33 非对称加密和使用SSH建立安全连接

更新时间：2019-07-31 09:23:07

> 读书而不思考，等于吃饭而不消化。
>
> ——波尔克

内容简介

1. 前言
2. 非对称加密
3. 用 SSH 创建一个安全的通信管道
4. 用 SSH 进行连接
5. 用密钥实现自动身份验证
6. 总结
7. 第四部分第四课预告

1. 前言

上一课 带你玩转Linux和Shell编程 | 第四部分第二课：连接到远程终端和对称加密 中，我们初步认识了加密技术，了解了加密的必要性，介绍了对称加密的方法。

但是对称加密有一个致命缺陷：必须谨慎地传递密钥。但这几乎是不可能的：因为首先得把密钥传递过去。如果用明文传递密钥，就会存在安全隐患。

所以我们来学习非对称加密，它可以为我们加密用于对称加密的密钥。

接着我们再学习如何用 SSH 来建立安全连接。

2. 非对称加密

非对称加密，英语是 Asymmetric-Key Encryption。asymmetric 是“非对称的”的意思，key 是“钥匙/密钥”的意思，encryption 是“加密”的意思。所以全称其实是“非对称密钥加密”。

- 对称加密方法中，我们只用一个密钥来进行加密和解密。这也是“对称”一词的由来。
- 非对称加密方法中，我们用一个密钥来进行加密，用另一个密钥来解密。因为两个密钥不一样，所以是“非对称”。

非对称加密有两个密钥：

- 一个是"公钥"（Public Key），用于加密。public 是英语“公共的，公开的”的意思。
- 一个是"私钥"（Private Key），用于解密。private 是英语“私有的”的意思。

公钥只用来加密。因此，用非对称加密的算法，我们就只能用私钥来解密。

我们请求电脑为我们生成这一对密钥：一个私钥和一个公钥。它们总是成对出现。

> 但是您不要问我生成这一对密钥的原理，也不要问我为什么它们总是成对出现。因为原理很复杂，我们暂时只要知道怎么用就好了。如果你想深入了解，自己去网上搜索资料即可。

假设我们有如下一对密钥：

- 公钥：**2K49c8uE**
- 私钥：**5z3sR6Ln**

为了加密，我们要用到公钥，如下图所示：

用 **2K49c8uE** 这个公钥加密

而解密呢，公钥就派不上用场了，必须用私钥才行。“公私分明，方为大德”嘛。如下图所示：

用 **5z3sR6Ln** 这个私钥解密

这就是为什么我们称这种方法为非对称的原因：需要两个不同的密钥。其中一个用于加密，另一个用于解密。

公钥可以在网络上以明文传输，毕竟是公开的密钥嘛。即使公钥被不怀好意的黑客截获也无所谓。

但是，用于解密的私钥却不能被公开传输，需要保管好。

当然了，非对称算法绝不止一种。在这个大家族中，最有名的要数 RSA 算法了。

1977 年，三位数学家 Rivest、Shamir 和 Adleman 设计了一种算法，可以实现非对称加密。这种算法用他们三个人的名字的首字母命名，叫做 **RSA 算法**。

3. 用 SSH 创建一个安全的通信管道

SSH 结合使用非对称加密和对称加密两种方法

SSH 以如下顺序使用两种加密方法：非对称加密和对称加密。

1. 首先，使用非对称加密，安全地传输对称加密的密钥。
2. 之后，就一直使用对称加密的密钥来作为加密和解密的手段。

聪明如你一定会好奇：“那为什么不只用非对称加密呢？”

当然可以只用非对称加密，但是有一个缺陷：非对称加密太消耗电脑资源了。非对称加密比对称加密要慢大概 100 ~ 1000 倍。

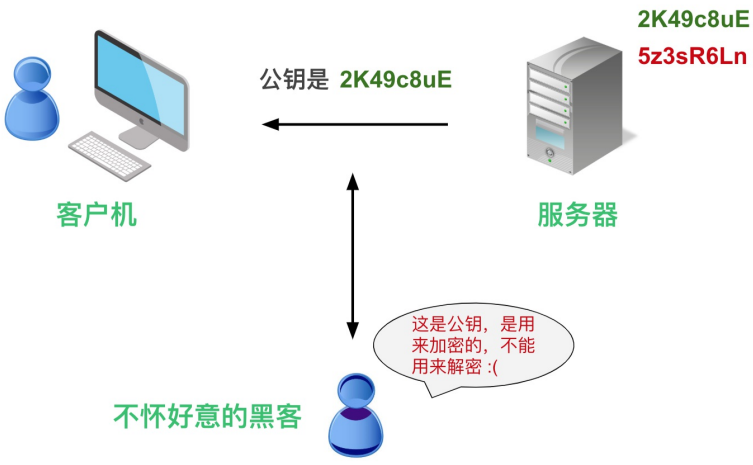
因此，两台电脑之间首先交换对称加密的密钥（用非对称加密的方式），之后就可以用对称加密来通信了，会更快捷。

