ios学习--RSA加密与破解 (SSL/TLS协议)

移动开发 来源:zzzzzddddxxxxx 2016-10-09 01:23 185℃ 0评论

目录(?) [+]

TLS名为传输层安全协议(Transport Layer Protocol)，这个协议是一套加密的通信协议。它的前身是SSL协议(安全套接层协议，Secure Sockets Layer)。这两个协议的工作方式类似，但TLS协议针对SSL协议进行了一些改善。SSL/TLS协议利用加密的方式，在开放的互联网环境中实现了加密通信，让通信的双方可以安心的说悄悄话。。

加密

SSL协议的基础是加密技术。加密和解密是自古就有技术了。比如说古代的男女偷偷发生私情，不能被相互之间有血海深仇的两个家族知道。男孩问女孩要不要一起私奔。女孩第二天传来答复，上面写着：

> K FQ

男孩拿着这串字符翻来覆去想了半天，没明白女孩的意思，就以为女孩不愿放弃优渥的生活和他私奔。直到十年后，男孩忽然灵光一闪，发现如果把每个字母都替换成字母表上提前两个的字母的话，这三个字符就变成了：

> I DO

这种加密方法是将原来的某种信息按照某个规律打乱。打乱的方式

热门推荐

Android开发人员不得不收集的工具类
2016-09-24 12:18 2119次浏览

专业程序员的7个特质
2016-10-11 10:47 1020次浏览

Android 安全架构及权限控制机制剖析
2016-10-01 05:25 979次浏览

一个美国人到中国当产品经理的心得：中国App设计真好！
2016-10-05 09:19 775次浏览

论程序员读书的重要性
2016-09-22 01:44 616次浏览

【Android】新的工具类DiffUtil，让RecyclerView上天
2016-10-09 01:48 581次浏览

我对程序员身体健康的一点感悟
2016-10-01 05:40 545次浏览

从程序员的角度分析微信小程序
2016-10-01 05:40 472次浏览

Android Studio 2.2发布：改进平台支持 速度提升更智能
2016-10-01 11:26 462次浏览

分类目录

系统运维 云计算
项目管理 综合
关于

热门标签 更多标签

ios(0) swift(0) android(0)
python(0) java(0) php(0)
html5(0) mysql(0) redis(0)
javascript(0) jquery(0) linux(0)

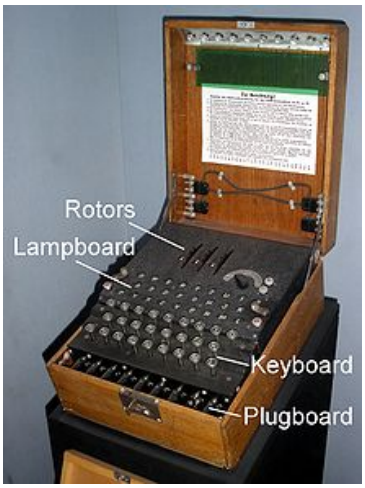
热门评论

论程序员读书的重要性
2016-09-22 01:44 3个评论

我对程序员身体健康的一点感悟
2016-10-01 05:40 2个评论

称为加密算法，而打乱过程中的参数就叫做密钥(cipher code)。上面女孩的加密方式是把原字母替换为字母表上后固定位的字母。而密钥就是固定的位数2了。发出信息的人根据密钥来给信息加密，而接收信息的人利用相同的密钥，来给信息解密。就好像一个带锁的盒子。发送信息的人将信息放到盒子里，用钥匙锁上。而接受信息的人则用相同的钥匙打开。加密和解密用的是同一个密钥，这种加密称为对称加密（symmetric encryption）。

如果一对一的话，那么两人需要交换一个密钥。理论上，如果密钥绝对安全，而且加密算法绝对复杂的话，对称加密是很难破解的。但通信双方很难绝对保证密钥的安全。一旦有其他人窃取到密钥，那么所有通信都变得不安全了。特别在一对多的话，如果共用同一套密钥，那么某一方通信的破解就意味着所有通信的破解。二战中盟军的情报战成果，很多都来自于破获这种对称加密的密钥。盟军破解了某个德国特工的加密手法，那么也就了解到纳粹总部的加密手法了。



