


# stevenczp

- [博客园](#)
- [首页](#)
- [新随笔](#)
- [联系](#)
- [管理](#)
- [订阅](#) 

随笔- 121 文章- 0 评论- 17

## Cryptography I 学习笔记 --- 密钥交换

1. 使用可信第三方 (Trusted third parties) 进行密钥交换。

a. Alice与TTP之间的密钥是K1, Bob与TTP之间的密钥是K2。

b. Alice向TTP发起一个与Bob交换密钥的请求。TTP生成一个随机密钥Kab, 然后将Kab用K1加密, 得到E1, 将Kab用K2加密, 得到E2。

c. TTP将E1与E2一起发送给Alice。Alice用K1解密E1, **得到密钥Kab**。

d. Alice再将E2发送给Bob, Bob用K2解密E2, 也**得到密钥Kab**。

e. 现在Alice与Bob之间就有一个共享的密钥Kab了。

TTP的缺点是完全中心化, 如果TTP被攻破, 会导致所有传递过的密钥暴露。

2. Merkle算法

第一个被发明的**无需依赖TTP的对称加密条件下的**密钥交换算法。

核心思想: Alice产生n个很难的问题Q, 每个问题附带一个密钥K, 然后将这些问题全部转发给Bob, Bob随机挑选一个问题Qi求解, 将解算的结果返回给Alice, Alice拿到Bob的运算结果后可以在常数时间内知道Bob求解的问题Qi。然后Alice与Bob就可以使用Qi所对应的密钥Ki进行通信了。攻击者不知道Bob选中的是哪个问题, 必须要求解所有的问题才行, 在n很大的情况下, 可以认为是安全的。

范例: Alice创建 $2^{32}$ 个密钥对, 第i个密钥对是128bit的随机数Xi与Ki, 将密钥对拼接并加入识别码, 得到明文Pi (eg:  $P_i = \text{"Puzzle\#Xi+Ki"}$ ), 将Pi用i加密, 得到密文Ei。Alice在本地保存这 $2^{32}$ 个密钥对, 然后将 $2^{32}$ 个Ei发送给Bob。Bob随机选取一个Ei, 然后用0到 $2^{32}-1$ 这 $2^{32}$ 个密钥对Ei进行解密, 如果解密后的结果是以Puzzle开头, 那么认为解密成功。Bob**得到 Xi与Ki**, 然后将Xi发回给Alice。Alice收到Xi后查表也**可以得到Ki**。此时Alice与Bob得到了共享密钥Ki, 于是可以安全通信了。

缺点：这个算法并不实用，因为即使 $n=2^{32}$ ，那么攻击者也只需要 $2^{64}$ 次计算即可解密。如果把 $n$ 设为 $2^{64}$ ，攻击次数则上升到 $2^{128}$ ，安全固然是安全了，但是Alice需要花太多时间来生成谜题，然后将这些谜题转发给Bob了。

3. 基于对称加密的密钥交换体系最好也只能做到**平方鸿沟**（quadratic gap，攻击者的时间复杂度是参与者的时间复杂度的平方）了，真正实用的不依赖于TTP的密钥交换体系，需要用到非对称密钥体系。

#### 4. Diffie-Hellman协议

a. Alice或者Bob生成一个2000bit的大质数 $p$ ，然后生成一个介于1到 $p$ 之间的整数 $g$ ，然后将 $p$ 与 $g$ 公开

b. Alice随机选择一个介于1到 $p-1$ 之间的整数 $a$ ，Bob随机选择一个介于1到 $p-1$ 之间的整数 $b$

c. Alice计算 $A = g^a \pmod{p}$ ，Bob计算 $B = g^b \pmod{p}$

d. Alice与Bob交换 $A$ 与 $B$

e. Alice与Bob之间的共享密钥是 $g^{ab} \pmod{p}$ ，分别可以由 $B^a \pmod{p}$ 与 $A^b \pmod{p}$ 得到

证明： $A = g^a \pmod{p} = g^a - k \cdot p \implies A^b \pmod{p} = (g^a - k \cdot p)^b \pmod{p} \implies$   
对 $(g^a - k \cdot p)^b$ 做二项式展开，可以知道除了第一项为 $g^{ab}$ 以外，其他的项都带有因数 $p$ ，这些项都可以在 $\pmod{p}$ 的操作中被约掉  $\implies A^b \pmod{p} = g^{ab} \pmod{p}$