> 非对称加密只是在通信之初用于交换对称加密的密钥。

让我们用图解的方式来解释一下 SSH 是如何创建一个安全的通信管道的：

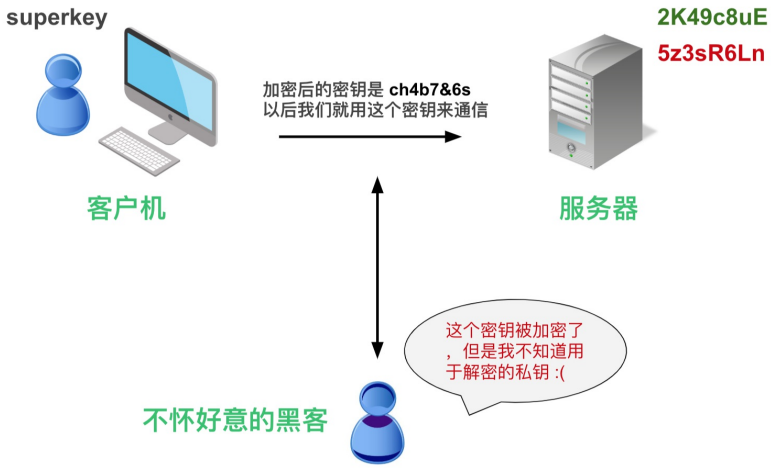
首先，我们要交换一个对称加密的密钥，但是我们又不能以明文方式传输这个密钥，不然黑客截获之后就可以用其来解密了。因此，我们要用非对称加密的方式来加密用于对称加密的密钥（希望您还没晕）。

服务器将非对称加密的公钥以明文方式传输给客户机，使客户机可以用公钥来加密。如下图（用绿色标明公钥，红色标明私钥）：

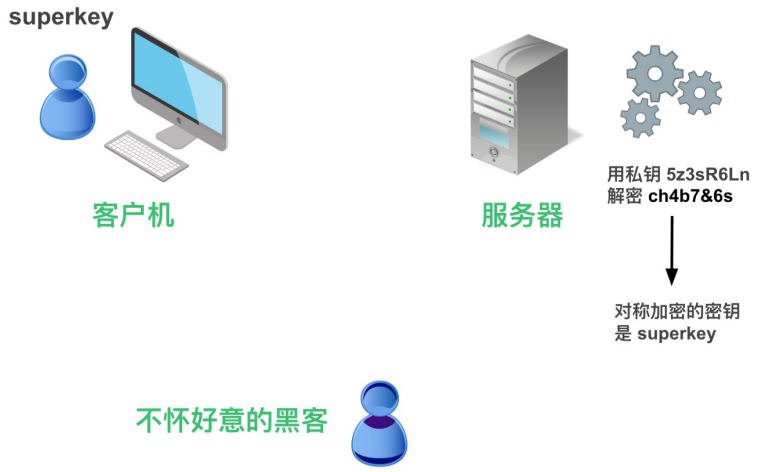


客户机收到服务器传给它的公钥之后，就会用公钥来加密自己的对称加密的密钥，假设是 superkey。如下图所示：

然后，客户机把经过公钥加密后的对称加密密钥传给服务器。黑客可以截获这个加密后的密钥，但是他没办法解密，因为他没有用于解密的私钥，这个私钥只有服务器知道。如下图所示（假设 superkey 这个对称加密密钥，经过公钥 2K49c8uE 加密后，变成了 ch4b7&6s ）：

