

1 网络编程安全解读与实战

1.1 HTTPS超文本传输安全协议

1.1.1 什么是HTTPS

1、HTTP协议回顾

引入

HTTP 协议于 1991 年引入，至今已有近 30 年的历史

自第一个文档化版本（后来称为 0.9）以来，它已经经历了一段相当长的历程（5年）。

HTTP 的第一个版本只能传输超文本标记语言（HTML）文件，因此我们称之为超文本传输协议。

超文本传输协议：

是一个基于请求与响应，无状态的，应用层的协议

基于TCP/IP协议传输（明文）数据，互联网上应用最为广泛的一种网络协议,所有的WWW都必须遵守这个标准。

我们使用浏览器访问一个网站页面，在浏览器的地址栏中我们会看到一串URL

http://www.domain.com

通信协议 域名

域名地址都很好理解，不同的域名地址表示网站中不同的页面，而通信协议，简单来说就是浏览器和服务

器之间沟通的语言。

HTTP发展历程

引入

1.1

1997 年发布的 HTTP/1.1 引入了一些其它的改进。除了添加像 OPTIONS 这样的方法外，它还引入了 Keep-Alive 头。它允许一个连接对多个 HTTP 请求保持打开状态。因为这点，连接不必在每次请求之后关闭，然后再重新打开。

2.0

HTTP 2.0拥有更优异的表现，现在网络变得的复杂，甚至演变成了独有的应用，媒体的播放量，增进交互的脚本大小也增加了许多，更多的数据通过HTTP请求被传输，由此HTTP 2.0为网络效率做了大量的优化

版本	产生时间	内容	发展现状
HTTP/0.9	1991年	不涉及数据包传输, 规定客户端和服务端之间通信格式, 只能GET请求	没有作为正式的标准
HTTP/1.0	1996年	传输内容格式不限制, 增加PUT、PATCH、HEAD、OPTIONS、DELETE命令	正式作为标准
HTTP/1.1	1997年	持久连接(长连接)、节约带宽、HOST域、管道机制、分块传输编码	2015年前使用最广泛
HTTP/2	2015年	多路复用、服务器推送、头信息压缩、二进制协议等	逐渐覆盖市场

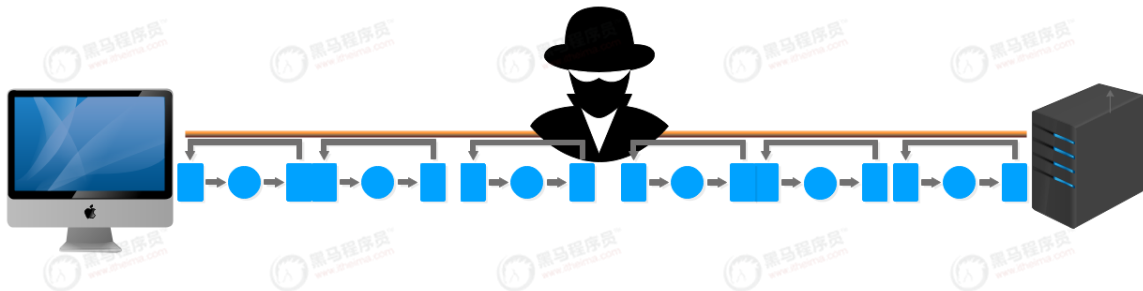
tips:

HTTP 3.0 目前处于制订和测试阶段, 是未来的全新的 HTTP 协议, HTTP 3.0 协议运行在 QUIC 协议之上, 是在 UDP 的基础上实现了可靠传输

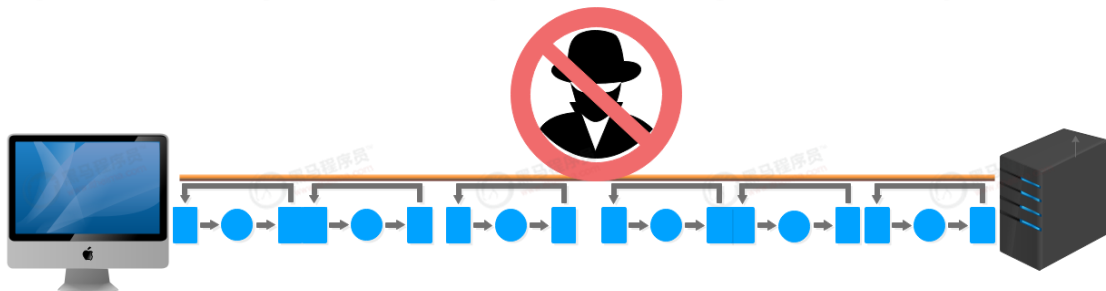
HTTP 3.0 不是 HTTP 2.0 的拓展, HTTP 3.0 将会是一个全新的协议

总结

1. HTTP协议始于1989年蒂姆·伯纳斯-李的一篇论文
2. HTTP/0.9是个简单的文本协议,只能获取文本资源(get获取纯文本)
3. HTTP/1.0确立了现在使用的大部分技术,但不是正式标准
4. HTTP/1.1是目前互联网使用最为广泛的协议,功能也非常完善(第一个标准文档).
5. HTTP/2基于Google的SPDY协议,注重性能改善,但还未普及
6. HTTP/3基于Google的QUIC协议,是将来的发展方向



2、HTTPS协议



https

编辑

收藏 | 2267 | 310

本词条由“科普中国”科学百科词条编写与应用工作项目 审核。

HTTPS（全称：Hyper Text Transfer Protocol over SecureSocket Layer），是以安全为目标的 HTTP 通道，在 HTTP 的基础上通过传输加密和身份认证保证了传输过程的安全性^[1]。HTTPS 在 HTTP 的基础下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。HTTPS 存在不同于 HTTP 的默认端口及一个加密/身份验证层（在 HTTP 与 TCP 之间）。这个系统提供了身份验证与加密通讯方法。它被广泛用于万维网上安全敏感的通讯，例如交易支付等方面^[2]。

中文名	超文本传输安全协议	目 标	安全的HTTP通道
外文名	HTTPS	应 用	安全数据传输

HTTPS协议可以理解为HTTP协议的升级，就是在HTTP的基础上增加了数据加密。在数据进行传输之前，对数据进行加密，然后再发送到服务器。这样，就算数据被第三者所截获，但是由于数据是加密的，所以你的个人信息让然是安全的。

这就是HTTP和HTTPS的最大区别。

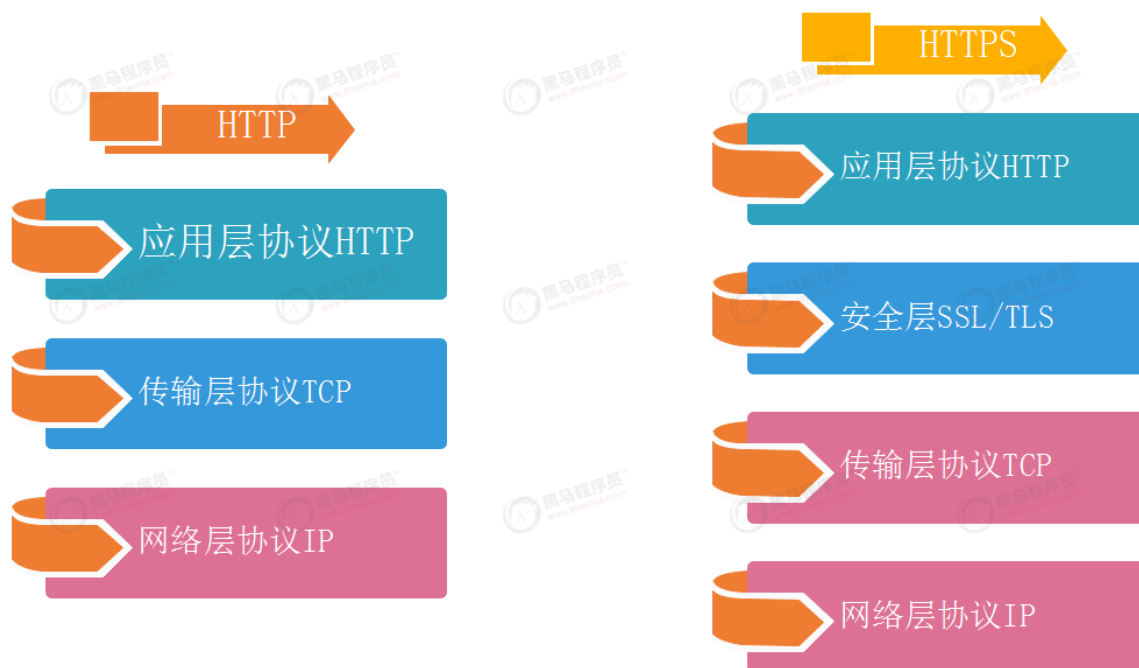


1.1.2 HTTPS与HTTP的区别

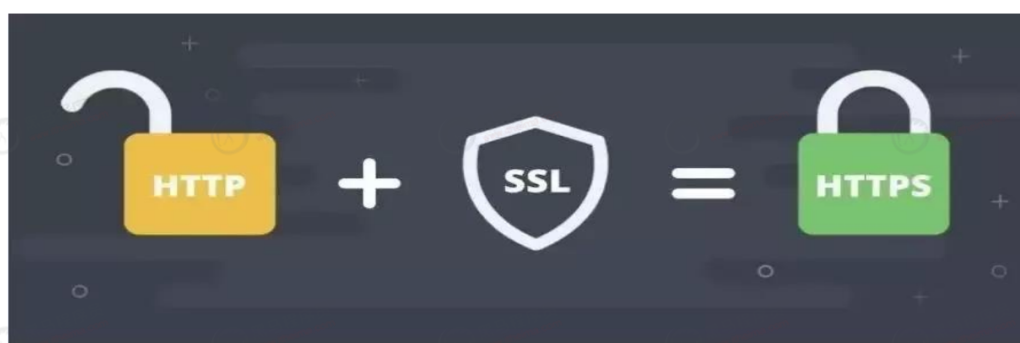
- 1、https协议需要申请CA,证书,http不需要证书
- 2、http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl/tls加密传输协议。
- 3、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。
- 4、http的连接很简单，是无状态的；HTTPS协议是由SSL/TLS+HTTP协议构建的加密传输
- 5、http直接与四层交互，https先给安全层交互，然后在与四层交互

1.2 加密原理与算法剖析

1.2.1 加密原理介绍



简单理解



如上图所示 HTTPS 相比 HTTP 多了一层 SSL/TLS

ssl [编辑](#)

本词条由“科普中国”科学百科词条编写与应用工作项目 审核。

SSL(Secure Sockets Layer 安全套接字协议),及其继任者传输层安全 (Transport Layer Security, TLS) 是为网络通信提供安全及数据完整性的一种安全协议。TLS与SSL在传输层与应用层之间对网络连接进行加密。

中文名	安全套接字协议	缩 写	SSL
外文名	Secure Sockets Layer	应 用	在传输层对网络连接进行加密

TLS (Transport Layer Security, 传输层安全)：

其前身是 SSL，它最初的几个版本 (SSL 1.0、SSL 2.0、SSL 3.0) 由网景公司开发

1999年从 3.1 开始被 IETF 标准化并改名，发展至今已经有 TLS 1.0、TLS 1.1、TLS 1.2 三个版本。

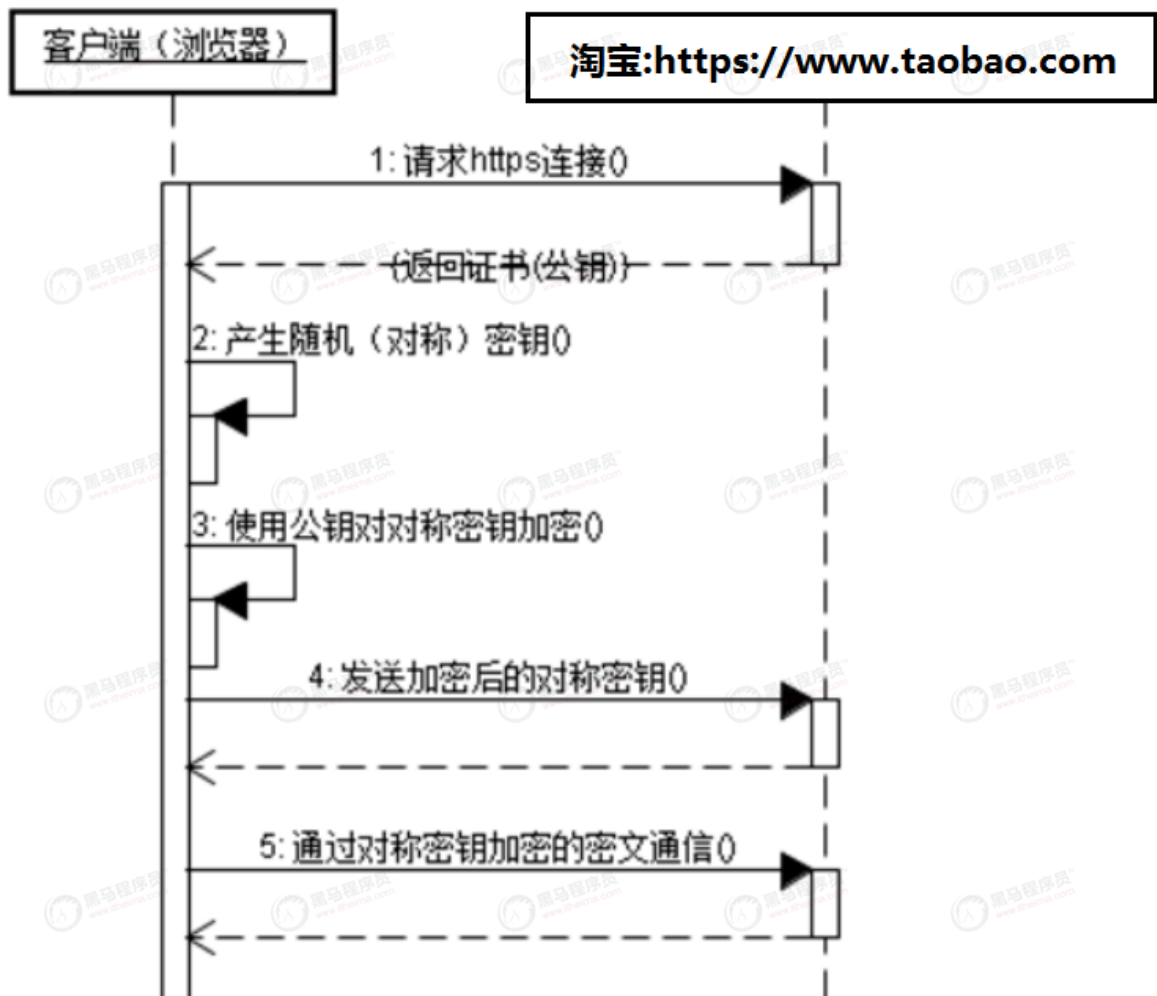
TLS 1.3 改动会比较大，目前还未大规模推广，目前使用最广泛的是TLS 1.2



1.2.2 加密后通信流程

客户端在使用HTTPS方式与Web服务器通信时的步骤

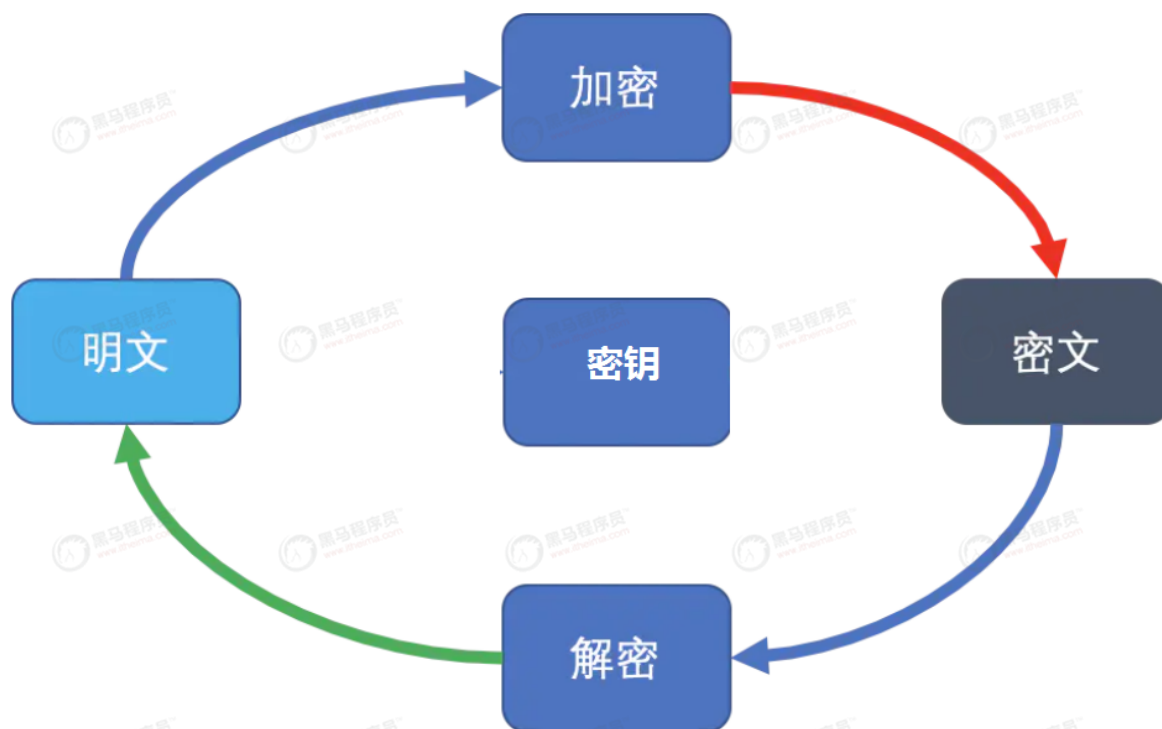
访问www.taobao.com



1.2.3 加密算法剖析

要建立一个密码体制，需要由五个空间组成，分别是：

- 明文M：加密前或解密后的信息；
- 密文C：明文加密后的信息；
- 密钥K：由加密密钥和解密密钥组成；
- 加密E：从明文到密文的变换；
- 解密D：从密文到明文的变换



1、对称加密：

或者也叫单钥加密，是指加密密钥和解密密钥相同的加密方式。

对称加密的主要优势是：加密、解密运算速度快，效率高；

代表算法：AES算法、DES算法、RC4，3DES、Triple-DES、RC2等；

• 特点：

1. 发送端和接收端共享相同的密钥才能通信，一定程度上保证了安全。
2. 对于枚举攻击，密码长度越长越难以破解

对称加密经典应用：

0010101010101010 明文

0001010001001011 密钥

0011111011100001 密文（异或后）

0010101010101010 明文（解密后）

2、非对称加密

加密使用的密钥和解密使用的密钥是不相同的，分别称为：公钥、私钥，公钥和算法都是公开的，私钥是保密的。

非对称加密算法性能较低，但是安全性超强，由于其加密特性，非对称加密算法能加密的数据长度也是有限的。

例如：RSA、DSA、ECDSA、DH、ECDHE

局限性：加密、解密运算效率较低；

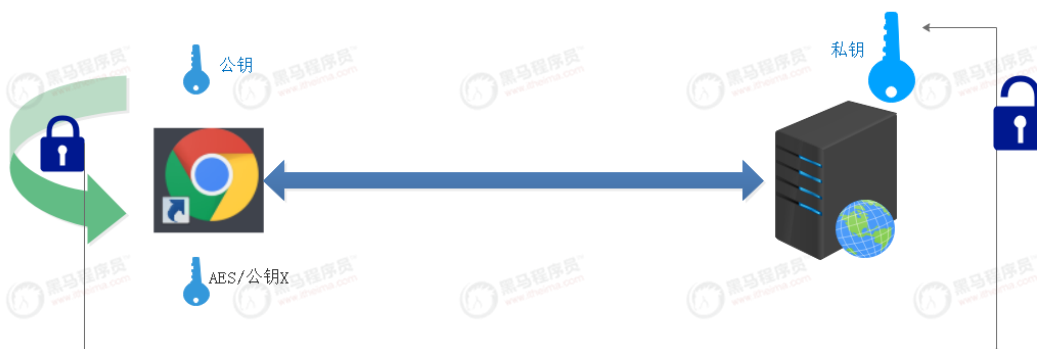
非对称加密算法：RSA，DSA（有一个公匙和一个私匙）

- 特点：

1. 编码密钥众所周知，但只有特定主机才知道私有解码密钥，解码密钥保密，保证了传输安全。
2. 所有的想向主机发送报文的人都可以使用相同的编码密钥，解决了对称加密密钥数量多管理不便的问题。