二战中德军的传奇加密机：Enigma

对称加密的薄弱之处在于给了太多人的钥匙。如果换一种思路，只给特工锁，而总部保有钥匙，那就容易了。特工将信息用锁锁到盒子里，谁也打不开，除非到总部用唯一的一把钥匙打开。只是这样的话，特工每次出门都要带上许多锁，太容易被识破身份了。总部老大想了想，干脆就把造锁的技术公开了。特工，或者任何其它人，可以就地取材，按照图纸造锁，但无法根据图纸造出钥匙。钥匙只有总部的那一把。上面的关键是锁和钥匙工艺不同。知道了锁，并不能知道钥匙。这样，总部可以将"造锁"的方法公布给所有用户。每个用户可以

Android官方开发文档Training系列课程中文版：布局性能优化之ListView的优化
2016-09-20 20:28 2个评论

一个完整直播app功能分析
2016-09-28 22:24 1个评论

Android leakcanary内存泄漏检测和一般的解决方案
2016-09-20 00:35 1个评论

Android7.0适配教程与心得
2016-10-12 10:01 0个评论

互联网合并浪潮中，内部员工该何去何从？
2016-10-12 09:34 0个评论

乔布斯去世整整五年了，我想写一下他神秘的妻子
2016-10-12 09:25 0个评论

嗯！你这里改一下。噢！那里也改一下
2016-10-12 09:25 0个评论

友情链接

IT问道

用锁来加密自己的信用卡信息。即使被别人窃听到，也不用担心：只有总部才有钥匙呢！非对称加密中，给所有人用的锁被称为公钥(public key)，总部自己保留的钥匙被称为私钥(private key)。这样一种钥匙和锁分离的加密算法就叫做非对称加密(asymmetric encryption)。

非对称加密

对称加密的原理相对比较直观，而非对称加密听起来就有些神奇。经过非对称加密产生的密文，就算知道加密的方法，也无法获知原文。实现了非对称加密的经典算法是RSA算法。它来自于数论与计算机计数的奇妙结合。我们从下面的情境中体验一下RSA算法的妙处。

我是潜伏在龙凤大酒楼的卧底。想让下面信息以加密的方式发送到总部：

> A CHEF HIDE A BED

厨子藏起来了一张床！这是如此的重要，需要立即通知总部。千万重要的是，不能让反革命的厨子知道。

第一步是转码，也就是将英文转换成某个对应的数字。这个对应很容易建立，比如：

A	B	C	D	E	F	G	H	I
1	2	3	4	5	6	7	8	9

将上面的信息转码，获得下面的数字序列：

```
A CHEF HIDE A BED 1 3856 8945 1 254
```

这串数字完全没有什么秘密可言。厨子发现了这串数字之后，很容易根据数字顺序，对应字母表猜出来。

为了和狡猾的厨子斗智斗勇，我们需要对这串数字进一步加密。使用总部发给我们的锁，两个数字：3和10。我们分为两步处理。第一步是求乘方。第一个数字是3，也就是说，总部指示我们，求上面数字串的3次方：

原字符串：1 3 8 5 6 8 9 4 5 1 2 5 4

三次乘方：1 27 512 125 216 512 729 64 125 1 8 125 64

第二步是求余数。第二个上锁的数字是10，将上面每个三次乘方除以10，获得其余数：

余数： 1 7 2 5 6 2 9 4 5 1 8 5 4

将这串数字发回总部。中途被厨子偷看到，但一时不能了解其中的意思。如果还是像刚才一样对应字母表的话，信息是：

AGBEFBIDEAHED

信息到了总部。总部开始用神奇的钥匙来解读。这个钥匙是3。在这个简单的粒子里，钥匙不小心和之前锁中的一个数字相同。但这只是巧合。复杂的情况下很容易让锁和钥匙不同。解锁过程也是两步。第一步求钥匙次的乘方，即3次方。第二步求它们除以10（锁之一）的余数。

