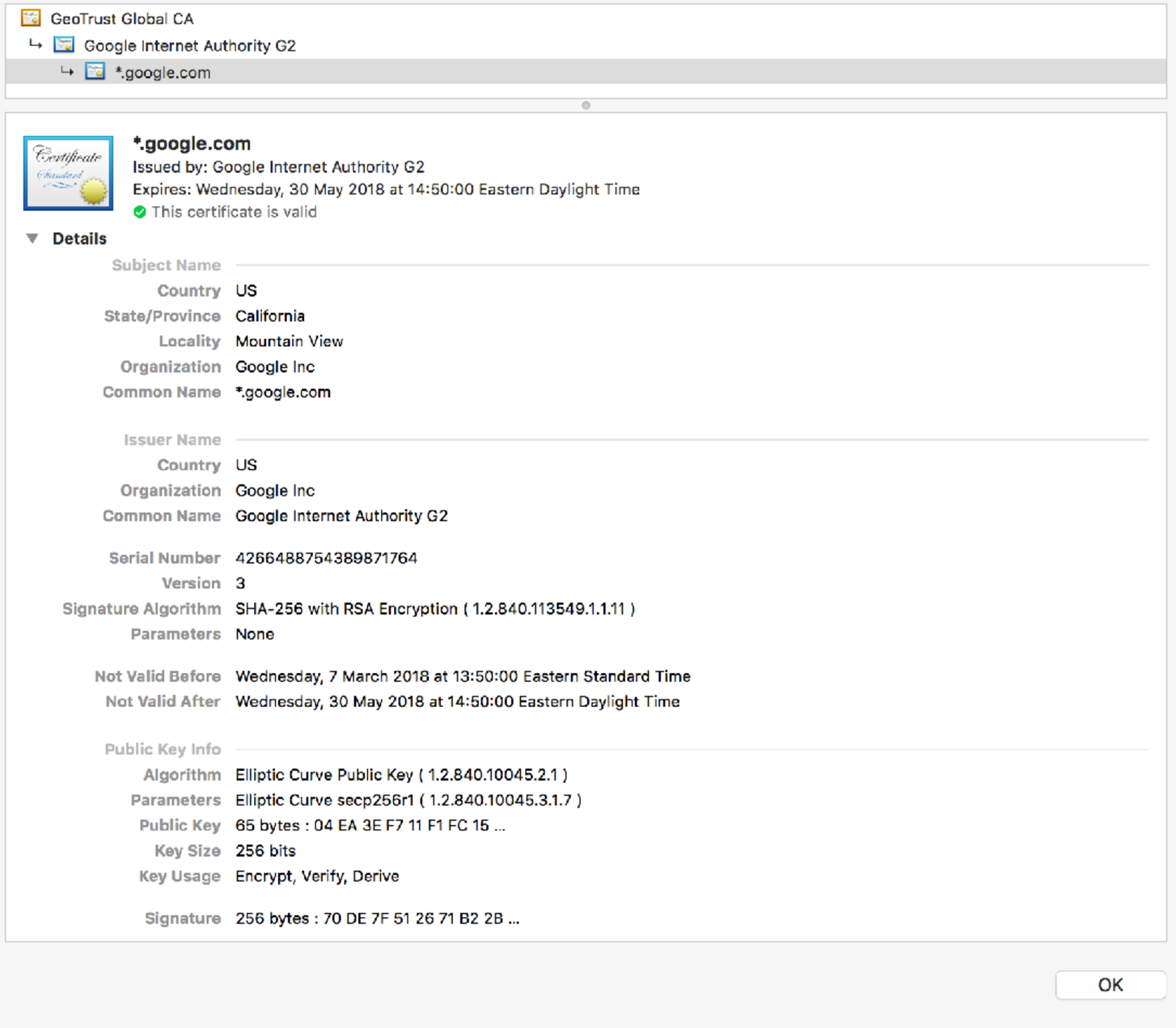
CA CERTIFICATE

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

