又很久没更新文章了啊……虽然这次疫情把大家都留在了家里，不过我自己的时间反而更少了，倒是花了大把的时间在峡谷的边路游走。

本文聊聊 HTTPS 的一些东西，和大家扯扯 SSL 证书的整个工作流程。希望大家有一些基本的常识：

- https 使用了非对称加密和对称加密，为什么要使用对称和非对称加密？非对称加密的原理是什么？这种简单的问题默认读者已经了解了。
- 非对称加密涉及到一对公钥和私钥组合，它们是一一对应的关系，不存在一个私钥对应多个公钥这种情况。
- CA 是 Certification Authority 的缩写，它代表世界上那些权威的证书颁发机构。

CA 需要做什么

我们在申请一个 https 证书的时候，要在市场上选择一家 CA 来给你签发证书，那么 CA 的工作是什么呢?

CA 要验证这个域名真的是你的：通常就是通过 DNS 记录或者就是你在指定 URI 下放置一个特殊文件，让 CA 可以在外网环境下访问到它。

CA 是一个非常关键的角色，因为它签出来的任何证书都是被信任的，所以这要求每个 CA 都不能胡来。

> 试想一下，如果某个 CA 私自给某个黑客签发了 *.taobao.com 的证书，那么黑客就能利用这个证书实现中间人攻击了。
>
> 有没有发生过 CA 瞎搞的事情呢？有一个典型的案例就是赛门铁克，由于它签发了大量的不合规的证书，导致了 Chrome 封杀。国内的 CA 也发生过，那就是沃通。
>
> 商业上来说，我觉得 CA 还是比较赚钱的，一个域名一年的花费就是几百至几千人民币，所以这些 CA 一定会努力维护自己的信誉。

大家应该也知道，Let's Encrypt 机构提供了免费的证书，那有什么区别呢?

Let's Encrypt 它只验证了这个域名是你的，然后就可以给你免费签发证书，这个证书的有效期是 3 个月，到期要自己去更新。因为它只验证了你的域名，所以这类证书又称为 DV 证书（Domain Validation）。

而一些收费的 CA 可以给你签发 OV 证书（Organization Validation）或 EV 证书（Extended Validation），他们不仅会验证这个域名真的是你的，还会**人工验证**你的公司是否符合他们的各项签发标准，所以收费也比较贵。通常这些证书的有效期是 1 年。

对于浏览器来说，通常会根据你的证书是 DV 还是 OV，来呈现不同的样式，所以有一种花钱的证书更香的感觉。

但是从技术上来说，它们都是提供一样的保护级别的，在最新的 Chrome 上，它没有区别对待，一律显示一个锁。

关键词：CA 要有底线、守信誉；证书分 DV、OV、EV。

证书组成

在介绍证书内容之前，我们先看看怎么向 CA 申请一个证书，这有助于你了解整个系统的运作过程。

证书申请

首先我们要生成一个 CSR，它的全称是 Certificate Signing Request，这个文件包含了你要申请的证书的各种信息，这和在某个 CA 的后台填写一个申请表单是一个意思，只是这样可以规范所有 CA 遵守一致的规则。

这里我们需要使用一个叫 **openssl** 的软件，执行下面的命令：

```
openssl req -new -newkey rsa:2048 -nodes -keyout www.javadoop.com.key -out
www.javadoop.com.csr
```

进入交互界面后，需要你填写国家、城市、公司名字、部门名字、申请的域名、邮箱地址。有些 CA 支持中文，大部分不支持，这里建议都是用英文字符。

然后我们就会得到两个**文本文件**：

```
www.javadoop.com.csr
www.javadoop.com.key
```

其中一个是 CSR 文件，用来发给 CA 申请证书的，另一个是私钥。私钥需要好好保存，等证书申请完成以后要用。

从域名的角度，证书分为单域名、通配符域名、多域名证书。这里我以单域名 `www.javadoop.com` 为例子，通配符和多域名也很好理解，在填写域名的时候按照格式填就可以了。

