第3章 密码技术应用

密码学是消息加密、身份认证、完整性检验等网络安全技术的理论基础，也是网络通信安全的重要支撑。本章将介绍系统安全访问与口令破解、公钥基础设施（Public Key Infrastructure，PKI）、完善保密协议（Pretty Good Privacy，PGP）等加密与身份认证技术，带领读者了解密码学在网络安全的应用。

基于口令的认证技术，是基于密码技术的典型应用。口令的选取、注册存储与验证是基于口令认证技术的关键。弱口令往往是系统安全的最短板，口令破解则是攻击者最常用的技术手段。PKI则用于网络通信中的身份认证的实现，其核心技术双钥密码使网络中的通信双方无须协商密钥即可进行加密的信息交换。PKI将用户身份与其公钥绑定，使公钥所有者的身份可信，提供身份认证、数据完整性、数据保密性等安全服务，因而被广泛地应用于安全浏览器、电子数据交换、安全电子邮件中。PGP则是使用多种密码学技术实现的一套用于消息加密与消息验证的应用程序，在安全电子邮件、安全Web应用中提供消息机密性和来源认证服务。与PKI不同，PGP支持通过信任网络分布式地实现公钥信赖管理。

通过本章的实验，读者将掌握口令破解、PKI以及PGP的基本原理与工具使用方法。

3.1 口令破解

3.1.1 实验目的

熟悉Linux系统的安全访问机制，了解典型账号口令破解技术的基本原理、常用方法及相关工具，掌握防范此类攻击的方法及措施。

3.1.2 实验内容

查看Linux系统账户信息文件，了解其安全访问机制，利用John the Ripper软件破解账户口令，了解字典攻击的原理及不足。

3.1.3 实验原理

1. Linux中的口令存储

Linux是一个多用户多任务的分时操作系统，允许多个用户共享使用同一台计算机资源。用户的账号在该机制下，一方面，可以帮助用户组织文件，保障安全性；另一方面，可以帮助系统管理员对当前使用计算机的用户进行跟踪，控制它们访问系统资源的权限。通常，系统将所有用户的账户信息存储在公共文件/etc/passwd中。该文件中每一行代表一个账户，不同域使用冒号（:）隔开，代表不同信息，各域的含义如图3-1所示，读者可以通过man

5 passwd 命令查看各个域的具体含义。

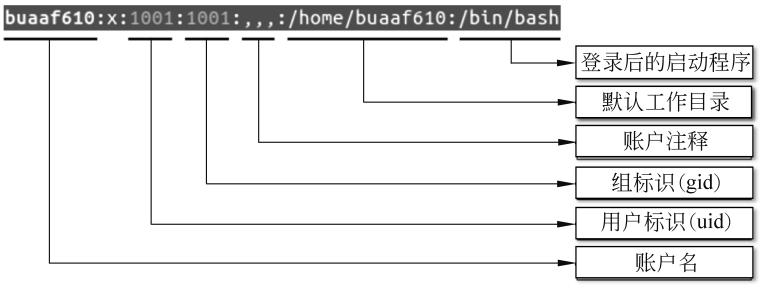

图 3-1　passwd 文件各域信息

- **账户名**是代表用户账号的字符串，其长度通常不超过 8 个字符。
- **口令**只存放一个特殊的字符，如“x”或“*”，真正加密后的用户口令存放在/etc/shadow 文件中。
- **用户标识(uid)**是一个整数，是用户在系统中的唯一标识号，通常情况下用户标识与账户名一一对应。当多个账户名对应同一用户标识的情况出现时，尽管它们可以有不同的口令、主目录等，系统会将它们视作同一个用户。0 是超级用户 root 的标识，1～99 由系统保留作为管理账号，普通用户的标识号从 100 开始。
- **组标识(gid)**是用户所属用户组的标识号，对应着/etc/group 文件中的一条记录。
- **账户注释**记录用户的个人信息，包括用户的真实姓名、电话与地址等。
- **默认工作目录**是当前用户的起始工作目录。在 Linux 系统中，所有用户的工作目录都被组织在一个特定的目录下，且目录的名称与用户的账户名称对应。各用户对自己的工作目录有读、写、执行的权限。
- **登录后的启动程序**是用户登录到系统后运行的命令解释器或某个特定程序，即 shell。Linux 默认的 shell 是 bash，此外还有 sh，csh，ksh，tcsh 等。

如上文所述，账户口令信息存储在受保护文件/etc/shadow 中，非 root 用户无法打开。同样，该文件中每一行代表一个账户，与/etc/passwd 文件中的账户一一对应。每一行的不同域使用冒号隔开，代表不同信息，各域的含义如图 3-2 所示，其中，口令时间相关信息包括口令最后被修改的时间、修改口令的最小期限、修改口令的最大期限、口令过期提醒时间、口令过期失效时间、账户过期时间等，可以通过 man 5 shadow 命令查看各个域的具体含义。

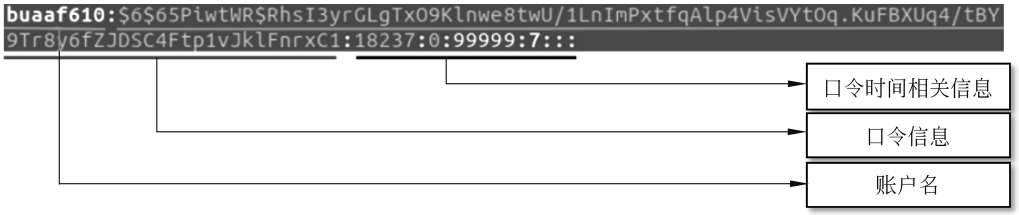

图 3-2　shadow 文件各域信息

从 shadow 文件内容中可以看出，账户的口令并不以明文形式存储。Linux 系统存储的是口令加盐之后的哈希值，其口令信息的格式为 idsalt$hash，其中，id 代表不同的哈希