对称、非对称区别

对称加密算法相比非对称加密算法来说，加解密的效率要高得多。但是缺陷在于对于密钥的管理上，以及在非安全信道中通讯时，密钥交换的安全性不能保障。所以在实际的网络环境中，会将两者混合使用。



3、散列算法（或音译为哈希）

Hash算法特别的地方在于它是一种单向（输出不依赖于输入）算法，用户可以通过hash算法对目标信息生成一段特定长度的唯一hash值，却不能通过这个hash值重新获得目标信息。

因此Hash算法常用在不可还原的密码存储、信息完整性校验等。

将任意长度的信息转换为较短的固定长度的值，通常其长度要比信息小得多，且算法不可逆。

例如：MD5、SHA-1、SHA-2、SHA-256 等

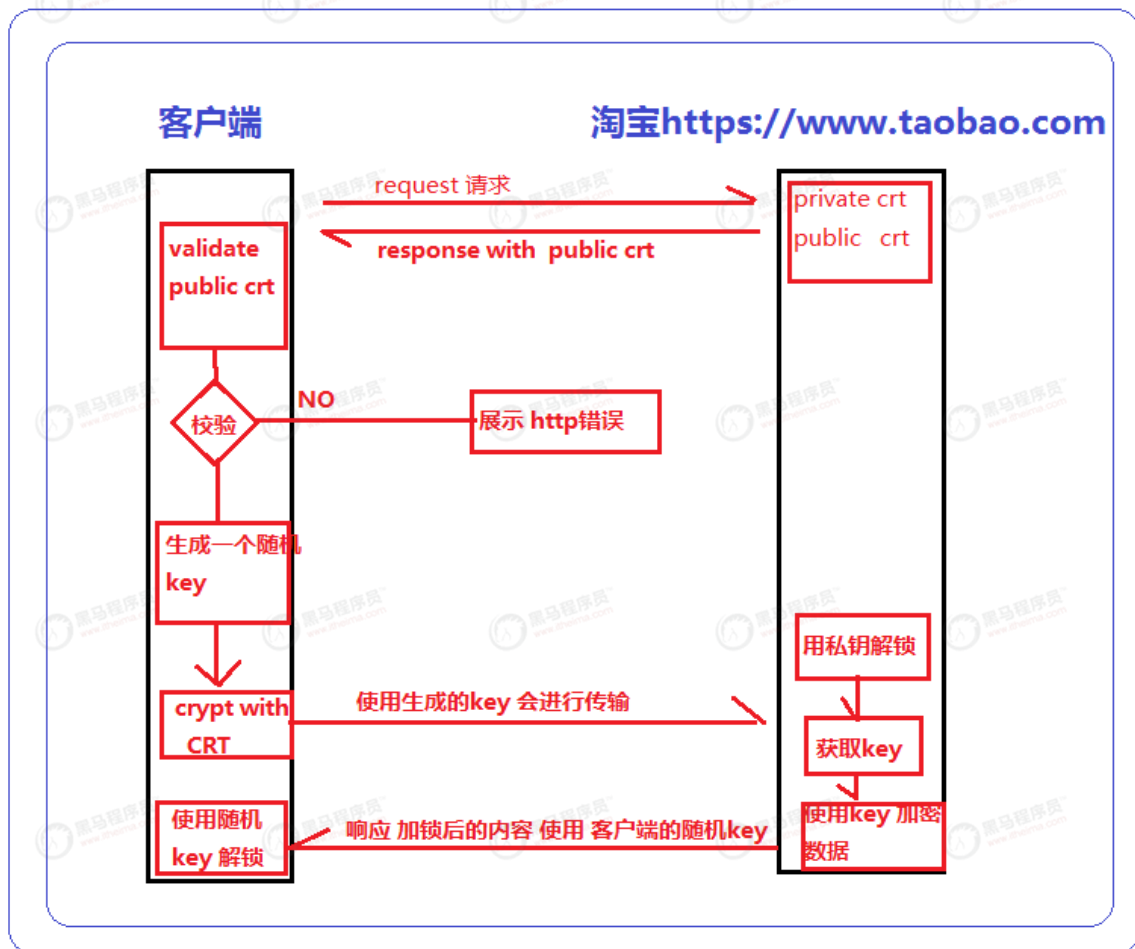
信息-摘要算法

MD5 ("ADSFASFA")=SSSSSS

把一个不定长的数据通过摘要算法处理成一个定长的数据

1.2.4 HTTPS协议SSL加密解密流程

1、加密解密流程

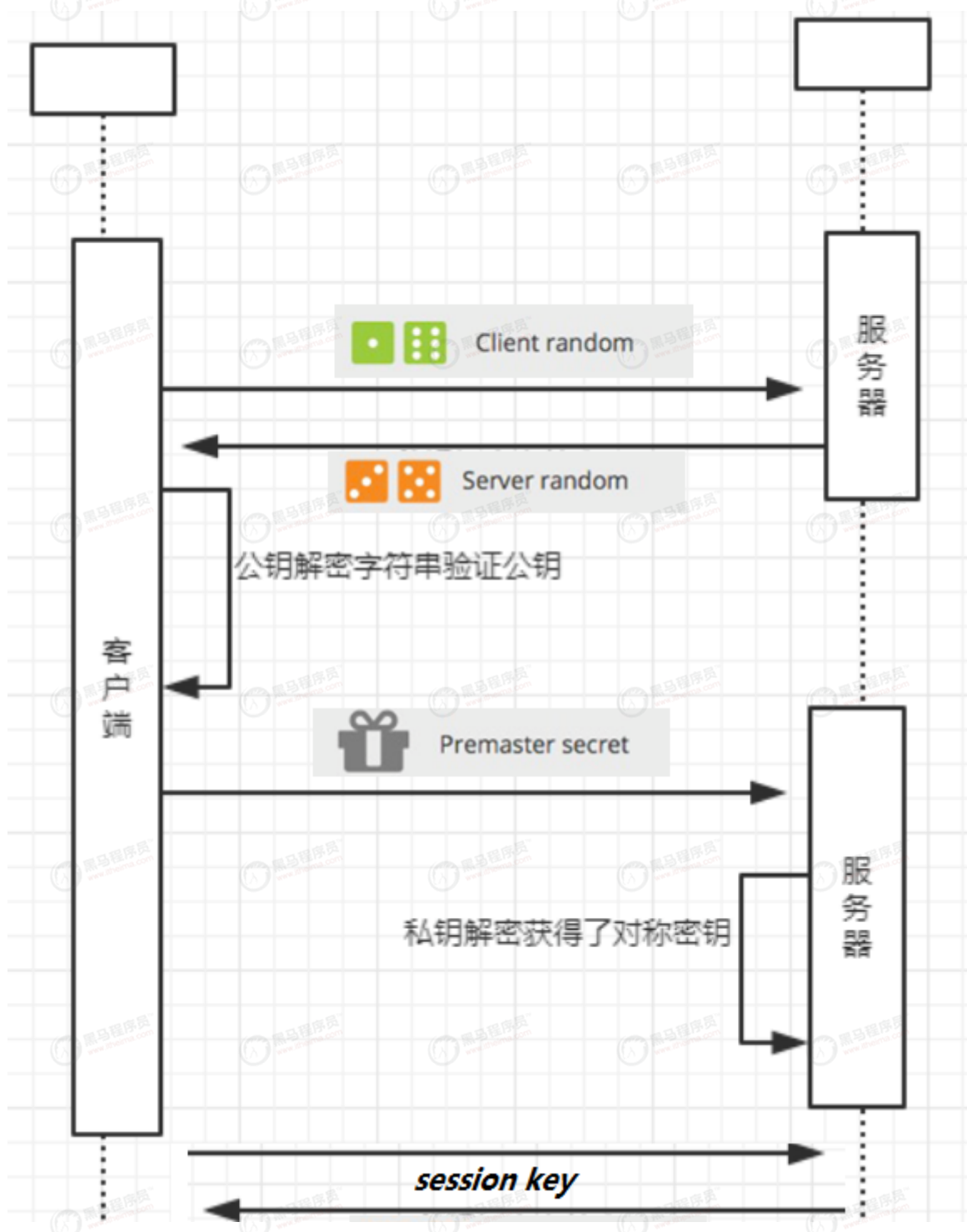


2、会话密钥生成流程详解（重要）

以上第1、2、4部分【生成的key或者随机值】是核心中的核心

也就是3次随机数生成会话密钥的过程

重点在说下



三个随机数：

- 1、客户端给出协议版本号，一个随机数（client random），以及客户端支持的加密方式
- 2、服务端确认双方使用的加密方式，并给出数字证书，以及一个服务器生成的随机数（server random）
- 3、客户端确认证书有效，然后生成一个新的46字节随机数（称为预备主密钥premaster secret），使用数字证书的公钥加密这个随机数，发送给服务端

总结：

服务端用非对称加密算法RSA生成公钥和私钥，把公钥放在证书里发送给客户端，私钥自己保存

客户端接收到公钥后，首先向一个权威的服务器检查证书的合法性，如果证书合法，客户端产生一段随机数，这个随机数就作为通信的密钥，我们称之为对称密钥，用公钥加密这段随机数，然后发送到服务器

服务器用密钥解密获取对称密钥，然后，双方就已对称密钥进行加密解密通信了

3、CA证书

公钥被截取了怎么办

CA证书

CA证书顾名思义就是由CA（Certification Authority，也叫“证书授权中心”）机构发布的数字证书

编码格式

X.509规范中一般推荐使用PEM格式来存储证书相关的文件。证书文件的文件名后缀一般为 .crt 或 .cer

对应私钥文件的文件名后缀一般为 .key。证书请求文件的文件名后缀为 .csr。有时候也统一用pem作为文件名后缀

编码格式总结：

- X.509 标准DER(Distinguished Encoding Rules)编码，后缀为：.der .cer .crt
- X.509 BASE64编码(PEM格式)，后缀为：.pem .cer .crt

CA证书具体包括以下内容

颁发者

使用者

版本

签名算法

签名哈希算法

使用者

公钥

指纹

指纹算法

.....

证书策略

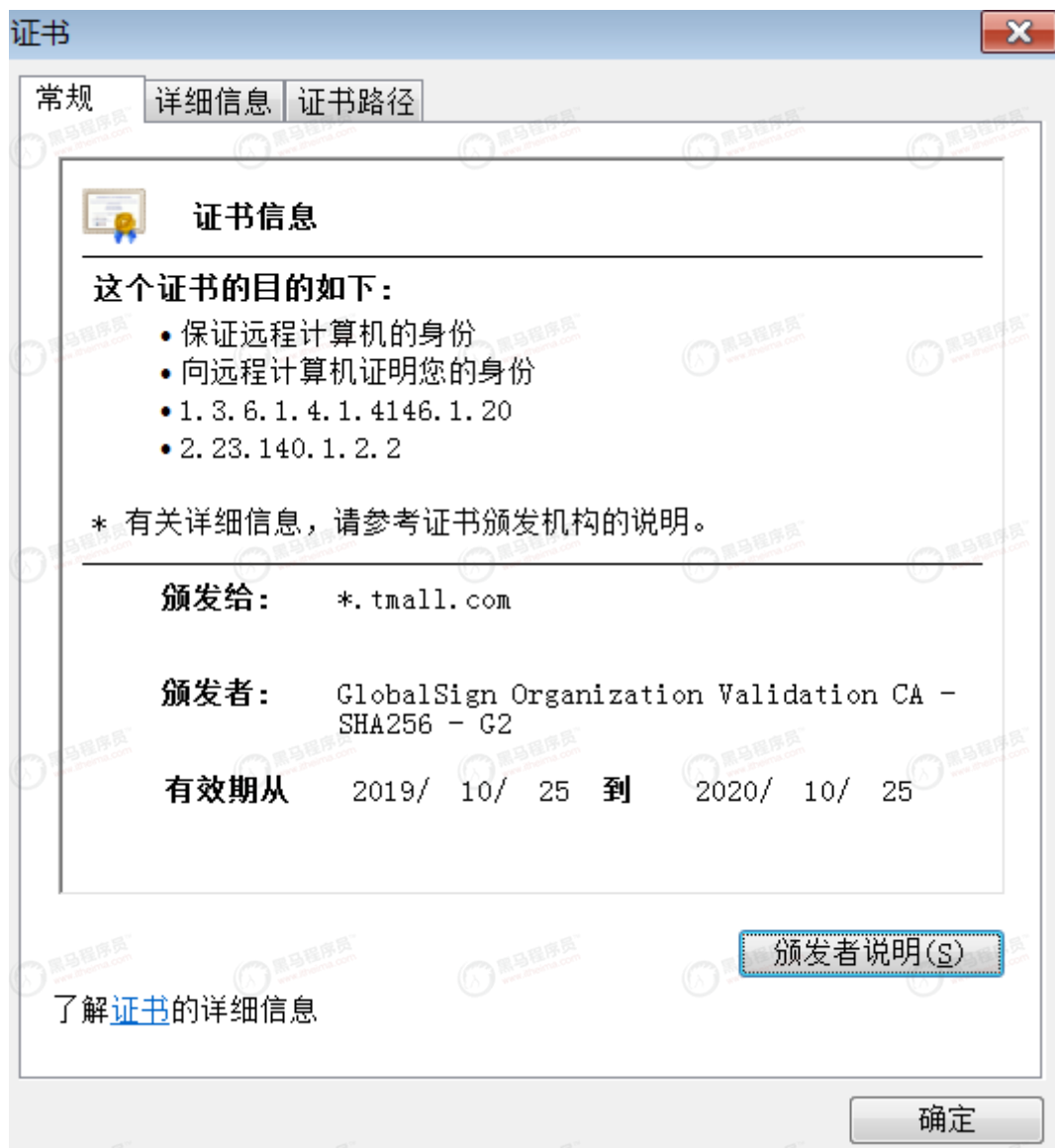
由对象标识符和限定符组成，这些对象标识符说明证书的颁发和使用策略有关

X509v3 Certificate Policies:

Policy: 1.3.6.1.4.1.4146.1.20

CPS: <https://www.globalsign.com/repository/>

Policy: 2.23.140.1.2.2



版本号(Version Number)：CA证书一个规范的版本号，目前为版本3，值为0x2；

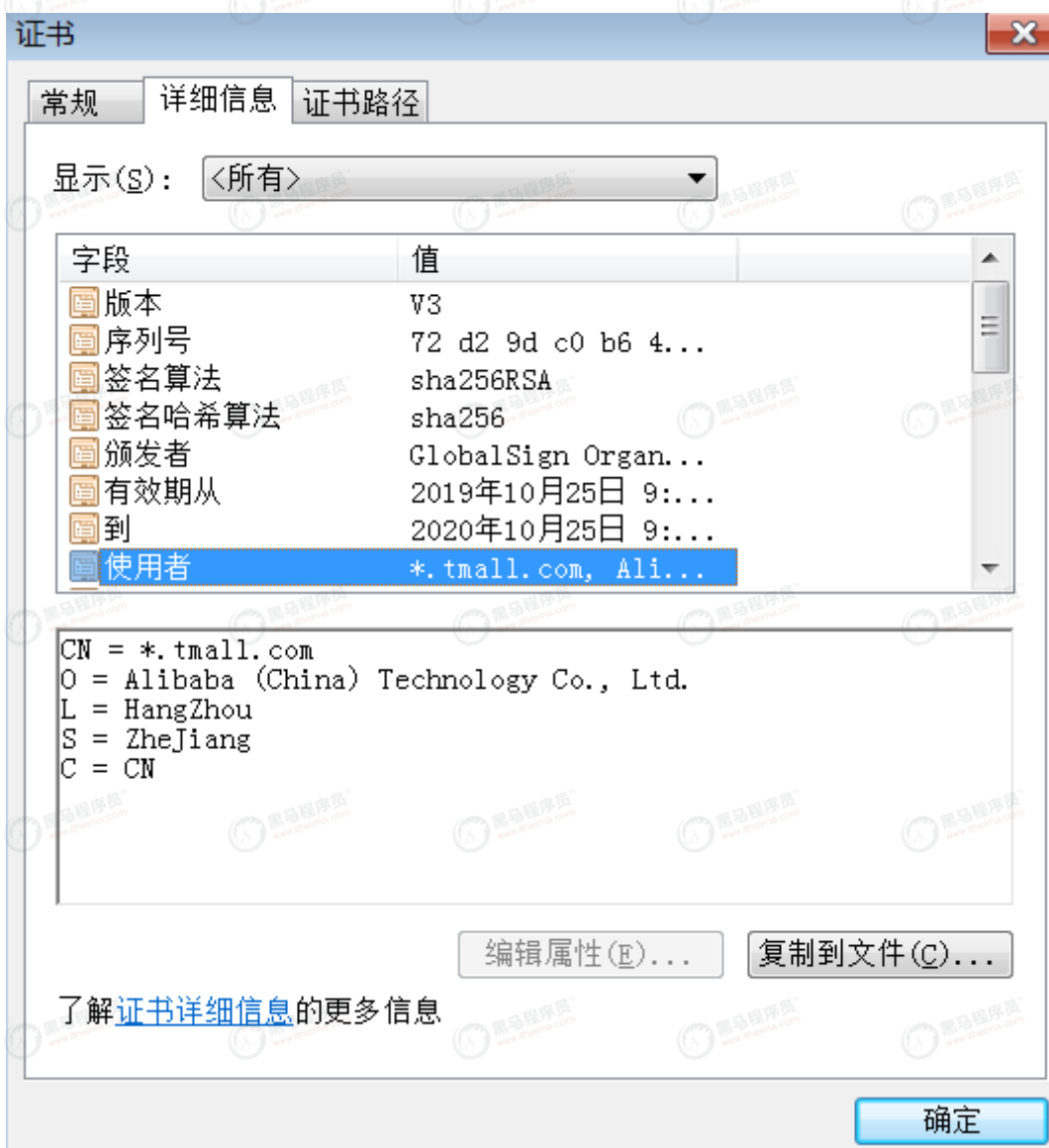
序列号 (Serial Number)：由CA维护的，为它所发的每个证书分配一个序列号，用来追踪和撤销证书。只要拥有签发者信息和序列号，就可以唯一标识一个证书，最大不能过20个字节；