我们来看一下生成出来的 `www.javadoop.com.csr` 文件，它的内容是这样的：

```
-----BEGIN CERTIFICATE REQUEST-----
MIIC2DCCAcACAQAwgZIxCzAJBgNVBAYTAkNOMREwDwYDVQQIDAhTaGFuZ0hhaTER
MA8GA1UEBwwIU2hhbmdIYWkxFTATBgNVBAoMDEphdmFkb29wIEluYzELMAkGA1UE
CwwCSVQxFzAVBgNVBAMMDiouamF2YWRvb3AuY29tMSAwHgYJKoZIhvcNAQkBFhF0
ZXN0QGphdmFkb29wLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AMucZll8pn6z+IObP0re4mhC7VySQjC84qBRsr6rOCbcoRjDfAgAmxQbFabMgZ4E
l1uEU+MhIZU0UfW5jZv+A3HW3fxGt++9f/kH3jj0tGpinMoR+HJXjAYtCaYzLCTy
u/dEhXZv2TqUegRW1J5/2naAzq5VtaL2lN1fTOKpXxeYnduTma7kzZYAaOT5XO5x
jxnIhm9ogaRdGPrdsj88WcaHXjaAE+ERwK0l0/o9nDm5DnY3IGAQNz0Ft0TdQxjj
GZtBFjl46FPg8KS9YgiUInw///c6k459XvMTHGmSO1eXe/7yA/uUt4SLNIGsfO8K
5GV7P0qxc2beDiHWht3lDDcCAwEAAaAAMA0GCSqGSIb3DQEBCwUAA4IBAQAANyqP
H84rTYt5MRe/vKR4+/ZDVZa867vCt9zlGrm/3Qz/sXyFpTlTzhMCvuE0ZS1vzKTJ
9jzTimaIVF1pKWhgNsCfw4MuUjxXT6PRuHmCYWZLST6JxaXZdfzZXe+v5UlJW2UI
4/l9A1z7wc+tFlxc+GhQcvB5iaFLvmh6VMabMsHEZUWw5RA5enVhDHMvrlyRB2GG
oN1nvzptFxms5T6qVjEj4VXiQLgPPIx54owiBtyP55uKYH6Nf3JkfQjfFFKcTkZI
yhpVsgr300wAZfdWZI1z99442omCEALvFy/4DQixcnKkowOrHpVoAcyI7VlBfl/8
4cID+7D+EnlMU6xF
-----END CERTIFICATE REQUEST-----
```

我们找个网站在线 decode 一下这个内容，比如我们用这个网站 https://decoder.link/result 进行 decode，可以得到下面的内容：

```
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = CN, ST = ShangHai, L = ShangHai, O = Javadoop Inc, OU = IT, CN = www.javadoop.com, emailAddress = test@javadoop.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:c8:3f:ad:7c:df:a6:ad:6f:1e:82:14:de:cd:d9:
                    d1:64:1b:06:a1:9c:79:85:cb:66:34:af:87:c0:78:
                    86:10:64:04:a0:53:11:62:b4:47:49:ab:cc:93:e1:
                    b9:c9:46:cc:0c:ce:fa:a5:7c:54:97:4f:98:92:ed:
                    57:c2:23:08:48:fa:87:7c:d8:fe:08:6e:97:fd:65:
                    e8:3f:87:fa:19:38:20:6f:b3:35:04:7e:5b:2c:65:
                    3d:de:9d:57:c9:14:06:34:54:f3:72:90:77:f7:8d:
                    eb:2b:12:c5:31:ed:ce:44:0c:a3:3c:5f:63:72:16:
                    8f:1e:38:74:6e:80:19:fa:b1:83:76:62:b2:d3:dc:
                    0d:ec:69:32:1d:93:e1:6f:6d:a3:ba:74:62:7c:21:
                    a3:2a:c6:11:3d:6a:90:e4:28:c8:d5:95:a3:e1:92:
                    70:3c:3c:5f:13:54:f5:ac:6b:d1:e8:cd:75:6a:82:
                    3c:8d:81:35:1e:da:23:b7:9d:e7:c4:78:a9:54:89:
                    b5:89:5b:88:5a:f2:a2:42:b8:a8:b6:80:b9:dc:4e:
                    24:09:65:8f:81:d9:9b:f5:7d:86:41:bf:6e:8d:4e:
                    61:a4:b2:d5:cb:fa:cc:ce:27:f1:5f:03:36:3c:ad:
                    cd:f2:ea:65:5e:42:b8:8f:c8:0d:c2:53:16:39:4c:
                    c8:d5
                Exponent: 65537 (0x10001)
        Attributes:
            a0:00
    Signature Algorithm: sha256WithRSAEncryption
         8d:71:ac:d2:62:33:6c:c8:db:64:6f:a9:d1:11:38:c9:f5:02:
         e3:bd:cc:c1:92:6b:19:69:3f:2d:07:42:fe:e3:96:c7:dc:52:
         bd:f9:3b:1d:2f:7b:74:a1:f5:b7:a6:bc:e1:d6:cd:ad:6a:38:
         c1:6d:5e:d7:ce:1b:84:47:fd:8d:e9:9b:cb:0d:8e:d9:e1:8c:
         7b:24:d4:be:cf:b8:78:e3:77:1c:7c:81:8a:ad:b7:db:0d:a6:
         70:b4:7d:8d:8b:55:03:8f:f2:08:28:0a:0f:3e:f8:a4:b7:1a:
         f0:17:04:e3:08:15:35:c6:be:fc:eb:c8:29:c0:b3:73:52:65:
         42:93:f3:fd:d6:23:70:b2:4b:3d:3f:4a:9f:bb:32:9d:5c:74:
         d6:3d:de:3d:c5:f9:b1:6b:47:6f:fb:34:92:40:50:ca:47:4a:
         71:30:5d:6d:a6:69:67:1d:df:c9:bd:06:9b:7c:c7:77:0c:24:
         03:9d:5f:17:a8:de:c8:12:1a:b3:aa:b1:ba:f4:22:c1:e1:c4:
         2a:47:2f:10:45:ca:fd:60:b7:21:4c:00:af:1c:a0:d0:fb:dc:
         39:ee:c6:0d:06:da:e3:e6:d9:d1:f3:21:39:bb:dc:e4:a8:56:
         f7:41:9f:76:72:00:83:ca:c0:9e:05:a0:2a:93:02:5e:90:44:
         09:aa:15:3e
```

