防御手段

知识要点

1. 密码安全

- 泄露渠道
  - 数据库被偷
  - 服务器被入侵
  - 通讯被窃听
  - 内部人员泄露
  - 其他网站（撞库）
- 防御
  - 严禁明文存储
  - 单向变换
  - 变换复杂度要求
  - 密码复杂度要求
  - 加盐（防拆解）
- 哈希算法
  - 明文 - 密文 - ——对应
  - 雪崩效应 - 明文小幅变化 密文剧烈变化
  - 密文 -明文无法反推
  - 密文固定长度 md5 sha1 sha256
- 密码传输安全
  - https传输
  - 频次限制
  - 前端加密意义有限 - 传输层加密 不会泄露 但不代表不能登录
- 摘要加密的复杂度
  - md5反查

    https://www.cmd5.com/

```
// /app/password.js
const crypto = require('crypto')
const hash = (type, str) => crypto.createHash(type).update(str).digest('hex')
const md5 = str => hash('md5',str)
const sha1 = str => hash('sha1',str)
const encryptPassword = (salt, password) => md5(salt + 'abced@#4@%#$7' + password)
const psw = '123432! @#! @#@! #'
console.log('md5', md5(psw))
console.log('sha1', sha1(psw))
module.exports = encryptPassword
```

两种强化方式

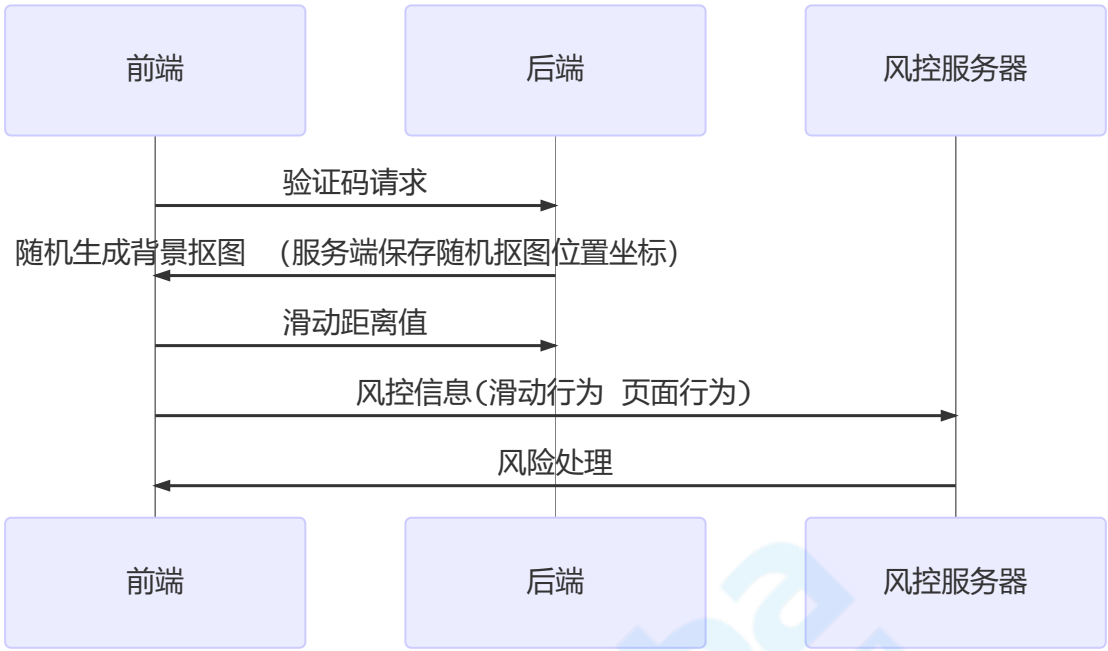
```
// index.js
  const encryptPassword = require('./password')
 if (res.length !== 0 && res[0].salt === null) {
        console.log('no salt ..')
        if (password === res[0].password) {
            sql = `
                update test.user
                set salt = ?,
                password = ?
                where username = ?
            `
            const salt = Math.random() * 99999 + new Date().getTime()
            res = await query(sql, [salt, encryptPassword(salt, password), username])
            ctx.session.username = ctx.request.body.username
            ctx.redirect('/?from=china')
        }
    } else {
        console.log('has salt')
        if (encryptPassword(res[0].salt, password) === res[0].password) {
            ctx.session.username = ctx.request.body.username
            ctx.redirect('/?from=china')
        }
    }
```