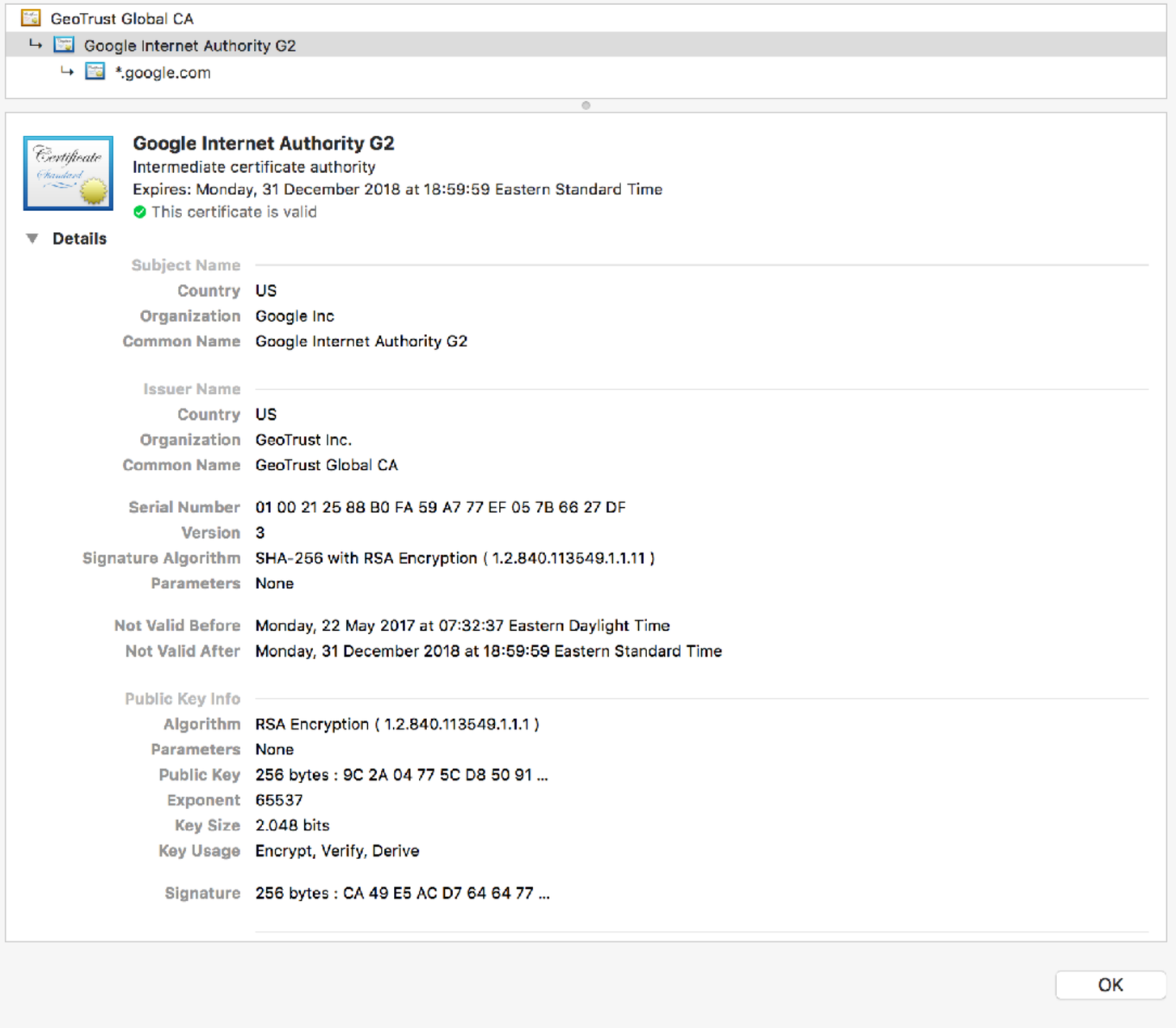
CA - DOMAIN VS EXTENDED VALIDATION



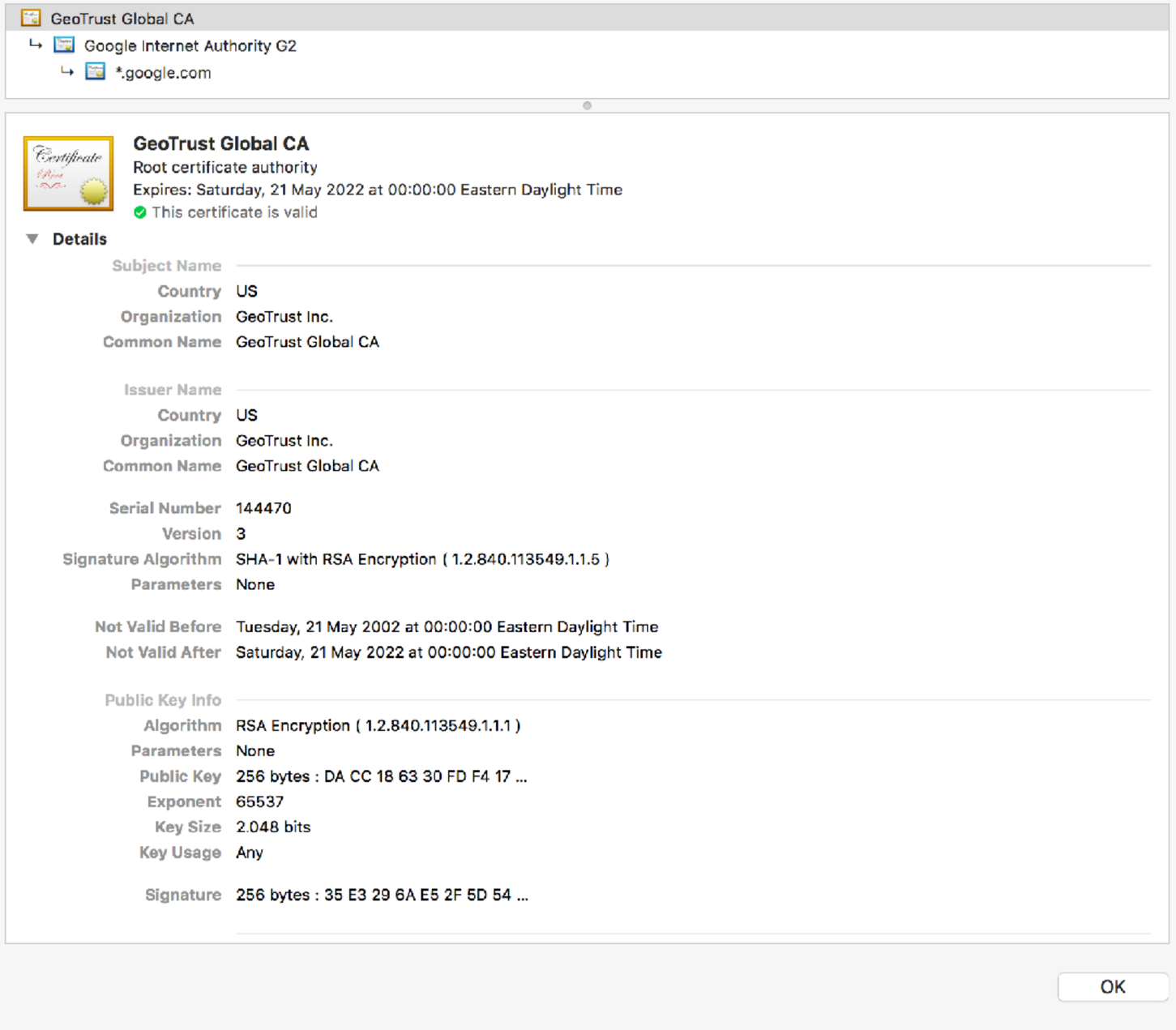