第一部分：Subject 中是我们填写的域名的基本信息，这里面，我们只需要关注 CN 字段就行：`www.javadoop.com`。CN 是 Common Name 的缩写。

第二部分：Subject Public Key Info 是公钥部分，这里指定了服务器使用的加密算法是 RSA，公钥长度是 2048 位。

第三部分：签名，使用了 sha256WithRSAEncryption 算法。也就是说首先将上面的所有信息进行 sha256 散列得到 hash 值，然后使用 RSA 算法**对 hash 值进行加密**，而加密的秘钥就是之前生成的私钥。

> 为什么这里要加第三部分的签名？其实就是为了防止你的 CSR 文件在发给 CA 的过程中被中间人拦截，然后修改了里面的信息再发给 CA。
>
> CA 的校验过程是：利用里面的公钥将签名进行解密得到里面的散列值，然后 CA 也会利用 CSR 里面的信息计算一遍散列值，如果两者相等，那么说明证书没有被中间人修改过，反之就是被修改过。

关键词：CSR 文件，包含了证书申请的信息，加密和散列算法，以及公钥。同时我们现在有了私钥。

证书组成

CA 收到我们的 CSR 文件以后，CA 会进行审核，前面说过了，审核这个域名是不是你的，如果需要，还有人工审核公司信息。审核通过后，它就会发给我们证书文件了，每家 CA 出来的文件名可能略有不同，但是表达的信息是一样的。

主要有以下几个文件：

- 域名证书：`www.javadoop.com.pem` 或叫 `cert.pem`
- 证书链：`fullchain.pem`

这些文件可能是 `.pem` 也可能是 `.crt` 后缀，但都是文本文件，可以直接打开查看它们的信息：

域名证书的文件内容通常是这样的：

```
-----BEGIN CERTIFICATE-----
MIIFTzCCBDegAwIBAgISAy4b8ie6L/ACCJt/V7x/OR0iMA0GCSqGSIb3DQEBCwUA
MEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQD
......
Ecbqh4AoB33mZhp9ptJb1N1RSlZREI0FlbX0kUd6VowKUPhH8Iex6jxQpJHwRkpq
YJaWKrUxGWuJurOcN7b3HXn6yw==
-----END CERTIFICATE-----
```