加密信息：1 7 2 5 6 2 9 4 5 1

8 5 4

三次乘方：1 343 8 125 216 8 729 64 125 1 5 12 125 64 （这里用的是钥匙的“3”）

除十得余：1 3 8 5 6 8 9 4 5 1 2 5 4

正是我们发送的信息。对应字母表，总部可以立即知道原来的信息。就此，我们简单的体验了RSA算法的使用过程。鉴于这里篇幅有限，这里不再详细解释RSA算法的原理。如果有兴趣，可以参考我的另一篇文章：“不给力啊，老湿！”：RSA加密与破解

SSL协议

可以看到，非对称加密从安全性上要强过对称加密。但天下没有免费的午餐。非对称加密的运算成本同样也比较高。为了兼顾效率和安全，SSL协议同时使用了非对称和对称加密。它用对称加密算法来加密信息本身。但对于安全性比较脆弱的对称加密密钥，则采用非对称加密的方式来传输。

SSL协议分为客户端和服务器端。通信的核心步骤很简单：

1. 双方利用明文通信的方式确立使用的加密算法。

2. 利用非对称算法通信，交换一个密钥。

3. 该密钥用于对称加密算法，加密接下来的通信正文。

可以看到，SSL协议的关键是用一个非常安全的方式来交换一个对称密钥。交换的过程会比上面的描述更加复杂一些。

1. 客户发起请求时，除了说明自己支持的非对称加密算法，还会附加一个客户端随机数(client random)。

2．服务器回复请求时，会确定非对称加密算法和哈希函数，并附上公钥。此外，服务器端还会在此次通信中附加一个服务器端随机数(server random)。

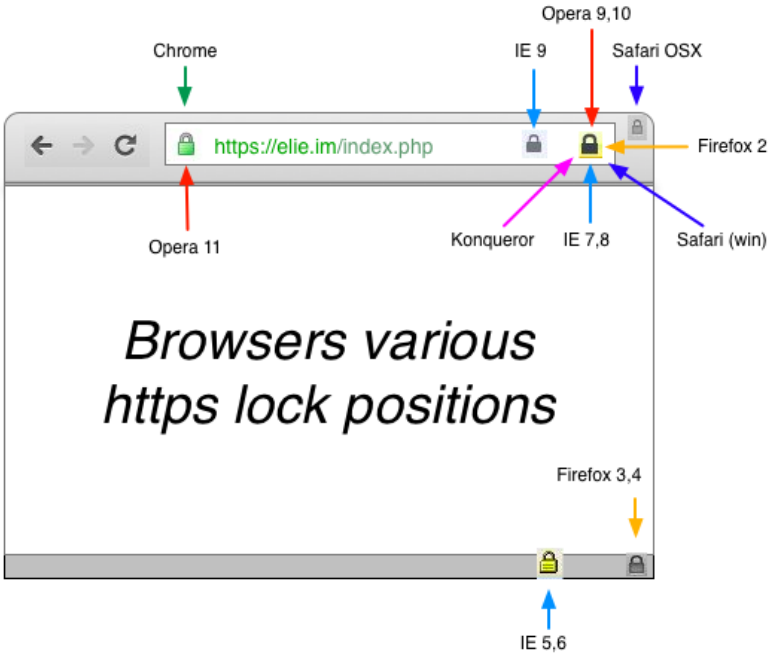
3．客户端会产生第三个随机数(Premaster secret)，然后利用服务器确定的非对称加密算法和公钥来加密这个随机数，再发送给服务器端。

4．客户端用自己的私钥解密第三个随机数。

4．这样，客户端和服务器端都知道了三个随机数。双方各自用商量好的哈希函数从三个随机数获得对称加密的密钥。

即使明文通信的时候，某些信息被窃听，但第三步的非对称加密通信部分可以保证窃听者无法完整的获得三个随机数。这样，窃听者还是不知道对称加密的密钥���什么。这样，对称加密的密钥就在一个安全的环境中获得了。为了进一步安全，服务器的公钥会包含在一个数字证书中发送给客户。这样，客户还可以通过数字证书来验证服务器的身份，以免服务器本身出现问题。

