数字证书和 PKI

概念

非对称密钥

非对称加密（Asymmetric Key Cryptography）的核心原理在于：**生成一对数学关联的密钥（密钥 A 和密钥 B），并确保它们具有单向解密的特性——即用密钥 A 加密的数据只能通过密钥 B 解密，反之亦然。**

- **公钥（Public Key）**：可以公开分发的密钥，用于加密数据或验证签名，任何人都可以使用。
- **私钥（Private Key）**：严格保密，仅由密钥持有者保存，用于解密数据或生成签名。

非对称加密算法在生成密钥对时，会基于特定的数学运算生成两把密钥。**公钥和私钥在生成时就已具备明确的数学关联和功能区分**，它们是本质不同的文件。以 RSA 算法为例，**私钥**包含**关键素数 p 和 q**，而**公钥**则由**模数 n = p · q 和公钥指数 e** 组成。

数字证书

数字证书（Digital Certificate）是一种**用于证明身份和确保通信安全**的**电子文档。**证书是一个结构化的文件，包含公钥、持有者的身份信息（如域名、组织名称）、颁发者信息、有效期等，并由 CA 使用其私钥签名以证明可信。

在实际应用中，数字证书、TLS 证书或简称证书**通常都指代这种 X.509 格式的文件。**

X.509 证书的结构由三个主要部分组成：证书主体（TBSCertificate）、签名算法标识和签名值。

以下是一个标准的 x509 终端证书：

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            05:bf:65:ff:fe:50:af:c1:e3:c0:6b:2a:5b:0e:d9:ad:05:34
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=US, O=Let's Encrypt, CN=R11
        Validity
            Not Before: May  9 01:05:37 2025 GMT
            Not After : Aug  7 01:05:36 2025 GMT
        Subject: CN=*.xiaoshae.cn
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:d9:d5:22:a4:b8:10:5d:7c:be:fe:5e:ec:8e:b1:
                    9f:c5:f6:5f:54:9d:f8:86:9f:fc:eb:1a:2b:0c:f1:
                    8b:69:16:ec:b0:d4:17:01:65:7a:5d:50:9b:d4:74:
```

```
                    97:e0:94:86:97:d0:a5:74:7b:db:28:d0:97:6e:97:
                    59:8e:37:4e:68:97:b8:30:38:04:38:93:ca:50:3d:
                    8e:6a:31:3d:21:56:21:40:57:b3:71:09:49:75:cb:
                    5d:14:cb:4a:8f:91:1f:d3:fc:f2:c5:3f:cd:61:1b:
                    9f:8b:3f:85:4f:90:21:71:52:98:f3:3f:a5:01:db:
                    11:2c:b1:77:db:7c:56:5b:96:5a:29:3c:ab:0b:d5:
                    4a:d8:6f:a4:1b:e5:3b:87:1b:4d:49:ee:cd:37:c7:
                    42:2d:a0:06:38:6c:1b:94:56:da:d6:22:35:01:79:
                    ac:46:e5:4f:5f:13:57:50:13:03:c5:43:8d:56:a8:
                    ff:02:6d:6f:30:1d:70:dd:d2:f2:5f:eb:f2:a2:25:
                    d8:3e:eb:3e:0a:40:1a:b1:af:bb:4f:47:87:1e:af:
                    b9:c4:ec:64:76:79:48:a2:81:83:2a:d8:f1:21:cf:
                    2f:d0:41:cd:b6:40:79:fa:f5:65:48:3e:32:c6:36:
                    b7:67:c7:ed:56:e4:b9:73:b9:69:f0:49:d9:b7:7d:
                    a6:9f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Key Identifier:
                CF:86:22:D5:73:E4:0A:86:FF:DC:28:4C:E2:F3:BA:92:96:51:17:76
            X509v3 Authority Key Identifier:
                C5:CF:46:A4:EA:F4:C3:C0:7A:6C:95:C4:2D:B0:5E:92:2F:26:E3:B9
            Authority Information Access:
                CA Issuers - URI:http://r11.i.lencr.org/
            X509v3 Subject Alternative Name:
                DNS:*.xiaoshae.cn, DNS:xiaoshae.cn
            X509v3 Certificate Policies:
                Policy: 2.23.140.1.2.1
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:http://r11.c.lencr.org/53.crl
            CT Precertificate SCTs:
                Signed Certificate Timestamp:
                    Version   : v1 (0x0)
                    Log ID    : 1A:04:FF:49:D0:54:1D:40:AF:F6:A0:C3:BF:F1:D8:C4:
                                67:2F:4E:EC:EE:23:40:68:98:6B:17:40:2E:DC:89:7D
                    Timestamp : May  9 02:04:07.964 2025 GMT
                    Extensions: none
                    Signature : ecdsa-with-SHA256
                                30:45:02:20:59:BE:36:DF:E0:DC:A9:A6:0E:BC:9B:59:
                                A9:0D:F5:6C:21:BB:0D:CB:9F:FA:B4:E8:9C:61:4A:4D:
                                A5:71:1A:0C:02:21:00:BE:D9:BE:C6:77:27:5D:39:28:
                                E9:0B:DF:ED:D7:2D:58:12:31:6D:73:64:CA:2F:27:04:
                                8B:2F:5C:09:C4:91:67
                Signed Certificate Timestamp:
                    Version   : v1 (0x0)
                    Log ID    : ED:3C:4B:D6:E8:06:C2:A4:A2:00:57:DB:CB:24:E2:38:
                                01:DF:51:2F:ED:C4:86:C5:70:0F:20:DD:B7:3E:3F:E0
```

```
                    Timestamp : May  9 02:04:09.442 2025 GMT
                    Extensions: none
                    Signature : ecdsa-with-SHA256
                                30:44:02:20:06:F3:07:DA:CB:3D:1E:C1:1E:E2:FD:7B:
                                F2:63:96:8F:E6:D0:13:6D:5C:63:1E:E9:6C:F6:5C:C2:
                                78:B7:FF:9B:02:20:46:49:9A:52:32:A4:93:24:8F:50:
                                F1:61:C7:B0:27:41:15:4D:88:19:80:15:99:7D:63:1B:
                                13:06:07:6B:DE:B9
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        31:4e:7b:07:d5:25:e0:be:91:4e:ff:d9:b5:59:7e:77:62:44:
        46:92:09:4a:b6:55:6c:16:01:c7:5c:ee:9a:9e:0f:e5:4b:92:
        4d:28:de:56:4f:e7:49:1e:b5:2e:eb:05:9d:28:cb:95:39:85:
        f3:b6:75:53:d6:b4:5c:2d:b4:c9:01:bd:d0:42:0f:cc:1c:4d:
        bc:67:94:37:67:15:c9:67:5d:f3:e0:62:56:84:a7:d8:7c:3b:
        fa:3a:e6:ea:96:5e:82:e4:71:cc:59:ac:5c:0a:30:ad:49:5b:
        aa:12:7a:83:ea:a5:78:61:e9:8b:3e:72:ef:be:62:d3:40:76:
        32:4a:df:c0:3e:a2:c1:29:51:89:aa:56:fe:74:54:c1:d6:de:
        4c:ba:1b:97:bf:20:74:11:8a:a0:f7:76:f5:a3:06:1a:24:0f:
        72:d2:28:38:c7:b5:90:be:2a:7e:c6:97:1f:b9:64:99:7e:74:
        b9:70:32:87:a3:dc:ef:59:c6:e0:f2:5b:1a:9d:bd:2c:91:39:
        00:22:6f:1f:83:4c:10:97:79:3e:7d:b3:b7:01:0d:3f:9a:b5:
        70:fe:a1:3a:92:db:04:6c:07:63:0f:68:1d:52:a6:d0:f7:31:
        f8:92:cc:c1:c7:a1:d0:c9:50:fa:03:44:d8:6a:e0:3b:6f:a7:
        fc:c1:5b:a2
```

标准 **x509 证书** 主要由三部分组成：**证书主体（TBS Certificate）**、**签名算法（Signature Algorithm）** 和 **签名值（Signature Value）**。

如果严格按照字段层级进行分类，则为**证书主体**（对应 Data）、**签名算法**（**Signature Algorithm**）和**签名值**（**Signature Value**）。

其中，公钥部分是证书主体中的一个子字段。

基本字段