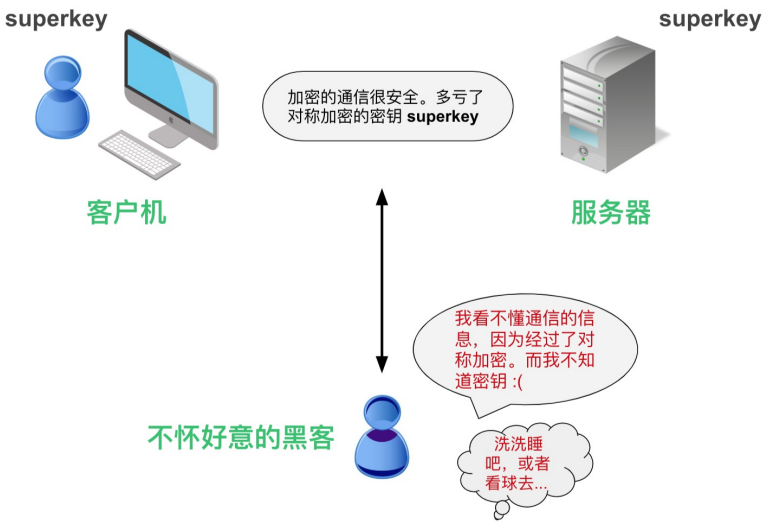


服务器接收到客户机传来的密钥，使用自己的私钥来解密，就得到了对称加密的密钥。如下图所示：



现在，客户机和服务器都知道了对称加密的密钥是 superkey，关键是他们从没在网络间以明文传递过这个密钥。

因此，从现在开始他们可以用对称加密的方式互相发送加密的信息（用 superkey 来加密和解密），不用再担心被黑客获取信息了。如下图所示：



以上就是 SSH 的工作原理了。是不是很巧妙地创建了一条安全通信的管道呢，设计这些的 IT 前辈们真的非常睿智。

现在既然客户机和服务器已经有了安全的通信方式，客户机就可以放心地将自己的登录名和密码传输给服务器，以连接服务器了。如下图：

那么，须要知道以上这些原理才能使用 SSH 吗？

当然不是啦，以上这些都是自动完成的。为了连接远程电脑，你要做的只是输入登录名和密码就好了。

> 现在全球都在使用 SSH，几乎没有人再用 Telnet 了。

4. 用 SSH 进行连接

> 我们接下来的命令都是以 Debian 系列（Ubuntu 为代表）来演示的，其他 Linux 发行版的命令格式类似。

好了，理论谈得够多了，该实战了。你会发现 SSH 使用起来很简单，因为电脑替我们做了大部分工作。

接下来我们分为两种情况：

你已经租用了一台服务器，那这台服务器因为已经配置好了作为 SSH 服务器，所以你什么也不需要做。估计站长大多是这种情况吧。

你没有租用一台服务器（大多数人的情况）。我们就来看看怎么把你自己的电脑配置成 SSH 服务器。

将你的电脑配置成 SSH 服务器

假如你要将自己的电脑配置成 SSH 服务器，以便自己或别人以后可以远程用 SSH 登录你的电脑，你可以这么做：

首先，安装 openssh

> OpenSSH 是 SSH 协议的免费开源实现。open 是“开放的”的意思。

OpenSSH 的官网是 https://www.openssh.com 。

OpenSSH 分客户端和服务端：

- 客户端：openssh-client
- 服务端：openssh-server

client 是“客户”的意思，server 是“服务器”的意思。

如果你只是想用 SSH 远程连接到别的机器，那只需要安装 openssh-client（Ubuntu 默认已经安装。如果没有，则用 `sudo apt install openssh-client` 来安装）。

如果要使你自己的机器开放 SSH 服务，则需要安装 openssh-server，运行如下命令来安装服务端：

```
sudo apt install openssh-server
```

安装完成后，它会自动开启 sshd 这个精灵进程（Daemon Process，或称为“守护进程”。是一种运行在后台的特殊进程）。

你也可以手动开启 sshd：

```
# Ubuntu 系统
sudo service ssh start
```

```
# Debian 系统
sudo /etc/init.d/ssh start
```

start 是英语“启动，开始”的意思。

要停止的话：

```
# Ubuntu 系统
sudo service ssh stop
```

```
# Debian 系统
sudo /etc/init.d/ssh stop
```

stop 是英语“停止”的意思。

如果你要对 SSH 的配置做修改，可以用文本编辑器（例如 Nano）修改 `/etc/ssh/ssh_config`，然后运行

```
sudo /etc/init.d/ssh reload
```

或

```
sudo service ssh reload
```

来使修改生效。reload 是英语“重新载入”的意思。

从一台 Linux 电脑上通过 SSH 连接

以下操作的前提是：你的系统上已经安装了 OpenSSH 的客户端（openssh-client）。如果没有，则用 `sudo apt install openssh-client` 来安装。