2. 人机验证 与 验证码

样式和反面教材 https://veui.net/

```
$('.verify-code font').text()
```

> http://www.lisa33xiaoq.net/1232.html



滑动验证码实现原理

1.服务端随机生成抠图和带有抠图阴影的背景图片，服务端保存随机抠图位置坐标；

2.前端实现滑动交互，将抠图拼在抠图阴影之上，获取到用户滑动距离值；

3.前端将用户滑动距离值传入服务端，服务端校验误差是否在容许范围内；

3. HTTPS 配置

> https 和密码学 https://www.cnblogs.com/hai-blog/p/8311671.html 浏览器如何验证SSL证书 http://wemedia.ifeng.com/70345206/wemedia.shtml

HTTP的弱点

```
#查看需要经过的节点
traceroute www.baidu.com
```

危害

- 窃听
  - 密码 敏感信息
- 篡改
  - 插入广告 重定向到其他网站(JS 和 Head头)

时代趋势

> - 目前全球互联网正在从HTTP向HTTPS的大迁移
> - Chrome和火狐浏览器将对不采用HTTPS 加密的网站提示不安全
> - 苹果要求所有APP通信都必须采用HTTPS加密
> - 小程序强制要求服务器端使用HTTPS请求

特点

- 保密性 (防泄密)
- 完整性（防篡改）
- 真实性（防假冒）

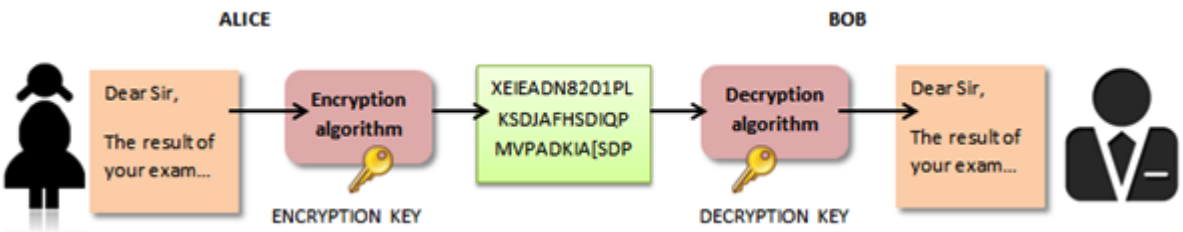
HTTP + SSL = HTTPS

什么是SSL证书

SSL证书由浏览器中"受信任的根证书颁发机构"在验证服务器身份后颁发,具有网站身份验证和加密传输双重功能

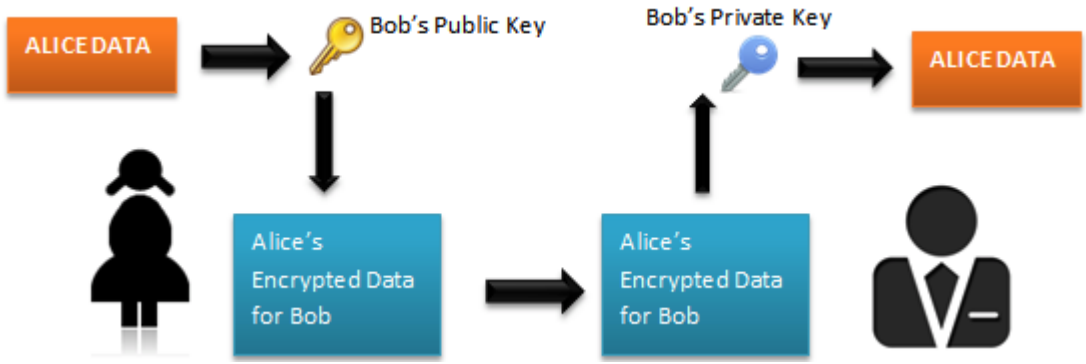
密码学

对称加密



> 对称加密的一大缺点是密钥的管理与分配，换句话说，如何把密钥发送到需要解密你的消息的人的手里是一个问题。在发送密钥的过程中，密钥有很大的风险会被黑客们拦截。现实中通常的做法是将对称加密的密钥进行非对称加密，然后传送给需要它的人。
>
> DES