如果攻击者想用从 $p, g, g^a, g^b$ 计算得到 $g^{ab}$ ，已知的最好算法是GNFS(General Number Field Sieve，一般数域筛法)，其时间复杂度为 $O(n^{\frac{1}{3}})$ 立方根，这是一个亚指数时间复杂度算法。

补充：基于椭圆曲线（elliptic curve）的Diffie-Hellman协议有更强大的时间复杂度，下表是相同破解难度下的密钥长度对比

cipher key size    modulus size    elliptic curve size

80 bits            1024 bits        160 bits

128 bits           3072 bits        256 bits

256 bits(AES)    15360 bits       512 bits

#### 5. 原始的Diffie-Hellman协议只能阻止窃听攻击，无法阻止中间人攻击

如果有中间人完全劫持了Alice与Bob之间的信道，Alice发送的 $A$ 被中间人篡改改为 $A'$ 后发送给Bob，Bob发送的 $B$ 被篡改改为 $B'$ 后发送给Alice。于是Alice认为密钥是 $g^{b'a} \pmod{p}$ ，Bob认为密钥是

$g^{a'b} \pmod p$ 。由于中间人知道 $a'$ 与 $b'$ ，他可以轻易的计算出这两个密钥。现在他需要做的事情只是把Alice发出的密文 $E_a$ 用 $g^{b'a} \pmod p$ 解密，即可得到明文 $P$ ，再把明文用密钥 $g^{a'b} \pmod p$ 加密后得到的密文 $E_b$ 转发给Bob即可。

6. Diffie-Hellman具有**非互动性**，比方Alice, Bob, Charlie, David都在自己的Facebook上公开了自己的公钥( $g^a, g^b, g^c, g^d$ )，那么如果Alice想要与David通信，她只需要去David的公共主页上看一眼他的公钥 $g^d$ ，就能立刻计算出密钥 $g^{ad}$ ，然后就能与David安全通信了。

## 7. 基于公钥加密算法的密钥交换体系

- Alice生成一个密钥对： $pk$ 与 $sk$ ，然后向Bob发布她的公钥 $pk$
- Bob接到 $pk$ 后生成一个随机密钥 $k$ ，然后用 $pk$ 对 $k$ 进行加密，得到密文 $E$
- Bob将 $E$ 发送给Alice
- Alice用私钥 $sk$ 对 $E$ 解密，即可得到 $k$
- Alice与Bob可以用 $k$ 进行安全通信了

8. 这种密钥交换体系无法抵御中间人攻击。

如果中间人完全劫持了Alice与Bob之间的信道，他可以将Alice所发布的公钥 $pk$ 替换为自己生成的密钥对 ( $pk'$ 与 $sk'$ ) 中的公钥 $pk'$ ，然后将 $pk'$ 发送给Bob。Bob用 $pk'$ 对 $k$ 加密得到 $E'$ 。中间人用私钥 $sk'$ 对 $E'$ 解密得到 $k$ ，然后再用Alice的公钥 $pk$ 对 $k$ 进行加密得到 $E$ ，再将 $E$ 回传给Alice即可。

标签: [密码学](#)

好文要顶 关注我 收藏该文



stevenczp

关注 - 1

粉丝 - 34

+加关注

0

0

« 上一篇: [近期计划](#)

» 下一篇: [Linux Performance Observability Tools](#)

posted @ 2017-03-14 00:33 [stevenczp](#) 阅读(344) 评论(0) [编辑](#) [收藏](#)

[刷新评论](#)[刷新页面](#)[返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)， [访问](#) 网站首页。

## 最新 IT 新闻:

- [虎扑直男“炮轰”B站UP主：1千万播放量到底该捐多少](#)
  - [用着我一样的设备 NASA科学家们在家操控火星探测器](#)
  - [腾讯安全与北京市方正公证处战略合作](#)
  - [AI做“军师”？先赢过Reddit用户再说吧](#)
  - [星海SA2云服务器助力腾讯广告“千人千面”，广告计算提速25%](#)
- » [更多新闻...](#)