```
Data:
    Version: 3 (0x2)
    Serial Number:
        05:bf:65:ff:fe:50:af:c1:e3:c0:6b:2a:5b:0e:d9:ad:05:34
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=US, O=Let's Encrypt, CN=R11
    Validity
        Not Before: May  9 01:05:37 2025 GMT
        Not After : Aug  7 01:05:36 2025 GMT
    Subject: CN=*.xiaoshae.cn
```

Version（版本号）：表示证书的 X.509 版本号。当前值为 3（最新）。

Serial Number（序列号）：数字证书的唯一标识符，该序列号随机生成。由证书颁发机构生成。

Signature Algorithm（签名算法）：**证书颁发机构**声明对当前证书签名时使用的算法。Data 外部为证书颁发机构签名时实际使用的算法。两者通常相同。

Issuer（颁发者）：签发此证书的**证书颁发机构的信息**。通常为上层证书的 **Subject**。根证书 Issuer 与 Subject 值相同。

Validity（有效期）：证书的有效时间范围，包含生效时间 (Not Before) 和到期时间 (Not After)。

- Not Before：生效时间
- Not After：到期时间

Subject（主体）：证书持有者的标识。

Subject 字段

Subject 字段用于标识证书持有者（实体），包含与公钥关联的身份信息。**该字段必须为非空的 X.500 可分辨名称（DN）**，由多个**属性-值对（AVP）**组成，其 ASN.1 结构与 Issuer 字段保持一致。

必须支持的属性类型（Mandatory）

国家（countryName，简称 C）：表示证书持有者所在国家，如 `C=CN`。

组织（organizationName，简称 O）：表示所属机构，如 `O=Example Inc.`。

组织单元（organizationalUnitName，简称 OU）：表示机构内的部门，如 `OU=Security Team`。

可分辨名称限定符（dnQualifier）：用于区分同名实体，通常为随机生成的值。

州/省（stateOrProvinceName，简称 ST）：表示所在地区，如 `ST=Beijing`。

通用名称（commonName，简称 CN）：通常为域名（如 `CN=www.example.com`）或个人姓名（如 `CN=张三`）。

序列号（serialNumber）：作为唯一标识符，确保实体可被精准识别。

建议支持的属性类型（Recommended）

地区/城市（localityName，简称 L）：表示所在城市或地区，如 `L=Shanghai`。

职位（title）：标识持有者的职位或头衔，如 `title=CTO`。

姓氏（surname，简称 SN）：表示持有者的姓氏，如 `SN=Zhang`。

名字（givenName，简称 GN）：表示持有者的名字，如 `GN=San`。

缩写（initials）：用于姓名缩写，如 `initials=ZS`。

别名（pseudonym）：提供匿名身份标识，适用于隐私保护场景。

代际限定符（generationQualifier）：用于区分同名家族成员，如 `generationQualifier=Jr.`。

TLS DV 数字证书 subject 字段

```
CN = grok.com
```

```
CN = xiaoshae.cn
```

TLS OV 数字证书 subject 字段

```
CN = WR2
O = Google Trust Services
C = US
```

```
CN = *.ccb.com
O = China Construction Bank
ST = 北京市
C = CN
```

```
CN = *.www.gov.cn
O = 国务院办公厅秘书局
L = 北京
ST = 北京
C = CN
```

```
CN = qwen.ai
O = 阿里巴巴（中国）网络技术有限公司
L = 杭州市
ST = 浙江省
C = CN
```

```
CN = *.shanghai.gov.cn
O = 上海市大数据中心
L = 上海市
ST = 上海市
C = CN
```

TLS EV 数字证书 subject 字段

```
CN = www.cmbchina.com
O = China Merchants Bank Co., Ltd
L = Shenzhen
ST = Guangdong Province
C = CN
serialNumber = 9144030010001686XA
businessCategory = Private Organization
jurisdictionLocalityName = Futian District
jurisdictionStateOrProvinceName = Guangdong Province
jurisdictionCountryName = CN
```

```
CN = www.boc.cn
O = Bank of China Limited
ST = Beijing
C = CN
serialNumber = 911000001000013428
businessCategory = Private Organization
jurisdictionStateOrProvinceName = Beijing
jurisdictionCountryName = CN
```

公钥字段

公钥字段是基本字段中的一部分。Subject Public Key Info 包含证书持有者的公钥信息，包括公钥算法和公钥本身。

```
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
            00:d9:d5:22:a4:b8:10:5d:7c:be:fe:5e:ec:8e:b1:
            ...（共2048位，256字节）...
        Exponent: 65537 (0x10001)
```

Public Key Algorithm（算法标识）：公钥使用的加密算法，这里是 RSA。

Public-Key（密钥长度）：标识密钥的长度，实际不包含在证书中，而是通过 Modulus 计算得出。

Modulus（模数）：RSA的 `n` 值（大整数，此处为2048位）

Exponent（指数）：RSA的 `e` 值（通常为65537）

扩展字段

X.509 版本 3 引入的扩展字段，提供了额外的功能和信息。

```
X509v3 extensions:
    X509v3 Key Usage: critical
        Digital Signature, Key Encipherment
    X509v3 Extended Key Usage:
        TLS Web Server Authentication, TLS Web Client Authentication
    X509v3 Basic Constraints: critical
        CA:FALSE
    X509v3 Subject Key Identifier:
        CF:86:22:D5:73:E4:0A:86:FF:DC:28:4C:E2:F3:BA:92:96:51:17:76
    X509v3 Authority Key Identifier:
        C5:CF:46:A4:EA:F4:C3:C0:7A:6C:95:C4:2D:B0:5E:92:2F:26:E3:B9
    Authority Information Access:
        CA Issuers - URI:http://r11.i.lencr.org/
    X509v3 Subject Alternative Name:
        DNS:*.xiaoshae.cn, DNS:xiaoshae.cn
    X509v3 Certificate Policies:
```

```
        Policy: 2.23.140.1.2.1
    X509v3 CRL Distribution Points:
        Full Name:
            URI:http://r11.c.lencr.org/53.crl
    CT Precertificate SCTs:
        Signed Certificate Timestamp:
            Version   : v1 (0x0)
            Log ID    : 1A:04:FF:49:D0:54:1D:40:AF:F6:A0:C3:BF:F1:D8:C4:
                        67:2F:4E:EC:EE:23:40:68:98:6B:17:40:2E:DC:89:7D
            Timestamp : May  9 02:04:07.964 2025 GMT
            Extensions: none
            Signature : ecdsa-with-SHA256
                        30:45:02:20:59:BE:36:DF:E0:DC:A9:A6:0E:BC:9B:59:
                        ...
        Signed Certificate Timestamp:
            Version   : v1 (0x0)
            Log ID    : ED:3C:4B:D6:E8:06:C2:A4:A2:00:57:DB:CB:24:E2:38:
                        01:DF:51:2F:ED:C4:86:C5:70:0F:20:DD:B7:3E:3F:E0
            Timestamp : May  9 02:04:09.442 2025 GMT
            Extensions: none
            Signature : ecdsa-with-SHA256
                        30:44:02:20:06:F3:07:DA:CB:3D:1E:C1:1E:E2:FD:7B:
                        ...
```

Key Usage（公钥用途）：指定证书公钥的用途，值 **Digital Signature, Key Encipherment**。这里允许用于数字签名和密钥加密。

Extended Key Usage（扩展公钥用途）：进一步指定公钥的用途，值 **TLS Web Server Authentication, TLS Web Client Authentication** 表示证书可用于 TLS 服务器身份验证和客户端身份验证。

Basic Constraints（基本约束）：指示该证书是否为 CA 证书，此处为 FALSE，表示这不是 CA 证书。防止该证书被用作 CA 来签发其他证书，增强安全性。

Subject Key Identifier（ SKI 主体密钥标识符）：证书中公钥的唯一标识符，通常是公钥的哈希值。

Authority Key Identifier（AKI 授权密钥标识符）：标识签发该证书的 CA 公钥的唯一标识符。

Authority Information Access（AIA）：提供 CA 证书的下载地址。

Subject Alternative Name（SAN）：列出证书适用的其他域名（SAN），包括通配符域名和主域名。扩展证书的适用范围，允许证书用于多个域名，现代浏览器通常优先检查 SAN 而非 Subject CN。

Certificate Policies：指定证书遵循的策略，值 **Policy: 2.23.140.1.2.1** 是 Let's Encrypt 的策略，符合 CA/Browser Forum 的域验证 (DV) 证书要求。

CRL Distribution Points：提供证书吊销列表 (CRL) 的下载地址。允许客户端检查证书是否被吊销。

CT Precertificate SCTs：证书透明性 (Certificate Transparency, CT) 的签名时间戳，证明证书已记录到 CT 日志中。增强证书透明性，防止未经授权的证书签发，现代浏览器要求 HTTPS 证书包含 SCT。

SKI 扩展

在 x509 v3 版本中存在一个 **Subject Key Identifier (SKI)** 扩展，该扩展字段的值类型一般为 SHA-1 哈希值，或直接取公钥的密钥标识符。标识当前证书的**公钥唯一性。**

```
X509v3 Authority Key Identifier:
    C5:CF:46:A4:EA:F4:C3:C0:7A:6C:95:C4:2D:B0:5E:92:2F:26:E3:B9
```

对于 RSA 密钥类型，**SKI** 值为 RSA公钥的 Modulus 和 Exponent 的 SHA-1 哈希值，以下式计算过程：

1. **将公钥的 `Modulus` 和 `Exponent` 编码为 DER 格式**（遵循 ASN.1 规则）。
2. **对 DER 编码后的二进制数据计算 SHA-1 哈希值**，结果即为 **SKI。**

数字签名

签名算法与签名值

签名原理

1. 将**证书主体（TBS Certificate）**转换为 ASN.1 DER 格式的二进制数据。
2. 对二进制数据计算其哈希值（此证书签名算法使用 **SHA-256** 哈希算法）。
3. 使用**私钥对计算出的哈希值进行加密**，加密后的结果即构成了证书中的 **Signature Value 字段的内容。**

证书中不存储原始哈希值，仅存储加密后的签名值（即 `Signature Value`）。**验证时**，验证方需要**自行对 TBS Certificate 进行 DER 编码并计算其哈希值。**

签名验证原理

- **从当前证书的 `Issuer` 字段中获取其上一级数字证书**，并从中提取出用于验证的公钥（例如，此处为 Let's Encrypt R11 的公钥）。
- 将当前证书中的**证书主体（TBS Certificate）转换为 ASN.1 DER 格式的二进制数据。**
- 依据证书中指明的**签名算法（Signature Algorithm）**，对这个 DER 编码后的数据计算其哈希值（此证书为 SHA-256 算法）。
- 利用之前获取到的上一级证书的公钥，**解密当前证书的 `Signature Value` 字段，从而得到原始的哈希值。**
- 将解密得到的原始哈希值与重新计算的哈希值进行比对。**若两者一致，则表明该证书的签名有效。**

证书链

在数字证书中，**上层证书不会完整嵌入在当前证书中**，而是通过引用关联。**确定上一层级证书（即颁发者CA的证书）**主要通过以下字段和机制实现：

Issuer 字段

声明当前证书的颁发者身份（即上层数字证书的信息）。