不对称加密



- 产生一对秘钥
- 公钥负责加密
- 私钥负责解密
- 私钥无法解开说明公钥无效 - 抗抵赖
- 计算复杂对性能有影响(极端情况下 1000倍)

  常见算法 RSA（大质数）、Elgamal、背包算法、Rabin、D-H、ECC（椭圆曲线加密算法）。

> RSA原理
>
> http://www.ruanyifeng.com/blog/2013/06/rsa_algorithm_part_one.html

SSH公钥登录原理

- 密码口令登录

  通过密码进行登录，主要流程为：

  1、客户端连接上服务器之后，服务器把自己的公钥传给客户端

  2、客户端输入服务器密码通过公钥加密之后传给服务器

  3、服务器根据自己的私钥解密登录密码，如果正确那么就让客户端登录

- 公钥登录

  公钥登录是为了解决每次登录服务器都要输入密码的问题，流行使用RSA加密方案，主要流程包含：

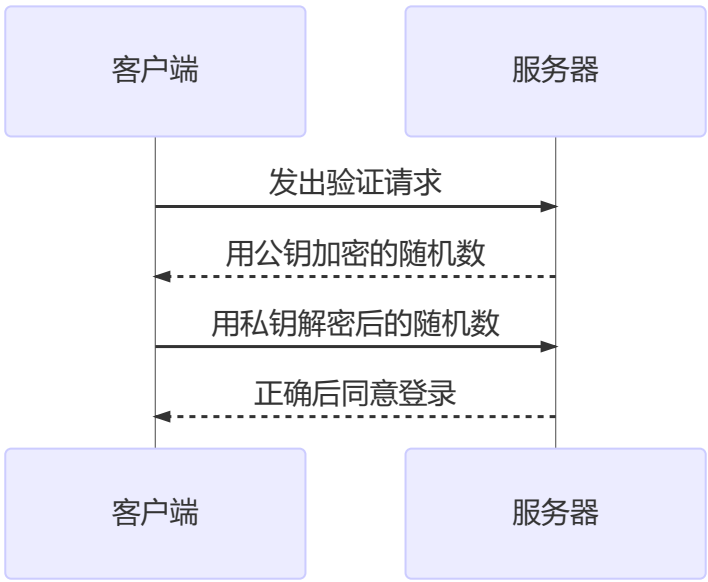
  1、客户端生成RSA公钥和私钥

  2、客户端将自己的公钥存放到服务器

  3、客户端请求连接服务器，服务器将一个用公钥加密随机字符串发送给客户端

  4、客户端根据自己的私钥加密这个随机字符串之后再发送给服务器

  5、服务器接受到加密后的字符串之后用公钥解密，如果正确就让客户端登录，否则拒绝。

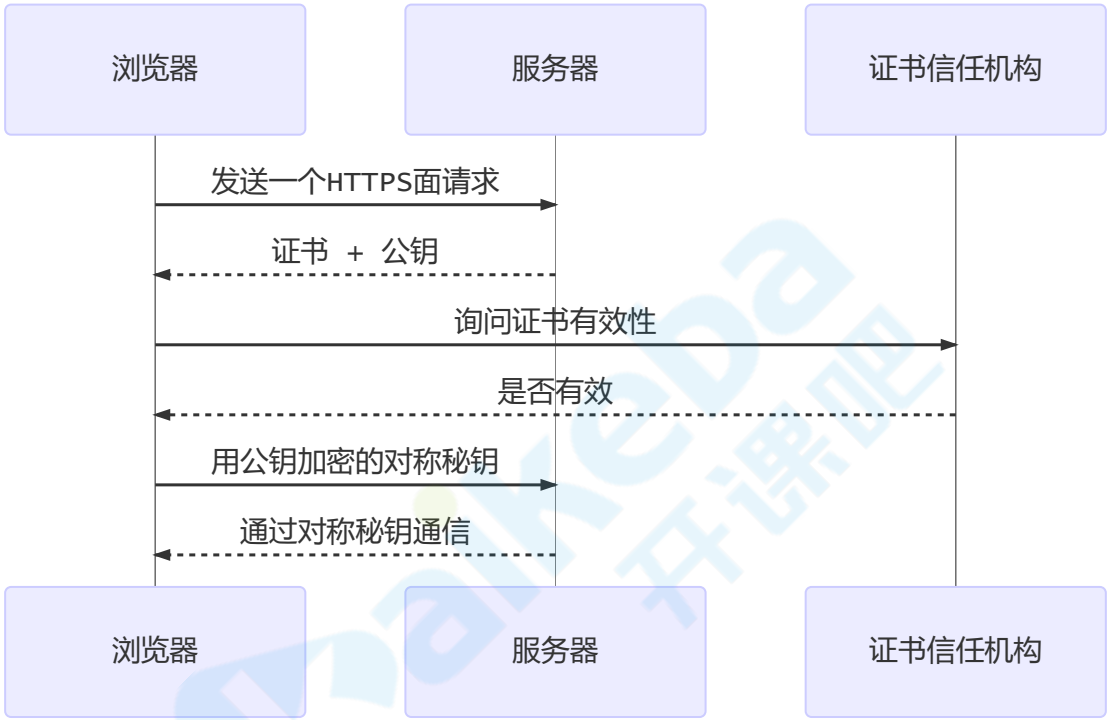


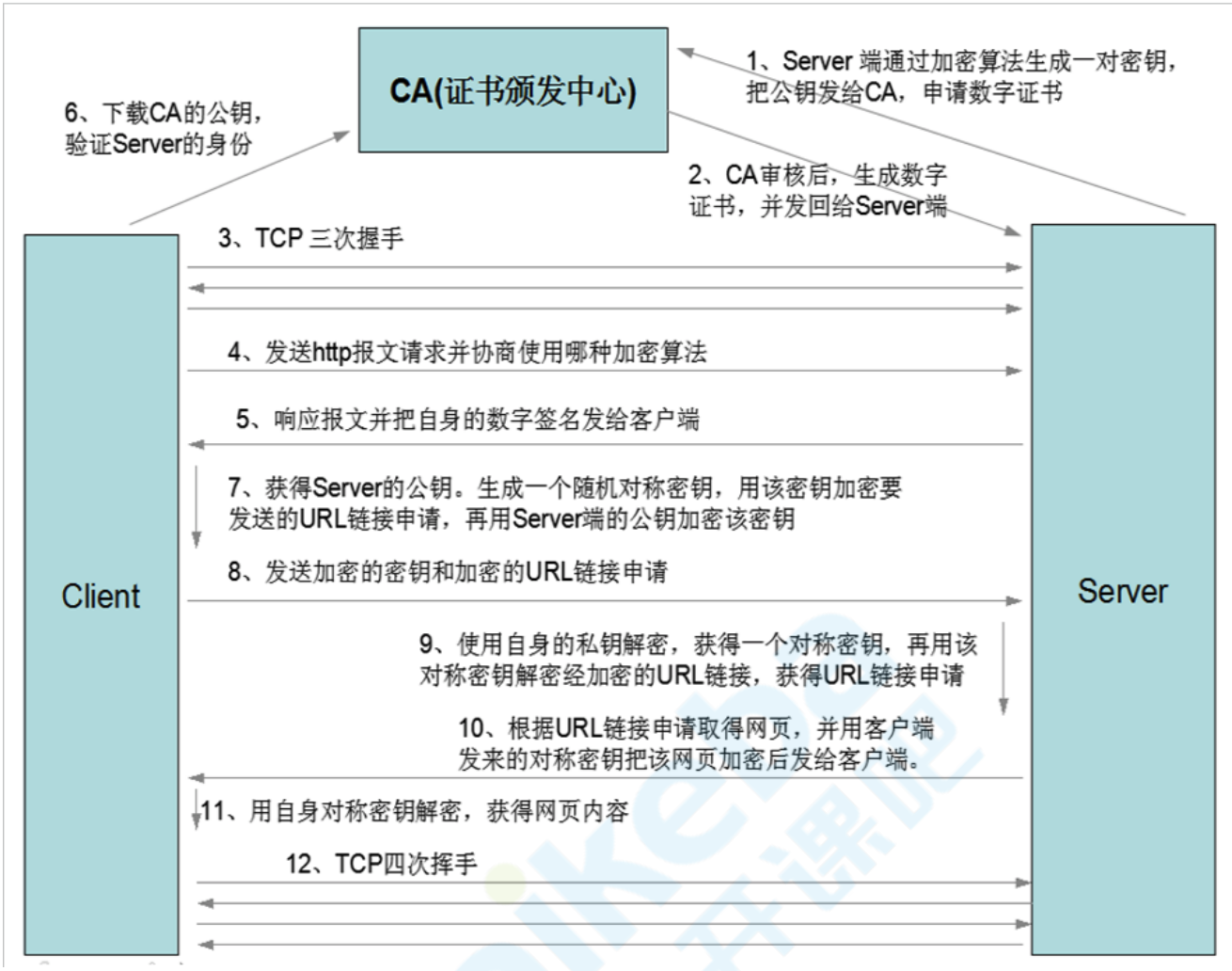
这样就不用使用密码了。

```
# 生成公钥
ssh-keygen -t rsa -P ''

xubin@xubindeMBP:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/xubin/.ssh/id_rsa):
/Users/xubin/.ssh/id_rsa already exists.
Overwrite (y/n)? yes
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/xubin/.ssh/id_rsa.
Your public key has been saved in /Users/xubin/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:IeFPfrcQ3hhP64SRTAFzGIHl2ROcopl5HotRi2XNOGk xubin@xubindeMBP
The key's randomart image is:
+---[RSA 2048]----+
|      .o*@=o     |
|     ..oEB=o     |
|      o@=+O .    |
|      B=+o @ .   |
|       =So* *    |
|      . o. = .   |
|            o    |
|                 |
|                 |
+----[SHA256]-----+

# 查看公钥
cat .ssh/id_rsa.pub
```

```
# 将公钥拷贝到服务器
scp ~/.ssh/id_rsa.pub root@47.98.252.XXX:/root

# 将公钥加入信任列表
cat id_dsa.pub >> ~/.ssh/authorized_keys
```