证书链文件通常是这样的：

```
-----BEGIN CERTIFICATE-----
MIIFTzCCBDegAwIBAgISAy4b8ie6L/ACCJt/V7x/OR0iMA0GCSqGSIb3DQEBCwUA
MEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQD
.......
Ecbqh4AoB33mZhp9ptJb1N1RSlZREI0FlbX0kUd6VowKUPhH8Iex6jxQpJHwRkpq
YJaWKrUxGWuJurOcN7b3HXn6yw==
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----
MIIEkjCCA3qgAwIBAgIQCgFBQgAAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/
MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT
.......
PfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iwBi8saCbGS5jN2p8M+X+Q7UNKEkROb3N6
KOqkqm57TH2H3eDJAkSnh6/DNFu0Qg==
-----END CERTIFICATE-----
```

这里贴了部分内容，是想要告诉大家：证书链文件的第一部分，和证书文件的内容是一模一样的。

> 如果 CA 还给你发了 `chain.pem` 文件，其实它的内容肯定就是证书链文件内容裁减掉第一部分的证书内容而已。

很多 CA 还会根据你的证书配置环境是 Nginx、IIS、Tomcat 等，给我们分为好几个不同的文件，但是说到底我们需要的其实就是一个证书链。

我上面贴的证书链就两部分内容，但是大家要知道，这个链是可以更长的。

关键词：CA 给我们颁发的证书，其实就是一个证书链文件

证书内容

接下来，我们来说说证书里面的内容到底是什么?

其实证书链上的每一个节点的内容都是类似的，我以 javadoop.com 的证书为例，将其进行 decode 后的结果如下：

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            03:76:e6:d1:15:1e:bb:72:04:ae:f9:f5:d6:ad:0b:f9:ab:81
    Signature Algorithm: sha256WithRSAEncryption
        // 证书颁发机构
        Issuer: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
        // 证书的有效期
        Validity
            Not Before: Apr 20 07:30:33 2020 GMT
            Not After : Jul 19 07:30:33 2020 GMT
        // 证书申请信息
        Subject: CN = javadoop.com
        // 公钥
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:d2:34:73:fb:83:e3:f0:82:f0:c8:ce:ab:50:e7:
                    d4:a4:14:e9:e2:a3:f1:61:bb:7e:db:b1:c9:e7:01:
                    49:c4:9c:89:e8:f0:25:bf:0d:1d:e0:e3:8d:df:7e:
                    16:ea:ab:0c:a9:5a:04:fb:ef:e5:09:4d:6e:21:5c:
```

```
                    d8:e8:39:38:46:69:91:49:04:6a:e1:83:6c:8e:5f:
                    16:7b:a4:d5:45:6e:16:a2:81:12:65:7e:a5:dc:60:
                    05:73:23:91:59:b3:1c:13:0b:25:15:2b:c0:27:80:
                    e6:ce:f8:d9:85:c0:0e:96:74:36:1a:78:fd:4c:2f:
                    82:66:16:00:ea:42:65:c3:25:ea:df:0d:5e:67:97:
                    65:85:0f:17:87:41:8a:3d:df:d2:93:b7:04:2d:15:
                    49:11:67:77:5b:49:29:6d:13:4b:c5:4f:9b:aa:e4:
                    1d:69:2b:8b:b3:c2:08:47:bf:f7:19:28:af:54:8b:
                    f3:6d:33:ad:4b:21:5e:42:07:08:15:03:06:b8:e5:
                    bc:93:fc:14:2f:d0:6d:e7:a0:34:86:8e:63:16:f3:
                    7f:e0:09:cd:1a:6b:ab:87:2d:b9:26:5f:17:5f:72:
                    10:05:5f:d1:d8:96:ab:0b:db:b5:25:15:99:49:8a:
                    0c:99:a7:e7:02:22:aa:92:29:50:7d:d3:44:25:7a:
                    18:97
                Exponent: 65537 (0x10001)
        // 这部分内容我们忽略
        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:

36:89:CF:2A:C8:0B:34:8F:9D:0D:F8:D8:DD:DF:4F:B6:89:19:77:DF
            X509v3 Authority Key Identifier:

keyid:A8:4A:6A:63:04:7D:DD:BA:E6:D1:39:B7:A6:45:65:EF:F3:A8:EC:A1

            Authority Information Access:
                OCSP - URI:http://ocsp.int-x3.letsencrypt.org
                CA Issuers - URI:http://cert.int-x3.letsencrypt.org/

            X509v3 Subject Alternative Name:
                DNS:javadoop.com, DNS:www.javadoop.com
            X509v3 Certificate Policies:
                Policy: 2.23.140.1.2.1
                Policy: 1.3.6.1.4.1.44947.1.1.1
                  CPS: http://cps.letsencrypt.org

