冬冬他哥哥

公告

昵称： 谢呈勖
园龄： 7年7个月
粉丝： 78
关注： 28
+加关注

< 2023年4月 >

日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

C#(6)
wpf(4)
NSIS(2)
image(2)
linux(2)
qt(2)
C#调用C++(2)
平台调用(2)
wxs(2)
win10(1)
更多

随笔分类

C#(67)
C#异步编程(22)
C#中的事件(6)
C#中的委托(4)
C++(101)
C++实现设计模式(1)

Http协议之Https

http协议之详解（点我）

http协议之https（点我）

http协议之libcurl（点我）

一、简介与原理

http协议是明文传输的，因此很容易被截取和解析，泄漏个人数据。https协议是在http和tcp之间多添加了一层，进行身份验证和数据加密。

HTTPS 原理

① 客户端将它所支持的算法列表和一个用作产生密钥的随机数发送给服务器

② 服务器从算法列表中选择一种加密算法，并将它和一份包含服务器公用密钥的证书发送给客户端；该证书还包含了用于认证目的的服务器标识，服务器同时还提供了一个用作产生密钥的随机数 [2] ；

③ 客户端对服务器的证书进行验证（有关验证证书，可以参考数字签名），并抽取服务器的公用密钥；然后，再产生一个称作 pre_master_secret 的随机密码串，并使用服务器的公用密钥对其进行加密（参考非对称加 / 解密），并将加密后的信息发送给服务器 [2] ；

④ 客户端与服务器端根据 pre_master_secret 以及客户端与服务器的随机数值独立计算出加密和 MAC密钥（参考 DH密钥交换算法） [2] ；

⑤ 客户端将所有握手消息的 MAC 值发送给服务器 [2] ； ⑥ 服务器将所有握手消息的 MAC 值发送给客户端

二、密码学基础

明文： 明文指的是未被加密过的原始数据。
密文：明文被某种加密算法加密之后，会变成密文，从而确保原始数据的安全。密文也可以被解密，得到原始的明文。
密钥：密钥是一种参数，它是在明文转换为密文或将密文转换为明文的算法中输入的参数。密钥分为对称密钥与非对称密钥，分别应用在对称加密和非对称加密上。

对称加密：对称加密又叫做私钥加密，即信息的发送方和接收方使用同一个密钥去加密和解密数据。对称加密的特点是算法公开、加密和解密速度快，适合于对大数据量进行加密，常见的对称加密算法有DES、3DES、TDEA、Blowfish、RC5和IDEA。
其加密过程如下：**明文 + 加密算法 + 私钥 => 密文**
解密过程如下： **密文 + 解密算法 + 私钥 => 明文**

对称加密中用到的密钥叫做私钥，私钥表示个人私有的密钥，即该密钥不能被泄露。
其加密过程中的私钥与解密过程中用到的私钥是同一个密钥，这也是称加密之所以称之为“对称”的原因。由于对称加密的算法是公开的，所以一旦私钥被泄露，那么密文就很容易被破解，所以对称加密的缺点是密钥安全管理困难。

非对称加密：非对称加密也叫做公钥加密。非对称加密与对称加密相比，其安全性更好。对称加密的通信双方使用相同的密钥，如果一方的密钥遭泄露，那么整个通信就会被破解。而非对称加密使用一对密钥，即公钥和私钥，且二者成对出现。私钥被自己保存，不能对外泄露。公钥指的是公共的密钥，任何人都可以获得该密钥。用公钥或私钥中的任何一个进行加密，用另一个进行解密。

COM技术(4)
C语言(1)
Docker技术(9)
Linux(16)
Nodejs(3)
NSIS(6)
QT(3)
react框架开发(3)
RPA(1)
更多

随笔档案

2023年3月(5)
2023年2月(1)
2022年12月(1)
2022年11月(5)
2022年10月(4)
2022年9月(11)
2022年4月(3)
2022年3月(2)
2022年2月(2)
2022年1月(5)
2021年12月(4)
2021年11月(2)
2021年10月(6)
2021年9月(14)
2021年8月(6)
更多

阅读排行榜

1. c++引用lib和dll的方法总结(21977)
2. LPCTSTR 用法(13780)
3. C#中的异步和同步(13545)
4. C# Task的暂停与终止(12582)
5. c++11の异步方法 及线程间通信(12312)

评论排行榜

1. .Net环境下调用ProtoBuf(5)
2. c++のurlmon实现下载文件并进度回调(2)
3. .Net的内存回收(1)
4. WPF 中ContextMenu 在mvvm模式中的绑定问题(1)
5. WPF的动态资源和静态资源(1)