假设你要以用户名 user，用 SSH 协议登录远程服务器（主机名是 host），只要一条简单命令就可以了：

```
ssh user@host
```

当然了，这里的 host（主机名），也可以用主机的 IP 地址来替换，例如：

```
ssh root@123.45.67.890
```

如果本地用户名与远程用户名一致，登录时可以省略用户名：

```
ssh host
```

SSH 的默认端口是 22。也就是说，你的登录请求会被送进远程服务器的 22 端口。

使用 p 参数，可以修改这个端口（p 是 port 的缩写，表示“端口”）：

```
ssh -p 250 user@host
```

上面这条命令表示：SSH 直接连接远程服务器的 250 端口。

如果你是第一次登录远程服务器，系统会出现类似下面的提示：

```
The authenticity of host 'host (12.18.429.21)' can't be established.
RSA key fingerprint is 98:2e:d7:e0:de:9f:ac:67:28:c2:42:2d:37:16:58:4d.
Are you sure you want to continue connecting (yes/no)?
```

这段话的意思是：无法确认 host 服务器（IP 地址是 12.18.429.21）的真实性，只知道它的公钥指纹（`98:2e:d7:e0:de:9f:ac:67:28:c2:42:2d:37:16:58:4d`），你还想继续连接吗？

所谓"公钥指纹"，是因为公钥长度较长（这里采用 RSA 算法，长达 1024 位），很难比对，所以对其进行 MD5 计算，将它变成一个 128 位的指纹（也就是上例中的`98:2e:d7:e0:de:9f:ac:67:28:c2:42:2d:37:16:58:4d` 这一串十六进制数），再进行比较，就容易多了。

很自然的一个问题就是：用户怎么知道远程服务器的公钥指纹应该是多少？回答是没有好办法，远程服务器必须在自己的网站上贴出公钥指纹，以便用户自行核对。

假定经过风险衡量以后，用户决定接受这个远程服务器的公钥（输入 yes，回车）：

```
Are you sure you want to continue connecting (yes/no)? yes
```

系统会出现一句提示，表示 host 主机（也就是我们要操作的远程电脑）已经得到认可：

```
Warning: Permanently added 'host,12.18.429.21' (RSA) to the list of known hosts.
```

然后，会要求输入密码：

```
Password: (enter password)
```

如果密码正确，就可以登录了。

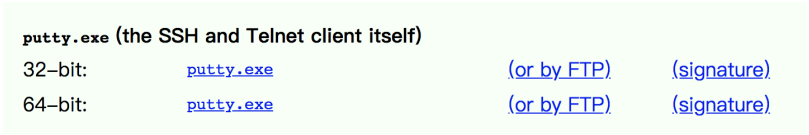
当远程服务器的公钥被接受以后，它就会被保存在文件 $HOME/.ssh/known_hosts 之中（HOME 是环境变量，通常保存了用户家目录的绝对路径，比如我的 HOME 就是 /home/oscar）。下次再连接这台服务器时，系统就会认出它的公钥已经保存在本地了，从而跳过警告部分，直接提示输入密码。

每个 SSH 用户都有自己的 known_hosts 文件。此外系统也有一个这样的文件，通常是 /etc/ssh/ssh_known_hosts，保存一些对所有用户都可信赖的远程服务器的公钥。