今年来使用越来越广泛的HTTPS协议就是在SSL/TLS协议的基础上进行通信。HTTP协议在通信过程中要经过多重路由，很容易被窃听。经过SSL协议加密的信息就算被窃听，也只能被通信目的地的人解读，从而保证了信息的安全。所以，如果所访问的网站没有使用HTTPS协议，那么在输入银行账号和密码之类的敏感信息时，就要三思而后行了。



当浏览器出现锁的符号时，说明访问的资源使用了HTTPS通信

为了演示方便，选用了简单的锁和钥匙。锁和钥匙只是凑巧相同。为此，我们做一个小练习。

练习：总部新公布出来的锁是2987(次乘方)和3937(为除数)。

1） 作为特工，用上面的算法为信息加密(你可能需要一些编程来计算，尝试用Python的数学计算功能？)。

猜到钥匙是什么了呢？不是上面两个数字中的任何一个，而是143！

2） 作为值班人员，验证143是钥匙，可以解密信息。

为了简便，你可以只检验一个简单的信息，比如"IE"。

下面是我根据这个练习写的一个Python小程序。这里的转码用的是ASCII编码标准，而不是上面的A对应1，B对应2。


```
    # By Vamei

#==== Agent ======== # coding covert: string to number #
By ASCII convention
def convert(original): return map(ord, original) # the in
put is a list of integers
def encrypt(input_list): f = lambda x: (x**2987)%3937
    return map(f, input_list) #==== Headquarter ===== #
the input is the result of the encrypt function
def decrypt(encrypted_list): f = lambda x: (x**143)%3937
    return map(f, encrypted_list) # convert numbers back
to a string
def inv_convert(decrypt_list): f = lambda x: str(unichr(x
)) result = map(f, decrypt_list) return "".join(result) #
Test
message = "Go to hell!" secret =
encrypt(convert(message)) print(secret) public =
inv_convert(decrypt(secret)) print(public)
```

费马与欧拉

发觉自己被愚弄了，厨子很生气，后果很严重。厨子发奋看了书，知道了这个加密方法叫RSA，是三为发明人 R. Rivest, A. Shamir和L. Adelman名字首字母合起来的。RSA算法是1977年发明的。全称是RSA Public Key System。这个"Public Key"是公共密钥，也就是我们上面说的锁。再读下去，厨子大窘。这个1977年的，现代计算机加密的RSA算法，居然源于17世纪。

1. 费马小定律

RSA的原理借助了数论中的"欧拉定理"(Euler's theorem)。17世纪的费马首先给出一个该定理的特殊形式，即"费马小定理":

p是一个正的质数，a是任意一个不能被p整除的整数。那么， ap-1-1 ap11能被p整除。

我们并不需要太深入了解费马小定理，因为等下就会看到这个定理的"升级版"。但这个定理依然很美妙，它优美的得到乘方和整除的某种特殊关系。使用一个例子来说明它。比如 p=7，a=3 p7a3。那么费马小定律表示， 37-1-1 3711可以被7整除。

事实上，上面的数字计算得到 36-1=728 361728，它确实可以被7整除。

练习：尝试一个其它的例子，比如 p=5，a=4 p5a4，验证费马小定律是否成立。

*** 数学小贴士：

1) 除 (divide)，商和余数：两个整数相除，有一个为整数的商，和一个余数。比如 10/3=3,余1 10331。我们用一个特别的方式记录这一叙述：

10≡1(mod3) 101mod3

也可以写成另一种方式：

[10]3=[1]3 10313

这一表述方式与"10除以3，得3余1"这样的方式并没有什么区别。但采用标准的数学方式更容易和别人交流。

如果我们知道：

[a]n=[b]n anbn

那么存在某个整数t，且：

a=nt+b antb

2) 整除 (divisible)：当一个整数a除以另一个整数b，余数为0时，那么我们说a可以被b整除。比如说，4可以被2整除。即

[4]2=[0]2 4202

3) 质数 (prime number)：一个质数是只能被 ±1 1和这个数自身整除的整数（不包括 ±1 1）。比如 2，3，5，7，11，13 23571113等等。