            CT Precertificate SCTs:
                Signed Certificate Timestamp:
                    Version   : v1 (0x0)
                    Log ID    :
E7:12:F2:B0:37:7E:1A:62:FB:8E:C9:0C:61:84:F1:EA:

7B:37:CB:56:1D:11:26:5B:F3:E0:F3:4B:F2:41:54:6E
```

```
                    Timestamp : Apr 20 08:30:33.880 2020 GMT
                    Extensions: none
                    Signature : ecdsa-with-SHA256
30:46:02:21:00:90:A7:C6:DD:84:2C:F8:90:38:BC:C7:
BB:4E:A7:46:38:68:5D:F2:9D:40:57:9E:12:9D:F6:85:
42:B5:E2:6A:9A:02:21:00:DE:B7:EA:51:98:34:D0:70:
CF:4E:94:F4:B0:32:13:70:68:2A:D7:8C:38:9D:13:FD:
                                EF:5F:B1:09:B8:1E:36:D9
                Signed Certificate Timestamp:
                    Version   : v1 (0x0)
                    Log ID    :
07:B7:5C:1B:E5:7D:68:FF:F1:B0:C6:1D:23:15:C7:BA:
E6:57:7C:57:94:B7:6A:EE:BC:61:3A:1A:69:D3:A2:1C
                    Timestamp : Apr 20 08:30:33.929 2020 GMT
                    Extensions: none
                    Signature : ecdsa-with-SHA256
30:46:02:21:00:DC:21:FE:44:A5:EB:03:27:D4:3A:25:
62:D1:2D:9B:F8:8C:D6:B8:27:3D:CE:17:C9:25:8A:71:
E0:4A:7F:7B:08:02:21:00:9F:13:05:62:9C:79:F4:33:
19:4B:B9:BA:86:64:87:28:20:B7:C3:28:06:26:B7:52:
                                95:2C:EF:69:13:A5:14:D1
    // 签名
    Signature Algorithm: sha256WithRSAEncryption
         7f:3a:c5:1d:88:67:6d:40:c1:34:ac:01:9c:44:32:b1:8a:6d:
         1d:c6:bc:e7:52:5c:e4:42:0b:14:17:bd:47:4b:36:52:5c:eb:
         94:11:99:9b:82:b6:2b:db:43:b1:85:a2:38:54:32:e7:9c:8f:
         53:b2:6f:c8:7f:ef:bb:17:c5:c1:08:ec:65:33:0b:8c:dc:e2:
         20:c0:29:99:20:cc:a4:9a:80:9e:f6:ef:04:34:bc:0d:f2:53:
         ce:98:2e:a6:a8:33:c0:24:0e:ca:d4:f9:fc:99:cc:d3:df:96:
         12:f5:69:b3:f1:72:33:26:81:87:16:71:0e:35:a2:91:29:44:
         ce:33:fb:95:81:de:5b:a3:8c:07:ff:b6:f7:90:c0:3f:62:f2:
         3e:e4:45:cc:41:43:f6:27:2f:f0:b3:bb:8e:3d:ce:37:05:e5:
         30:be:c9:a4:32:7a:6c:a8:57:eb:ce:66:46:9b:fd:b5:13:1b:
         61:bf:05:7e:6f:9e:e0:99:77:8f:5a:eb:a5:79:3e:3a:57:dc:
         38:2b:3d:3e:39:31:f1:46:f2:b6:b1:4f:80:06:6e:4f:3b:ff:
         94:fb:74:b7:6d:01:ea:ae:5f:a7:3f:d3:24:eb:ea:7e:c4:ef:
         7c:97:4f:be:eb:f9:87:fd:64:b9:e7:0f:3c:b1:c1:51:4a:fe:
         86:eb:8d:95
```

证书中主要包含：

- 证书颁发机构：用于寻找链中的下一个验证节点
- 证书的有效期：比如浏览器要根据这个值来判断证书是否已过期
- 证书申请信息：比如浏览器要判断改证书是否可用于当前访问的域名
- 公钥：用于后续和服务端通信的秘钥，这个公钥和当初生成 CSR 时的公钥是一个东西，因为只有它是和服务器的私钥是一对的
- 签名：用于验证证书内容没有被篡改

这里简单说一说这个证书里面的公钥和签名：

前面在介绍生成 CSR 的时候，我们说过了，签名部分，是服务器使用私钥加密 hash 值得到的，同时在 CSR 中包含了公钥，这样 CA 在收到这个文件后，可以用 CSR 文件中的公钥来解密签名，进而做校验。

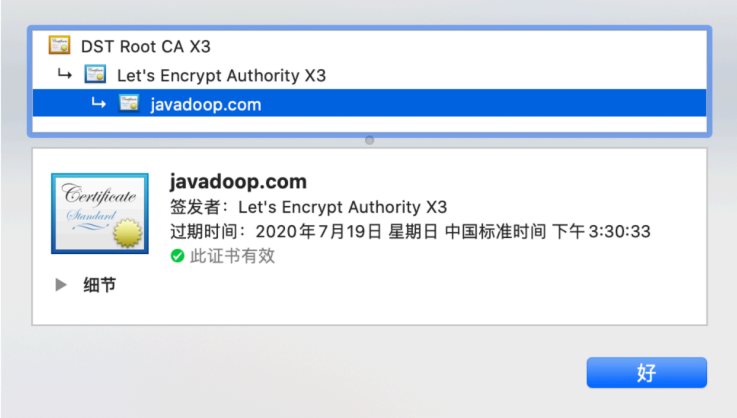
而这里不一样，这个证书是 CA 给我们的，自然这个签名也是 CA 使用它自己的私钥进行加密的，但是这里的公钥是我们服务器的公钥，显然不能用于解密签名。

那对于用户浏览器来说，在收到这个证书以后，怎么校验这个证书的签名呢？显然浏览器需要得到 CA 的公钥。下一节我们就将详细描述这个过程。

HTTPS 验证过程

其实，前面已经把大部分内容都说完了，我们只需要知道最后一步，怎么得到 CA 的公钥就可以了。

我们以一个实际的例子来分析，下面将使用 javadoop.com 这个域名的证书来分析。



首先，我们可以看到，这个证书链由 3 个证书组成。`javadoop.com` 证书由中间证书 `Let's Encrypt Authority X3` 签发，中间证书由 `DST Root CA X3` 签发，而 `DST Root CA X3` 是一个受信任的根证书。

流程如下：

1、用户访问 `https://javadoop.com`，服务器返回 CA 给的证书链，其中包含了 `javadoop.com` 证书以及中间证书；

2、浏览器首先需要判断 javadoop.com 的证书是不是可信的，关键的一步就是要解密证书的签名部分。因为证书是由中间证书签发的，所以要用中间证书里面的公钥来进行解密；

3、第 2 步初步判断了 javadoop.com 的证书是合法的，但是，这个是基于中间证书合法的基础上来的，所以接下来要判断中间证书是否是合法的；

4、根据中间证书里面的信息，可以知道它是由 `DST Root CA X3` 签发的，由于证书链只有两个节点，所以要到操作系统的根证书库中查找，由于这个证书是一个使用非常广泛的根证书，所以在系统中可以找到它。然后利用根证书的公钥来解密中间证书的签名部分，进而判断中间证书是否合法，如果合法，整个流程就通了；

我们思考一下：

- 这个系统要工作好，关键就是最终一定要走到本地根证书库，一环验证一环，实现整个链路证书的可信任；
- 中间证书有多少层都可以，只要能一直传递到根证书就行；
- 本地的根证书是由操作系统内置的，如果你的使用场景中，根证书不在系统预装里面，需要手动导入根证书；

另外，我这里使用了操作系统内置这个说法，其实也不准确吧，各大浏览器厂商可以自己内置这个根证书库，这样我想信任谁就信任谁，而不是听 Microsoft、Apple... 这些操作系统厂商的。

脑洞大开一下，如果你想开一家 CA 公司，技术上是没什么成本的，但是你要说服各大操作系统、浏览器厂商，把你家的根证书内置到里面，这就有点难了。当然，还有另一条路可以走，那就是不要搞根证书，基于某个 CA 搞个中间证书，然后用这个中间证书去签发证书就可以了。

小结

本来我还想聊聊 Java 相关的内容的，不过怕是本文的信息量也挺大了，就不折腾大家了。

大家碰到不懂的地方，请仔细分析下我写的内容，如果觉得我哪里写得不易懂，或者有错误，欢迎大家指正。