签名算法 (Signature Algorithm)：数字签名所采用的算法，如：

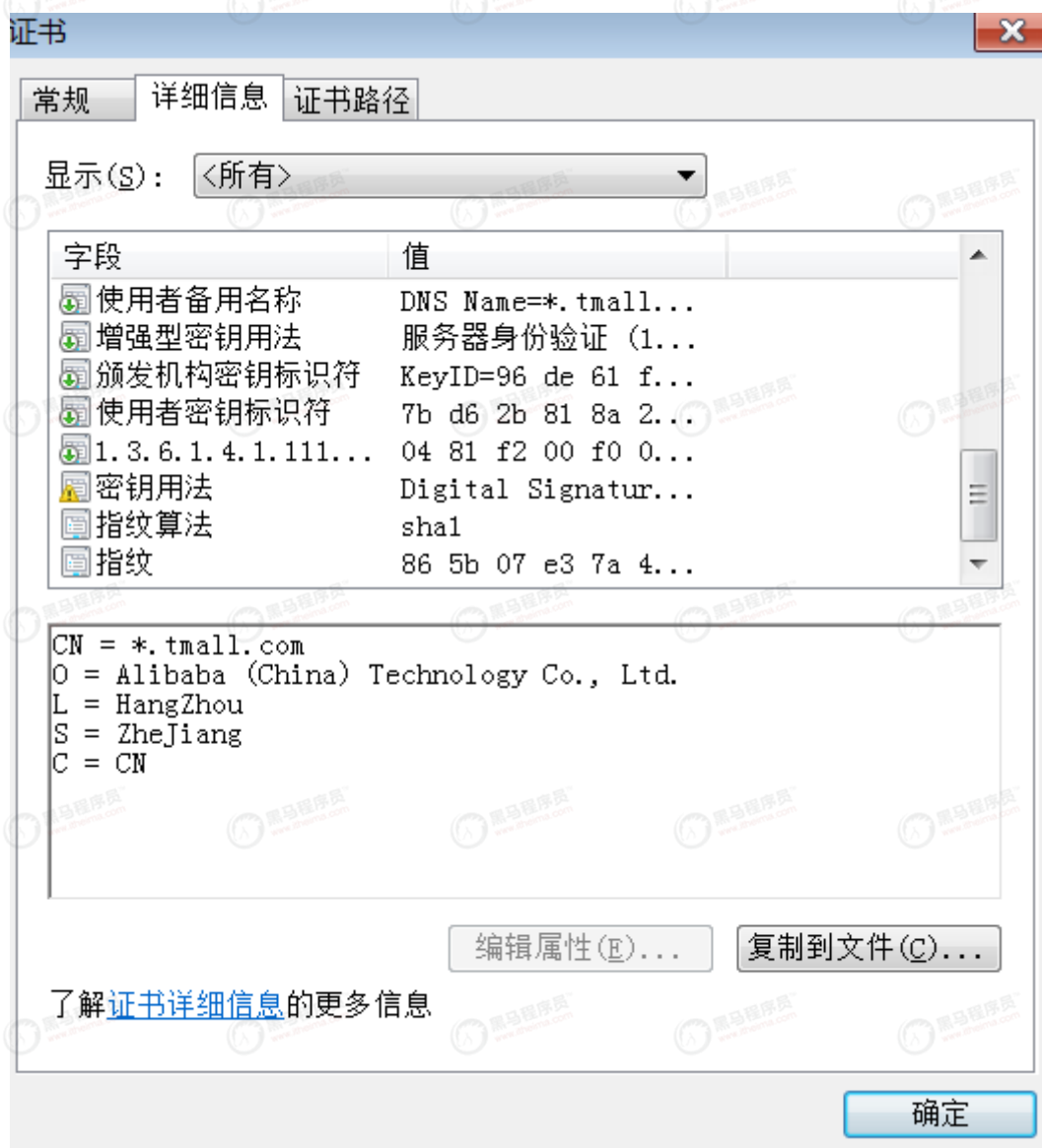
sha256-with-RSA-Encryption
ccdsa-with-SHA256；

颁发者 (Issuer)：发证书单位的标识信息，如 " C=CN, ST=Beijing, L=Beijing, O=org.example.com, CN=ca.org.example.com " ；

有效期(Validity): 证书的有效期很，包括起止时间。







服务器提供服务器的公钥给ca机构，生成证书，证书一般包含以下内容：

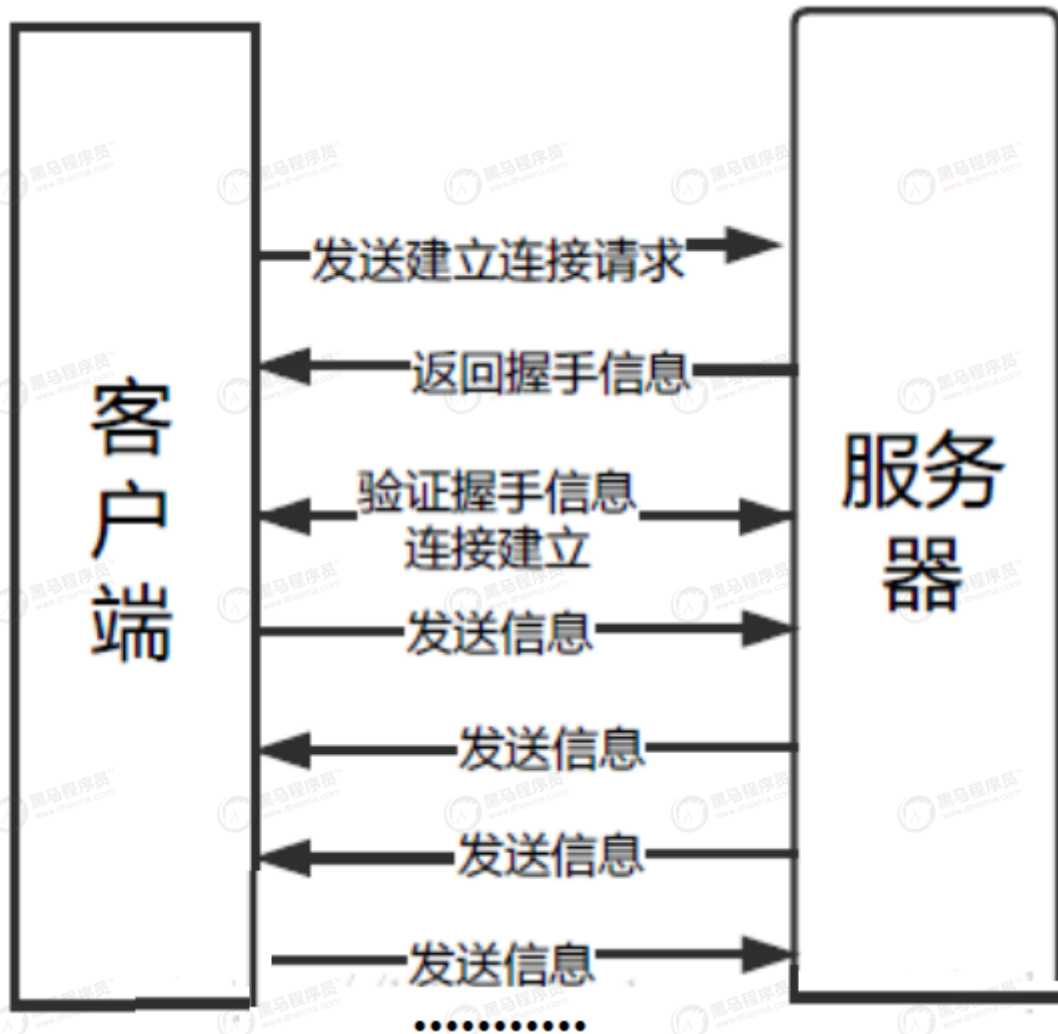
- ◆Issuer (证书的发布机构)
- ◆Valid from , Valid to (证书的有效期)
- ◆Public key (公钥)
- ◆Subject (主题，使用者)
- ◆Signature algorithm (签名所使用的算法)
- ◆Thumbprint, Thumbprint algorithm (指纹以及指纹算法)

生成证书的过程一般是：把Issuer (证书的发布机构),Public key (公钥),Subject (主题),Valid from,Valid to(证书的有效期)等信息以明文的形式写到证书里面作为内容，然后用一个指纹算法（淘宝是sha1）计算出这些数字证书内容的一个指纹（也就是签名），并把指纹和指纹算法用自己的私钥进行加密，然后生成证书

1.3 SSL/TLS实战与开发

1.3.1 WebSocket实战与应用

websocket通信



1、发送连接请求

客户端通过一个格式为：ws://host:port/的请求地址发起WebSocket连接请求，并由JavaScript实现WebSocket API与服务器建立WebSocket连接，其中host为服务器主机IP地址或域名，port为端口。

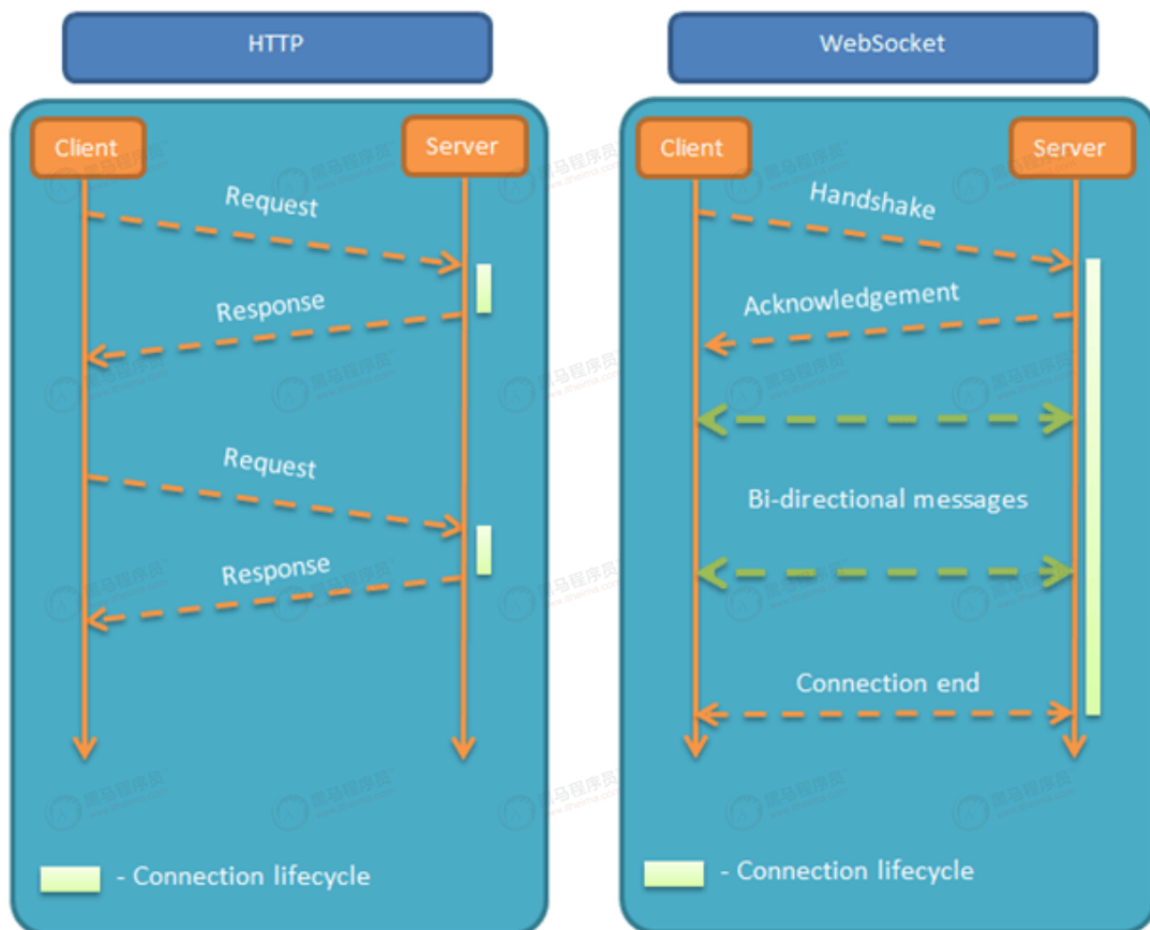
2、握手

当服务器收到请求后，会解析请求头信息，根据升级后的协议判断该请求为WebSocket请求，并取出请求信息中的Sec-WebSocket-Key字段的数据按照某种算法重新生成一个新的字符串序列放入响应头Sec-WebSocket-Accept中

Sec-WebSocket-Accept：服务器接受客户端HTTP协议升级的证明

3、WebSocket建立连接客户端接收服务器的响应后，同样会解析请求头信息，取出请求信息中的Sec-WebSocket-Accept字段，并用服务器内部处理Sec-WebSocket-Key字段的算法处理之前发送的Sec-WebSocket-Key，把处理得到的结果与Sec-WebSocket-Accept进行对比，数据相同则表示客户端与服务器成功建立WebSocket连接，反之失败

传统HTTP客户端与服务器请求响应模式如下图所示



上图对比可以看出，相对于传统HTTP每次请求-应答都需要客户端与服务端建立连接的模式

WebSocket是类似Socket的TCP长连接通讯模式。一旦WebSocket连接建立后，后续数据都以帧序列的形式传输。

WebSocket与HTTP的关系

WebSocket就像HTTP一样，则是一个典型的应用层协议。

Socket是传输控制层接口，WebSocket是应用层协议。

相同点：

1. 都是一样基于TCP的，都是可靠性传输协议。
2. 都是应用层协议。

不同点：

1. websocket是双向通信协议，模拟Socket协议，可以双向发送或接受信息。HTTP是单向的。
2. websocket是需要浏览器和服务器握手进行建立连接的。而http是浏览器发起向服务器的连接，服务器预先并不知道这个连接。

WebSocket在建立握手时，数据是通过HTTP传输的。但是建立之后，在真正传输时候是不需要HTTP协议的。

在WebSocket中，只需要服务器和浏览器通过HTTP协议进行一个握手的动作，然后单独建立一条TCP的通信通道进行数据的传送。

为什么需要 WebSocket?

全双工通讯、替代轮询

可以做什么?

1.即时通讯

2.消息推送

总结

Socket是传输控制层接口，WebSocket是应用层协议。

与http关系

WebSocket就像HTTP一样，则是一个典型的应用层协议

2) Springboot集成WebSocket

WebSocket API 是 HTML5 标准的一部分，但这并不代表 WebSocket 一定要用在 HTML 中，或者只能在基于浏览器的应用程序中使用

1、引入起步依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
```

2、开启WebSocket支持端点

```
@Bean
public ServerEndpointExporter serverEndpointExporter() {
    return new ServerEndpointExporter();
}
```

3、创建server核心类

```
package com.itheima.websocket.server;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import org.apache.commons.lang.StringUtils;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.springframework.stereotype.Component;
```

```

import javax.websocket.OnClose;
import javax.websocket.OnMessage;
import javax.websocket.OnOpen;
import javax.websocket.Session;
import javax.websocket.server.PathParam;
import javax.websocket.server.ServerEndpoint;
import java.io.IOException;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * @Class: WebSocketServer
 * @Package com.itheima.websocket.server
 * @Description: 连接、消息管理
 * @Company: http://www.itheima.com/
 */
@ServerEndpoint("/ws/{userId}")
@Component
public class WebSocketServer {
    //日志
    static Log log = LogFactory.getLog(WebSocketServer.class);
    //在线数量
    private static final AtomicInteger onlineCount = new AtomicInteger(0);
    //处理客户端连接socket
    private static Map<String, WebSocketServer> websocketMap = new
ConcurrentHashMap<>();
    //会话信息
    private Session session;
    //用户信息
    private String userId = "";

    /*
     * @Description: 打开WebSokcet连接
     * @Method: onOpen
     * @Param: [userId, session]
     * @Update:
     * @since: 1.0.0
     * @Return: void
     */
    @OnOpen
    public void onOpen(@PathParam("userId") String userId, Session session) {
        //处理session和用户信息
        this.session = session;
        this.userId = userId;
        if (websocketMap.containsKey(userId)) {
            websocketMap.remove(userId);
            websocketMap.put(userId, this);
        } else {
            websocketMap.put(userId, this);
            //增加在线人数
            addOnlineCount();
        }
        try {
            //处理连接成功消息的发送
            sendMessage("Server>>>>远程webSocket连接成功");
        }
    }
}

```



```
log.info("用户" + userId + "成功连接, 当前的在线人数为" +
getOnlineCount());
```

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
}
```

```
/*
```

```
 * @Description: 关闭连接
 * @Method: onClose
 * @Param: []
 * @Update:
 * @since: 1.0.0
 * @Return: void
 *
```

```
 */
```

```
@OnClose
```

```
public void onClose() {
```

```
    if (websocketMap.containsKey(userId)) {
        websocketMap.remove(userId);
        subOnlineCount();
    }
```

```
    log.info("用户退出....");
}
```

```
}
```

```
public static synchronized void subOnlineCount() {
    websocketServer.onlineCount.decrementAndGet();
}
```

```
/*
```

```
 * @Description:消息中转
 * @Method: onMessage
 * @Param: [message, session]
 * @Update:
 * @since: 1.0.0
 * @Return: void
 *
```

```
 */
```

```
@OnMessage
```

```
public void onMessage(String message, Session session) {
```

```
    if (StringUtils.isEmpty(message)) {
        try {
```

```
            //解析消息
```

```
            JSONObject jsonObject = JSON.parseObject(message);
            String toUserId = jsonObject.getString("toUserId");
            String msg = jsonObject.getString("msg");
            if (StringUtils.isEmpty(toUserId) &&
```

```
websocketMap.containsKey(toUserId)) {
```

```
                websocketMap.get(toUserId).sendMessage(msg);
```

```
            }
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }
```

```
}
```



```

/*
 * @Description: 服务端向客户端发送数据
 * @Method: sendMessage
 * @Param: [s]
 * @Update:
 * @since: 1.0.0
 * @Return: void
 *
 */
public void sendMessage(String s) throws IOException {
    this.session.getBasicRemote().sendText(s);
}

/*
 * @Description: 获取在线人数的数量
 * @Method: getOnlineCount
 * @Param: []
 * @Update:
 * @since: 1.0.0
 * @Return: java.util.concurrent.atomic.AtomicInteger
 *
 */
public static synchronized AtomicInteger getOnlineCount() {
    return onlineCount;
}

/*
 * @Description: 增加在线人数
 * @Method: addOnlineCount
 * @Param: []
 * @Update:
 * @since: 1.0.0
 * @Return: void
 *
 */
public static synchronized void addOnlineCount() {
    websocketServer.onlineCount.incrementAndGet();
}

/*
 * @Description: 服务器消息推送
 * @Method: sendInfo
 * @Param: [message, userId]
 * @Update:
 * @since: 1.0.0
 * @Return: void
 *
 */
public static boolean sendInfo(String message, @PathParam("userId") String
userId) throws IOException {
    boolean flag=true;
    if (StringUtils.isEmpty(userId) && websocketMap.containsKey(userId))
    {
        websocketMap.get(userId).sendMessage(message);
    } else {
        log.error("用户" + userId + "不在线");
    }
}

```

```

        flag=false;
    }
    return flag;
}
}

```

Tips:

1、addOnlineCount和getOnlineCount为static；表示当前这个变量是有状态的（有状态：线程间资源不被隔离），多线程共享onlineCount

2、addOnlineCount和getOnlineCount为synchronized；表示当前这个方法如果单纯操作原子变量无需synchronized、如果方法体内使用非原子操作则无需synchronized、如果需要做方法同步需要synchronized

此处为基础内容

4、创建控制器

用户打开页面和服务端消息推送

```

@RequestMapping("im")
public ModelAndView page() {
    return new ModelAndView("ws");
}
/*
 * @Description: 消息推送
 * @Method: pushToWeb
 * @Param: [message, toUserId]
 * @Update:
 * @since: 1.0.0
 * @Return: org.springframework.http.ResponseEntity<java.lang.String>
 *
 */
@RequestMapping("/push/{toUserId}")
public ResponseEntity<String> pushToWeb(String message, @PathVariable String toUserId) throws Exception {

    boolean flag = WebSocketServer.sendInfo(message, toUserId);
    return flag == true ? ResponseEntity.ok("消息推送成功...") :
    ResponseEntity.ok("消息推送失败，用户不在线...");
}