```
Issuer: C=US, O=Let's Encrypt, CN=R11
```

AKI 扩展

Authority Key Identifier 扩展标识**上层数字证书的公钥**，此扩展的值为**上层数字证书的 SKI**。

```
X509v3 Authority Key Identifier:
    C5:CF:46:A4:EA:F4:C3:C0:7A:6C:95:C4:2D:B0:5E:92:2F:26:E3:B9
```

AIA 扩展

Authority Information Access (AIA) 扩展 用于指定证书链中**上一层数字证书的下载地址**，其值为一个 **URL**。

```
Authority Information Access:
    CA Issuers - URI:http://r11.i.lencr.org/
```

PKI

公钥基础设施是一套**用于管理数字证书和公钥加密的系统**，其核心组件包括：

- **证书颁发机构（CA）**：负责颁发和撤销数字证书。
- **注册机构（RA）**：协助 CA 验证申请者的身份。
- **数字证书**：基于 X.509 标准，包含公钥、身份信息和 CA 签名。
- **私钥和公钥**：用于加密、解密和签名。
- **证书撤销列表（CRL）**：记录被撤销的证书。
- **信任链**：由根 CA、中间 CA 和终端实体证书组成。

OpenSSL 提供了实现上述组件的工具，允许用户创建自己的 CA、管理证书生命周期、生成 CRL 并验证信任链。

openssl

OpenSSL 提供了多个命令用于生成非对称密钥（私钥和公钥）以及相关参数，主要涉及以下命令：

早期 OpenSSL 使用专用命令**（genrsa / gendsa / ecparam）**生成密钥，操作分散且复杂。现代版本改用通用命令 **genpkey**（支持多种算法）和 **pkey**（统一管理密钥），简化流程并提高灵活性。**pkeyutl** 取代 **rsautl** 实现通用加密/签名，整体设计更简洁高效。

genpkey

openssl genpkey 是一个用于生成私钥或密钥对的通用命令，支持多种公钥算法（如 RSA、EC、DSA、DH 等），是 OpenSSL 中推荐的非对称密钥生成工具，取代了旧的专用命令（如 genrsa、gendsa）。以下是其所有参数的详细说明：

-algorithm alg

指定使用的公钥算法。支持的算法：

- 私钥生成：RSA、RSA-PSS、EC、X25519、X448、ED25519、ED448。
- 参数生成（需配合 -genparam）：DH、DSA、EC。

```
openssl genpkey -algorithm EC -out eckey.pem
```

必须在 -pkeyopt 之前指定。与 -paramfile 互斥。

-paramfile filename

指定参数文件，用于基于已有参数生成私钥。

```
openssl genpkey -paramfile dsaparam.pem -out dsakey.pem
```

与 -algorithm 互斥，参数文件决定算法类型。

-genparam

生成算法参数而非私钥。

支持的算法：DH、DSA、EC。

必须在 **-algorithm、-paramfile 或 -pkeyopt** 之前指定。生成的参数可用于后续密钥生成。

```
openssl genpkey -genparam -algorithm DH -out dhparam.pem
```

-out filename

指定私钥或参数的输出文件。

如果未指定，输出到标准输出（stdout）。

```
openssl genpkey -algorithm RSA -out key.pem
```

-outform DER|PEM

指定输出格式，PEM（文本格式，Base64 编码）或 DER（二进制格式）。默认：PEM。

仅适用于密钥输出，**使用 -genparam 生成参数时，-outform 被忽略，输出格式固定为 PEM**。PEM 格式更常见，易于阅读和传输。

```
openssl genpkey -algorithm RSA -out key.der -outform DER
```

-pass arg

指定输出私钥的加密密码来源。

与 **-cipher** 配合使用，加密私钥以增强安全性。

参考 openssl-passphrase-options(1)，支持格式如 pass:password、env:var、file:filename 等。

```
openssl genpkey -algorithm RSA -out key.pem -cipher aes256 -pass pass:secure123
```

-cipher

指定加密私钥时使用的对称加密算法（如 aes256、des3）。

需要与 **-pass** 配合使用，加密算法必须是 **EVP_get_cipherbyname()** 支持的算法。

```
openssl genpkey -algorithm RSA -out key.pem -cipher aes256 -pass pass:secure123
```

-verbose

在生成密钥时显示“状态点”（progress dots），表示生成进度。

```
openssl genpkey -algorithm RSA -verbose -out key.pem
```

-quiet

禁止显示“状态点”，保持输出简洁。

```
openssl genpkey -algorithm RSA -quiet -out key.pem
```

与 **-verbose** 互斥，适合脚本或自动化任务。

-text

以明文形式打印私钥、公钥或参数的详细信息（不加密），连同 PEM 或 DER 结构。

```
openssl genpkey -algorithm RSA -out key.pem -text
```

-config configfile

指定配置文件，覆盖默认的 **openssl.cnf**。

配置文件可定义默认参数、算法选项等，详见 config(5)。

```
openssl genpkey -algorithm RSA -out key.pem -config custom.cnf
```

-pkeyopt opt:value

设置特定算法的选项，具体选项因算法而异。