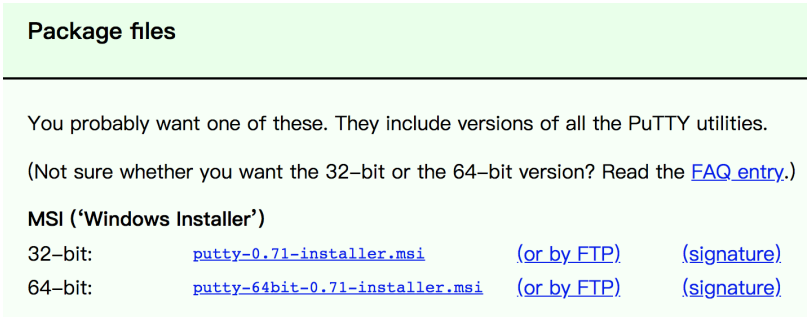
从一台 Windows 电脑上通过 SSH 连接

从 Windows 电脑要连接到远程的 SSH 服务器，有不少软件可以帮助我们。不过一般常用的是 PuTTY 这个软件。

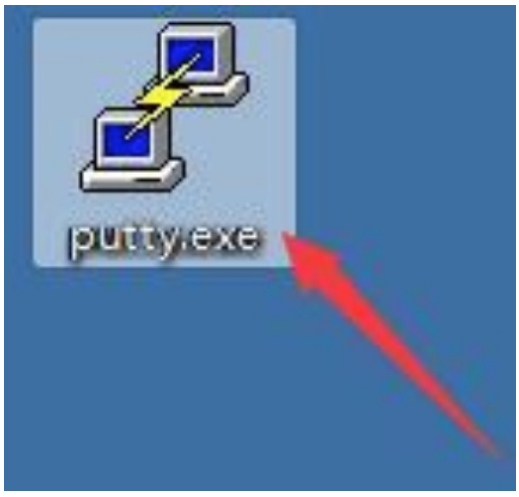
PuTTY 这个软件可以从它的官网（ https://putty.org ，尽量不要去第三方软件网站下载，以防有内置恶意程序）下载可直接运行的可执行程序 putty.exe，一般下载 64-bit（64 位） 的那个：



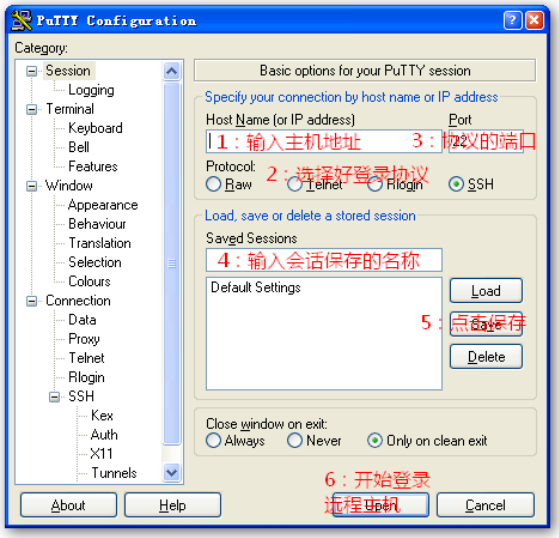
也可以下载安装程序（例如 putty-64bit-0.71-installer.msi）来安装：



下载或安装完毕后双击软件图标打开软件。

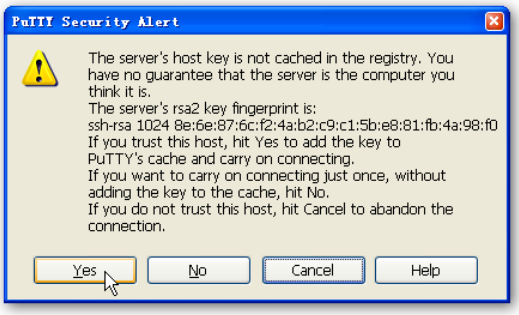


打开 PuTTY 后，可以看到如下窗口



在上图的“1：输入主机地址”的方框处填入要连接的服务器的 IP 地址。

你可以保存本次的 session（“会话”的意思）设置（上图中 4 和 5 两步），以方便下次登录。也可以不保存，直接点击 Open 按钮（上图中第 6 步），如果是首次连接，会弹出以下窗口，点击 yes（“是”）即可：