网站如何通过加密和用户安全通讯

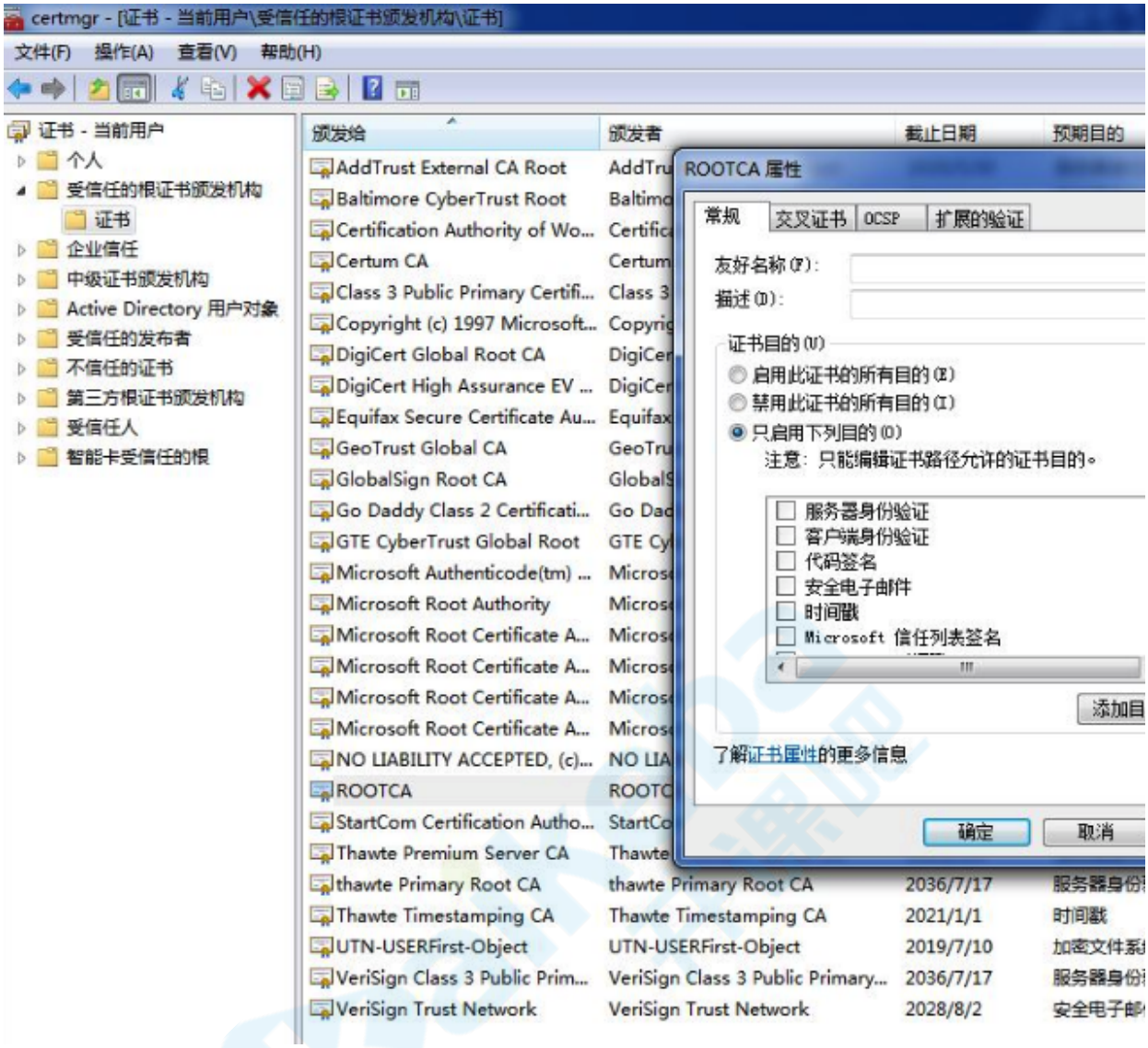




根证书在哪里

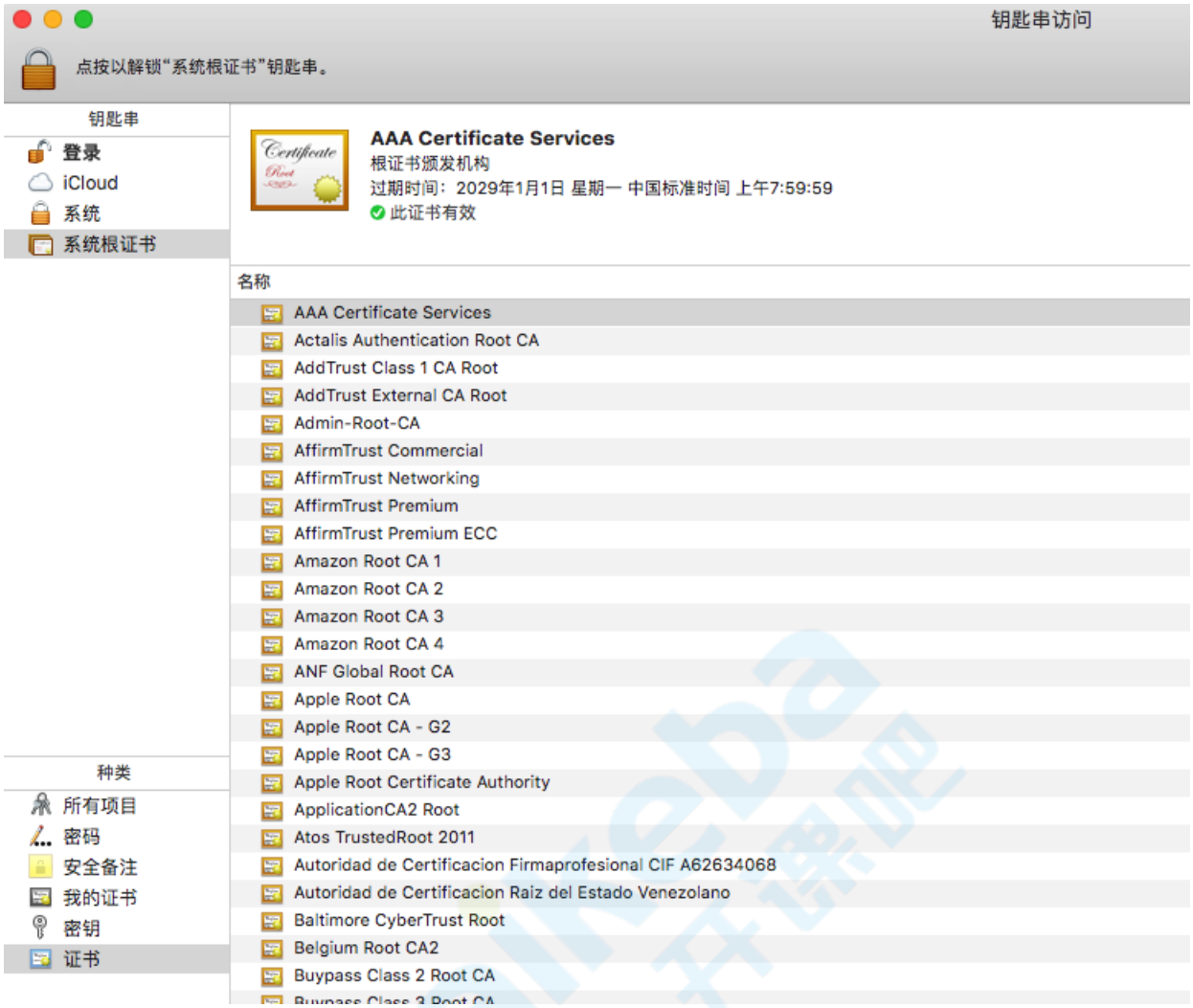
> windows

> 在Windows下按Windows+ R, 输入certmgr.msc，在“受信任的根证书颁发机构”-“证书中”找到“ROOTCA”，截止日期2025/08/23，单击右键，属性，可以查看其属性“禁用此证书的所有目的”


certmgr - [证书 - 当前用户\受信任的根证书颁发机构\证书]
文件(F) 操作(A) 查看(V) 帮助(H)
证书 - 当前用户
个人
受信任的根证书颁发机构
证书
企业信任
中级证书颁发机构
Active Directory 用户对象
受信任的发布者
不信任的证书
第三方根证书颁发机构
受信任人
智能卡受信任的根
颁发给
颁发者
截止日期
预期目的
AddTrust External CA Root
Baltimore CyberTrust Root
Certification Authority of Wo...
Certum CA
Class 3 Public Primary Certifi...
Copyright (c) 1997 Microsoft...