```
openssl genpkey -algorithm RSA -out key.pem \
    -pkeyopt rsa_keygen_bits:4096 \
    -pkeyopt rsa_keygen_primes:3
```

可通过 **openssl genpkey -algorithm XXX -help** 查看某算法支持的选项。详见下文的“密钥生成选项”和“参数生成选项”。

密钥选项

RSA 密钥生成选项

rsa_keygen_bits:numbits

指定 RSA 密钥的位数，**默认值为 2048 位。**建议至少使用 2048 位，推荐 3072 或 4096 位以增强安全性。

用法示例：**-pkeyopt rsa_keygen_bits:3072**。

rsa_keygen_primes:numprimes

设置生成 RSA 密钥的素数个数，默认为 2。多素数 RSA 可提升密钥生成速度，但需注意安全性评估。

用法示例：**-pkeyopt rsa_keygen_primes:3**。

rsa_keygen_pubexp:value

指定 RSA 公钥指数，默认值为 65537。支持十进制或十六进制（如 0x 前缀），常用值为 3 或 65537。

用法示例：**-pkeyopt rsa_keygen_pubexp:3**。

EC 密钥生成选项

ec_paramgen_curve:curve

指定椭圆曲线名称，支持 NIST 标准曲线如 P-256（secp256r1）、P-384（secp384r1）等。

用法示例：**-pkeyopt ec_paramgen_curve:P-256**。

查看完整曲线列表可使用命令：**openssl ecparam -list_curves**。

ec_param_enc:encoding

设置椭圆曲线参数的编码格式，默认为 **named_curve**（仅引用曲线名称），也可选 **explicit**（包含完整参数）。

用法示例：**-pkeyopt ec_param_enc:named_curve**。

私钥能推导出公钥，本质上是**通过私钥的数学参数计算出公钥**，私钥中存储了生成公钥所需的全部信息。

pkey

`openssl pkey` 是 OpenSSL 工具集中的一个核心组件，专门用于处理公钥和私钥。**它支持多种功能**，包括密钥格式转换、密钥验证、加密解密操作以及密钥信息提取等，适用于各类密钥管理场景。

该命令的选项可分为三大类：**通用选项**用于设置基础操作参数，**输入选项**用于指定密钥来源及其格式，**输出选项**则控制密钥的导出方式及内容展示。

通用选项

-check

此选项用于检查密钥对中公钥和私钥组件的一致性。

主要用于检查私钥的数学一致性，**不适用于公钥**。

-pubcheck

此选项用于检查公钥或密钥对中公钥组件的正确性。

仅适用于某些特定算法（如 DSA、ECDSA），**不适用于 RSA 公钥**。

输入选项

-in filename|uri

指定输入文件或 URI，包含要处理的公钥或私钥。如果未指定，默认为标准输入。 如果输入的密钥是加密的且未提供 -passin，会提示输入密码。

-inform DER|PEM|P12|ENGINE

指定输入密钥的格式，默认根据文件内容自动检测。

- PEM：Base64 编码的文本格式（常见）。
- DER：二进制格式。
- P12：PKCS#12 格式（常用于证书和密钥的打包）。
- ENGINE：通过加密引擎加载密钥。

-pubout

仅输出公钥部分（即使输入包含私钥）。与 **-text** 结合时，等效于 **-text_pub**。

未指定 **-pubout** 参数则输出内容为私钥，如果指定 **-pubout** 参数则输出内容为公钥。无法同时输出公钥和私钥。

-passin arg

指定输入密钥的密码来源。格式见 **openssl-passphrase-options(1)**，常见格式包括：

- **pass:password**：直接指定密码。
- **file:filename**：从文件中读取密码。
- **env:var**：从环境变量读取密码。
  示例：-passin pass:secret123

-pubin

指定输入文件为公钥（而不是默认的私钥）。如果输入仅包含私钥，会自动提取其公钥部分。

输出选项

-out filename

指定输出文件，保存编码后的密钥或文本信息。如果未指定，输出到标准输出。

注意：输出文件会覆盖输入文件（如果文件名相同），但文件 I/O 非原子操作。

-outform DER|PEM

指定输出密钥的格式，默认是 PEM。

- PEM：文本格式，适合大多数应用场景。
- DER：二进制格式，常用于需要严格格式的场景。

-cipher

使用指定的加密算法加密输出的 PEM 私钥（如 aes128、des3）。需要结合 **-passout** 提供密码。

注意：DER 格式不支持加密。

-passout arg

指定输出文件的密码来源，格式同 -passin。

示例：**-passout pass:secret123**

-traditional

使用传统的私钥格式，而非默认的 PKCS#8 格式。

PKCS#8 是更现代的格式，支持多种算法和加密。

-text

以明文形式输出密钥的详细信息（如 RSA 的模数、指数或 EC 的参数）。可与编码输出结合，但不能与 DER 格式结合。

-text_pub

仅输出公钥部分的明文信息，不能与 DER 格式结合。

-noout

不输出编码后的密钥，仅输出文本信息（需结合 **-text** 或 **-text_pub**）。

- 仅使用 **-text**（未指定 **-noout**）：输出 PEM 格式密钥及明文信息。
- 同时使用 **-text 和 -noout**：仅输出明文信息，不显示 PEM 格式密钥。
- 单独使用 **-noout**（未指定 **-text** 或 **-text_pub**）：无任何输出。

-ec_conv_form arg

（仅限椭圆曲线密钥）指定椭圆曲线点的编码格式：

- compressed（默认）：压缩格式，占用空间小。
- uncompressed：未压缩格式，包含完整点坐标。
- hybrid：混合格式。

注意：由于专利问题，二进制曲线的压缩格式默认禁用，需在编译时定义 OPENSSL_EC_BIN_PT_COMP 启用。

-ec_param_enc arg

（仅限椭圆曲线密钥）指定椭圆曲线参数的编码方式：

- named_curve（默认）：使用曲线 OID（如 secp256r1）。
- explicit：显式编码曲线参数（如素数、生成点等）。

注意：implicitlyCA 目前未实现。

x509

openssl x509 是一个多功能的证书处理命令，用于显示、转换、编辑信任设置、生成证书或证书请求，并支持自签名或作为“微型CA”签名。

输入、输出和通用选项

-new

从头生成一个新证书，而不是基于现有证书或请求。需配合 -set_subject 指定主体名称，公钥可通过 -force_pubkey 指定，默认使用 -key 或 -signkey 提供的密钥（自签名）。

openssl 3.0 and latest

-req

指定输入为PKCS#10证书请求（默认期望输入为证书）。请求需自签名，扩展默认不复制，可通过-extfile指定。

-in filename|uri

指定输入文件或 **URI**，读取证书或证书请求（与 **-req** 配合使用）。默认从标准输入读取。

注意：不能与 **-new** 选项一起使用。

-inform DER|PEM

指定输入文件格式，默认为PEM。支持DER或PEM。

-passin arg

指定输入文件（如私钥或证书）的密码来源。

-x509toreq

将证书转换为PKCS#10证书请求，需使用 -key 或 -signkey 提供私钥进行自签名，公钥放入请求的 subjectPKInfo 字段。

默认不复制输入证书的扩展，可通过 -extfile 添加扩展。

-copy_extensions arg

处理从证书到请求（-x509toreq）或从请求到证书（-req）的扩展复制行为：

- none：忽略扩展（默认）。
- copy 或 copyall：复制所有扩展（生成请求时不复制主体标识和颁发者密钥标识扩展）。可结合 -ext 进一步限制复制的扩展。

-vfyopt nm:v

传递验证操作的签名算法选项，具体名称和值因算法而异。

-key filename|uri / -signkey filename|uri

指定用于签名新证书或请求的私钥，公钥自动放入证书或请求（除非使用 **-force_pubkey** ）。

- 设置颁发者名称为主体名称（自颁发）。
- 除非使用 -preserve_dates，否则有效期起始时间为当前时间，结束时间由 -days 决定。
- 不能与 -CA 一起使用。-signkey是-key的别名。

-keyform DER|PEM|P12|ENGINE

指定私钥输入格式，默认未指定。

-out filename

指定输出文件名，默认输出到标准输出。

-outform DER|PEM

指定输出格式，默认PEM。

-nocert

不输出证书内容，仅输出由其他选项（如打印选项）请求的内容。

-noout

禁止输出，仅打印由其他选项（如-text、-serial等）指定的信息。

证书输出选项

-set_serial n

设置证书序列号（十进制或以 0x 开头的十六进制）。可与 -key、-signkey 或 -CA 一起使用。若与 -CA 一起使用，则不使用 -CAserial 指定的序列号文件。

-next_serial

将序列号设置为输入证书序列号加1。

-set_issuer arg

设置证书的颁发者名称，格式同 **-set_subject**。

openssl version 3.3 and latest

-set_subject arg / -subj arg
设置证书的主体名称，格式为`/type0=value0/type1=value1/...`，支持反斜杠转义特殊字符，空值允许，多值RDN用+分隔。

例如：`/DC=org/DC=OpenSSL/DC=users/UID=123456+CN=John Doe`

可与 **-new** 和 **-force_pubkey** 一起使用生成新证书。

openssl version 3.3 and latest

-subj 是 **-set_subject** 的别名。

openssl version 3.3 and latest

-days arg

设置新证书从今天起的有效期（天数），默认 30 天。

不能与 **-preserve_dates** 或 **-not_after** 一起使用。

-not_before date

显式设置证书生效日期，格式为 **YYMMDDHHMMSSZ（ASN1 UTCTime）**或 **YYYYMMDDHHMMSSZ（ASN1 GeneralizedTime）**，或 **today**。不能与 **-preserve_dates** 一起使用。

openssl version 3.4 and latest

-not_after date

显式设置证书到期日期，格式同上。不能与 **-preserve_dates** 一起使用，优先于 **-days**。

openssl version 3.4 and latest

-preserve_dates

签名时保留输入证书的 **notBefore** 和 **notAfter** 日期，不能与 **-days**、**-not_before** 或 **-not_after** 一起使用。

-force_pubkey filename

设置证书或请求的公钥为指定文件中的公钥，而不是输入或 -key 中的公钥。适用于生成自颁发但非自签名的证书（如DH密钥）。

-clrext

生成新证书或请求时，不保留输入的扩展。生成请求时，主体标识和颁发者密钥标识扩展不包含。

-extfile filename

指定包含X.509扩展的配置文件。

-extensions section

指定 **-extfile** 中要添加的扩展部分，详见 x509v3_config(5) 。

-sigopt nm:v

签名操作时传递给签名算法的选项，可多次使用，具体选项因算法而异。

-badsig

签名时故意破坏签名，用于测试。

-digest

`digest` 选项不是独立使用的，它通常与其他选项（如 `-fingerprint` 用于生成证书指纹）结合使用。

用于指定计算证书指纹时使用的哈希算法。

如果未指定，或仅指定 `-fingerprint` 未指定 `-digest`，则输出证书指纹时，默认使用 SHA1 算法进行计算。签名算法会使用默认的摘要算法（通常是 SHA256）。

示例：

```
openssl x509 -in ca-cert.pem -text -noout -fingerprint
```

计算证书指纹时使用 SHA-1

```
openssl x509 -in ca-cert.pem -text -noout -fingerprint -sha512
```

计算证书指纹时使用 SHA-512

```
openssl x509 -new -key private.key -out cert.pem -days 3650 -subj "/C=CN"
```

数字签名时，计算证书指纹哈希算法为 SHA-256