******

费马是一名律师，也是一名业余数学家。他对数学贡献很大，堪称"业余数学家之

王”。比如他和帕斯卡的通信算是概率论的开端。还有“费马大定理”，或者称为“费马猜想”。费马有在书边写注释的习惯。他在页边角写下了费马猜想，并说：

> 我发现了一个美妙的证明，但由于空白太小而没有写下来。

费马自己的证明没有再被发现。“费马猜想”的证明是300多年后，以现代数学为工具证得的，而这些数学工具在费马的时代是不存在的。这导致现代的数学家怀疑费马是不是在吹牛。费马小定理是费马的另一个定理。在费马那里，也还是个猜想。证明要等到欧拉。

程序员们：注释要完整啊！

2．欧拉定律

时间流过一百年。欧拉是18世纪的瑞典数学家。这位数学巨人写了75本数学专著，几乎把当时所有的数学领域都征服了一遍。欧拉后来被叶卡捷琳娜二世邀请到俄国。据说，无神论者狄德罗造访俄国，他宣称上帝并不存在，靠雄辩击败了整个俄国宫廷。欧拉曾醉心神学，对上帝很虔诚。欧拉看不下去了，上前说，“先生， $e^{i\pi}+1=0$ $e^{i\pi}10$，所以上帝存在。请回答！”狄德罗败给这个问题，灰溜溜的走了。

（这个传说的可信度不高，因为狄德罗本人也是一位颇有造诣的数学家。）

欧拉定理(Euler's theorem)是欧拉在证明费马小定理的过程中，发现的一个适用性更广的定理。

首先定义一个函数，叫做欧拉Phi函数，即 ϕ(n) ϕn，其中，n是一个正整数。

ϕ(n)=总数(从1到n−1，与n互质的整数) ϕn1n1n

比如5，那么1，2，3，4，都与5互质。与5互质的数有4个。 ϕ(5)=4 ϕ54

再比如6，与1，5互质，与2，3，4并不互质。因此， ϕ(6)=2 ϕ62

对于一个质数p来说，它和1，2，3，...，p - 1都互质，所以 ϕ(p)=p-1 ϕpp1。比如 ϕ(7)=6,ϕ(11)=10 ϕ76ϕ1110

*** “互质”的数学小贴士：

1) 因子 (factor)：每个整数都可以写成质数相乘的形式，每个这样的质数称为该整数的一个因子。

2) 互质 (relative prime)：如果两个整数没有公共因子，这两个质数互质。

******

欧拉定理叙述如下：

如果n是一个正整数，a是任意一个非0整数，且n和a互质。那么， aϕ(n)−1 aϕn1可以被n整除。 (1)

由于质数p有 ϕ(p)=p-1 ϕpp1。因此，从欧拉定理可以推出费马小定理。我们可以只使用欧拉定理，把费马小定理抛到脑后了。我们用一个例子简单的检验欧拉定理。如果n是6，那么 ϕ(6)=2 ϕ62。让a是11，和6互质。 112−1 1121为120，确实可以被n，也就是6整除，符合欧拉定理。

数学中还有一个关于Phi函数的推论：

m和n是互质的正整数。那么， ϕ(mn)=ϕ(m)ϕ(n) ϕmnϕmϕn (2)

RSA西游记

下面我们要进入实质的证明。除了上面的(1)和(2)推论，还需要提前说明一个问题，即：

[ab]n=[a]n[b]n abnanbn (3)

证明：假设a和b除以n的余数为 c1,c2 c1c2。a和b可以写成 a=nt1+c1,b=nt2+c2 ant1c1bnt2c2。那么， ab=n2t1t2+nt1c2+nt2c1+c1c2 abn2t1t2nt1c2nt2c1c1c2。因此ab除以n的余数为 c1c2 c1c2。即 [ab]n=[a]n[b]n abnanbn。

根据此可以推论， [am]n=[a]mn amnanm。

演一出叫做"西游记"的大戏，选角开始：

先选择两个质数p和q，分别是沙和尚和白龙马。让 n=pq npq，n是唐僧。一路向西，唐僧靠的是沙和尚和白龙马出力：一个背行李，一个驮人。

而 k=ϕ(n)=(p-1)(q-1) kϕnp1q1。这里使用了(2)以及"质数p的Phi函数值为p-1"。k是八戒，也就是Phi(唐僧)，就是唐僧的一个跟屁虫。