END CERTIFICATE



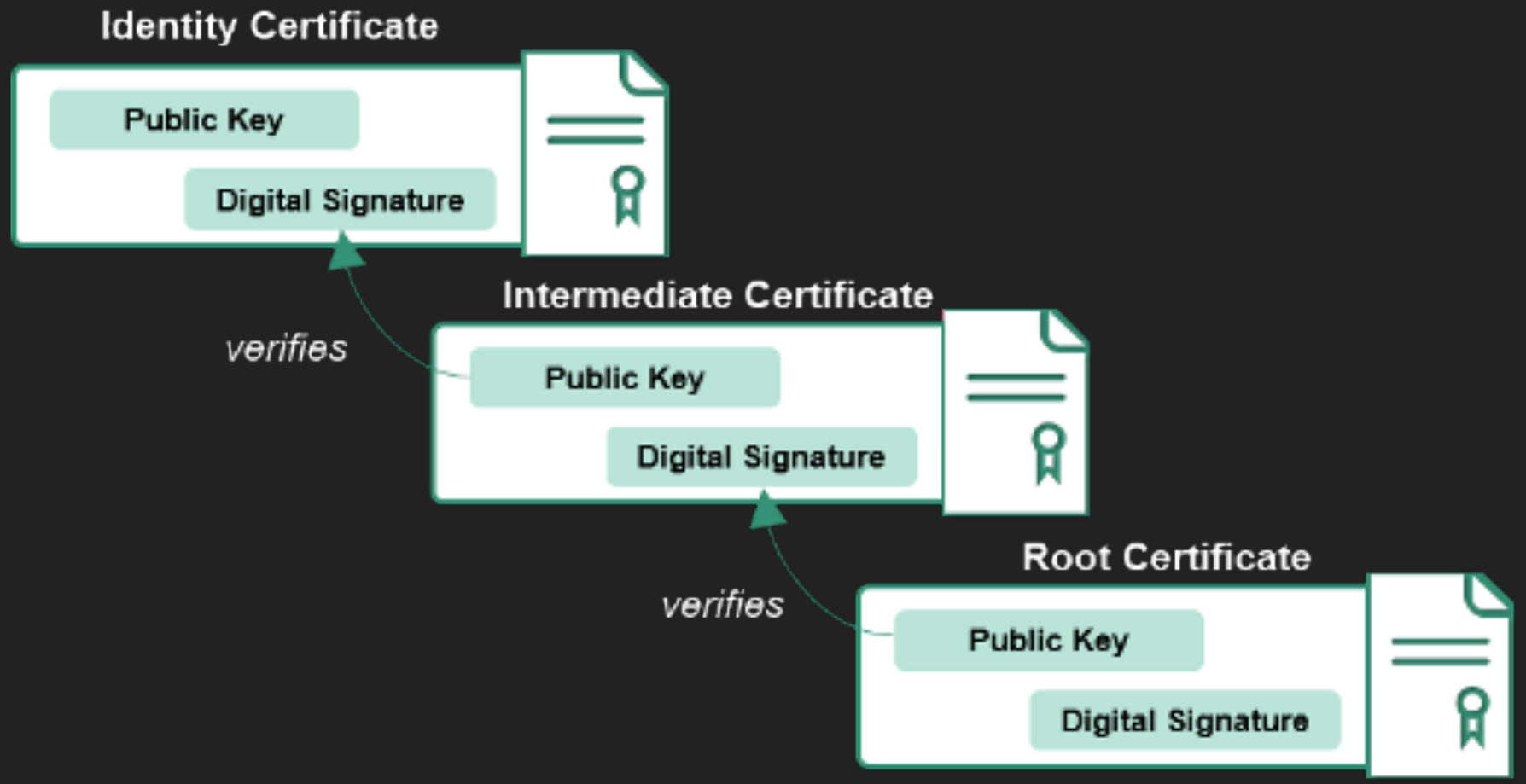
INTERMEDIATE CERTIFICATE