```
openssl x509 -new -key private.key -out cert.pem -days 3650 -subj "/C=CN" -sha512
```

数字签名时，计算证书指纹哈希算法为 SHA-512

微型 CA 选项

-CA filename|uri

指定 CA 证书，用于签名新证书。设置新证书的颁发者为 CA 的主题。

-CAkey filename|uri

指定 CA 私钥，用于签名。若未提供，私钥需包含在 -CA 输入中。

-CAform DER|PEM|P12

CA 证书的格式。

-CAkeyform DER|PEM|P12|ENGINE

CA 私钥的格式。

-CAserial filename

指定序列号文件，存储上次使用的序列号（十六进制）。默认文件名为 CA 证书文件名加 .srl。

-CAcreateserial

如果序列号文件不存在，创建并使用随机序列号。

证书检查选项

-checkend arg

检查证书是否在未来 arg 秒内到期（返回非零表示即将到期）。

-checkhost host

验证证书是否匹配指定主机名。

-checkemail email

验证证书是否匹配指定电子邮件地址。

-checkip ipaddr

验证证书是否匹配指定 IP 地址。

证书打印选项

-text

以文本形式打印证书的完整信息，包括公钥、签名算法、扩展等。

-certopt option

自定义 -text 的输出格式，支持多个选项（如 no_header、no_pubkey、no_extensions）。见“Text Printing Flags”部分。

-fingerprint

计算并打印证书的指纹（DER 编码的摘要，通常是 SHA1）。

-serial

打印证书序列号。

-subject

打印主题名称。

-issuer

打印颁发者名称。

-startdate / -enddate / -dates

分别打印证书的生效日期、到期日期或两者。

-nameopt option

控制主题或颁发者名称的显示格式（如 RFC2253、oneline）。见 openssl-namedisplay-options(1)。

-ext extensions

打印指定扩展（如 subjectAltName, keyUsage）。支持逗号分隔的扩展列表。

-purpose

测试证书扩展并输出其用途（如 SSL 客户端、服务器等）。

-pubkey

打印证书的公钥（PEM 格式）。

-modulus

打印公钥的模数（适用于 RSA 密钥）。

req

openssl req 是 OpenSSL 3.5 中的一个命令，主要用于创建和处理 PKCS#10 格式的证书请求（CSR），也可以生成自签名证书，例如用作根 CA。

openssl req 命令用于：

- **生成证书请求（CSR）**：创建 PKCS#10 格式的 CSR，包含公钥和主题信息，用于向证书颁发机构（CA）申请证书。
- **生成自签名证书**：通过 -x509 选项生成自签名证书，常用于测试或作为根 CA。
- **验证和查看 CSR**：检查 CSR 的内容或验证其自签名。

通用选项

-help

显示命令的帮助信息，列出所有可用选项。

-verbose

打印操作的详细信息，适合调试。

-quiet

减少操作的输出信息，适合脚本或批量处理。

-batch

启用非交互模式，直接使用配置文件或命令行参数，不提示用户输入。

输入/输出选项

-in filename

指定输入文件（CSR 或证书），默认从标准输入读取。如果使用 -x509 或 -CA，此选项非必需。

-inform DER|PEM

输入文件格式，默认为 PEM。

-out filename

指定输出文件（CSR 或证书），默认输出到标准输出。

-outform DER|PEM

输出文件格式，默认为 PEM。

-passin arg

输入私钥或证书的密码来源，详见 openssl-passphrase-options(1)。

-passout arg

输出文件的密码来源，详见 openssl-passphrase-options(1)。

证书请求生成选项

-new

生成新的 CSR 证书请求，提示用户输入主题字段（由配置文件或命令行指定）。

在 CSR（证书签名请求）中，可指定的字段分为**基础字段**和 **x509v3 扩展字段**两类。

基础字段包括：version、serial number、Signature Algorithm、Issuer、Validity（有效时间）、Subject、Subject Public Key Info（公钥信息）。

CSR 证书请求中仅能自定义的字段是 **Subject** 和 **Subject Public Key Info**，其余字段通常由 CA（证书颁发机构）在签发时填充。

x509v3 扩展字段更为丰富，对于 **TLS 证书**，关键扩展包括：

- **subjectAltName=DNS:example.com**（指定可用的域名）
- **extendedKeyUsage=clientAuth**（定义证书用途，如客户端认证）

在 CSR（证书签名请求）中，虽然可以指定**基础字段**和**扩展字段**，但最终生效与否完全取决于 **CA（证书颁发机构）的签发策略**。

对于 **DV TLS 数字证书**（域名验证型证书），即使 CSR 中指定了 **O（Organization，组织）** 等字段，CA 通常**仅保留 CN（Common Name，证书所有者）** 字段，并且 **CN 必须是一个有效的域名**。如果 **CN 字段不在 subjectAltName（SAN）扩展中**，CA 通常会**忽略 CN**，转而从 **subjectAltName** 中选择一个域名作为证书主体。

-subj arg

直接指定 CSR 的主题（DN），格式为 /type0=value0/type1=value1/...（如 /C=CN/O=MyOrg/CN=example.com）。

注意：OpenSSL 在生成证书时，会严格按照命令行中指定的字段顺序处理，不会自动调整或重排序。

-newkey arg

生成新私钥并用于 CSR 或证书。格式包括：

- **rsa:nbits**：生成指定位数的 RSA 密钥（默认 2048 位）。
- **algname[:file]**：使用指定算法（如 dsa、ec、gost2001）生成密钥，可指定参数文件。
- **param:file**：从文件中读取算法参数生成密钥。

-keyout filename

指定新生成私钥的输出文件。如果未提供 -key，默认使用配置文件中的 default_keyfile。

-noenc

生成私钥时不加密（替代已废弃的 -nodes 选项）。

-cipher name

指定私钥加密的算法，默认使用 AES-256-CBC（OpenSSL 3.5 新增，默认从 3DES 改为 AES-256）。

-pkeyopt opt:value

设置公钥算法选项，例如设置 EC 曲线的参数，详见 openssl-genpkey(1)。

-key filename|uri

指定现有私钥文件。私钥用于签名（支持 PEM、DER、P12 格式）。

如果是生成 CSR 证书请求：

- 该私钥的**公钥**会被提取并包含在 CSR 中，作为未来证书的公钥。
- 该私钥的**私钥**则用于对 CSR 的内容进行签名，证明请求者拥有该公钥所对应的私钥。

如果是生成新的数字证书（且没有指定 -CA 和 -CAkey）：

- 该私钥的**公钥**会被提取并作为新证书的公钥。
- 该私钥的**私钥**则用于对新证书进行自签名。

如果是生成新的数字证书（且指定了 -CA 和 -CAkey）：

- 该私钥的**公钥**会被提取并作为新证书的公钥。
- 该私钥的**私钥**不会用于对新证书进行签名，新证书的签名将由 `-CAkey` 指定的 CA 私钥来完成。

自签名证书选项

-x509

生成自签名证书而非 CSR，默认使用 X.509 v3（除非指定 -x509v1）。

-x509v1

生成 X.509 v1 证书（不含扩展）。

-CA filename|uri

指定 CA 证书，用于签名新证书，模拟“微型 CA”模式。

-CAkey filename|uri

指定 CA 的私钥，与 -CA 配合使用。

-days n

设置证书有效期（天数），默认 30 天。如果指定 -not_after，则优先使用。

-not_before date

设置证书的起始时间，格式为 YYMMDDHHMMSSZ 或 YYYYMMDDHHMMSSZ，支持 today。

-not_after date

设置证书的到期时间，格式同上，支持 today。

-set_serial n

设置自签名证书的序列号（十进制或以 0x 开头的十六进制）。

扩展和配置选项

-config filename

指定配置文件，覆盖默认配置文件（通常为 openssl.cnf）。

-section name

指定配置文件中的特定节（默认 req 节，OpenSSL 3.0 新增）。

-extensions section

指定证书扩展的配置文件节（用于 -x509）。

-reqexts section

指定 CSR 扩展的配置文件节（等同于 -extensions，OpenSSL 3.2 起为别名）。

-addext ext

添加特定扩展到 CSR 或证书，格式为 key=value（如 subjectAltName=DNS:example.com），可多次使用。

指定多个域名：

```
-addext subjectAltName=DNS:*.xiaoshae.cn,DNS:xiaoshae.cn
```

-copy_extensions arg

控制是否将 CSR 中的扩展复制到证书（none：忽略，copy 或 copyall：复制）。**无其他选项。**

-precert

生成带“毒性扩展”的预证书（用于证书透明度日志，RFC6962），需配合 -new。

签名和验证选项

-digest

指定签名使用的摘要算法（默认由配置文件指定，某些算法如 Ed25519 忽略此选项）。

-sigopt nm:v

传递签名算法的选项（算法相关）。

-vfyopt nm:v

传递验证算法的选项（算法相关）。

-verify

验证 CSR 的自签名，如果失败，程序立即退出（OpenSSL 3.3 起返回退出码 1）。

输出和格式化选项

-text

以文本形式打印 CSR 或证书内容。

-subject

打印 CSR 或证书（若使用 -x509）的主题。

-pubkey

打印公钥。

-modulus

打印公钥的模数（仅 RSA）。

-noout

不输出编码后的 CSR 或证书。

-nameopt option

自定义主题或颁发者名称的显示格式，详见 openssl-namedisplay-options(1)。

-reqopt option

自定义 -text 输出的格式，详见 openssl-x509(1)。

-utf8

将字段值解释为 UTF-8 字符串（默认 ASCII）。

-newhdr

在 PEM 输出中添加 NEW 标记（某些 CA 或软件要求）。

ca

openssl ca 命令是 OpenSSL 工具集中的一个功能，用于**模拟证书颁发机构（CA）的操作。**它可以用来**签署证书请求（CSR）、生成证书吊销列表（CRL），并维护一个记录已颁发证书及其状态的文本数据库。**尽管它是 OpenSSL 的一个示例性工具，但其功能足以支持基本的 CA 操作。

openssl ca 是一个用于管理 CA 的命令行工具，主要功能包括：

- **签署证书请求（CSR）**：根据提供的 CSR 文件生成证书。
- **生成证书吊销列表（CRL）**：创建或更新 CRL，记录被吊销的证书。
- **维护证书数据库**：记录已颁发的证书及其状态（如有效、吊销等）。
- **处理 Netscape SPKAC**：签署 Netscape 格式的公钥和挑战请求。
- **支持多种格式**：支持 PEM、DER 等格式的输入和输出。

命令语法

```
openssl ca [选项] [证书请求文件...]
```

选项：控制 CA 操作的行为，如指定配置文件、输入输出文件、签名算法等。

证书请求文件：可以指定单个 CSR 文件（通过 -in 选项）或多个 CSR 文件（通过 -infiles 或在命令末尾列出）。

指定多个 CSR 文件命令示例