## 公告

昵称: [stevenczp](#)

园龄: [8年4个月](#)

粉丝: [34](#)

关注: [1](#)

[+加关注](#)

[< 2020年4月 >](#)

日 一 二 三 四 五 六

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 7 8 9

## 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](#)
- [更多链接](#)

## 随笔档案

- [2018年8月\(1\)](#)
- [2018年7月\(1\)](#)
- [2018年3月\(2\)](#)
- [2017年12月\(4\)](#)
- [2017年11月\(2\)](#)
- [2017年10月\(1\)](#)
- [2017年9月\(15\)](#)
- [2017年8月\(8\)](#)
- [2017年7月\(28\)](#)
- [2017年6月\(1\)](#)
- [2017年5月\(3\)](#)
- [2017年4月\(6\)](#)
- [2017年3月\(18\)](#)
- [2017年2月\(5\)](#)
- [2017年1月\(4\)](#)
- [2016年12月\(1\)](#)
- [2016年11月\(2\)](#)
- [2016年10月\(1\)](#)
- [2016年4月\(1\)](#)
- [2016年2月\(3\)](#)
- [2015年11月\(3\)](#)
- [2015年10月\(2\)](#)
- [2015年9月\(3\)](#)
- [2015年8月\(1\)](#)
- [2015年2月\(1\)](#)
- [2014年8月\(4\)](#)

## 最新评论

- [1. Re:如何在windows平台下使用hsdis与jitwatch查看JIT后的汇编码](#)
- @stevenczp OK 谢谢博主 我这两天也是研究JVM执行 而且下午又看到了volatile这个东西...
- --夏漪
- [2. Re:如何在windows平台下使用hsdis与jitwatch查看JIT后的汇编码](#)
- @夏漪 下一篇文章中有用到这个分析技术 没明白你说的是啥, 最后一张图里, 左边的是 source (java 源码), 中间的是 bytecode (字节码), 右边的是 assembly (汇编码) ...
- --stevenczp
- [3. Re:如何在windows平台下使用hsdis与jitwatch查看JIT后的汇编码](#)
- 请问下博主,什么情况下会用到JIT后的汇编码,是查看代码执行过程吗? 然后使用方法中,e小点,界面中间一栏应该是执行码(操作码),博主说的最后一栏应该是指令码...
- --夏漪
- [4. Re:MySQL InnoDB MVCC深度分析](#)
- 你好, 这篇博客写得简洁又通透! 请问可以转载吗? 想转载在csdn上.
- --heqianwu
- [5. Re:HashMap在Java1.7与1.8中的区别](#)
- jdk1.7中, 有的版本是用hashcode的取模运算, 有的是用下面的位运算, 位运算的性能要高于取模运算static int indexFor(int h, int length) { return ...
- --okguo

## 阅读排行榜

- [1. HashMap在Java1.7与1.8中的区别\(19079\)](#)
- [2. Netty源码学习 \(二\) NioEventLoopGroup\(10784\)](#)
- [3. MySQL创建存储过程/函数需要的权限\(7915\)](#)
- [4. Netty源码学习 \(五\) ChannelInitializer\(6269\)](#)
- [5. 为什么分布式数据库中不使用uuid作为主键? \(6119\)](#)

## 评论排行榜

- [1. 2016校招记录\(7\)](#)
- [2. HashMap在Java1.7与1.8中的区别\(4\)](#)
- [3. 如何在windows平台下使用hsdis与jitwatch查看JIT后的汇编码\(3\)](#)
- [4. Guava源码学习 \(四\) 新集合类型\(2\)](#)
- [5. MySQL InnoDB MVCC深度分析\(1\)](#)

## 推荐排行榜

- [1. HashMap在Java1.7与1.8中的区别\(3\)](#)
- [2. MySQL InnoDB MVCC深度分析\(2\)](#)
- [3. Ticket Lock, CLH Lock, MCS Lock\(1\)](#)
- [4. Netty源码学习 \(一\) Netty线程模型\(1\)](#)
- [5. 重复造轮子之RSA算法\(一\) 大素数生成\(1\)](#)

Copyright © 2020 stevenczp

Powered by .NET Core on Kubernetes