```

5、创建消息发送HTML

```

<!DOCTYPE html>
<html>
<head>

```

```

<meta charset="utf-8">
<link href="../css/index.css" rel="stylesheet">

</head>
<script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.js"></script>
<script>
    var socket;
    function openWebSocket() {
        if (typeof(WebSocket) == "undefined") {
            console.log("对不起，您的浏览器不支持WebSocket");
        } else {
            var websocketUrl = "ws://localhost:9999/ws/im/" +
                $("#userId").val();
            if (socket != null) {
                socket.close();
                socket = null;
            }
            socket = new WebSocket(websocketUrl);
            //打开
            socket.onopen = function () {
                console.log("Client>>>>WebSocket已打开");
            };
            //获取消息
            socket.onmessage = function (msg) {
                console.log(msg.data);
                $("#msg").val(msg.data)
            };
            //关闭
            socket.onclose = function () {
                console.log("Client>>>>WebSocket已关闭");
            };
            //发生错误
            socket.onerror = function () {
                console.log("Client>>>>WebSocket发生了错误");
            }
        }
    }

    function sendMessage() {
        if (typeof(WebSocket) == "undefined") {
            console.log("对不起，您的浏览器不支持WebSocket");
        } else {
            socket.send('{"toUserId":"' + $("#toUserId").val() + '", "msg":"' +
                $("#msg").val() + '"}');
        }
    }
</script>

<body>
<div id="panel">
    <div class="panel-header">
        <h2>即时通讯IM</h2>
    </div>
    <div class="panel-content">
        <div class="user-pwd">
            <button class="btn-user">发</button>
            <input id="userId" name="userId" type="text" value="张三">

```

```

</div>
<div class="user-pwd">
  <button class="btn-user">收</button>
  <input id="toUserId" name="toUserId" type="text" value="李四">
</div>
<div class="user-msg">
  <input id="msg" name="msg" type="text" value="">
</div>
<div class="panel-footer">
  <button class="login-btn" onclick="openWebSocket()" >连接
WebSocket</button>
  <button class="login-btn" onclick="sendMessage()">发送消息</button>

</div>
</div>

</body>

</html>

```

6、单播运行效果如下

1、张三建立连接

<http://localhost:9999/ws/wspage/>

打开效果



建立WebSocket连接

即时通讯IM

发

张三

收

李四

Server>>>远程WebSocket连接成功

连接WebSocket

发送消息



黑马程序员™
www.itheima.com



博学谷
www.boxuegu.com

F12查看ws



1. GET请求的地址不是类似/http/，而是以ws://开头的地址；
2. 请求头Upgrade: websocket和Connection: Upgrade表示这个连接将要被转换为WebSocket连接；
3. Sec-WebSocket-Key是由浏览器随机生成的，是用于标识这个连接，并非用于加密数据；
4. Sec-WebSocket-Version指定了WebSocket的协议版本

Sec-WebSocket-Extensions:请求扩展

General是一个 http 请求以及响应，这里的 general 可以理解成为 http 请求 和 响应的请始行 的一个公共部分

(交换请求)



▼ Response Headers

[view source](#)

Connection: upgrade

Date: Fri, 26 Jun 2020 14:49:27 GMT

Sec-WebSocket-Accept: Sz+qgI3EyyvQF758KGkLSVH6F8CY=

Sec-WebSocket-Extensions: permessage-deflate;client_max_window_bits=15

Upgrade: websocket

响应代码101表示本次连接的HTTP协议即将被更改，更改后的协议就是Upgrade: websocket指定的WebSocket协议。

版本号 and 子协议规定了双方能理解的数据格式，以及是否支持压缩等等。如果仅使用WebSocket的API，就不需关心这些

2、李四建立连接

<http://localhost:9999/ws/wspage/>

首页打开效果



建立WebSocket连接



3、张三对话李四

即时通讯IM

发

张三

收

李四

你好，我是张三，请问您是李四吗？

连接WebSocket

发送消息



黑马程序员™
www.itheima.com



博学谷
www.boxuegu.com

查看李四窗口是否收到消息

即时通讯IM

发

李四

收

张三

你好，我是张三，请问您是李四吗？

连接WebSocket

发送消息



黑马程序员™
www.itheima.com



博学谷
www.boxuegu.com

4、李四回复张三

即时通讯IM

发

李四

收

张三

我是李四，请问您找我有什么事情吗？|

连接WebSocket

发送消息



黑马程序员™
www.itheima.com



博学谷
www.boxuegu.com

张三接收消息



5、服务端消息推送

<http://localhost:9999/ws/push/张三?message=618> 开宝箱领红包 超多聚划算爆款折上再五折

← → ↺ ① localhost:9999/ws/push/张三?message=618%20开宝箱领红包%20%20超多聚划算爆款折上再五折

消息推送成功...

接收消息



1.3.2 自签名证书HTTPS开发

1) SSL证书生成

密钥库

1) 生成keystore证书

JDK中keytool是一个证书管理工具，可以生成自签名证书

```
keytool -genkey -alias czbk -keypass 123456 -keyalg RSA -keysize 1024 -validity 365 -keystore c:/czbk.keystore -storepass 123456
```

下面内容可以全部一样

比如itheima

```
C:\Users\My>keytool -genkey -alias czbk -keypass 123456 -keyalg RSA -keysize 1024 -validity 365 -keystore c:/czbk.keystore -storepass 123456
您的名字与姓氏是什么?
[Unknown]: itheima
您的组织单位名称是什么?
[Unknown]: czbk
您的组织名称是什么?
[Unknown]: czbkLtd
您所在的城市或区域名称是什么?
[Unknown]: beijing
您所在的省/市/自治区名称是什么?
[Unknown]: beijing
该单位的双字母国家/地区代码是什么?
[Unknown]: zh
CN=itheima, OU=czbk, O=czbkLtd, L=beijing, ST=beijing, C=zh是否正确?
```

查看生成结果



名称	修改日期	类型	大小
czbk.keystore	2020/6/24 14:13	KEYSTORE 文件	2 KB
用户	2019/9/28 12:53	文件夹	

tips

后续使用，将生成文件复制到spring boot项目中的resources目录中

命令解释：

keytool

-genkey

-alias tomcat(别名)

-keypass 123456(别名密码)

-keyalg RSA(生证书的算法名称，RSA是一种非对称加密算法)

-keysize 1024(密钥长度，证书大小)

-validity 365(证书有效期，天单位)

-keystore c:/czbk.keystore(指定生成证书的位置和证书名称)

-storepass 123456(获取keystore信息的密码)

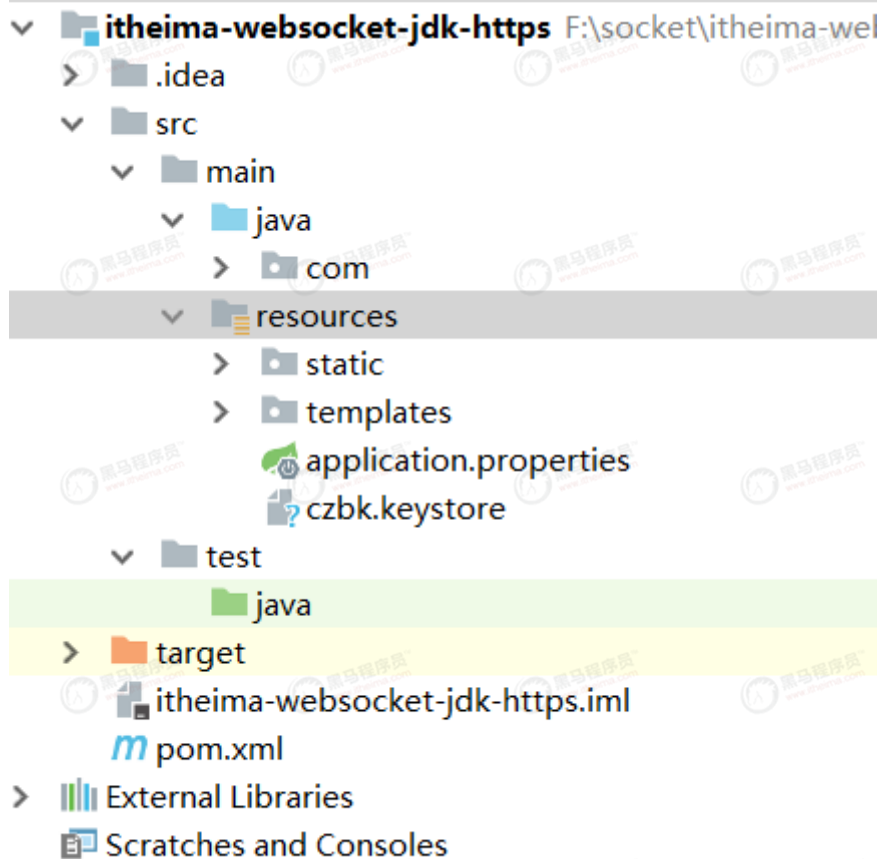
-storetype (指定密钥仓库类型)

2) springboot集成SSL

将xxx.keystore拷贝到项目src/java/resources目录下.配置application.properties

```
# 端口
http.port=7777
server.port=8888
# 指定签名文件, 对应生成的密钥库文件
server.ssl.key-store=classpath:czbk.keystore
# 指定签名密码, 设置的密钥库指令
server.ssl.key-store-password=123456
# 指定密钥仓库类型, JKS
server.ssl.key-store-type=JKS
# 指定别名, 生成密钥库的时候进行了设定
server.ssl.key-alias=czbk
```

2、将czbk.keystore放到classpath下面



3、增加配置类

```
package com.itheima.websocket.config;

import org.apache.catalina.Context;
import org.apache.catalina.connector.Connector;
import org.apache.tomcat.util.descriptor.web.SecurityCollection;
```

```

import org.apache.tomcat.util.descriptor.web.SecurityConstraint;
import org.springframework.beans.factory.annotation.Value;
import
org.springframework.boot.web.embedded.tomcat.TomcatServletWebServerFactory;
import org.springframework.boot.web.servlet.server.ServletWebServerFactory;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * @Class: HttpRedirectHttps
 * @Package com.itheima.websocket.config
 * @Description: http访问重定向https
 * @Company: http://www.itheima.com/
 */
@Configuration
public class HttpRedirectHttps {

    @Value("${http.port}")
    Integer httpPort;
    @Value("${server.port}")
    Integer httpsPort;

    /**
     * @Description: 配置一个TomcatServletWebServerFactory 的BEAN
     * 然后添加一个Tomcat中的Connector（监听80端口），并将请求转发到80端口
     * @Method: tomcatServletWebServerFactory
     * @Param: []
     * @Update:
     * @since: 1.0.0
     * @Return:
     org.springframework.boot.web.embedded.tomcat.TomcatServletWebServerFactory
     *
     */
    @Bean
    public TomcatServletWebServerFactory servletContainer() {
        TomcatServletWebServerFactory tomcat = new
TomcatServletWebServerFactory() {
            @Override
            protected void postProcessContext(Context context) {
                SecurityConstraint securityConstraint = new
SecurityConstraint();
                securityConstraint.setUserConstraint("CONFIDENTIAL");
                SecurityCollection collection = new SecurityCollection();
                collection.addPattern("/");
                securityConstraint.addCollection(collection);
                context.addConstraint(securityConstraint);
            }
        };
        tomcat.addAdditionalTomcatConnectors(createTomcatConnector());
        return tomcat;
    }

    /**
     * @Description:

```


* Spring Boot 不支持同时启动 HTTP 和 HTTPS ，为了解决这个问题，我们这里可以配置一个请求转发，当用户发起 HTTP 调用时，自动转发到 HTTPS 上

```
*
*
* @Method: createTomcatConnector
* @Param: []
* @Update:
* @since: 1.0.0
* @Return: org.apache.catalina.connector.Connector
*
*/
public Connector createTomcatConnector() {
    Connector connector = new
Connector("org.apache.coyote.http11.Http11NioProtocol");
    connector.setScheme("http");
    //Connector监听的http的默认端口号
    connector.setPort(httpPort);
    connector.setSecure(false);
    //监听到http的端口号后转向到的https的端口号,也就是项目配置的port
    connector.setRedirectPort(httpsPort);
    return connector;
}
}
```

CONFIDENTIAL:要求用使SSL

4、https安全验证

<http://localhost:7777/im>

跳转到

<https://localhost:8888/im>

谷歌

谷歌浏览器发现有红色警告，这是因为我们自己生成的 https 证书不被浏览器认可，不过没关系，我们直接点击继续访问就可以了（实际项目中只需要更换一个被浏览器认可的 https 证书即可）。



您的连接不是私密连接

攻击者可能会试图从 **localhost** 窃取您的信息（例如：密码、通讯内容或信用卡信息）。[了解详情](#)

NET-ERR_CERT_AUTHORITY_INVALID

☐ 将您访问的部分网页的网址、有限的系统信息以及部分网页内容发送给 Google，以帮助我们提升 Chrome 的安全性。[隐私权政策](#)

[隐藏详情](#)

[返回安全连接](#)

此服务器无法证明它是**localhost**；您的计算机的操作系统不信任其安全证书。出现此问题的原因可能是配置有误或您的连接被拦截了。

[继续前往localhost（不安全）](#)

警告：面临潜在的安全风险

https://localhost:8888/im

80%

...

☆

移动设备上的书签

警告：面临潜在的安全风险

Firefox 检测到潜在的安全威胁，因此没有继续访问 localhost。若您访问此网站，攻击者可能会尝试窃取您的密码、电子邮件、信用卡等信息。

[详细了解...](#)

[返回上一页（推荐）](#)[高级...](#)

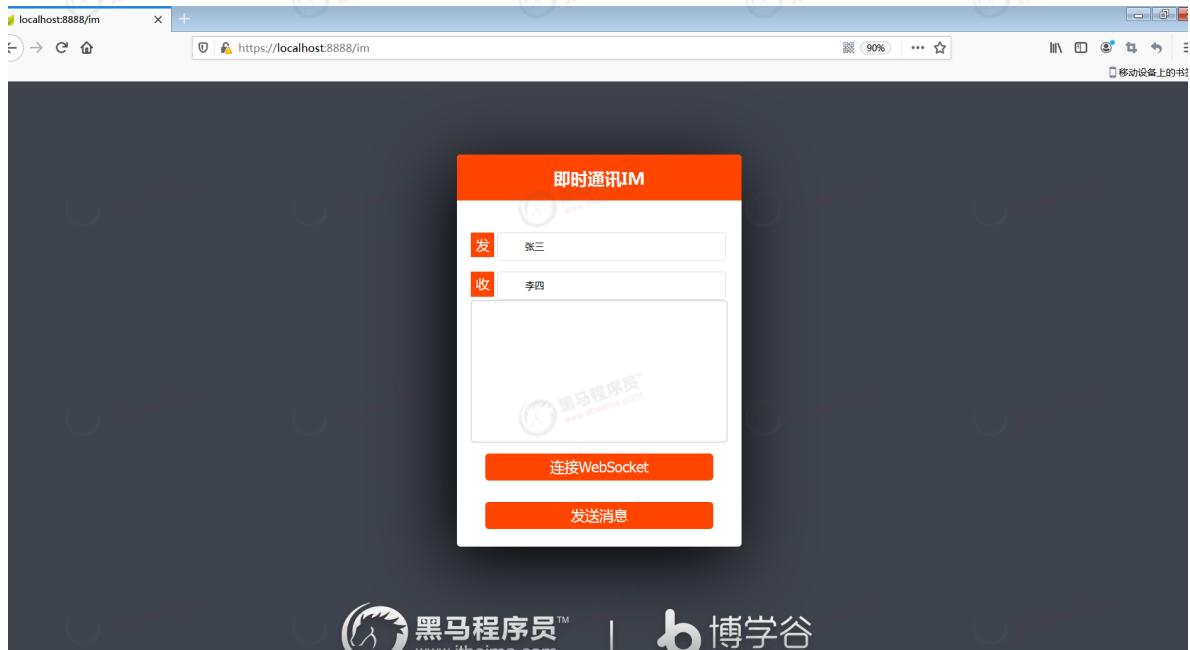
各个网站通过证书证明自己的身份。Firefox 不能信任此网站，它使用的证书对 localhost:8888 无效。

错误代码：MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT

[查看证书](#)

[返回上一页（推荐）](#)[接受风险并继续](#)

强制访问



https改造完毕后不能通讯



解决方案

修改前端代码

```
var websocketUrl = "wss://localhost:8888/ws/" + $("#userId").val();
```

连接成功



3) Nginx代理SSL

springboot程序不加入ssl任何配置

- 1、解决http重定向到https
- 2、解决Wss服务问题
- 3、关于Nginx版本问题，版本应该大于1.3

关于nginx

自1.3 版本开始, Nginx就支持 WebSocket, 并且可以为 WebSocket 应用程序做反向代理和负载均衡。WebSocket 和 HTTP 是两种不同的协议, 但是 WebSocket 中的握手和 HTTP 中的握手兼容, 它使用 HTTP 中的 Upgrade 协议头将连接从 HTTP 升级到 WebSocket, 当客户端发过来一个 Connection: Upgrade请求头时, 其实Nginx是不知道的。所以, 当 Nginx 代理服务器拦截到一个客户端发来的 Upgrade 请求时, 需要我们显式的配置Connection、Upgrade头信息, 并使用 101 (交换协议) 返回响应, 在客户端、代理服务器和后端应用服务之间建立隧道来支持 WebSocket。

1、启动nginx

```
start nginx
```

常见的错误:

(1)端口号被占用

(2)nginx文件夹路径含中文

查看任务进程是否存在, dos或打开任务管理器都行

```
tasklist /fi "imagename eq nginx.exe"
```

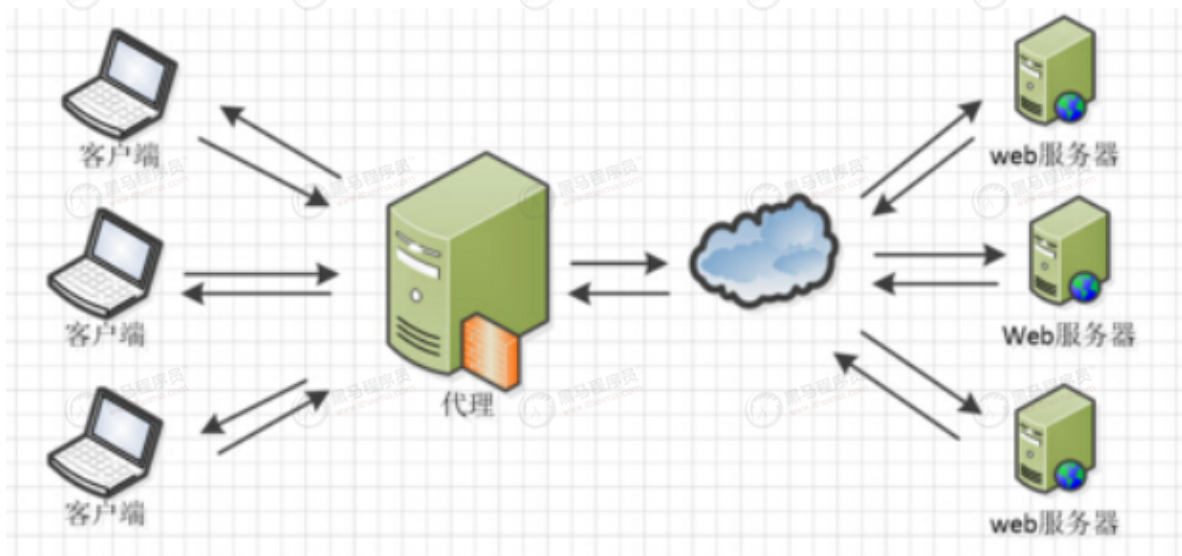
```
C:\nginx-1.4.7>tasklist /fi "imagename eq nginx.exe"
```