```
openssl ca ... -infiles csr1.pem csr2.pem csr3.pem
```

通用选项

-help

显示命令帮助信息。

-verbose

输出详细的操作信息，便于调试。

-config filename

指定配置文件路径，默认值参考 openssl(1) 的“COMMAND SUMMARY”部分。

-name section 或 -section section

指定配置文件中使用的 CA 部分，覆盖默认的 default_ca 设置。

-batch

启用批处理模式，自动签署证书而不提示用户确认。

输入输出选项

-in filename

指定包含单个证书请求（CSR）的输入文件。

-inform DER|PEM

指定输入证书请求的格式，默认未指定（见 openssl-format-options(1)）。

-out filename

指定输出证书的文件，默认输出到标准输出（以 PEM 格式，除非使用 -spkac 则为 DER 格式）。

-outdir directory

指定输出证书的目录，证书文件名以十六进制序列号加 .pem 后缀命名。

-infiles

表示后续参数为多个证书请求文件的列表，需放在选项最后。**与 -in 参数冲突。**

-spkac filename

处理 Netscape 格式的 SPKAC 文件（包含公钥和挑战）。

-ss_cert filename

处理自签名证书的签署请求。

-notext

不输出证书的文本形式，仅输出编码格式（如 PEM 或 DER）。

证书相关选项

-cert filename

指定 CA 证书文件，必须与 -keyfile 匹配。

-certform DER|PEM|P12

指定 CA 证书的格式，默认未指定。

-keyfile filename|uri

指定 CA 私钥文件或 URI，必须与 -cert 匹配。

-keyform DER|PEM|P12|ENGINE

指定私钥格式，默认未指定。

-key password

指定私钥加密密码，需谨慎使用（命令行参数可能在某些系统上可见，建议使用 -passin）。

-passin arg

指定私钥或证书的密码来源（见 openssl-passphrase-options(1)）。

-selfsign

使用 CSR 中的私钥进行自签名，忽略其他私钥（与 -spkac、-ss_cert 或 -gencrl 冲突）。

-startdate date

显式设置证书的生效时间，格式为 YYMMDDHHMMSSZ（UTCTime）或 YYYYMMDDHHMMSSZ（GeneralizedTime）。

-enddate date

显式设置证书的到期时间，格式同上。

-days arg

指定证书有效期（天数）。

-md alg

指定签名使用的消息摘要算法（如 sha256），支持 openssl-dgst(1) 中列出的算法。对于不支持摘要的算法（如 Ed25519、Ed448），此选项被忽略。

-policy arg

指定 CA 策略，定义证书 DN 字段的匹配规则（见“策略格式”部分）。

-extensions section

指定配置文件中定义的证书扩展部分，默认使用 x509_extensions（生成 V3 证书，否则为 V1 证书）。

-extfile file

指定额外的配置文件以读取证书扩展。

-subj arg

覆盖请求中的主题名称，格式为 /type0=value0/type1=value1/...（支持多值 RDN 和转义字符）。

-utf8

将字段值解释为 UTF-8 字符串，默认按 ASCII 解释。

-preserveDN

保留请求中的 DN 顺序，默认按策略部分定义的顺序。

-noemailDN

从证书主题中移除 EMAIL 字段，仅将其放入扩展（如 subjectAltName）。

-msie_hack

为兼容旧版 IE 证书注册控件，启用特殊处理（已废弃，不推荐使用）。

-sigopt nm:v

传递签名算法的特定选项（如 SM2 的 distid）。

-vfyopt nm:v

传递验证签名算法的特定选项（如验证 CSR 自签名时的选项）。

-create_serial

如果无法从序列号文件读取序列号，则生成新的随机序列号。

-rand_serial

使用大随机数作为序列号，优先于序列号文件。

配置文件

openssl ca 的行为很大程度上依赖于配置文件（通常为 openssl.cnf）。配置文件中与 CA 相关的部分由 -name 或 default_ca 指定。以下是配置文件中的关键选项：

好的，这是对您提供的 OpenSSL 配置文件的完整中文翻译。翻译力求遵循"信、达、雅"的原则，既忠于原文的技术细节，又符合中文技术文档的语言习惯，同时保持了原有的格式和注释。

```
##################################################################
```

```
[ ca ]
default_ca  = CA_default        # 默认的 ca 节

####################################################################
[ CA_default ]

dir             = ./demoCA          # 所有文件的存放目录
certs           = $dir/certs        # 已签发证书的存放目录
crl_dir         = $dir/crl          # 已签发 CRL 的存放目录
database        = $dir/index.txt    # 数据库索引文件。
#unique_subject = no                # 设置为 'no' 以允许创建
                                    # 多个具有相同主题（subject）的证书。
new_certs_dir   = $dir/newcerts     # 新证书的默认存放位置。

certificate     = $dir/cacert.pem   # CA 证书
serial          = $dir/serial       # 当前的序列号文件
crlnumber       = $dir/crlnumber    # 当前的 CRL 编号文件
                                    # 若要生成 V1 版本的 CRL，必须注释掉此行
crl             = $dir/crl.pem      # 当前的 CRL 文件
private_key     = $dir/private/cakey.pem    # CA 的私钥

x509_extensions = usr_cert          # 要添加到证书中的扩展项

# 为了使用“传统”的（且极易出错的）格式，请注释掉以下两行。
name_opt        = ca_default        # 主题名称（Subject Name）选项
cert_opt        = ca_default        # 证书字段选项

# 扩展复制选项：请谨慎使用。
# copy_extensions = copy

# 要添加到 CRL 的扩展。注意：Netscape Communicator 无法处理 V2 版本的 CRL，
# 因此默认情况下此项被注释掉，以生成 V1 版本的 CRL。
# 要生成 V1 版本的 CRL，crlnumber 项也必须被注释掉。
# crl_extensions    = crl_ext

default_days        = 365           # 证书有效期（天）
default_crl_days    = 30            # 下一次 CRL 更新前的天数
default_md          = default       # 使用公钥的默认消息摘要算法
preserve            = no            # 是否保留传入的 DN（Distinguished Name）顺序

# 几种不同的方式来规定请求的外观应如何相似
# 对于 CA 类型，所列出的属性必须匹配，
# 而 optional（可选）和 supplied（提供）字段则名副其实 :-)
policy      = policy_match

# 用于 CA 的策略
[ policy_match ]
countryName             = match     # 必须匹配
stateOrProvinceName     = match     # 必须匹配
organizationName        = match     # 必须匹配
organizationalUnitName  = optional  # 可选
commonName              = supplied  # 必须提供
emailAddress            = optional  # 可选
```

```
55
56  # "anything"（任何内容）策略
57  # 在当前版本中，您必须列出所有可接受的“对象”类型。
58  [ policy_anything ]
59  countryName             = optional
60  stateOrProvinceName     = optional
61  localityName            = optional
62  organizationName        = optional
63  organizationalUnitName  = optional
64  commonName              = supplied
65  emailAddress            = optional
66
67  ####################################################################
68  [ req ]
69  default_bits        = 2048
70  default_keyfile     = privkey.pem
71  distinguished_name  = req_distinguished_name
72  attributes          = req_attributes
73  x509_extensions     = v3_ca # 添加到自签名证书的扩展
74
75  # 私钥的密码，如果未提供，则会提示输入
76  # input_password = secret
77  # output_password = secret
78
79  # 此项为允许的字符串类型设置一个掩码。有多种选项：
80  # default: PrintableString, T61String, BMPString.
81  # pkix   : PrintableString, BMPString (PKIX 在 2004 年前的建议)
82  # utf8only: 仅使用 UTF8Strings (PKIX 在 2004 年后的建议).
83  # nombstr : PrintableString, T61String (不含 BMPStrings 或 UTF8Strings).
84  # MASK:XXXX 一个字面的掩码值.
85  # 警告：旧版本的 Netscape 会在遇到 BMPStrings 或 UTF8Strings 时崩溃。
86  string_mask = utf8only
87
88  # req_extensions = v3_req # 添加到证书请求的扩展
89
90  [ req_distinguished_name ]
91  countryName                 = 国家名称 (2 字母代码)
92  countryName_default         = AU
93  countryName_min             = 2
94  countryName_max             = 2
95
96  stateOrProvinceName         = 州或省份名称 (全名)
97  stateOrProvinceName_default = Some-State
98
99  localityName                = 地区名称 (例如：城市)
100
101 O.organizationName          = 组织名称 (例如：公司)
102 O.organizationName_default  = Internet Widgits Pty Ltd
103
104 # 我们也可以这样做，但通常不需要 :-)
105 #1.organizationName     = 第二个组织名称 (例如：公司)
106 #1.organizationName_default = World Wide Web Pty Ltd
107
```

```
organizationalUnitName          = 组织单位名称 (例如：部门)
#organizationalUnitName_default =

commonName                      = 通用名称 (例如：服务器 FQDN 或您的姓名)
commonName_max                  = 64

emailAddress                    = 电子邮件地址
emailAddress_max                = 64

# SET-ex3                = SET 扩展编号 3

[ req_attributes ]
challengePassword               = 质询密码
challengePassword_min           = 4
challengePassword_max           = 20

unstructuredName                = 一个可选的公司名称

[ usr_cert ]

# 当 'ca' 签署一个请求时，会添加这些扩展。

# 此项设置有悖于 PKIX 指南，但某些 CA 如此操作，且部分软件需要此设置
# 以避免将最终用户证书误解为 CA 证书。

basicConstraints=CA:FALSE

# 这对于客户端证书的密钥用法是典型的。
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# PKIX 的建议，包含在所有证书中均无害。
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

# 以下内容用于 subjectAltName 和 issuerAltname。
# 导入电子邮件地址。
# subjectAltName=email:copy
# 另一种生成符合 PKIX 标准、未被弃用的证书的方式。
# subjectAltName=email:move

# 复制主题（subject）的详细信息
# issuerAltName=issuer:copy

# 此项为 TSA 证书所必需。
# extendedKeyUsage = critical,timeStamping

[ v3_req ]

# 添加到证书请求的扩展

basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

```

```
[ v3_ca ]