DigiCert Global Root CA
DigiCert High Assurance EV ...
Equifax Secure Certificate Au...
GeoTrust Global CA
GlobalSign Root CA
Go Daddy Class 2 Certificati...
GTE CyberTrust Global Root
Microsoft Authenticode(tm) ...
Microsoft Root Authority
Microsoft Root Certificate A...
Microsoft Root Certificate A...
Microsoft Root Certificate A...
Microsoft Root Certificate A...
NO LIABILITY ACCEPTED, (c)...
ROOTCA
StartCom Certification Autho...
Thawte Premium Server CA
thawte Primary Root CA
thawte Primary Root CA 2036/7/17 服务器身份
Thawte Timestamping CA
Thawte Timestamping CA 2021/1/1 时间戳
UTN-USERFirst-Object
UTN-USERFirst-Object 2019/7/10 加密文件系
VeriSign Class 3 Public Prim...
VeriSign Class 3 Public Primary... 2036/7/17 服务器身份
VeriSign Trust Network
VeriSign Trust Network 2028/8/2 安全电子邮
ROOTCA 属性
常规
交叉证书
OCSP
扩展的验证
友好名称(F):
描述(D):
证书目的(U)
启用此证书的所有目的(E)
禁用此证书的所有目的(I)
只启用下列目的(O)
注意：只能编辑证书路径允许的证书目的。
服务器身份验证
客户端身份验证
代码签名
安全电子邮件
时间戳
Microsoft 信任列表签名
添加目
了解证书属性的更多信息
确定
取消

Mac

钥匙串



> http://www.techug.com/post/https-ssl-tls.html HTTPS加密原理介绍

配置过程

- 修改开发机的host 前置

```
# 开发机的hosts文件 /etc/hosts
# 添加
127.0.0.1  www.josephxia.com
```