映像名称	PID	会话名	会话#	内存使用
nginx.exe	7628	Console	1	6,380 K
nginx.exe	9656	Console	1	6,680 K

```
C:\nginx-1.4.7>
```

使用 Nginx 反向代理来解决 WebSocket 的 wss 服务问题, 即客户端通过 Wss 协议连接 Nginx 然后 Nginx 通过 Ws 协议和 Server 通讯

也就是说 Nginx 负责通讯加解密, Nginx 到 Server 是明文的



```

#*****分割线*****
events {
    worker_connections 1024;
}
#*****分割线*****
http {
    map $http_upgrade $connection_upgrade {
        default upgrade;
        '' close;
    }
#*****分割线*****
    server {
        listen 80;
        server_name websocket.nginx.com;
        add_header Strict-Transport-Security max-age=15768000;
        return 301 https://$server_name$request_uri;

    }

#*****分割线*****
    server {
        listen 443 ssl;
        server_name websocket.nginx.com
        ssl on;
        ssl_certificate c:\czbk.crt;
        ssl_certificate_key c:\czbk.key;
        ssl_session_timeout 5m;
        ssl_protocols SSLv3 SSLv2 TLSv1 TLSv1.1 TLSv1.2;
        ssl_ciphers ECDHE-RSA-AES128-GCM-
        SHA256:ECDHE:ECDH:AES:HIGH:!NULL:!aNULL:!MD5:!ADH:!RC4;
        ssl_prefer_server_ciphers on;
#*****页面http访问的时候也可以使用/im*****
        #location /im/ {
            #proxy_pass http://localhost:8888/im;
        }

#*****http跳转与wss协议调用（自动判断请求协议）*****
        location / {
            proxy_pass http://localhost:8888;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "Upgrade";
            proxy_set_header X-Real-IP $remote_addr;
        }
    }
#*****分割线*****
}

```

map指令的作用：

#该作用主要是根据客户端请求中\$http_upgrade 的值，来构造改变\$connection_upgrade的值，即根据变量\$http_upgrade的值创建新的变量\$connection_upgrade，
#创建的规则就是{}里面的东西。其中的规则没有做匹配，因此使用默认的，即 \$connection_upgrade的值会一直是 upgrade。然后如果 \$http_upgrade为空字符串的话，那值会是 close。

核心配置

`ssl_certificate` 证书其实是个公钥，它会被发送到连接服务器的每个客户端

`ssl_certificate_key` 私钥是用来解密的，所以它的权限要得到保护但nginx的主进程能够读取。

`ssl_session_timeout`：客户端可以重用会话缓存中ssl参数的过期时间

ssl_protocols指令用于支持加密协议

`ssl_ciphers`选择加密套件，不同的浏览器所支持的套件（和顺序）可能会不同。这里指定的是OpenSSL库能够识别的写法，你可以通过 `openssl -v cipher 'RC4:HIGH:!aNULL:!MD5'`（后面是你所指定的套件加密算法）来看所支持算法。

加密套件 之间用冒号分隔，加密套件 前有感叹号的表示必须废弃。

`ssl_prefer_server_ciphers on`设置协商加密算法时，优先使用我们服务端的加密套件，而不是客户端浏览器的加密套件

访问流程：

- 1、<http://websocket.nginx.com/im/>访问进入到 listen 80;
- 2、接着跳转到listen 443
- 3、进入到443后分别（wss或者http）进行跳转

重载 Nginx 服务

主流数字证书都有哪些格式？

一般来说，主流的web服务软件，通常都基于两种基础密码库：OpenSSL和Java。

Tomcat、Weblogic、JBoss等，使用Java提供的密码库。通过Java的Keytool工具，生成Java Keystore（JKS）格式的证书文件。

Apache、Nginx等，使用OpenSSL提供的密码库，生成PEM、KEY、CRT等格式的证书文件

此处我们需要crt和key文件

我们需要将证书keystore导出crt和key

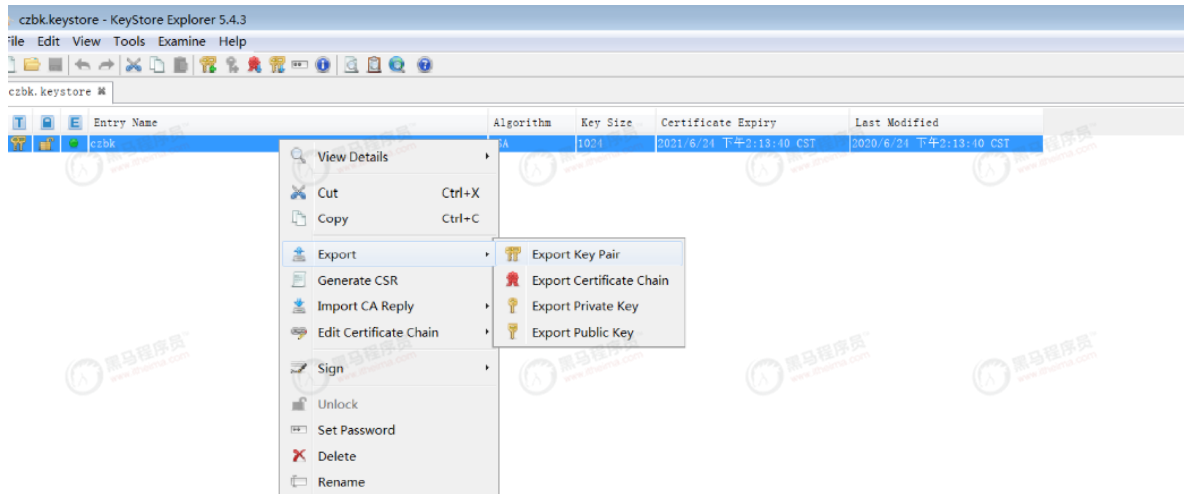
- 1、使用kse-543转p12
- 2、通过P12生成crt和key文件

2、安装kse-543-setup

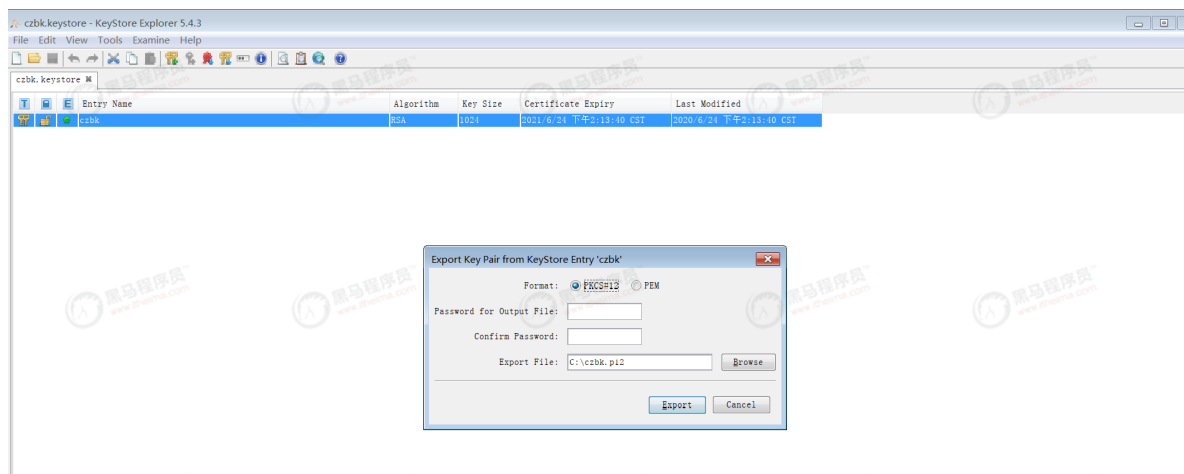
目的：将keystore转p12

kse-543-setup.exe	2020/6/25 15:33	应用程序	10,576 KB
netassist (1).zip	2020/6/16 14:12	360压缩 ZIP 文件	462 KB
nginx-1.4.7.zip	2020/6/24 22:07	360压缩 ZIP 文件	1,198 KB
WinPcap_4_1_3.exe	2020/6/19 14:33	应用程序	894 KB
Wireshark-win64-3.0.6.exe	2020/6/12 15:09	应用程序	57,887 KB

桌面打开快捷方式，导入keystore

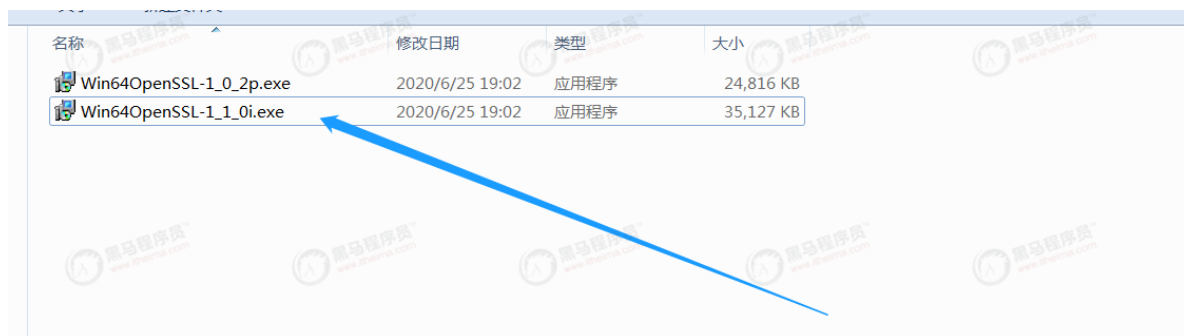


生成p12



通过P12生成crt和key文件(Nginx使用openssl的crt和key)

1、安装Win64OpenSSL-1_1_0i



3、设置配置文件

```
set RANDFILE=C:\OpenSSL-win64\.rnd  
  
set OPENSSL_CONF=C:\OpenSSL-win64\bin\openssl.cfg
```

4、进入C:\OpenSSL-Win64\bin

打开cmd窗口 查看版本

```
openssl version -a
```

5、生成key文件

无密码直接回车

```
openssl pkcs12 -in c:\czbk.p12 -nocerts -nodes -out c:\czbk.key
```

```
管理员: C:\Windows\system32\cmd.exe
C:\OpenSSL-Win64\bin>openssl version -a
OpenSSL 1.1.0i 14 Aug 2018
built on: reproducible build, date unspecified
platform: UC-MINGW64
options: bn(64,64) rc4(16x,int) des(long) idea(int) blowfish(ptr)
compiler: cl
OPENSSLDIR: "C:\Program Files\Common Files\SSL"
ENGINESDIR: "C:\Program Files\OpenSSL\lib\engines-1_1"
C:\OpenSSL-Win64\bin>openssl pkcs12 -in c:\czbk.p12 -nocerts -nodes -out czbk.key
Enter Import Password:
```

6、生成crt文件

```
openssl pkcs12 -in c:\czbk.p12 -nokeys -out c:\czbk.crt
```

```
C:\OpenSSL-Win64\bin>
C:\OpenSSL-Win64\bin>openssl pkcs12 -in c:\czbk.p12 -nokeys -out czbk.crt
Enter Import Password:
C:\OpenSSL-Win64\bin>
```

7、修改hosts文件

C:\Windows\System32\drivers\etc

```
127.0.0.1      websocket.nginx.com
```

9、修改前端和配置文件

```
var websocketUrl = "wss://websocket.nginx.com/ws/" + $("#userId").val();
```

配置文件端口修改

```
server.port=8888
```

10、常见问题

nginx报错

```
[emerg] 4920#200: bind() to 0.0.0.0:443 failed (10013: An attempt was made to access a socket in a way forbidden by its access permissions)
```

端口占用，通过下面命令查看

```
netstat -ano|findstr "443"
```

nginx报错

```
2020/06/26 13:10:31 [alert] 7524#3004: the event "ngx_master_7524" was not signaled for 5s
```

去除ssl密码

nginx中还是c:/czbk.key

```
openssl rsa -in c:/czbk.key -out c:/czbk.key.unsecure
```

11、访问验证

tips

删除springboot和ssl相关所有的代码，包含证书

http重定向到https

```
http://websocket.nginx.com/im/
```

▲ 不安全 | <https://websocket.nginx.com/im/>



开始建立websocket连接

▲ 不安全 | websocket.nginx.com/im/



4) TLS通信流程之抓包分析

张三发送给李四数据【wss加密传输测试==AAAA==12345】

```
https://websocket.nginx.com/im/
```

nginx设置https、代理及websocket连接60s自动关闭

proxy_read_timeout , websocket的有效时间，默认60s，可以设置大点

张三登录并发送



李四接收

即时通讯IM

发 李四

收 张三

wss加密传输测试==AAAA==12345

连接WebSocket

发送消息



黑马程序员™
www.itheima.com



博学谷
www.boxuegu.com

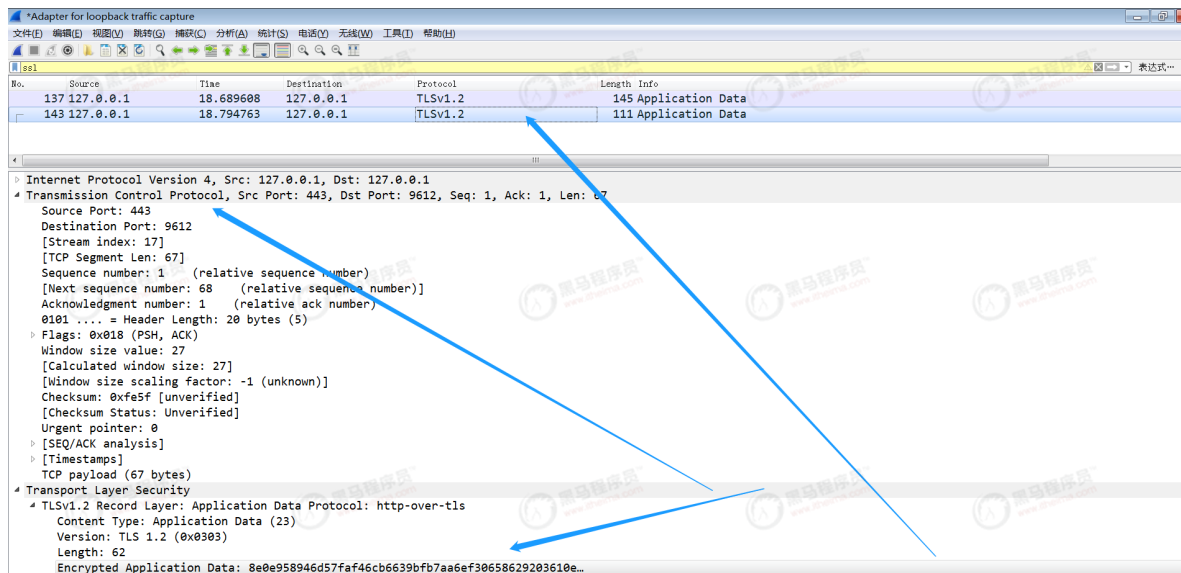
抓包分析

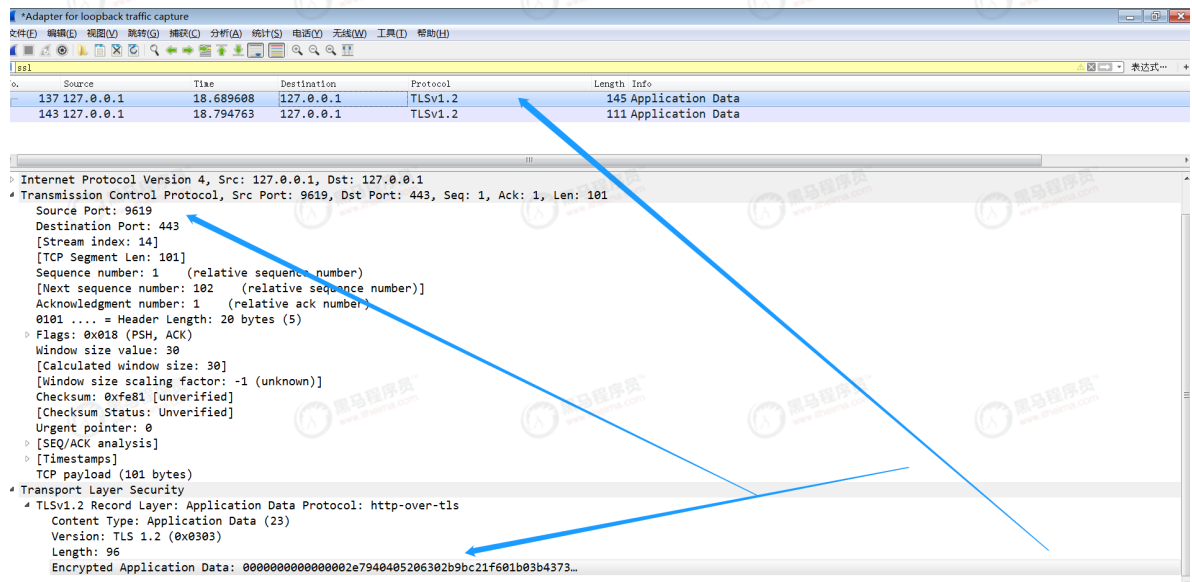
过滤

ssl

Application Data

应用层的数据，是加密的，使用密钥交换协议协商出来的密钥加密。因为我们的Wireshark不知道服务器的私钥，所以这段通信是安全的，我们在Wireshark中也无法解密这段信息





1.3.3 阿里云HTTPS开发

准备工作

- 阿里云ecs一个
- 域名一个 (www.itheima.cloud)
- ca证书一份 (用来支持https) (需要绑定域名)
- 本地打包好的springboot项目 (需要打包上传到阿里云)
- ftp客户端一个, 用来把jar传到阿里云服务器上

1) 阿里云域名申请

<https://mi.aliyun.com/>

快速搜索：不限 纯数字 2数字 3数字 4数字 5数字 6数字 顺子 | 纯字母 2字母 3字母 4字母 5字母 | 双声 三声 四声 五声 | 单拼

双拼 三拼 四拼 | 杂米 2杂 3杂

包含：多个关键词用逗号“,”隔开；如a,b,c ☐ 开头 ☐ 结尾 | 排除：排除关键词 ☐ 开头 ☐ 结尾 +

后缀：请填写后缀用逗号隔开 **不限** ☐ .xin ☐ .com ☐ .cn ☐ .net ☐ .com.cn ☐ .vip ☐ .top ☐ .cc ☐ .shop ☐ .club ☐ .wang

价格：不限 **0-50** 50-100 100-500 500-1000 1000-3000 3000-5000 5000-10000 10000以上 输入数字： —

长度：**不限** 1个 2个 3个 4个 5个 6个 7个 8个 9个 10个以上 输入数字： —

交易类型：**不限** [域名抢注](#) [万网预释放](#) [万网预订](#) [一口价（优选）](#) [一口价（万网）](#) [议价域名](#) [精品域名](#) 可用时间(天)： —

[保留词一口价](#) [违约认购](#) [竞价域名](#)

组成：

我的搜索：[\[保存当前搜索条件\]](#)

共有**357109**个搜索结果（最多显示5000个结果）

域名	当前价格	出售剩余时间	到期时间	交易类型	操作
☐ e3m28n.cn	¥1	46秒	2020-09-11	一口价（万网）	加入购物车 立即购买 查看详情
☐ c4414n.cn	¥1	46秒	2020-09-11	一口价（万网）	加入购物车 立即购买 查看详情
☐ 11975.top	¥5	46秒	2020-11-01	一口价（万网）	加入购物车 立即购买 查看详情