ROOT CERTIFICATE



CERTIFICATE CHAINS



SUBVERSION

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

ANDROID

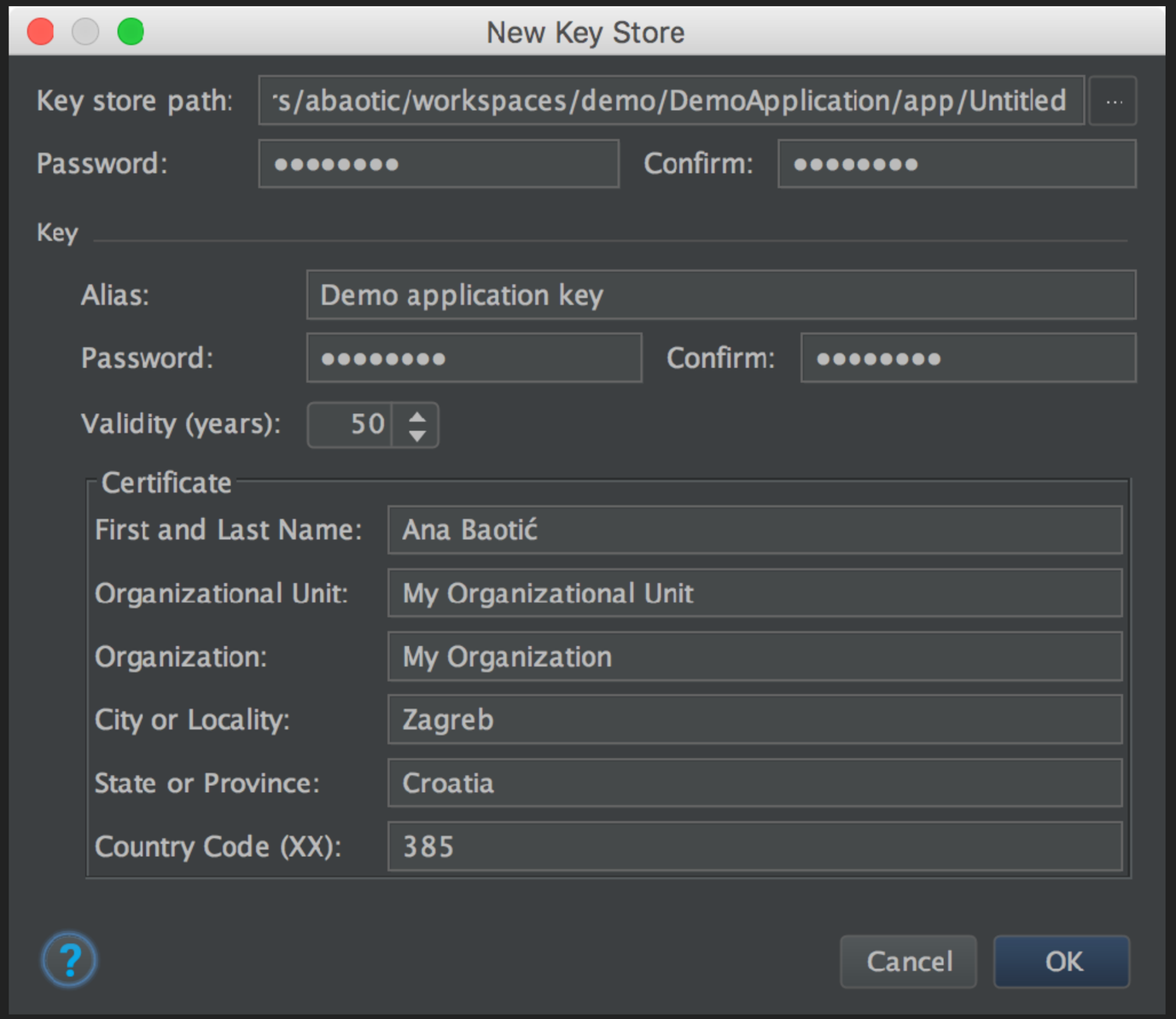
- Unknown CA
- Self-signed
- No intermediate cert on server