- 阿里云取得的真实证书 （域名 www.josephxia.com）
- docker模拟nginx环境

```
# 安全课程根目录
version: '3.1'
services:
  nginx:
    restart: always
    image: nginx
    ports:
      - 80:80
      - 443:443
    volumes:
      - ./conf.d/:/etc/nginx/conf.d
      - ./html/:/var/www/html/
```

- 原始的80端口服务

```
# conf.d/www.josephxia.com.conf

server {
    listen       80;
    server_name  www.josephxia.com;

    location / {
        root   /var/www/html;
        index  index.html index.htm;
    }

}
# 增加的部分
server {
    listen 443;
    server_name localhost;
    ssl on;
    root html;
    index index.html index.htm;
    # 公钥 + 证书
    ssl_certificate   conf.d/cert/www.josephxia.com.pem;
    # 私钥
    ssl_certificate_key  conf.d/cert/www.josephxia.com.key;
    ssl_session_timeout 5m;
    ssl_ciphers ECDHE-RSA-AES128-GCM-
SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_prefer_server_ciphers on;
    location / {
        root /var/www/html;
        index index.html index.htm;
    }
}
```

- 增加http -> https强制跳转

```
# conf.d/www.josephxia.com.conf
server {
    listen       80;
    server_name  www.josephxia.com;
    # location / {
    #     root   /var/www/html;
    #     index  index.html index.htm;
    # }

    location / {
        rewrite ^(.*) https://www.josephxia.com/$1 permanent;
    }
}
```

SSL证书分类

入门级 DVSSL - 域名有效 无门槛

企业型 OVSSL - 企业资质、个人认证

增强型EVSSL - 浏览器给予绿色地址栏显示公司名字

3. helmet中间件

> 英[ˈhelmɪt] 头盔
>
> > https://www.npmjs.com/package/koa-helmet

```
// npm i koa-helmet -s

const Koa = require("koa");
const helmet = require("koa-helmet");
const app = new Koa();

app.use(helmet());

app.use((ctx) => {
  ctx.body = "Hello World"
});

app.listen(4000);
```

- Strict-Transport-Security：强制使用安全连接（SSL/TLS之上的HTTPS）来连接到服务器。
- X-Frame-Options：提供对于“点击劫持”的保护。
- X-XSS-Protection：开启大多现代浏览器内建的对于跨站脚本攻击（XSS）的过滤功能。
- X-Content-Type-Options： 防止浏览器使用MIME-sniffing来确定响应的类型，转而使用明确的content-type来确定。
- Content-Security-Policy：防止受到跨站脚本攻击以及其他跨站注入攻击。

4. Session管理

对于cookie的安全使用，其重要性是不言而喻的。特别是对于动态的web应用，在如HTTP这样的无状态协议的之上，它们需要使用cookie来维持状态

- Cookie标示
  - secure - 这个属性告诉浏览器，仅在请求是通过HTTPS传输时，才传递cookie。
  - HttpOnly - 设置这个属性将禁止`javascript`脚本获取到这个cookie，这可以用来帮助防止跨站脚本攻击。
- Cookie域
  - domain - 这个属性用来比较请求URL中服务端的域名。如果域名匹配成功，或这是其子域名，则继续检查`path`属性。
  - path - 除了域名，cookie可用的URL路径也可以被指定。当域名和路径都匹配时，cookie才会随请求发送。
  - expires - 这个属性用来设置持久化的cookie，当设置了它之后，cookie在指定的时间到达之前都不会过期。

5. 浏览器安全控制

- X-XSS-Protection

  防止反射型XSS

- Strict-Transport-Security

  强制使用HTTPS通信

- CSP

> https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Content-Security-Policy
>
> https://juejin.im/post/5c6ad29ff265da2da00ea459

HTTP 响应头 **Content-Security-Policy** `允许站点管理者在指定的页面控制用户代理的资源。除了少数例外，这条政策将极大地指定服务源 以及脚本端点。这将帮助防止跨站脚本攻击 (Cross-Site Script)` (XSS).

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'; img-src
https://*; child-src 'none';">
```

安全防范的总结

https://www.tuicool.com/articles/7Ff2EbZ