选择好的域名直接购买

[域名交易](#) > wrdgpea.cn

wrdgpea.cn

[whois](#) | [百度](#) | [google](#) | [微博分享](#) | [微信分享](#)

2020-06-22 16:44:46发布 已浏览0次

米店：[老妖域名专卖Q191039399微信同号](#)(ID:10633)

价格：¥4

出售剩余时间：18秒

交易类型：一口价（万网）

数据来源：阿里云 / 万网

域名注册时间：[查询 whois](#)

域名过期时间：2021-01-02

出售发布时间：2020-06-22 16:44:46

出售到期时间：2020-06-23 14:04:00

介绍：特价cn域名,大量过V信2级不死,Q2级防红域名,低价出阿里/腾讯/华为/景安等实名认证被案域名,请进o91.cn或+Q号191039399_V信同号 [\[举报\]](#)

[立即购买](#)[加入购物车](#)

支付成功

支付完成

填写域名信息

核对确认订单信息

去支付

完成



恭喜，域名支付成功！

您可以到[我是买家](#)查看已买到的域名或者前往[域名交易市场](#)继续购买其他域名

- 提示
- 1、万网仅为域名交易的第三方担保平台，对在平台上发生的域名交易不提供购买发票；
 - 2、域名购买成功后，请尽快查看域名到期时间，注意及时续费。

域名	价格	购买结果
itheima.cloud	11.00	成功

查看域名状态

[https://dc.console.aliyun.com/next/index?](https://dc.console.aliyun.com/next/index?spm=5176.100251.recommends.ddomain.6ffe4f15tozyLa#/domain/details/info?saleId=DT49H3EX462ZAYP&domain=itheima.cloud)
[spm=5176.100251.recommends.ddomain.6ffe4f15tozyLa#/domain/details/info?](https://dc.console.aliyun.com/next/index?spm=5176.100251.recommends.ddomain.6ffe4f15tozyLa#/domain/details/info?saleId=DT49H3EX462ZAYP&domain=itheima.cloud)
[saleId=DT49H3EX462ZAYP&domain=itheima.cloud](https://dc.console.aliyun.com/next/index?spm=5176.100251.recommends.ddomain.6ffe4f15tozyLa#/domain/details/info?saleId=DT49H3EX462ZAYP&domain=itheima.cloud)

<

基本信息 / itheima.cloud

如何修改域名信息

基本信息

域名持有者过户

域名信息修改

域名持有者实名认证

DNS 修改

自定义DNS Host

域名转出

安全设置

域名证书下载

域名解析

账号间转移

带价PUSH

域名持有者(中文):

域名持有者(英文):

联系人邮箱:

姓名网址应用:

持有者实名认证:

实名认证证件类型:

实名认证证件号码:

注册商:

注册日期:

到期日期:

域名状态:

DNS服务器:

注册局安全锁:

SSL证书:

备注:

更改 (过户)

@qq.com 更改 (过户) 验证通过

认证成功

身份证

2

Alibaba Cloud Computing Ltd. d/b/a AliChina (www.net.cn) (阿里云计算有限公司) 索取域名转移码

2019-12-03 18:12:33

2020-12-03 18:12:33 续费

查看 whois

修改 DNS

未开启 购买

开启SSL证书

whois结果

域名 **itheima.cloud** 的注册信息 ?

以下信息获取时间：2020-06-22 22:56:14

[获取最新信息](#)

"itheima"还有**商标**可以注册，为防止被他人占用，建议立即注册进行保护！>>[注册商标](#)

所有者联系邮箱

[如要进行联系，请在线填写信息](#)

[了解更多](#)

[委托阿里云购买](#)

Registrant E-mail

注册商

Alibaba Cloud Computing Ltd. d/b/a HiChina (www.net.cn)

Sponsoring Registrar

注册日期

2019年12月03日

Registration Date(UTC)

到期日期

2020年12月03日

Expiration Date(UTC)

2020年12月03日前，域名可正常使用。请在2020年12月03日前及时续费延期。

域名状态
Domain Status

正常状态 (ok) ?
<https://icann.org/epp#ok>

DNS服务器
Name Server

DNS1: dns30.hichina.com
DNS2: dns29.hichina.com

2) 阿里云ECS服务器申请

[https://www.aliyun.com/activity/618/index?](https://www.aliyun.com/activity/618/index?spm=5176.12825654.a9ylfrljh.d111.e9392c4aU65uab&scm=20140722.2188.2.2170)

[spm=5176.12825654.a9ylfrljh.d111.e9392c4aU65uab&scm=20140722.2188.2.2170](https://www.aliyun.com/activity/618/index?spm=5176.12825654.a9ylfrljh.d111.e9392c4aU65uab&scm=20140722.2188.2.2170)

The screenshot shows the Alibaba Cloud 618 promotion page for ECS instances. The page features a top banner with the 618 logo and various promotional messages. Below the banner, there are several product cards for different ECS instance types, each with a detailed description, specifications, and pricing. The cards are arranged in a grid, and each card has a '立即购买' (Buy Now) button. The page also includes a sidebar with navigation links and a shopping cart icon in the bottom right corner.

产品系列	规格	配置	价格 (元/月起)
ECS计算型c6	2核4G	1M带宽, 40G高效云盘	¥1948.20
ECS通用型g6	2核8G	1M带宽, 40G高效云盘	¥2393.40
ECS计算型c5	2核4G	1~10M带宽, 40~100G可透云盘	¥699.84
ECS突发性能t5	1核2G	1~5M带宽, 40G高效云盘	¥89.76

ECS突发性能 t5 [更多配置购买](#)



地域: 华北 2（北京） 华北 2 可用区 H

实例: 1 CPU 2G 内存
ecs.t5-lc1m2.small

已选实例：ecs.t5-lc1m2.small 1核 2GB, 突发性能实例 t5 I/O 优化实例

突发性能实例：☐ 打开 t5 实例无性能约束模式 [?](#)

ⓘ不适用于长时间超过性能“基线”或企业稳定计算性能需求场景

操作系统: CentOS 7.4 64位

系统盘: 高效云盘 40 GB

网络类型: 专有网络

默认专有网络 默认交换机

如需使用其他专有网络，请选择已有专有网络，也可以自行到控制台创建>

带宽: 按固定带宽 1 Mbps

购买时长: 1 年

数量: 1

总配置费用

¥89.76 已省 ¥930.24 [优惠说明](#)

[加入购物车](#)

[立即购买](#)

确认订单

我的订单

产品名称	配置详情	付费方式	购买周期	数量	优惠	资费
云服务器ECS(包月)						优惠券 无
云服务器ECS(包月)	带宽：1Mbps, 按固定带宽 I/O 优化实例：I/O 优化实例 实例：1核 2GB 系统盘：高效云盘, 40GB 可用区：华北 2 可用区 H 地域：华北 2 网络类型：专有网络 操作系统：CentOS 7.4 64位	预付款	1年	1	¥930.24	¥89.76

☒ 我已阅读并同意 [《云服务器ECS\(包月\) 服务协议》](#)

总配置费用

¥89.76 节省 ¥930.24

[确认购买](#)

支付

回到旧版本支付

待支付订单

应付：¥89.76

订单号	产品	配置	数量	时长	总额
206267236570097	云服务器ECS(包月)	实例 突发性能实例 t5 系列 IV 1核 2GB I/O 优化实例 I/O 优化。 详情	1	1 年	¥89.76

其他方式支付

支付：¥89.76

支付宝

个人网银

企业网银

云分账

线下汇款

支付宝

蚂蚁花呗

实付金额 ¥89.76

支付

支付成功，查看ECS实例

<https://ecs.console.aliyun.com/?spm=5176.2020520132.products-grouped.decs.a1597218YHc0M5#/server/region/cn-beijing>

远程登录--重置密码

阿里云服务器购买之后，新的实例需要设置root登录密码之后才能正常操作，不然就登录不了。重置实例登录密码的时候，适用于在新创建时未设置密码或者忘记密码的情况。对于正在运行的实例，需要在重置实例登录密码之后重启实例才能使新的密码生效

ecs.console.aliyun.com/#/server/region/cn-beijing

阿里云 华北2（北京）

搜索文档、控制台、API、解决方案和资源

费用 工单 备案 企业 支持 官网

实例使用须知 创建实例

云服务器 ECS

概览

事件

标签

自动化部署 ROS

容器服务 Kubernetes

实例与镜像

实例

弹性容器实例 ECI

专有宿主机 DDH

实例列表

选择实例属性项搜索，或者输入关键字识别搜索

高级搜索

实例ID/名称	标签	监控	可用区	IP地址	状态	网络类型	配置	付费方式	操作
i-Zze9346f3ag0y4l3jg9c iZze9346f3ag0y4l3jg9cZ			华北2 可用区 H	101.201.232.138(公) 172.17.143.149 (私有)	运行中	专有网络	1 vCPU 2 GiB (I/O优化) ecs.t5-1c1m2.small 1Mbps	包年包月 2021年6月22日 23:59 到期	管理 续费 升级配置 更多

启动

停止

重启

重置实例密码

续费

按量付费转包年包月

释放设置

更多

共有1条，每页显示：20 条

重置密码

实例列表

[实例使用须知](#)[创建实例](#)

选择实例属性项搜索，或者输入关键字识别搜索									
高级搜索									
实例ID/名称	标签	监控	可用区	IP地址	状态	网络类型	配置	付费方式	操作
☐ i-2ze9346f3qg0yk13jg9c iZ2ze9346f3qg0yk13jg9cZ			华北 2 可用区 H	101.201.232.138(公) 172.17.143.149 (私有)	运行中	专有网络	1 vCPU 2 GiB (I/O优化) ecs.t5-ic1m2.small 1Mbps	包年包月 2021年6月22日 23:59 到期	管理 远程连接 升降配 续费 更多
☐ 启动 停止 重启 重置实例密码 续费 按量付费转包年包月 释放设置 更多									

重置实例密码

修改远程连接 (VNC) 密码

执行密码重置

重置密码 [重置密码须知](#)

提示：

- 重置实例密码后，只有在控制台重启实例才能生效。
- 如果实例已绑定密钥对，重置密码后系统可以通过新密码进行登录。
- 如果Windows实例默认管理员账号administrator被修改，则重置密码将会失败。

用户名：root (Linux) / administrator (Windows)

*新密码：

.....

8-30个字符，必须同时包含下面四项中的三项：大写字母、小写字母、数字、和特殊字符（仅支持下列特殊字符：() ` ~ ! @ # \$ % ^ & * - _ + = | { } [] ; ' < > , . ? / ）。其中，Windows 实例不能以斜线号 (/) 为密码首字符。

*确认密码：

.....|

您所选的 1 个实例 将执行重置密码操作，您是否确定操作？

提交

取消

手机验证



您绑定的手机：155****3206 ([更换手机](#))

* 校验码：

点击获取

确定

取消

提示

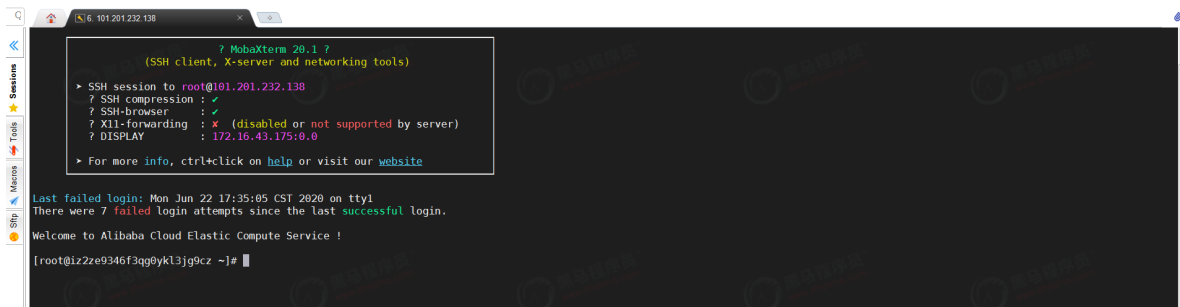


重置实例密码后，只有在控制台重启实例才能生效。

立即重启

取消

通过外网IP进行连接



3) 阿里云SSL证书申请

<https://common-buy.aliyun.com/?spm=5176.7968328.1266638..213d1232uEXCSm&commodityCode=cas#/buy>

1、选择SSL证书（应用安全）



2、购买证书



3、购买免费域名

单域名---DV/SSL---免费版

商品类型

云盾证书服务（包年）

证书扩展服务

选择域名类型

通配符域名（推荐）

单域名

多域名

选择证书类型

OV SSL

DV SSL

EV SSL

选择加密等级

基础版

免费版

选择证书品牌

DigiCert

保护一个单页面网站，例如 domain.com、ssl.domain.com、ssl.ssl.domain.com，如需要保护同根下的所有子域名，请购买通配符证书。
仅Globalsign OVI证书支持IP证书，其他品牌的单域名证书暂时不支持IP证书

通过验证您的域名，为您颁发证书，保护客户数据安全，同时可提高您网站的搜索排名。
DV SSL是企业/个人为**纯信息展示类网站**获得公信力的基础保证，拥有它才意味着您的**网站所有权已经过严格审查**。
如您的域名含有.edu、.gov、.org、.jp（国家缩写）、.pay、.bank、.live、.nuclear等**特殊词**，签发机构需要人工严格验证您的身份，为避免签发失败，**请直接**

仅供**个人或企业测试**所用，扩展性较差且具有一定的安全隐患。
不支持IP证书，IP证书支持仅Globalsign OV单域名证书
每个UID仅支持签发20张DV单域名证书，可以购买DV基础版获得更多的DV单域名证书

DigiCert是业界最受信任且广受赞誉的高信誉度证书提供商

总配置费用

¥0.00

立即购买

4、开始支付

云盾证书服务(包年)

配置详情	支付方式	数量	时长	折扣	价格
选择域名类型：单域名 选择证书类型：DV SSL 选择加密等级：免费版 选择证书品牌：DigiCert 选择域名个数：1个 证书托管服务：未开通 时长：1年	预付款	1	1年	¥0.00	配置费用 ¥0.00

服务协议

☒ 我已阅读并同意《云盾证书服务(包年)服务协议》

猜您喜欢（可选）

+ 云安全中心（ECS贴身安全防护，提供漏洞管理、病毒主动查杀、安全告警、基线检查、资产统一管理等功能）

+ 负载均衡（高性能，高可用，从容应对高并发场景）

应付款：¥0.00

去支付

5、支付成功

支付

确认订单

支付

支付完成

恭喜，支付成功！

请登录控制台，申请证书，签发成功后，可安装到网站/APP/邮箱等使用。

证书控制台

第一次用证书？您可能需要人工配置/安装服务

6、支付成功后在[未签发]

未签发

证书	绑定域名	已部署	状态	操作
DigiCert 免费版 SSL 实例: cas-cn-oew1pi6nk1r4 有效期限: 1 年 标签:未设置标签		--	已付款	证书申请 详情 升级 删除

7、点击证书申请，进行【证书与域名绑定】

证书申请

填写申请

验证信息

证书绑定域名

www.itheima.cloud

请输入完整的单个域名，域名格式例如：www.aliyun.com，IP证书仅Globalsign的OV单域名证书支持。

申请人姓名

申请人手机号

1

申请人邮箱

下一步

取消

证书申请



☒ 自动DNS验证

如果您绑定的域名在本账号下，可以使用自动验证。

☐ 手工DNS验证

需要登录域名管理控制台，将域名验证信息配置到域名解析列表中

☐ 文件验证

在没有域名管理权限的情况下，可以将TXT上传到域名服务器进行验证

CSR生成方式

i CSR文件是您的公钥证书原始文件，包含了您的服务器信息和您的单位信息，需要提交给CA认证中心审核。**建议您使用系统生成的CSR，避免因内容不正确而导致的审核失败。**

☒ 系统生成

☐ 手动填写

下一步

取消

点击下一步

证书申请



如果域名在阿里云，由域名管理员登录到域名管理控制台（dns.console.aliyun.com），按照以下提示信息添加到解析配置中，该验证记录在证书签发后才能删除，否则会因没有解析记录导致证书签发失败。

配置项目	配置项值
域名授权验证类型	DNS
记录类型	TXT ?
主机记录	_dnsauth ?
记录值	2020062100000033r1svcqofysifvfjc3wbbkswnczk74brdm6b9zdea2hkt2lf ?

验证

✓ 域名验证成功，域名验证记录在证书签发后再删除，否则会因没有解析记录导致证书签发失败。

提交审核

上一步

取消

提示

域名控制台添加的域名解析记录/上传域名所在服务器的验证文件，需要在证书签发成功后再删除，否则会因为缺少域名验证解析记录导致证书签发失败。

确定

提示

已经成功提交到CA公司，请您保持电话畅通，并及时查阅邮箱中来自CA公司的电子邮件。

确定

已签发

证书	绑定域名	已部署	到期时间	状态	操作
cert-3885227 DigiCert 免费版 SSL 实例: cas-cn-oww1pi6nk1r4 有效期限: 1 年 标签: 未设置标签	www.itheima.cloud itheima.cloud	--	2021年6月23日	已签发	详情 部署 下载 吊销 证书托管

SSL证书

概览

托管服务

资产中心

一键式HTTPS

联系人管理

证书总数 1

待申请的证书 0

已签发的证书 1

即将过期的证书 0

[购买证书](#) [上传证书](#) 全部状态 1 全部品牌

> 未签发

> 已签发

证书	绑定域名	已部署	到期时间
cert-3885227 DigiCert 免费版 SSL 实例: cas-cn-oww1pi6nk1r4 有效期限: 1 年 标签: 未设置标签	www.itheima.cloud itheima.cloud	--	2021年6月23日

证书下载

请根据您的服务器类型选择证书下载：

服务器类型	操作
Tomcat	帮助 下载
Apache	帮助 下载
Nginx	帮助 下载
IIS	帮助 下载
其他	帮助 下载

一键式HTTPS服务
不用安装证书，不用纠结各种安全套件的选择，不用担心私钥泄露，一键式HTTPS服务帮助您解决网站HTTPS问题。 [立即使用](#)

4) 域名与ECS服务器绑定

```
C:\Users\My>ping www.itheima.cloud
正在 Ping www.itheima.cloud [101.201.232.138] 具有 32 字节的数据:
来自 101.201.232.138 的回复: 字节=32 时间=6ms TTL=52
来自 101.201.232.138 的回复: 字节=32 时间=6ms TTL=52
来自 101.201.232.138 的回复: 字节=32 时间=6ms TTL=52
来自 101.201.232.138 的回复: 字节=32 时间=5ms TTL=52

101.201.232.138 的 Ping 统计信息:
数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 5ms, 最长 = 6ms, 平均 = 5ms
```

<https://dns.console.aliyun.com/?spm=5176.100251.111252.22.3a8b4f15HNfTsl#/dns/setting/itheima.cloud>

DNS解析:

新手引导

X

云解析DNS就是指把网站域名或应用资源转换为数字 IP地址，从而将最终用户的访问路由到相应的网站或应用资源上。

网站解析

邮箱解析

主机记录: 同时添加'www'及'@'记录

* 记录值:

添加成功后，您可以通过以下域名访问您的网站：

www.itheima.cloud

itheima.cloud

取消

确定

点击确定后出现www和@

← 解析设置 itheima.cloud

[② 如何设置解析](#)