# 一个典型 CA 的扩展


# PKIX 建议。

subjectKeyIdentifier=hash

authorityKeyIdentifier=keyid:always,issuer

basicConstraints = critical,CA:true

# 密钥用法：这对于 CA 证书是典型的。但由于它会阻止
# 该证书被用作测试性的自签名证书，因此默认情况下最好将其省略。
# keyUsage = cRLSign, keyCertSign

# 在主题备用名称中包含电子邮件地址：另一条 PKIX 建议
# subjectAltName=email:copy
# 复制颁发者（issuer）的详细信息
# issuerAltName=issuer:copy

# 一个扩展的 DER 十六进制编码：注意，仅限专家使用！
# obj=DER:02:03
# 其中 'obj' 是一个标准的或新增的对象
# 你甚至可以覆盖一个受支持的扩展：
# basicConstraints= critical, DER:30:03:01:01:FF

[ crl_ext ]

# CRL 扩展。
# 在 CRL 中，只有 issuerAltName 和 authorityKeyIdentifier 有意义。

# issuerAltName=issuer:copy
authorityKeyIdentifier=keyid:always

[ proxy_cert_ext ]
# 创建代理证书时应添加这些扩展。

# 此项设置有悖于 PKIX 指南，但某些 CA 如此操作，且部分软件需要此设置
# 以避免将最终用户证书误解为 CA 证书。

basicConstraints=CA:FALSE

# 这对于客户端证书的密钥用法是典型的。
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# PKIX 的建议，包含在所有证书中均无害。
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

# 以下内容用于 subjectAltName 和 issuerAltname。
```

```
# 导入电子邮件地址。
# subjectAltName=email:copy
# 另一种生成符合 PKIX 标准、未被弃用的证书的方式。
# subjectAltName=email:move

# 复制主题（subject）的详细信息
# issuerAltName=issuer:copy

# 要使其成为代理证书，此项确实需要设置。
proxyCertInfo=critical,language:id-ppl-anyLanguage,pathlen:3,policy:foo
```

示例

RSA 证书颁发机构

操作系统版本

```
NAME="openEuler"
VERSION="22.03 (LTS-SP4)"
ID="openEuler"
VERSION_ID="22.03"
PRETTY_NAME="openEuler 22.03 (LTS-SP4)"
ANSI_COLOR="0;31"
```

```
Linux Server1 5.10.0-216.0.0.115.oe2203sp4.x86_64 #1 SMP Thu Jun 27 15:13:44 CST 2024 x86_64 x86_64 x86_64 GNU/Linux
```

OpenSSL 版本

```
OpenSSL 1.1.1wa  16 Nov 2023
```

openssl ca 命令是 OpenSSL 工具集中的一个功能，用于**模拟证书颁发机构（CA）的操作。**它可以用来**签署证书请求（CSR）、生成证书吊销列表（CRL），并维护一个记录已颁发证书及其状态的文本数据库。**尽管它是 OpenSSL 的一个示例性工具，但其功能足以支持基本的 CA 操作。

首先在 `/etc/pki/tls` 目录下创建 CA 环境所需的目录结构：

```
# 进入 TLS 证书目录
cd /etc/pki/tls

# 创建 demoCA 主目录
mkdir demoCA

# 进入 demoCA 目录并创建子目录
cd demoCA
mkdir certs crl newcerts private
```

- `certs/`: 存放已颁发的证书
- `crl/`: 存放证书吊销列表
- `newcerts/`: 存放新颁发的证书副本
- `private/`: 存放 CA 的私钥文件

初始化 CA 数据库文件

```
touch index.txt serial
echo "1000" > serial
```

- `index.txt`: 证书数据库文件，记录所有颁发的证书信息
- `serial`: 证书序列号文件，每次颁发证书后会自动递增

编辑 /etc/pki/tls/openssl.cnf 文件，修改参数。

将第 45 行的 `dir = ./demoCA` 修改为 `dir = /etc/pki/tls/demoCA`

```
     38 ####################################################################
     39 [ ca ]
     40 default_ca      = CA_default            # The default ca section
     41
     42 ####################################################################
     43 [ CA_default ]
     44
-    45 dir             = ./demoCA              # Where everything is kept
+    45 dir             = /etc/pki/tls/demoCA   # Where everything is kept
     46 certs           = $dir/certs            # Where the issued certs are kept
     47 crl_dir         = $dir/crl              # Where the issued crl are kept
     48 database        = $dir/index.txt        # database index file.
     49 unique_subject  = no                     # Set to 'no' to allow creation of
     50                                         # several certs with same subject.
     51 new_certs_dir   = $dir/newcerts         # default place for new certs.
     52
     53 certificate     = $dir/cacert.pem       # The CA certificate
     54 serial          = $dir/serial           # The current serial number
     55 crlnumber       = $dir/crlnumber        # the current crl number
```

```
     56                                          # must be commented out to leave a V1
CRL
     57 crl             = $dir/crl.pem          # The current CRL
     58 private_key     = $dir/private/cakey.pem# The private key
```

取消第 68 行的 `copy_extensions = copy` 的注释

```
     67 # Extension copying option: use with caution.
-    68 # copy_extensions = copy
+    68 copy_extensions = copy     # 取消注释
```

这个参数指示 OpenSSL 在为证书请求（CSR）签名时，将 CSR 中包含的扩展信息（如 Subject Alternative Name, SAN）复制到最终颁发的证书中。这对于生成包含 SAN 扩展的服务器证书至关重要，因为 SAN 允许一个证书保护多个域名（例如 `*.lab.org` 和 `lab.org`）。

修改第 83 行的 `policy = policy_match` 为 `policy = policy_anything`

```
     80 # A few difference way of specifying how similar the request should look
     81 # For type CA, the listed attributes must be the same, and the optional
     82 # and supplied fields are just that :-)
-    83 policy          = policy_match
+    83 policy          = policy_anything
```

- `policy = policy_match`：默认策略，要求证书请求中的国家（C）、省份（ST）、地区（L）和组织（O）字段必须与 CA 证书中的对应字段匹配。
- `policy = policy_anything`：修改后的策略。这意味着 CA 在为证书请求签名时，不会强制要求证书请求中的字段（如国家、省份、组织等）必须与 CA 证书中的字段严格匹配。

生成 CA 私钥

```
openssl genpkey -algorithm RSA -out private/cakey.pem -pkeyopt rsa_keygen_bits:8192
```

- `-algorithm RSA`：指定生成 RSA 算法的私钥。
- `-out private/cakey.pem`：将私钥保存到 `demoCA/private/` 目录下，文件名为 `cakey.pem`。
- `-pkeyopt rsa_keygen_bits:8192`：指定 RSA 密钥的长度为 8192 位。

生成自签名 CA 证书，它包含了 CA 的公钥和身份信息。作为根 CA，它通常是自签名的。

```
openssl req -x509 -subj "/CN=ca-rsa.lab.org" -key private/cakey.pem -out cacert.pem -
days 3650
```

- `-x509`：指定生成一个自签名证书，而不是 CSR 证书请求。

- `-subj "/CN=ca-rsa.lab.org"`：指定证书的主题（Subject）。`CN` (Common Name) 通常用于标识证书的用途或所有者。这里设置为 `ca-rsa.lab.org`.
- `-key private/cakey.pem`：指定用于签名此证书的私钥文件。这里使用的是我们刚刚生成的 CA 私钥。
- `-out cacert.pem`： `cacert.pem` 将保存在 `demoCA/` 目录下。
- `-days 3650`：指定证书的有效期为 3650 天（约 10 年）。

将为 `*.lab.org` 域名生成一个 RSA 私钥。这个私钥将用于后续生成 CSR 证书请求。

```
openssl genpkey -algorithm RSA -out lab.org.key -pkeyopt rsa_keygen_bits:4096
```

- `-algorithm RSA`：指定生成 RSA 算法的私钥。
- `-out lab.org.key`：指定私钥的输出路径和文件名。`lab.org.key` 将保存在当前目录（即 `/etc/pki/tls/demoCA`）下。
- `-pkeyopt rsa_keygen_bits:4096`：指定 RSA 密钥的长度为 4096 位。

生成服务器证书请求，证书请求（CSR）包含了服务器的身份信息和公钥，用于向 CA 申请颁发证书。

```
openssl req -new -subj "/CN=*.lab.org/C=CN/ST=shanghai/L=shanghai/O=system/OU=system" -key lab.org.key -addext subjectAltName="DNS:*.lab.org,DNS:lab.org" -out lab.org.csr
```

- `-new`：表示生成一个新的证书请求。`-subj "/CN=*.lab.org/C=CN/ST=shanghai/L=shanghai/O=system/OU=system"`
  - `CN=*.lab.org`：通用名称，通常是服务器的主机名或域名。这里的 `*.lab.org` 表示这是一个通配符证书请求。
  - `C=CN`：国家（Country）为中国。
  - `ST=shanghai`：省份（State or Province）为上海。
  - `L=shanghai`：地区（Locality）为上海。
  - `O=system`：组织（Organization）为 system。
  - `OU=system`：组织单位（Organizational Unit）为 system。
- `-key lab.org.key`：指定用于生成此证书请求的私钥。这里使用的是我们刚刚生成的 `lab.org.key`。
- `-addext subjectAltName="DNS:*.lab.org,DNS:lab.org"` 添加主题备用名称（Subject Alternative Name, SAN）扩展。这是非常重要的一步，它允许一个证书保护多个域名。
  - `DNS:*.lab.org`：表示证书将适用于 `lab.org` 域下的所有子域名（例如 `www.lab.org`, `mail.lab.org`）。
  - `DNS:lab.org`：表示证书也将适用于 `lab.org` 根域名本身。
- `-out lab.org.csr`：指定证书请求的输出路径和文件名。`lab.org.csr` 将保存在当前目录（即 `/etc/pki/tls/demoCA`）下。

现在，我们已经有了 CA 环境、CA 证书和私钥，以及证书请求（CSR）。最后一步是使用我们的 CA 对服务器的 CSR 进行签名，从而颁发一个正式的服务器证书。