推荐排行榜

1. wpfのuri（让你完全明白wpf的图片加载方式以及URI写法）(4)
2. WPF自定义控件（一）の控件分类(3)

被公钥加密过的密文只能被私钥解密，过程如下：
明文 + 加密算法 + 公钥 => 密文， 密文 + 解密算法 + 私钥 => 明文
被私钥加密过的密文只能被公钥解密，过程如下：
明文 + 加密算法 + 私钥 => 密文， 密文 + 解密算法 + 公钥 => 明文

由于加密和解密使用了两个不同的密钥，这就是非对称加密“非对称”的原因。
非对称加密的缺点是加密和解密花费时间长、速度慢，只适合对少量数据进行加密。
在非对称加密中使用的主要算法有：RSA、Elgamal、Rabin、D-H、ECC（椭圆曲线加密算法）等。

三、https建立的过程

服务器端的公钥和私钥，用来进行非对称加密

客户端生成的随机密钥，用来进行对称加密

一个HTTPS请求实际上包含了两次HTTP传输，可以细分为8步。
1.客户端向服务器发起HTTPS请求，连接到服务器的443端口

2.服务器端有一个密钥对，即公钥和私钥，是用来进行非对称加密使用的，服务器端保存着私钥，不能将其泄露，公钥可以发送给任何人。

3.服务器将自己的公钥发送给客户端。

4.客户端收到服务器端的证书之后，会对证书进行检查，验证其合法性，如果发现发现证书有问题，那么HTTPS传输就无法继续。严格的说，这里应该是验证服务器发送的数字证书的合法性，关于客户端如何验证数字证书的合法性，下文会进行说明。如果公钥合格，那么客户端会生成一个随机值，这个随机值就是用于进行对称加密的密钥，我们将该密钥称之为client key，即客户端密钥，这样在概念上和服务器端的密钥容易进行区分。然后用服务器的公钥对客户端密钥进行非对称加密，这样客户端密钥就变成密文了，至此，HTTPS中的第一次HTTP请求结束。

5.客户端会发起HTTPS中的第二个HTTP请求，将加密之后的客户端密钥发送给服务器。

6.服务器接收到客户端发来的密文之后，会用自己的私钥对其进行非对称解密，解密之后的明文就是客户端密钥，然后用客户端密钥对数据进行对称加密，这样数据就变成了密文。

7.然后服务器将加密后的密文发送给客户端。

8.客户端收到服务器发送来的密文，用客户端密钥对其进行对称解密，得到服务器发送的数据。这样HTTPS中的第二个HTTP请求结束，整个HTTPS传输完成。

四、优缺点

优点

使用 HTTPS 协议可认证用户和服务器，确保数据发送到正确的客户机和服务器；

HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议，要比 HTTP 协议安全，可防止数据在传输过程中不被窃取、改变，确保数据的完整性 。

HTTPS 是现行架构下最安全的解决方案，虽然不是绝对安全，但它大幅增加了中间人攻击的成本 。

缺点

相同网络环境下，HTTPS 协议会使页面的加载时间延长近 50%，增加 10%到 20%的耗电。

HTTPS 协议还会影响缓存，增加数据开销和功耗 。

HTTPS 协议的安全是有范围 中间人攻击 伪造证书

http协议之详解（点我）

3. C# AOP的实现（利用.Net自带的轻量级框架RealProxy）(2)
4. WPF可视对象变换（RenderTransform）-----RotateTransform、TranslateTransform、ScaleTransform(2)
5. c++智能指针（unique_ptr 、shared_ptr、weak_ptr、auto_ptr）(2)

最新评论

1. Re:C# Task的暂停与终止
if (_cancelToken.IsCancellationRequested) { return; } 如果后面的逻辑需要执行10分钟猜完毕，岂不是要等待10分钟猜能正在取消？ 这个cancelT...
--wgscd
2. Re:WPF的动态资源和静态资源
666 这份介绍的太详细了
--彭二狗的牵引绳
3. Re:WPF 中ContextMenu 在mvvm模式中的绑定问题
您好！请问你下这个上下文动态绑定的有没有源码，可以发一份吗？564178640@qq.com
--天空之上的鱼儿
4. Re:WPF自定义控件（四）の自定义控件
能把源码发一下吗？ 多谢
--zhhu
5. Re:.Net的内存回收
string是引用类型，所有引用类型的默认值都是null。
--羽烈

http协议之https（点我）

http协议之libcurl（点我）

分类: 网络

谢呈勖
粉丝 - 78 关注 - 28

0 0

+加关注

« 上一篇： Http协议之详解
» 下一篇： WIndows进程通信（IPC）之管道通信

posted @ 2020-07-06 16:23 谢呈勖 阅读(1168) 评论(0) 编辑 收藏 举报