✓ 当前分配的DNS服务器是：dns30.hichina.com, dns29.hichina.com

添加记录

导入/导出

请求量统计

新手引导

全部记录 ▾

精确搜索 ▾

输入关键字

Q

高级搜索 ▾

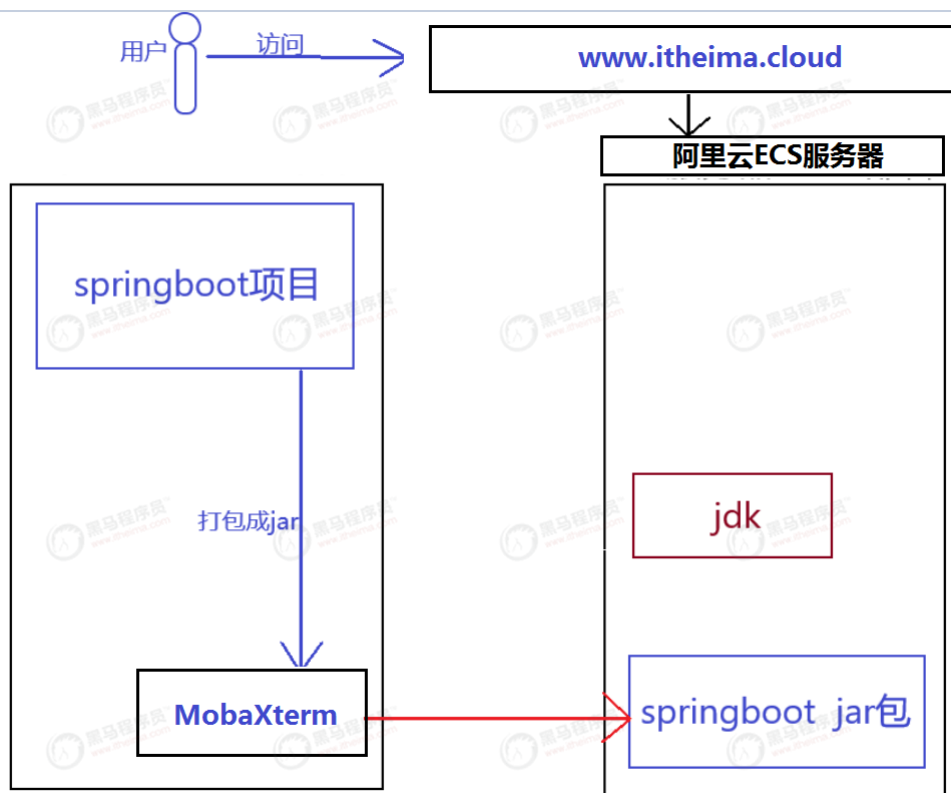
☐	主机记录	记录类型	解析线路(isp)	记录值	TTL	状态	备注	操作
☐	www	A	默认	101.201.232.138	10分钟	正常		修改 暂停 删除 备注
☐	@	A	默认	101.201.232.138	10分钟	正常		修改 暂停 删除 备注
☐	_dnsauth	TXT	默认	2020062100000033r1svcofyisfvafjc3wbbkswnczk74brdm6b9zdea2hkt2lf	10分钟	正常		修改 暂停 删除 备注

☐ 暂停 启用 删除 更换分组

共3条 < 1 > 10条/页 ▾

5) SpringBoot部署到阿里云

1、阿里云远程部署



1、配置文件修改

将证书文件复制到resources目录下

注意证书路径要带上 classpath:

```
http:
  port: 7777
server:
  # 端口 使用HTTPS默认端口
  port: 443
  #HTTPS加密配置
  ssl:
    #证书路径
    key-store: classpath:4108720_www.itheima.cloud.pfx
    #证书密码
    key-store-password: 5nLB0ixq
```

centos7 下杀死占用端口的进程

```
# 根据端口号得到其占用的进程的详细信息
netstat -lnp|grep 443

# 查看进程的详细信息
ps "pid"

#直接杀掉占用端口的进程    -9是强制关闭
kill -9 "pid"

或者

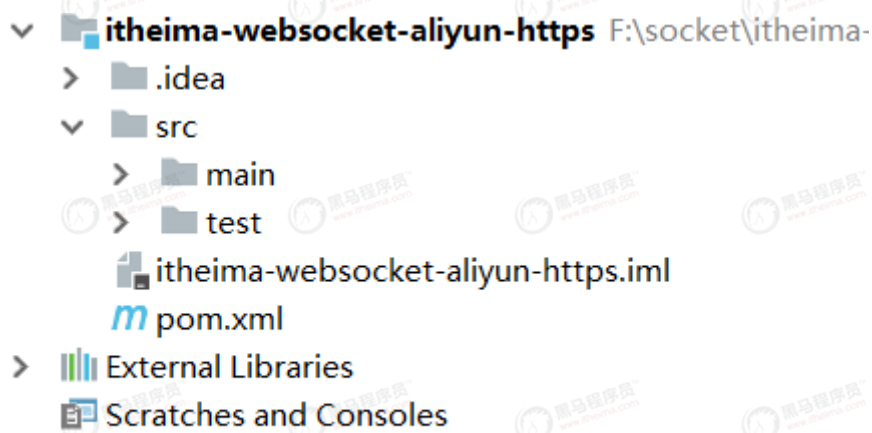
netstat -tunlp|grep 443

jobs 查看nohup运行的程序

ps -ef | grep java

kill- 9 id
```

- 将springboot项目打包（使用idea的package）

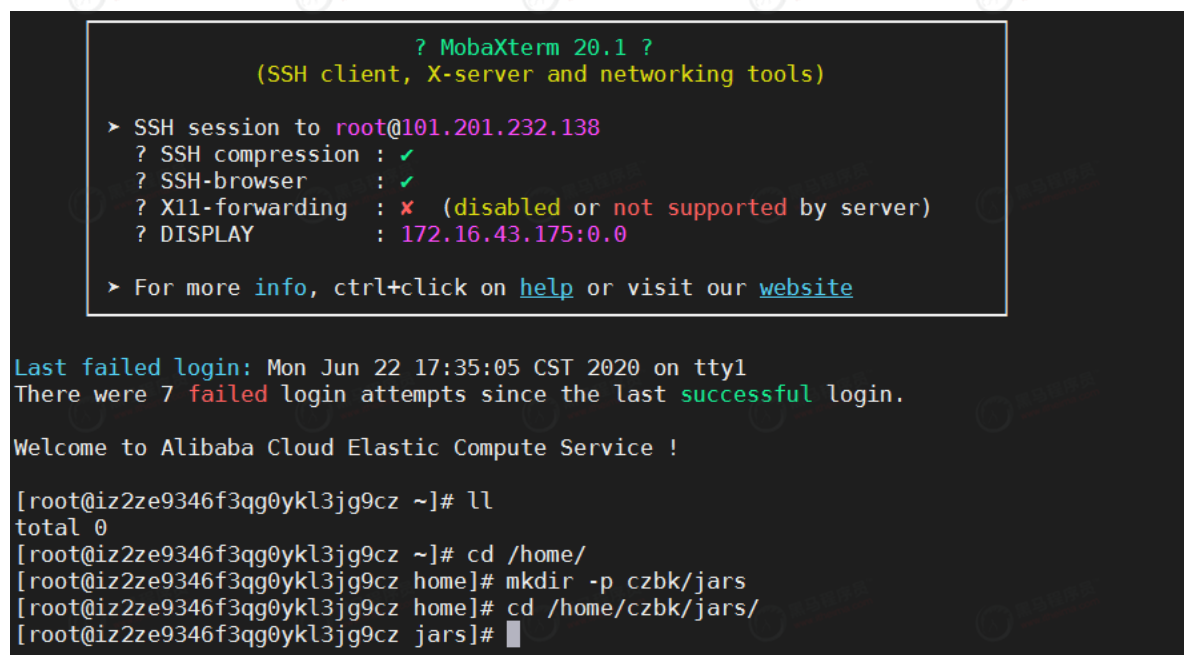


- 远程服务器创建目录

首先在服务器home目录下创建jars目录

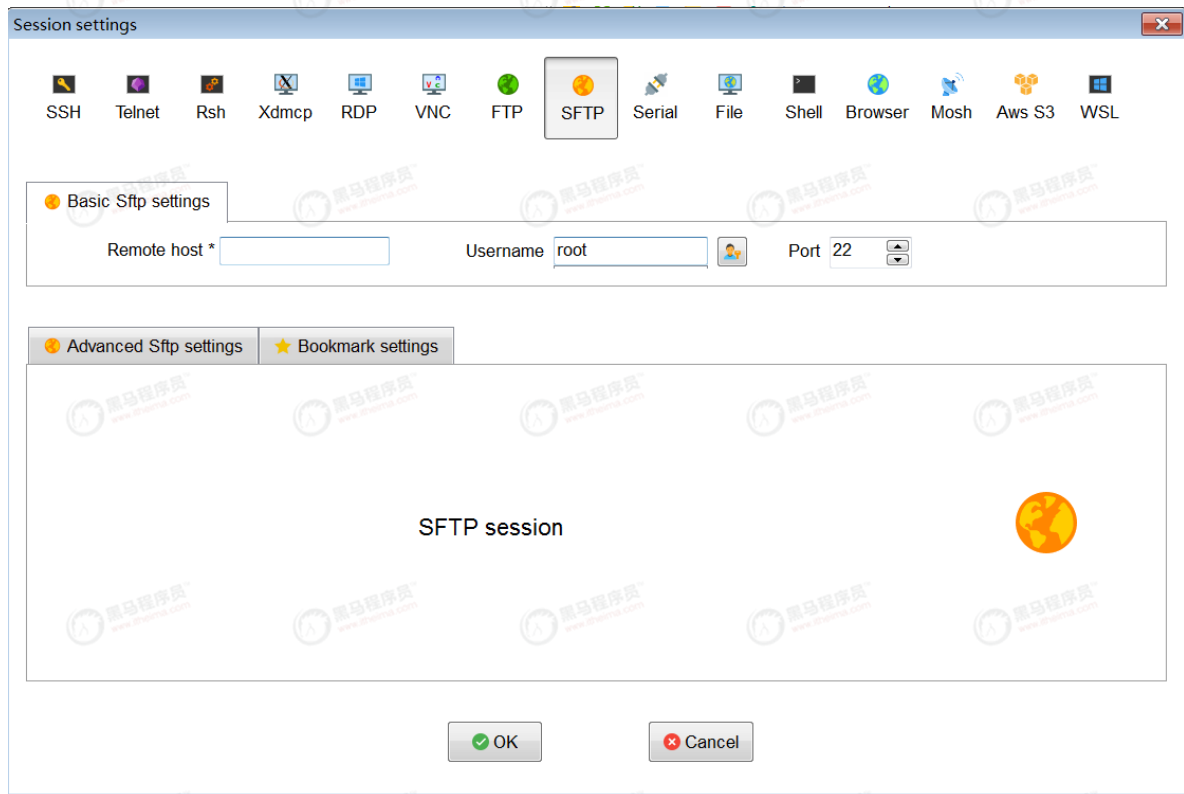
将jar包上传到创建的工程目录

```
cd /home/  
mkdir -p czbk/jars
```

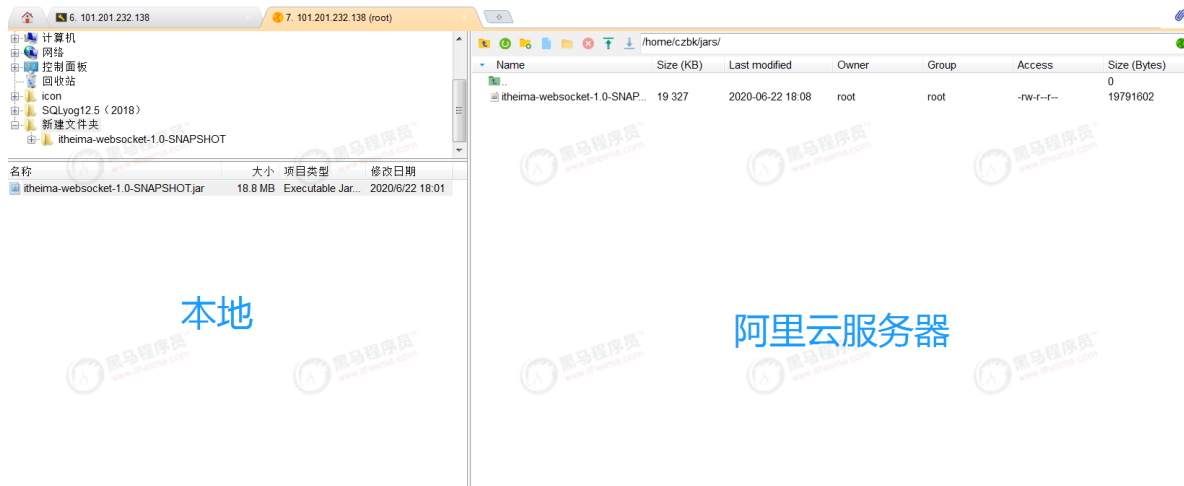


- 上传本地jar到远程ECS服务器

使用sftp连接远程服务器



将本地jar拖放到右边

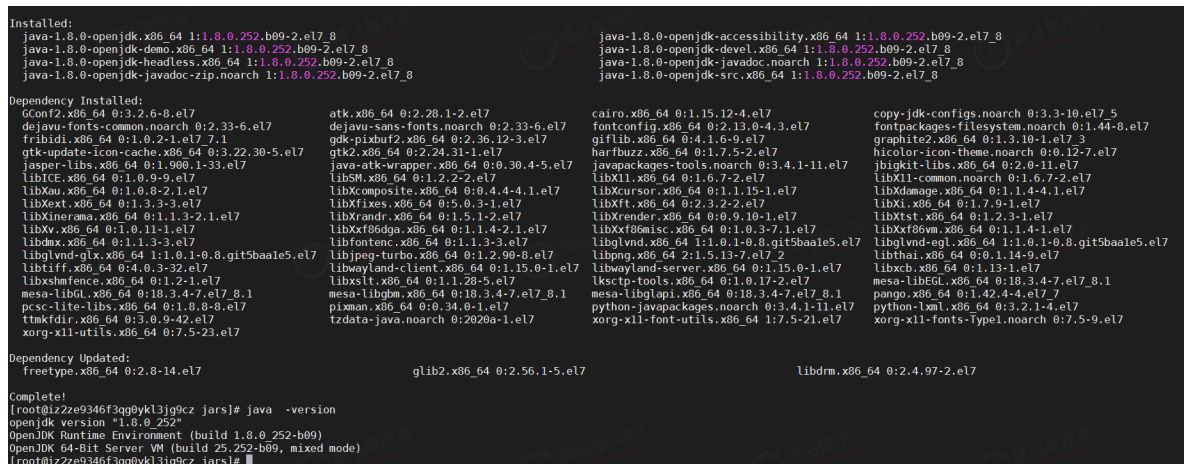


本地

阿里云服务器

```
yum install java-1.8.0-openjdk* -y
```

```
java -version
```



```
nohup java -jar itheima-websocket-aliyun-https-1.0-SNAPSHOT.jar &
```

3、访问阿里云HTTPS应用程序

<https://ecs.console.aliyun.com/?spm=a2c1d.8251892.recommends.decs.10565b76fQAfCc#/server/region/cn-beijing>

安全组：必须配置安全组，否则无法访问

入方向		出方向					
授权策略	优先级 ①	协议类型	端口范围 ①	授权对象 ①	描述	创建时间	操作
允许	100	自定义 TCP	目的:7777/7777	源:0.0.0.0/0	System created rule.	2020年6月29日 13:17:50	编辑 复制 删除
允许	100	自定义 TCP	目的:443/443	源:0.0.0.0/0	System created rule.	2020年6月29日 09:30:10	编辑 复制 删除
允许	100	自定义 TCP	目的:80/80	源:0.0.0.0/0	System created rule.	2020年6月23日 09:47:04	编辑 复制 删除
允许	100	自定义 TCP	目的:22/22	源:0.0.0.0/0	System created rule.	2020年6月22日 17:12:06	编辑 复制 删除
...	...	...	...	...	...	2020年6月22日	

端口范围填：80/80

授权对象填：0.0.0.0/0

就是将80号端口开放给所有的用户

优先级：1-100，数值越小，优先级越高

ip访问

<http://101.201.232.138:7777/im>

域名访问

<https://www.itheima.cloud/im>

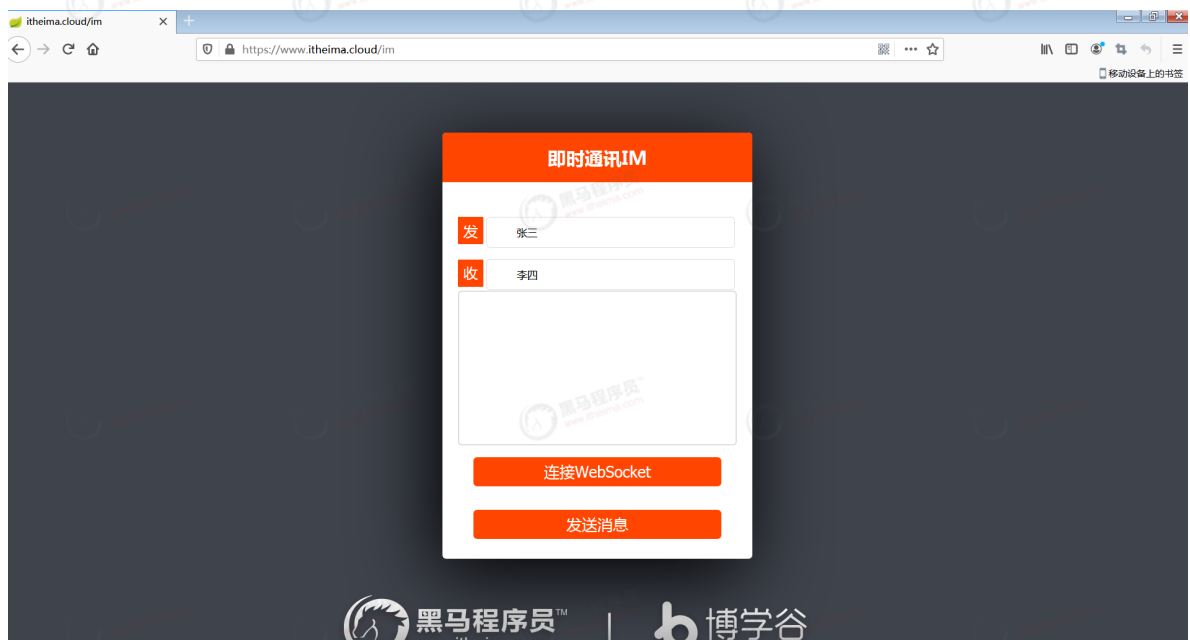
tips:

第一次部署证书的时候，是可以访问的，大约过滤两分钟后，被阿里云监测到后就无法访问了
因为域名尚未备案成功

转到http访问，就会有提示尚未备案的提示

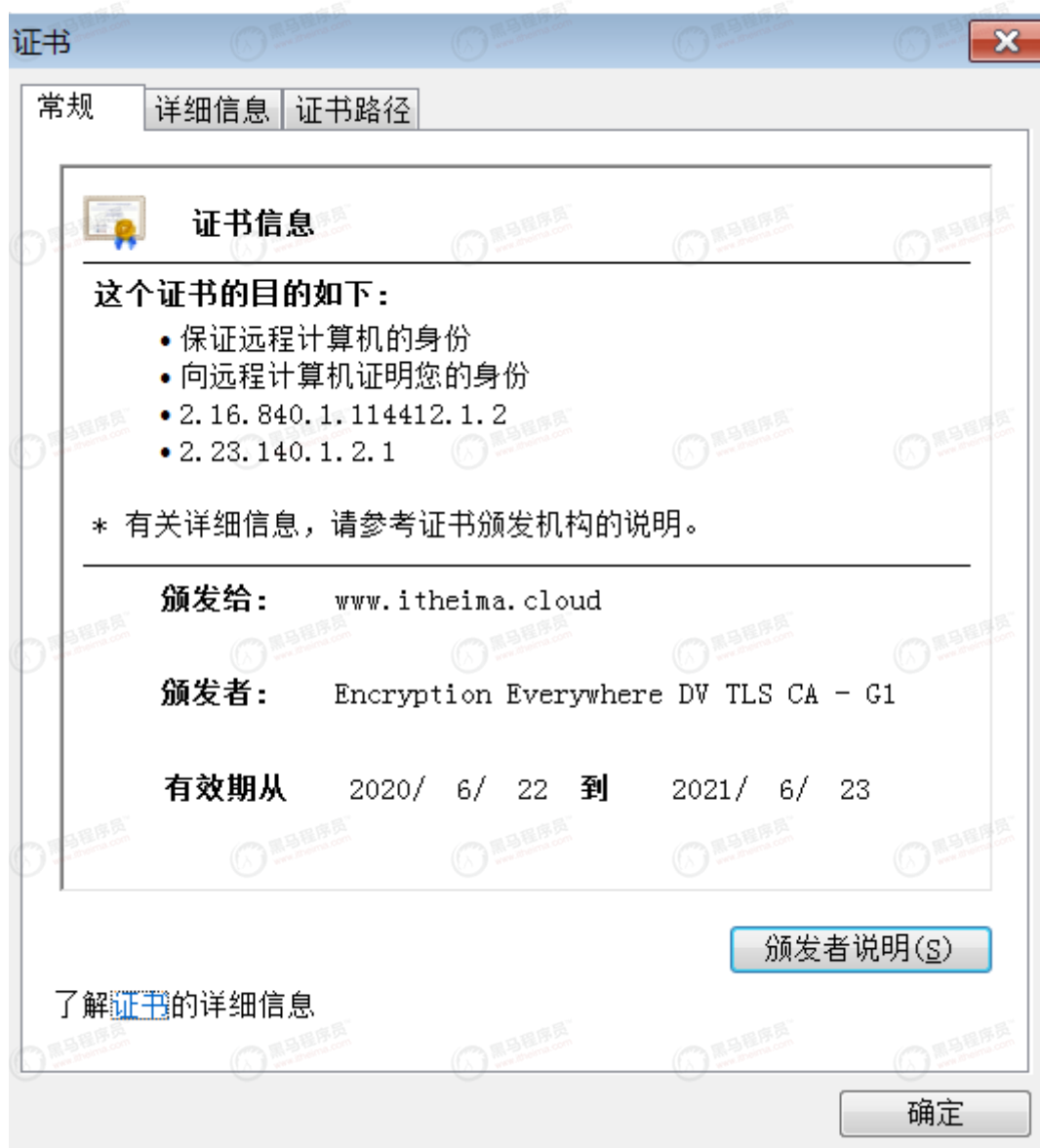


火狐





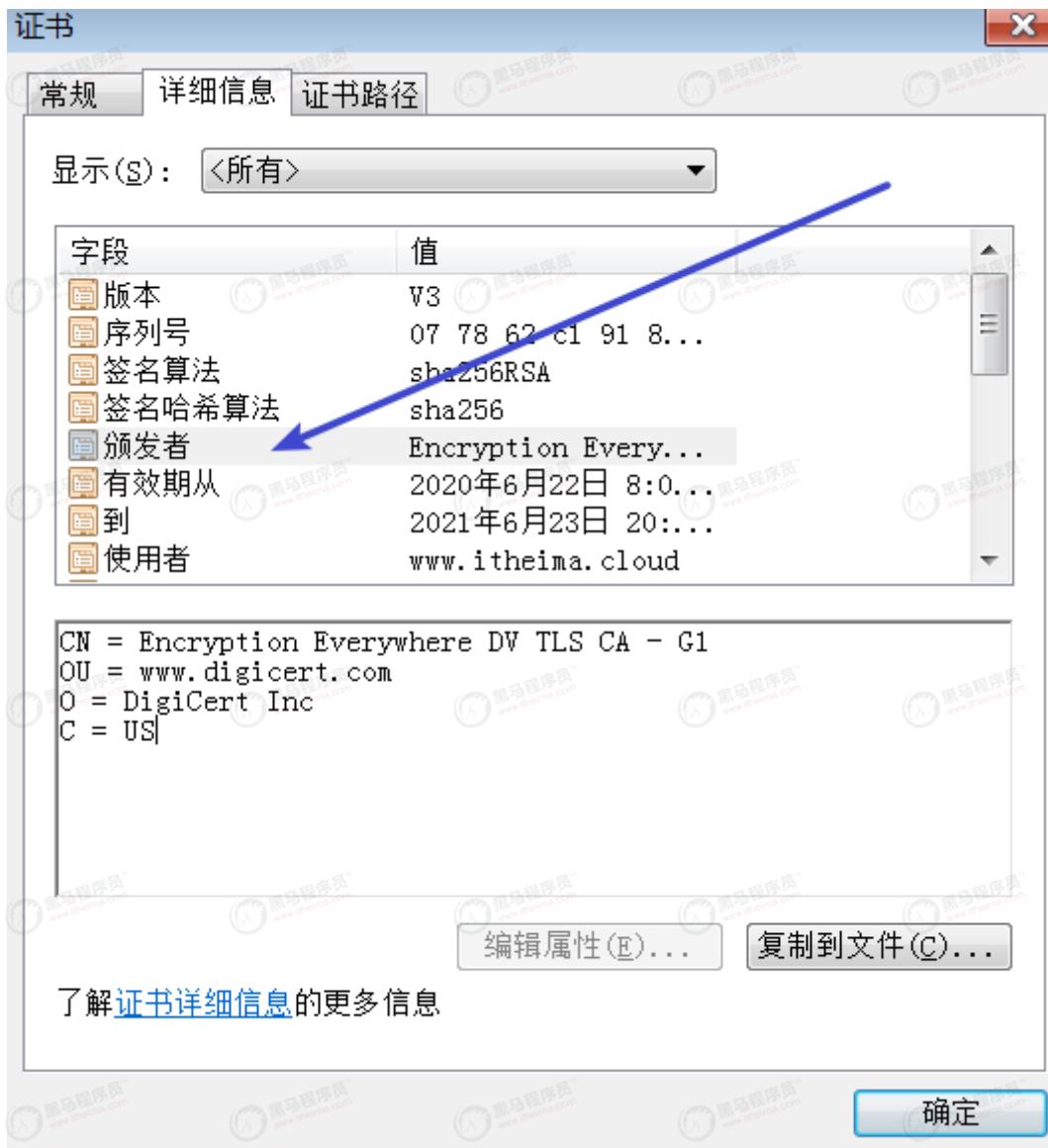
详细证书如下：



Encryption Everywhere DV TLS CA - G1 (通用名)

DV (domain validation) 域认证

Encryption Everywhere 这个是Symantec收购的Digicert安全业务的一部分



l Cn: common name 通用名

对象的属性为CN，例如一个用户的名字为：张三，那么“张三”就是一个CN。

l ou : OrganizationUnit 组织单位

o和ou都是ldap目录结构的一个属性，建立目录的时候可选新建o，ou等。在配置我司交换设备ldap的时候具体是配置ou，o还是cn等，要具体看ldap服务器的相应目录是什么属性。

l o: organizationName 组织名

证书



常规

详细信息

证书路径

显示(S): <所有>

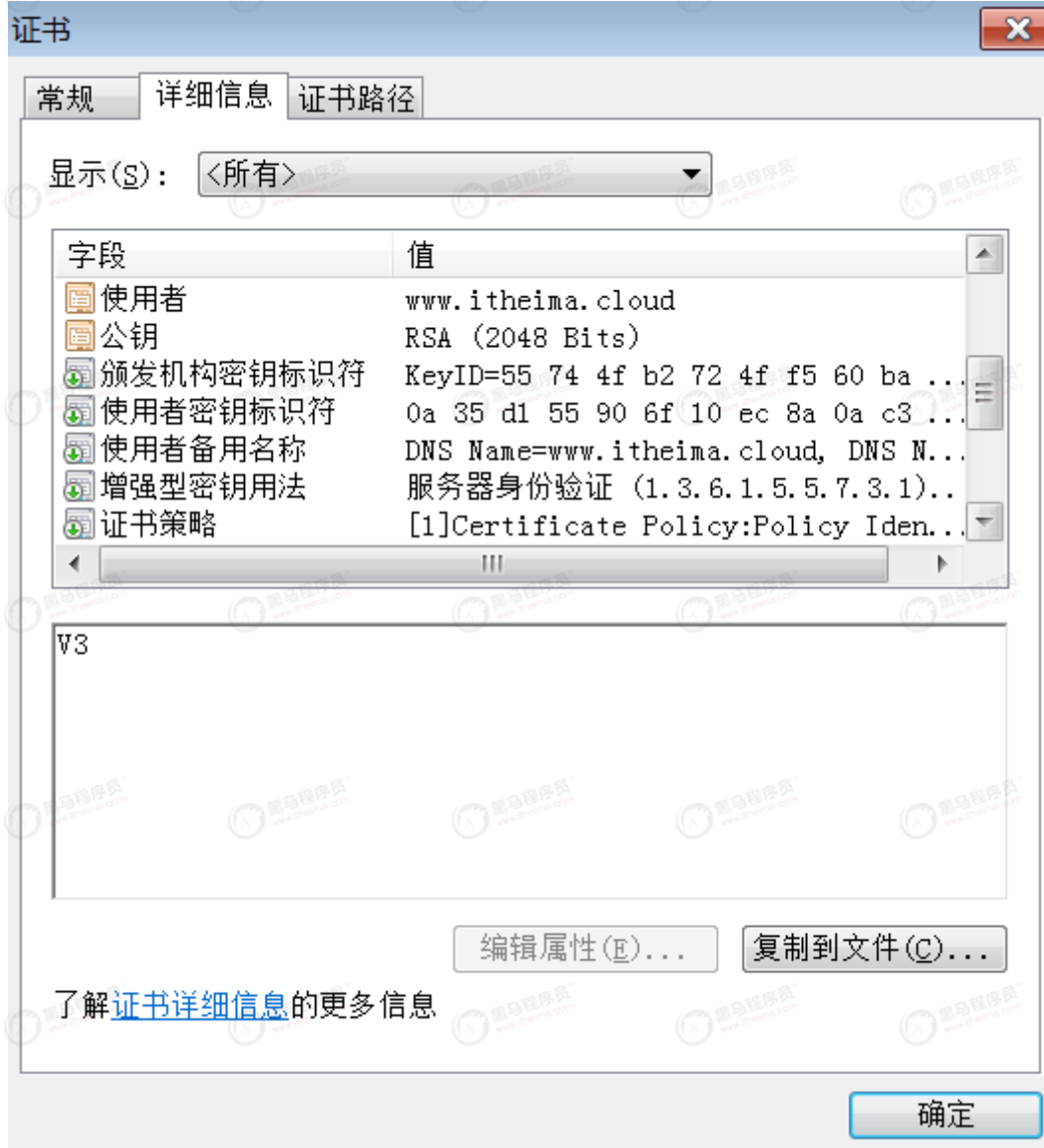
字段	值
版本	V3
序列号	07 78 62 c1 91 8...
签名算法	sha256RSA
签名哈希算法	sha256
颁发者	Encryption Every...
有效期从	2020年6月22日 8:0...
到	2021年6月23日 20:...
使用者	www.itheima.cloud

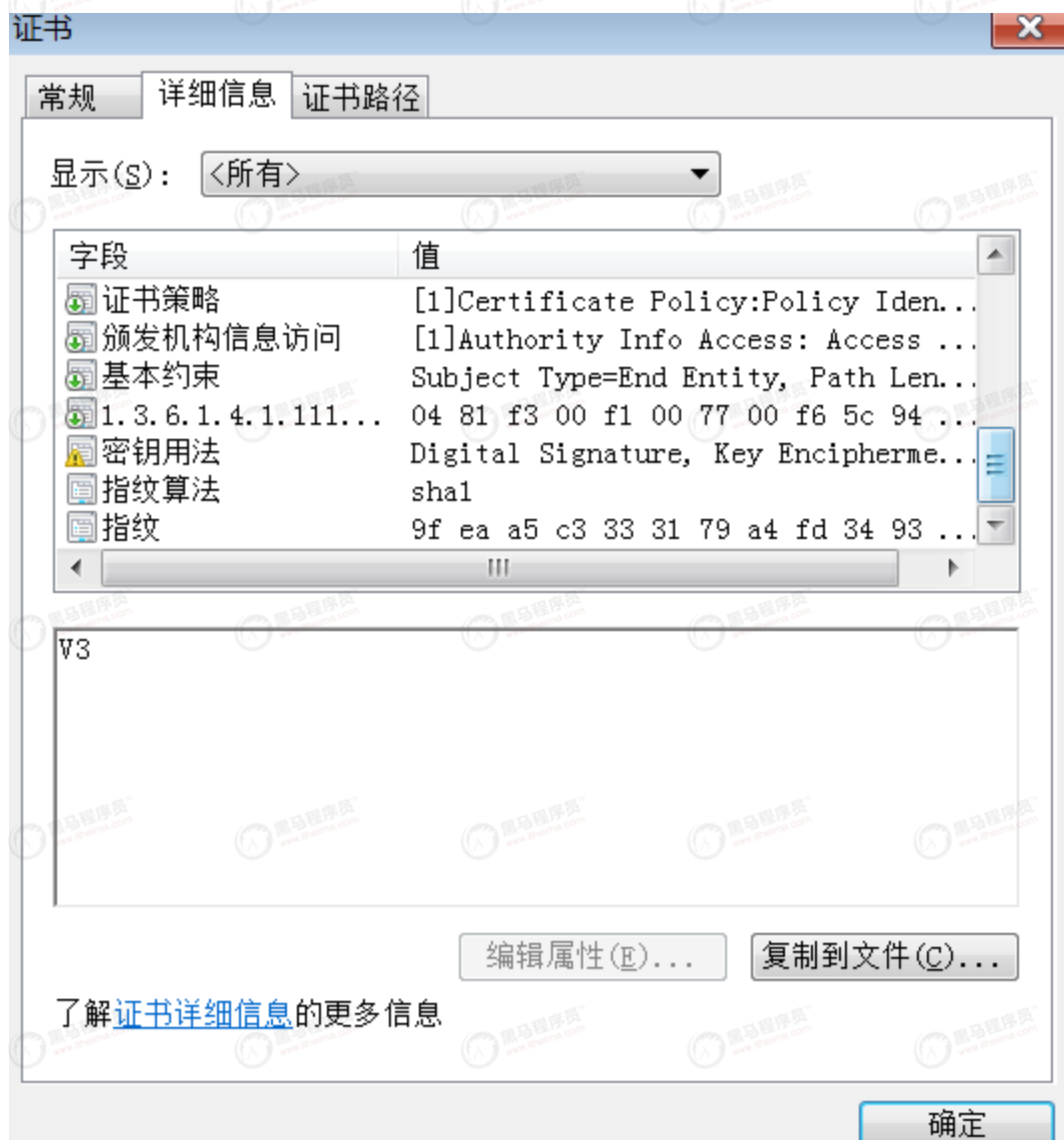
编辑属性(E)...

复制到文件(C)...

了解[证书详细信息](#)的更多信息

确定





直接通过ip访问【不安全】

http://101.201.232.138:7777/im
跳转
https://101.201.232.138/im

证书+域名=绑定



2、无法访问情况一

1、在线实时监测

<https://zijian.aliyun.com/?spm=a2c4g.11186623.2.17.41982da1r4nGxu#/domainDetect>

域名检测工具

www.itheima.cloud [立即检测\(3\)](#)

域名检查 ①

域名注册商:	Alibaba Cloud Computing Ltd. d/b/a HiChina (www.net.cn)	域名有效期:	🟢 域名在有效期内
域名状态:	🟢 域名状态正常		

DNS检查 ②

DNS服务商:	dns29.hichina.com, dns30.hichina.com (阿里云解析提供DNS服务)	本地DNS检测:	点击下载本地检测工具
DNS服务商解析结果:	A 101.201.232.138	223.5.5.5解析结果:	A 101.201.232.138
递归解析追踪:	🟢 域名递归解析正常	TTL生效时间:	域名TTL生效时间为 600 秒 提示: 如果域名记录修改不久, 请等待TTL生效时间后再次检测

网站检查 ②

备案检查:	🟡 网站未备案, 请咨询网站服务器提供商	工信部黑名单网站:	🟢 网站不在工信部黑名单中
Ping 检查:	🟢 Ping 检查正常	网站状态检查:	🔴 网站状态检查失败

2、阿里巴巴DNS检测工具--本地监测

https://help.aliyun.com/knowledge_detail/39834.html?spm=a2c1e.14465449.0.0.13ae1d68p0FPry#h2--dns-2

域名解析在云解析DNS上是否生效

判断方法: 如果下图中权威DNS的查询结果, 和您在云解析DNS设置的解析一致, 则代表解析记录在云解析DNS上已生效。如查询结果与您的设置不一致, 请 [提交工单](#) 联系阿里云售后为您处理

如下



云解析

https://dns.console.aliyun.com/?spm=5176.71615.J_5602601440.3.11ff5b8eod5Dzf#/dns/domainList

云解析DNS / 域名解析

解析不生效，一键检测

域名解析

功能概览

公告：.com/.net/.cn/.xyz/.top/.xyz/.vip/.club/.shop/.wang/.ren等域名注册成功后必须进行域名实名认证，否则会造成解析不生效，实名认证审核通过后的1-2个工作日解析可恢复使用。查看详情

【年中大促】云解析DNS爆款新用户年付75折；续费越久越优惠，1年8折/2年75折/3年7折！

全部域名

请求量统计

版本套餐管理

更多服务

添加域名

批量自动续费

批量操作

全部域名

域名快速搜索

Q

高级搜索

↓

☐

域名

标签

记录数

DNS服务器状态

付费版本

操作

☐

☆ itheima.cloud

◇

3

正常

免费版

解析设置 升级

☐

删除

更换分组

共1条

<

1

>

10条/页

← 解析设置 itheima.cloud

② 如何设置解析

当前分配的DNS服务器是：dns30.hichina.com, dns29.hichina.com

添加记录

导入/导出

请求量统计

新手引导

全部记录

精确搜索

输入关键字

高级搜索

☐	主机记录	记录类型	解析线路(isp)	记录值	TTL	状态	备注	操作
☐	www	A	默认	101.201.232.138	10分钟	正常		修改 暂停 删除 备注
☐	@	A	默认	101.201.232.138	10分钟	正常		修改 暂停 删除 备注
☐	_dnsauth	TXT	默认	2020062100000033r1svcqofysifvfajc3wbbskwnzczk74brdm6b9zdea2hkt2lf	10分钟	正常		修改 暂停 删除 备注

☐

暂停

启用

删除

更换分组

共3条

<

1

>

10条/页

3、无法访问情况二

如果出现下面的界面；需要备案

← → ↻ 不安全 | itheima.cloud/im

中文 | English



网站暂时无法访问

该网站未根据工信部相关法律进行备案
法律依据：《非经营性互联网信息服务备案管理办法》

如果您是网站普通访客：

请等待网站备案后尝试访问。

如果您是网站所有者，网站无法访问常见原因如下：

1. 网站未完成备案、未在接入商完成备案接入。
2. 网站内容与备案信息不符、备案信息不准确、网站存在不适宜传播的内容等。

[点击进入备案管理平台 >](#)