```
openssl ca -in lab.org.csr -batch -days 1825
```

```
demoCA/
├── cacert.pem
├── certs
├── crl
├── index.txt
├── index.txt.attr
├── index.txt.attr.old
├── index.txt.old
├── lab.org.csr
├── lab.org.key
├── newcerts
│   └── 1000.pem
├── private
│   └── cakey.pem
├── serial
└── serial.old

4 directories, 11 files
```

SM 证书颁发机构

首先在 `/etc/pki/tls` 目录下创建 CA 环境所需的目录结构：

```
# 进入 TLS 证书目录
cd /etc/pki/tls

# 创建 demoCA 主目录
mkdir demoCA

# 进入 demoCA 目录并创建子目录
cd demoCA
mkdir certs crl newcerts private
```

- `certs/`：存放已颁发的证书
- `crl/`：存放证书吊销列表
- `newcerts/`：存放新颁发的证书副本
- `private/`：存放 CA 的私钥文件

初始化 CA 数据库文件

```
touch index.txt serial
echo "1000" > serial
```

- `index.txt`：证书数据库文件，记录所有颁发的证书信息
- `serial`：证书序列号文件，每次颁发证书后会自动递增

编辑 /etc/pki/tls/openssl.cnf 文件，修改参数。

将第 45 行的 `dir = ./demoCA` 修改为 `dir = /etc/pki/tls/demoCA`

```
     38 ####################################################################
     39 [ ca ]
     40 default_ca      = CA_default            # The default ca section
     41
     42 ####################################################################
     43 [ CA_default ]
     44
-    45 dir             = ./demoCA              # Where everything is kept
+    45 dir             = /etc/pki/tls/demoCA   # Where everything is kept
     46 certs           = $dir/certs            # Where the issued certs are kept
     47 crl_dir         = $dir/crl              # Where the issued crl are kept
     48 database        = $dir/index.txt        # database index file.
     49 unique_subject  = no                     # Set to 'no' to allow creation of
     50                                         # several certs with same subject.
     51 new_certs_dir   = $dir/newcerts         # default place for new certs.
     52
     53 certificate     = $dir/cacert.pem       # The CA certificate
     54 serial          = $dir/serial           # The current serial number
     55 crlnumber       = $dir/crlnumber        # the current crl number
     56                                         # must be commented out to leave a V1
CRL
     57 crl             = $dir/crl.pem          # The current CRL
     58 private_key     = $dir/private/cakey.pem# The private key
```

将第 77 行的 `default_md = default` 修改为 `default_md = sm3`。

```
     75 default_days    = 365                  # how long to certify for
     76 default_crl_days= 30                   # how long before next CRL
-    77 default_md      = default              # use public key default MD
+    77 default_md      = sm3                  # use public key default MD
     78 preserve        = no                   # keep passed DN ordering
```

`default_md = sm3`：这个参数指定了 CA 默认使用的消息摘要算法（哈希算法）。`sm3` 是国密算法套件中的哈希算法，用于生成证书的指纹和签名。

取消第 68 行的 `copy_extensions = copy` 的注释

```
     67 # Extension copying option: use with caution.
-    68 # copy_extensions = copy
+    68 copy_extensions = copy     # 取消注释
```

这个参数指示 OpenSSL 在为证书请求（CSR）签名时，将 CSR 中包含的扩展信息（如 Subject Alternative Name, SAN）复制到最终颁发的证书中。这对于生成包含 SAN 扩展的服务器证书至关重要，因为 SAN 允许一个证书保护多个域名（例如 `*.lab.org` 和 `lab.org`）。

修改第 83 行的 `policy = policy_match` 为 `policy = policy_anything`

```
     80 # A few difference way of specifying how similar the request should look
     81 # For type CA, the listed attributes must be the same, and the optional
     82 # and supplied fields are just that :-)
-    83 policy          = policy_match
+    83 policy          = policy_anything
```

- `policy = policy_match`：默认策略，要求证书请求中的国家（C）、省份（ST）、地区（L）和组织（O）字段必须与 CA 证书中的对应字段匹配。
- `policy = policy_anything`：修改后的策略。这意味着 CA 在为证书请求签名时，不会强制要求证书请求中的字段（如国家、省份、组织等）必须与 CA 证书中的字段严格匹配。

CA 私钥是 CA 的核心，用于对所有颁发的证书进行签名。这里我们将使用国密 SM2 算法生成私钥。

```
openssl genpkey -algorithm EC -out private/cakey.pem -pkeyopt ec_paramgen_curve:sm2
```

- `-algorithm EC`：指定生成椭圆曲线（Elliptic Curve）算法的私钥。SM2 密钥是基于椭圆曲线的。
- `-out private/cakey.pem`：指定私钥的输出路径和文件名。`private/cakey.pem` 表示将私钥保存到 `demoCA/private/` 目录下，文件名为 `cakey.pem`。
- `-pkeyopt ec_paramgen_curve:sm2`：这是关键参数，它指定了椭圆曲线的名称为 `sm2`，即使用国密 SM2 曲线。

CA 证书是 CA 的公开部分，它包含了 CA 的公钥和身份信息。作为根 CA，它通常是自签名的，并使用 SM3 哈希算法进行签名。

```
openssl req -x509 -subj "/CN=ca-sm.lab.org" -days 3650 -out cacert.pem -key
private/cakey.pem
```

- `-x509`：指定生成一个自签名证书，而不是证书请求。
- `-subj "/CN=ca-sm.lab.org"`：指定证书的主题（Subject）。`CN` (Common Name) 通常用于标识证书的用途或所有者。这里我们将其设置为 `ca-sm.lab.org`，表示这是一个用于 `lab.org` 域的 SM 算法 CA 证书。
- `-days 3650`：指定证书的有效期为 3650 天（约 10 年）。CA 证书的有效期通常设置得较长。
- `-out cacert.pem`：指定自签名证书的输出路径和文件名。`cacert.pem` 将保存在 `demoCA/` 目录下。

- `-key private/cakey.pem`：指定用于签名此证书的私钥文件。这里使用的是我们刚刚生成的 SM2 CA 私钥。由于 `openssl.cnf` 中已设置 `default_md = sm3`，所以此证书将使用 SM3 算法进行签名。

接下来，我们将为 `*.lab.org` 域名生成一个服务器私钥，同样使用 SM2 算法。

```
openssl genpkey -algorithm EC -out lab.org.key -pkeyopt ec_paramgen_curve:sm2
```

- `-algorithm EC`：指定生成椭圆曲线算法的私钥。
- `-out lab.org.key`：指定私钥的输出路径和文件名。`lab.org.key` 将保存在当前目录（即 `/etc/pki/tls/demoCA`）下。
- `-pkeyopt ec_paramgen_curve:sm2`：指定椭圆曲线的名称为 `sm2`，确保生成的私钥是 SM2 私钥。

证书请求（CSR）包含了身份信息和公钥，用于向 CA 申请颁发证书。

```
openssl req -new -subj "/CN=lab.org/C=CN/ST=shanghai/L=shanghai/O=system/OU=system" -
addext subjectAltName="DNS:*.lab.org,DNS:lab.org" -out lab.org.csr -key lab.org.key
```

- `-new`：表示生成一个新的证书请求。`-subj "/CN=*.lab.org/C=CN/ST=shanghai/L=shanghai/O=system/OU=system"`
  - `CN=*.lab.org`：通用名称，通常是服务器的主机名或域名。这里的 `*.lab.org` 表示这是一个通配符证书请求。
  - `C=CN`：国家（Country）为中国。
  - `ST=shanghai`：省份（State or Province）为上海。
  - `L=shanghai`：地区（Locality）为上海。
  - `O=system`：组织（Organization）为 system。
  - `OU=system`：组织单位（Organizational Unit）为 system。
- `-key lab.org.key`：指定用于生成此证书请求的私钥。这里使用的是我们刚刚生成的 `lab.org.key`。
- `-addext subjectAltName="DNS:*.lab.org,DNS:lab.org"` 添加主题备用名称（Subject Alternative Name, SAN）扩展。这是非常重要的一步，它允许一个证书保护多个域名。
  - `DNS:*.lab.org`：表示证书将适用于 `lab.org` 域下的所有子域名（例如 `www.lab.org`，`mail.lab.org`）。
  - `DNS:lab.org`：表示证书也将适用于 `lab.org` 根域名本身。
- `-out lab.org.csr`：指定证书请求的输出路径和文件名。`lab.org.csr` 将保存在当前目录（即 `/etc/pki/tls/demoCA`）下。

最后一步是使用我们的 SM CA 对服务器的 CSR 进行签名，从而颁发一个正式的服务器证书。CA 将使用 SM3 算法对证书进行签名。

```
openssl ca -in lab.org.csr -batch -days 1825
```

- `-in lab.org.csr`：指定输入的证书请求文件。
- `-batch`：以非交互模式运行，跳过所有交互式提示。
- `-days 1825`：指定新颁发证书的有效期为 1825 天（约 5 年）。