选择任意d，并保证它与k互质。d是观音。观音姐姐在高老庄，真的是把八戒给"质"了一把。

取整数e，使得 [de]k=[1]k dek1k。也就是说 de=kt+1 dekt1，t为某一整数。e是悟空，横行无忌。

我们记得公开的用来上锁的两个数字，它们分别是悟空e和唐僧n。悟空威力大，负责乘方。唐僧太唠叨：一切妖怪见到它，就变成了余数。悟空和唐僧合作，就把世界搞乱了。

总部的观音姐姐d看不下去了。观音姐姐威力也大，也是乘方。再逼着唐僧重新唠叨。世界就恢复了。

善哉，善哉！

我们看一下这一魔幻大片“西游记”的现实主义原理。根据欧拉定理(1)，对于任意z，如果z与n互质，那么：

[zϕ(n)]n=[zk]n=[1]n zϕnnzkn1n

因此，

[zde]n=[zkt+1]n=[(zk)tz]n=[z]n zdenzkt1nzktznzn

上面主要使用了 de=kt+1 dekt1以及(3)。也就是说：

[zde]n=[z]n zdenzn

根据(3)的推论，有

([ze]n)d=[z]n zendzn

妖怪z，经过e和d的各一道，又变回了妖！上面过程中，悟空e和观音d忙得不亦乐乎，唐僧n就在一旁边唠叨边打酱油了。

这一等式，也正是我们加密又解密的过程 （加密：悟空次方 + 唐僧唠叨。解密：观音次方 + 唐僧唠叨）。悟空和唐僧是公钥，扔出去亮相。观音是私钥，偷偷藏起来，必要的时候才出来。

（上面都默认余数是最小正余数，也就是说，10除以3的余数为1，而不是4。尽管4也可以算是10的余数，即 [4]3=[10]3 43103。）

姐姐，饶了我吧。

3和8两个妖怪见到唐僧5，都被唠叨成了余数3。这样就观音姐姐就算法力无边，还是没法还原。为了让唐僧求余的时候，不会把数字弄混了，RSA算法要求所有妖怪z小于唐僧n。为了对足够多的字符转码加密，n必须大过最大的妖怪。

但唐僧n大更重要的原因是要保护马仔。想破解，必须找到观音。回顾我们选择角色的过程。我们可以这样破解：唐僧n是公开的，1） 先找到它的隐藏手下沙和尚和白龙马。2） 沙和尚和白龙马知道了，那么二师兄k就保不住了。3） de = kt + 1，即找到一个e，可以让de - 1被k整除。观音姐姐就找到了。

上面的整个破解过程中，最困难的是第一步，即找到两个隐藏的打手。通常，p和q都会选的非常大，比如说200位。这导致唐僧n也非常大，有400位。寻找一个400位数字的质数分解并不容易，我们要做的除法运算次数大约
为 10400----√/2 104002。这是 10199 10199次除法运算！天河2号每秒浮点运算是 1016 1016级别。那么，找到隐藏打手的工作，大约需要 10174 10174年……。这个活，看来只能佛祖干了。

练习 如果唐僧不够大的话，马仔就危险了。想想之前的厨子，知道悟空是3，唐僧是10。隐藏打手是谁？ 八戒呢？ 观音呢？

总之，带头大哥不够"罩"的话，团伙就要被一窝端了。

总结

数学可以是程序员军火库中有力的武器。加密、解密这一事关IT安全的大课题，却和数论这一纯粹数学学科发生奇妙的关系。RSA算法的数学基础在于欧拉定理。这一诞生了几百年没有什么实用性的数学理论，却在网络时代，找到自己的栖身之处。

RSA算法是非对称算法。公开的加密方式，私有的解密方式。RSA安全的关键在于很难对一个大的整数进行因子分解。

原文地址：http://www.cnblogs.com/vamei/p/3480994.html

还没有评论，沙发等你来抢

社交帐号登录: 微信 微博 QQ 人人 更多»

说点什么吧...

发布

IT问道正在使用多说