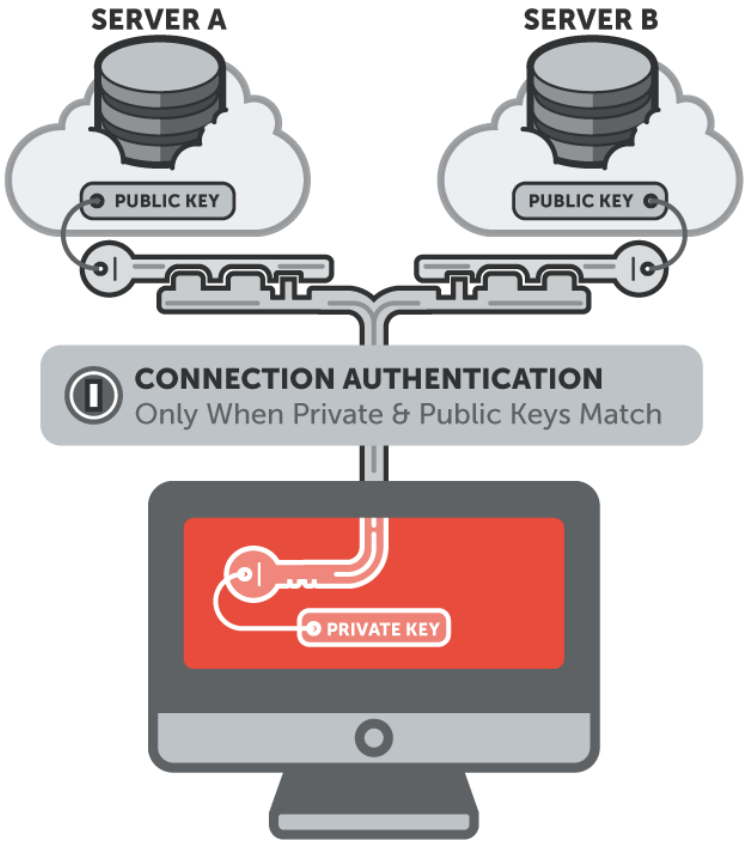


在弹出的命令行窗口中依次输入用户名和密码。注意 Linux 系统下输入的所有密码都是不可见的（也不会用星号表示），所以你不必以为是键盘坏了或者输入不起作用，其实已经输入了：

5. 用密钥实现自动身份验证

使用密码登录，每次都必须输入密码，非常麻烦。幸亏 SSH 还提供了公钥登录，可以省去输入密码的步骤。

> 所谓"公钥登录"，原理很简单，就是用户将自己的公钥储存在远程服务器上。登录的时候，发送一个经过公钥加密的随机数据给客户机，这个数据只能通过私钥解密，客户机将解密后的信息发还给服务器，服务器验证正确后即确认客户机是可信任的，从而建立起一条安全的信息通道，直接允许登录 Shell，不再要求密码。



这种方法要求用户必须提供自己的公钥。如果没有现成的，可以直接用 ssh-keygen 命令生成一个：

```
ssh-keygen
```

运行上面的命令以后，系统会出现一系列提示，可以一路回车。其中有一个问题是，要不要对私钥设置口令（passphrase），如果担心私钥的安全，这里可以设置一个。一般都不设置。

运行结束以后，在 $HOME/.ssh/ 目录下，会新生成两个文件：id_rsa.pub 和 id_rsa。前者是你的公钥，后者是你的私钥。

```
oscar@oscar-laptop: ~
File  Edit  View  Search  Terminal  Help
oscar@oscar-laptop:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/oscar/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/oscar/.ssh/id_rsa.
Your public key has been saved in /home/oscar/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:plcs+xyzF6l1kj1Ye93QkGkULDoO/QzeFx8fZhEChi8 oscar@oscar-laptop
The key's randomart image is:
+---[RSA 2048]----+
|          .o.+o=.|
|         .. . B. |
|         ... o o.|
|        .E=.  +=.|
|        S++* *o=*|
|       o +o @ * *|
|      . o oo = o |
|       . o.+.    |
|         +.      |
+----[SHA256]-----+
oscar@oscar-laptop:~$ ls .ssh
id_rsa  id_rsa.pub
oscar@oscar-laptop:~$
```

这时再运行下面的命令，将公钥传送到远程服务器 host 上面：

```
ssh-copy-id user@host
```

好了，从此你再登录，就不需要输入密码了。

如果还是不行，就打开远程服务器的 /etc/ssh/sshd_config 这个文件，检查下面几行前面的"#"注释是否已经去掉。

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
```

然后，重启远程服务器的 ssh 服务：

```
# Ubuntu 系统
sudo service ssh restart
```

```
# Debian 系统
sudo /etc/init.d/ssh restart
```

6. 总结

对称加密只用一个密钥来实现加密和解密。非对称加密则用两个不同的密钥来实现加密和解密，负责加密的称为公钥，负责解密的称为私钥。

在 Windows 下，为了远程连接到 Linux 系统，一般我们用 PuTTY 这个软件。

在 Linux 和 macOS 下，为了远程连接到 Linux 系统，我们可以用 ssh 命令，为它指定在远程 Linux 机器上的登录名（login）和远程 Linux 机器的 IP 地址。例如：

```
ssh oscar@79.27.172.59
```

通过 SSH 协议， 两台机器之间传递的信息会被加密，这样就保证了传输信息的安全性。SSH 使用了非对称加密和对称加密。

为了免去每次用 SSH 协议连接远程机器都要输入用户密码的麻烦，我们可以创建一个用于验证身份的密钥对（公钥和私钥）。公钥需要传输并储存到远程机器上，私钥则存在我们自己的电脑里。之后，我们的 SSH 连接就不需要输入密码了。

今天的课就到这里，一起加油吧！

32 连接远程终端和对称加密，安全快捷

34 文件传输，潇洒同步