SIGNING ANDROID APPLICATIONS

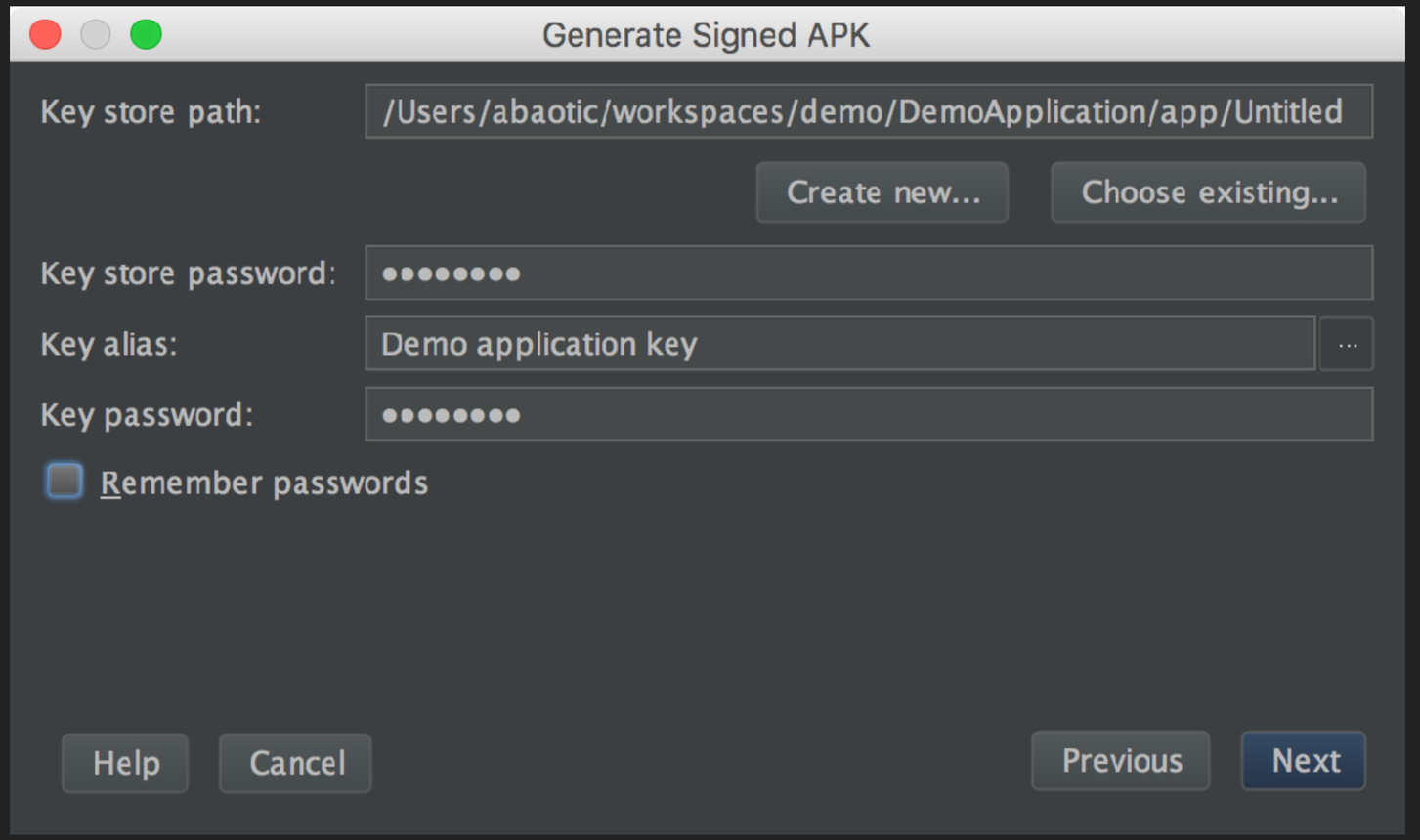
SIGNING ANDROID APPLICATIONS

- Generate a key and keystore
- Sign your application

SIGNING ANDROID APPLICATIONS - NEW KEY STORE



SIGNING ANDROID APPLICATIONS - SIGN THE APP



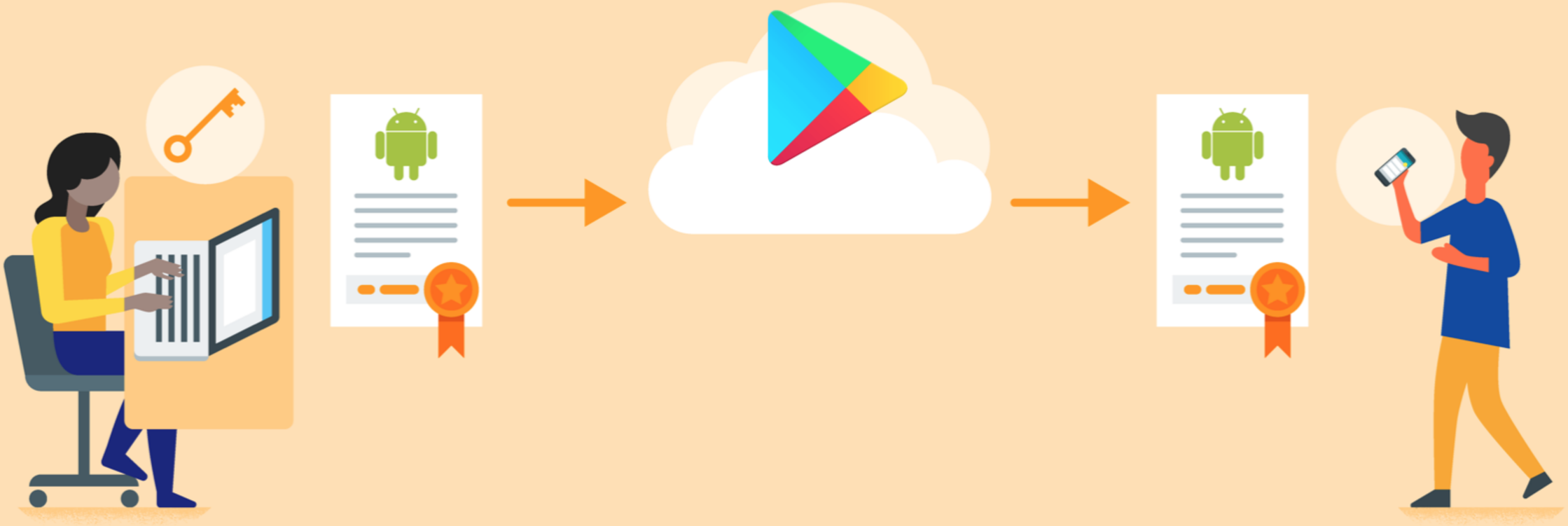
ALTERNATIVE

- 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

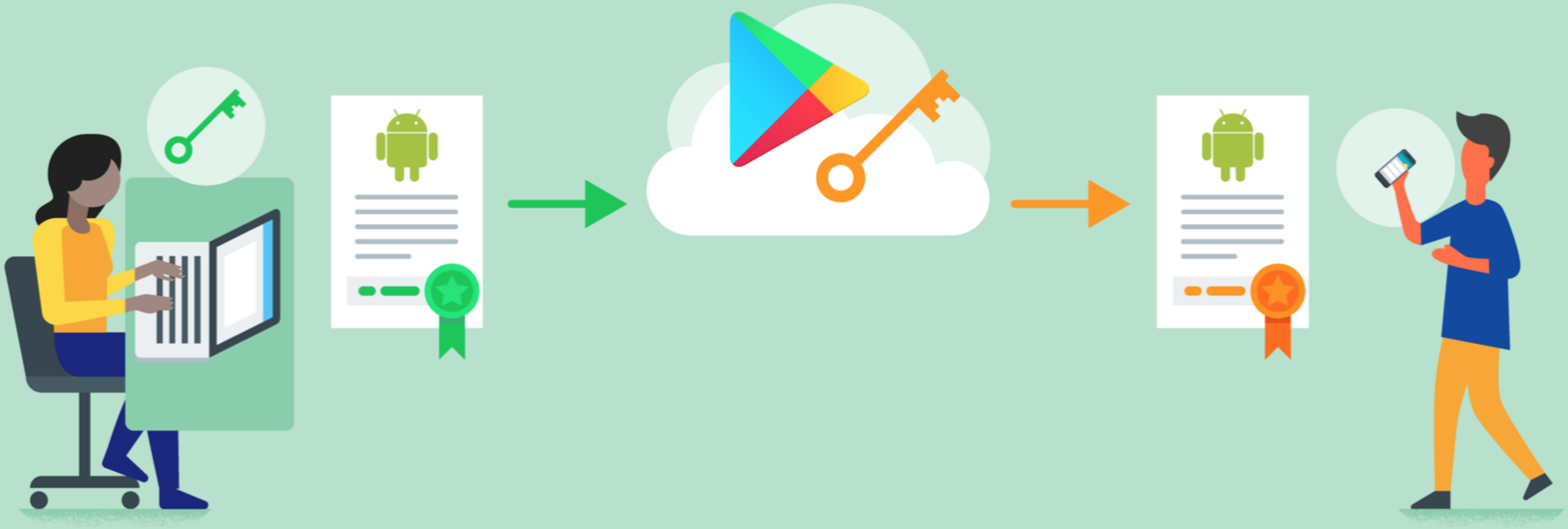
REALITY

- 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

