© Developer Q Distribute Develop Support News Discover Design Account **Developer Forums** ? Q Search by keywords or tags

Certificate Signing Requests Explained



This thread has been locked by a moderator.



© 5.5k

I regularly help developers with code signing problems and I find that a lot of those problems stem from a fundamental misunderstanding of how code signing requests work. This post is my attempt at explaining that.

Share and Enjoy

Quinn "The Eskimo!" @ Developer Technical Support @ Apple let myEmail = "eskimo" + "1" + "@" + "apple.com"

Certificate Signing Requests Explained

I see a lot of folks confused by how code signing requests (CSRs) work, and that causes problems down the line. This is my attempt to explain

the process and head off those problems. IMPORTANT This post covers the 'classic' certificate creation process described in Developer Account Help > Create certificates > Create a

certificate signing request. Things work differently if you use Xcode to create your certificates, and very differently if you use cloud-managed certificates. Here's a basic outline of the CSR process:

1. You run Keychain Access and choose Certificate Assistant > Request a Certificate from a Certificate Authority.

- 2. You run through the workflow as described in Developer Account Help > Create certificates > Create a certificate signing request.
- 3. This does two things:
- - o It generates a public / private key pair in your keychain. To see these, run Keychain Access and select "login" on the left and Keys at the top. Look for keys whose names match the Common Name you entered in step 2. It prompts you to save a certSigningRequest file (CSR). This contains a copy of the public key.
- 4. You upload the CSR file to the developer web site. 5. The developer web site issues you a certificate. In human terms this certificate says "Apple certifies that the subject of this certificate
- holds the private key that matches the public key embedded in this certificate."

Note The developer web site sets the subject information in the certificate based on your developer account. It ignores the subject

information in the CSR. So, you can enter any information you want in step 2. This is a good way to distinguish between different keys in your keychain. For example, you might set the Common Name field in step 2 to include a unique identifier that allows you to easily identify the public / private key pair generated in step 3. 6. You download the certificate and add it to your keychain.

At this point your keychain contains a digital identity, that is, a certificate and the private key that matches the public key embedded in that

certificate. To see this in Keychain Access, select "login" on the left and My Certificates at the top.

There's an industry-wide terminology problem here. Folks use the term certificate to mean two different things:

What's This My Certificates Thing?

A digital identity, that is, a certificate and its matching private key

- An actual certificate
- This industry-wide confusion extends into the Apple ecosystem. For example:

• The Security framework gets this right, drawing a clear distinction between a digital identity (SecIdentity) and a certificate

- (SecCertificate). Keychain Access uses My Certificates for digital identities.
- Other user-facing apps use different terms. For example, Apple Configurator uses signing identity (yay for them!). OTOH, the help for Apple Mail uses the term personal certificate.
- Xcode and its documentation uses the term *signing certificate* to denote a digital identity that can be used for code signing.
- This terminological inexactitude causes all sorts of problems. For example, imagine you're setting up a new Mac. You download your certificate from the developer web site and then wonder why you can't sign your code. That's because the developer web site gives you a certificate, not a

digital identity. Indeed, the developer web site can't give you a digital identity because it never got a copy of your private key [1]. [1] Again, we're talking about the classic certificate creation process here; this statement is not true for cloud-managed certificates.

Digital Identity Formation

1. Extracting the public key from the certificate. 2. Calculating a SHA-1 digest of that.

Apple platforms form a digital identity by:

- 3. Looking for a private key whose kSecAttrApplicationLabel attribute matches that SHA-1 hash.
- For more background on this, see my SecItem attributes for keys post.
- Note that it's perfectly valid for multiple certificates to match against the same private key, yielding a digital identity for each certificate. You

regularly see this when you renew a certificate.

A CSR is a PEM file (PEM is short for Privacy-Enhanced Mail) with the CERTIFICATE REQUEST label:

Version: 0 (0x0)

Data:

SEQUENCE {

INTEGER 0 SEQUENCE { SET {

SEQUENCE {

Looking Inside a CSR

----BEGIN CERTIFICATE REQUEST----MIICgjCCAWoCAQAwPTEcMBoGCSqGSIb3DQEJARYNZnJvZ0Bmcm9nLmNvbTEQMA4G

```
Ur9x5voYb6CafUBZMMiYw6aFXcgnsx4ZXxe8VEgNCarrQi+9tgitDD/bCuymT5Da
 2+t64DGjpVM2lwtwqvH6Qh6QdPjkUw==
 ----END CERTIFICATE REQUEST----
To see inside, run the openssl tool as shown below:
 % openssl req -in CertificateSigningRequest.certSigningRequest -text -noout
 Certificate Request:
```