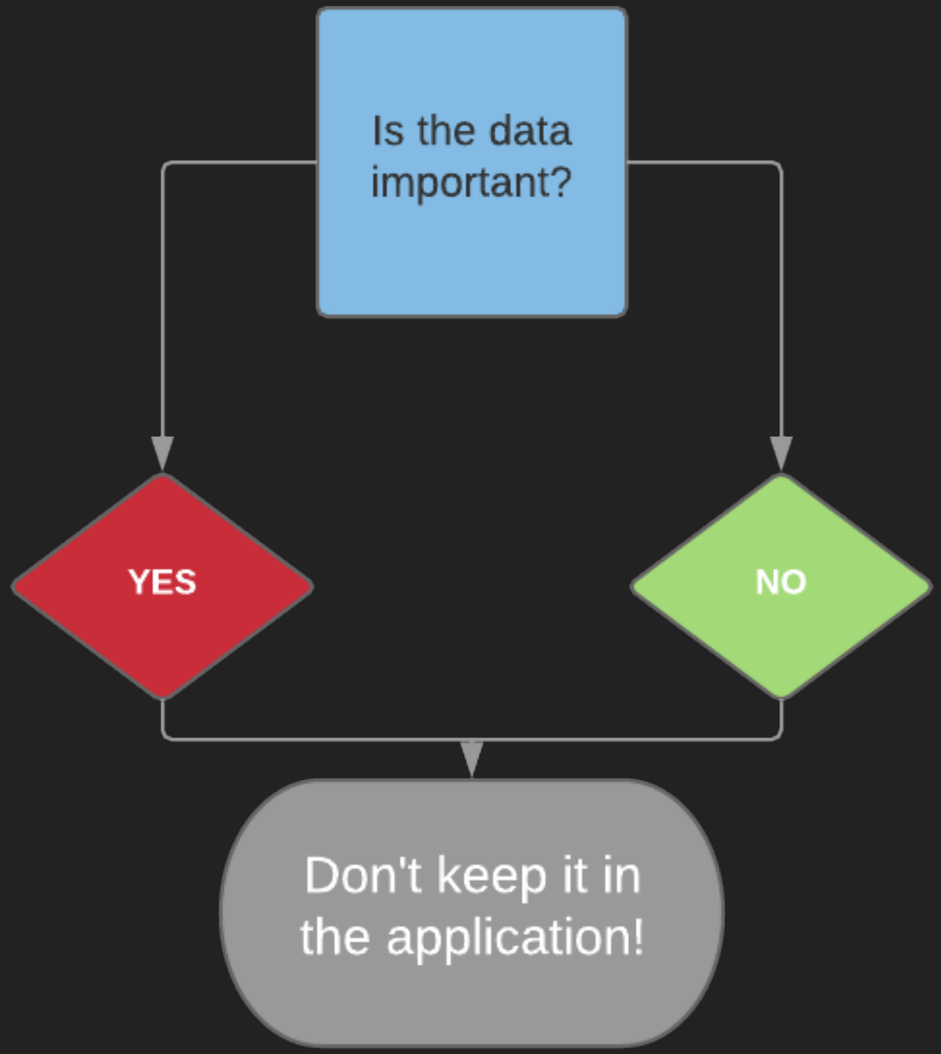
- Single point of failure

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
- System provided UI

- App-specific credentials
- 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 = ks.aliases();
```

LISTING ENTRIES

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

Enumeration<String> aliases = ks.aliases();
```

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

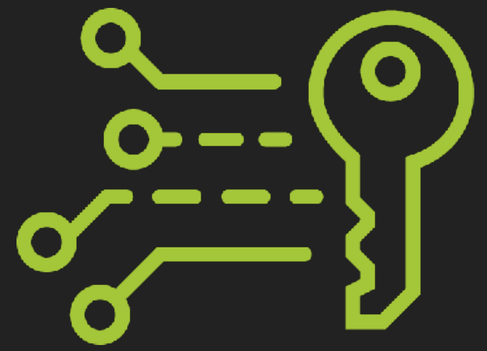
```
Signature s = Signature.getInstance("SHA256withECDSA");
s.initVerify(((PrivateKeyEntry) entry).getCertificate());
s.update(data);
boolean valid = s.verify(signature);
```

KEY ATTESTATION

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

ANDROID P(ANCAKE?)

- Key rotation
- StrongBox Keymaster (HSM)
- …



https://transparencyreport.google.com/

CHECK YOUR DOMAIN (NOW!)

https://www.ssllabs.com/ssltest/analyze.html

Q&A

THANK YOU!