Subject: emailAddress=mrgumby@opendoor.com, CN=Mr Gumby, C=US Subject Public Key Info: Public Key Algorithm: rsaEncryption Public-Key: (2048 bit) Modulus:

% cat CertificateSigningRequest.certSigningRequest

00:b1:b4:a0:15:4d:4a:d7:29:1d:ed:d6:b7:c2:7c: 28:b9:8a:58:a4:04:63:fe:45:b2:4f:db:bd:93:20: 4e:8b Exponent: 65537 (0x10001) Attributes: a0:00 Signature Algorithm: sha256WithRSAEncryption 80:f9:0e:73:8e:42:d8:3c:e3:e0:06:54:13:d7:48:ef:a8:71: 2f:74:e1:2e:cf:e7:ed:3e:64:b4:78:85:f4:ac:38:07:b1:15: 6b:3c:39:f9 For even more details, convert the file to DER form and then dump that as ASN.1: % openssl req -in CertificateSigningRequest.certSigningRequest -out CertificateSigningRequest.der -outform der % dumpasn1 -p -a CertificateSigningRequest.der SEQUENCE {

IA5String 'mrgumby@opendoor.com' SET { SEQUENCE { OBJECT IDENTIFIER commonName (2 5 4 3)

OBJECT IDENTIFIER emailAddress (1 2 840 113549 1 9 1)

UTF8String 'Mr Gumby' SET { SEQUENCE { OBJECT IDENTIFIER countryName (2 5 4 6) PrintableString 'US' SEQUENCE { SEQUENCE { OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1) NULL BIT STRING, encapsulates { SEQUENCE { INTEGER 00 B1 B4 A0 15 4D 4A D7 29 1D ED D6 B7 C2 7C 74 28 B9 8A 58 A4 04 63 FE 45 B2 4F DB BD 93 20 4E INTEGER 65537 [0] Error: Object has zero length. SEQUENCE { OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11) NULL

% cat public.pem ----BEGIN PUBLIC KEY----MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsbSgFU1K1ykd7da3wnx0

I'm using the dumpasn1 tool, available here.

80 F9 0E 73 8E 42 D8 3C E3 E0 06 54 13 D7 48 EF

ED 3E 64 B4 78 85 F4 AC 38 07 B1 15 6B 3C 39 F9

To extract the public key from the CSR, run this command:

BIT STRING

FymGqUEcwfIISlG1C9VXyMRPzDcMrzjHm4i9qI9NliYouYpYpARj/kWyT9u9kyB0 iwIDAQAB ----END PUBLIC KEY----To further explore that key, use the techniques in my On Cryptographic Key Formats post. Security Code Signing Signing Certificates

Add a Comment

Topics & Technologies

Accessibility

% openssl req -in CertificateSigningRequest.certSigningRequest -noout -pubkey -out public.pem

This site contains user submitted content, comments and opinions and is for informational purposes only. Apple disclaims any and all liability for the acts, omissions and conduct Agreement.

Platforms

iOS

Developer

Forums

of any third parties in connection with or related to your use of the site. All postings and use of the content on this site are subject to the Apple Developer Forums Participation

Programs

Apple Developer Program

Posted 8 months ago by (2) eskimo (1)

Apple Developer Enterprise Program iPadOS Accessories Curriculum macOS **App Extensions** Downloads App Store Small Business Program tvOS App Store Forums MFi Program watchOS Audio & Video Videos **News Partner Program Augmented Reality** Video Partner Program Tools Support Security Bounty Program Business Swift Support Articles Design Security Research Device Program SwiftUI Contact Us Distribution SF Symbols **Bug Reporting Events** Education System Status App Accelerators Swift Playgrounds **Fonts** TestFlight App Store Awards Games Account Xcode Apple Design Awards Health & Fitness Apple Developer **Xcode Cloud** Apple Developer Academies In-App Purchase **App Store Connect Entrepreneur Camp** Localization Certificates, IDs, & Profiles Tech Talks Maps & Location Feedback Assistant WWDC **Machine Learning** Security Safari & Web

Resources

Documentation

To view the latest developer news, visit News and Updates Copyright © 2022 Apple Inc. All rights reserved. Terms of Use Privacy Policy License Agreements