方式,包括1 == md5,5 == sha256,6 == sha512 等。

2. 口令破解攻击

通常,对于口令的破解攻击方法包括穷举攻击、字典攻击、混合攻击等,其中,穷举攻击穷举所有可能的口令组合,加密每个组合,并将已知的密文与加密结果进行比对,试图找到已知密文对应的明文;字典攻击是从字典文件(也称彩虹表)中寻找可能的口令;而混合攻击则是结合上述两种猜测方式进行攻击。常见的口令破解工具包括 Cain,John the Ripper,Pandora,LC5 等。

John the Ripper 软件是一个快速的口令破解工具,用于在已知密文的情况下尝试破解出明文。它支持破解目前大多数的加密算法,如 DES,MD4,MD5 等。它有 3 种破解模式,包括字典破解(Wordlist crack)模式、简单破解(Single crack)模式、增强破解(Incremental)模式。

(1) 字典破解模式

John the Ripper 实施字典攻击的模式,将字典文件与亟待破解的密码文件作为参数。字典模式还可以施加单词变化规则(Word mangling rules)——通过将其应用于单词列表文件中的每一行,使每个源单词衍生多个候选密码(如字母大小写变换),以增加破解概率。

(2) 简单破解模式

John the Ripper 将用户账户名、主目录名称等信息,以及其应用变换规则后得到的大量衍生字符串作为候选密码实施破解工作,该模式下破解速度最快。

(3) 增强破解模式

John the Ripper 尝试所有可能的密码组合,是最具威力的破解模式。然而尝试所有的字元组合,因此破解时间十分冗长。

John the Ripper 的命令行工具 john,具体参数选项如表 3-1 所示。

表 3-1 John the Ripper 命令行工具的参数及含义

参　数	含　义
-single	使用简单破解模式
-wordlist=FILE-stdin	字典模式,从 FILE 或标准输入中读取词汇
-rules	应用单词变换规则
-incremental[=MODE]	使用增强破解模式
-show	现实已破解的口令
-test	执行破解速度测试

3.1.4 实验步骤

本次实验继续在配置好的虚拟环境中操作。安装实验所需软件:

```
$sudo apt-get install john locate
```

其中,john 用于进行口令破解实验,locate 用于查找指定文件名的文件。请在虚拟机~/目录下创建 password 目录:

```
$mkdir password/
```

```
$cd password
```

1. 添加用户

按表 3-2 添加用户并设定口令。

表 3-2　实验添加用户名及口令

用　户　名	口　　令	用　户　名	口　　令
User1	Hello	User5	Hellodragon
User2	123	User6	123Hello
User3	Flower	User1	Hello
User4	Dragon		

添加用户账号并设定口令的命令如下：

```
// 添加用户账号
$sudo useradd [Username]
// 设定用户账户口令
$sudo passwd [Username]
// 根据提示输入口令
// 根据提示再次输入口令确认
// 请注意,两次输入口令都不会在屏幕上显示
```

任务 3.1　添加所有用户后，请利用以下命令查看口令文件内容与权限，截图记录查看结果。

```
$cat /etc/passwd
$sudo cat /etc/shadow
$ls -l /etc/passwd
$ls -l /etc/shadow
```

2. 利用 John the Ripper 软件破解口令

组合口令文件：

```
$sudo unshadow /etc/passwd /etc/shadow >test_shadow
```

查找字典表文件的路径：

```
$sudo updatedb
$sudo locate password.lst
```

运行 John the Ripper 软件破解刚刚创建的口令文件，其中，[list_path]为之前找到的字典表文件路径：

```
$john --wordlist=[list_path] test_shadow
```

运行过程中，按空格键可以查看运行进程，按 Ctrl＋C 键停止进程。完成后，查看破解结果：

```
$john --show test_shadow
```

任务 3.2　按以上步骤完成实验，截图记录实验过程，观察并解释破解结果。

3. 增强 John the Ripper 软件口令破解能力

在图形界面查看字典表文件,按 Ctrl+F 键查找单词,发现该文件存在单词 flower,但不存在单词 Flower。在运行 John the Ripper 软件时,可以利用-rules 参数应用单词变化规则尝试字典文件中单词的其他可能。

任务 3.3 利用-rules 参数再次破解口令文件,截图记录运行命令与破解结果,并回答:这次破解的运行时间如何?为什么运行时间出现这样的变化?

3.2 公钥基础设施

3.2.1 实验目的

熟悉 PKI 的基本组成,及其解决的安全问题;了解 CA 在 PKI 中的作用以及其签发证书的流程;了解数字证书的用途和 X.509 数字证书格式。

3.2.2 实验内容

在实验中实现建立 CA 以及 CA 签发证书过程,查看生成的证书,了解数字证书的构成;并将证书用于 HTTPS Web 服务器,了解浏览器验证网站证书的过程。

3.2.3 实验原理

1. 中间人攻击

公钥加密是现代加密通信的基础,但是它在公钥共享阶段很容易受到中间人(Man-in-the-Middle, MITM)攻击。中间人攻击发生在两个设备的通信中,攻击者与受害者建立单独的通信,并在受害者之间传递消息。受害者认为他们正在与对方进行秘密通信,而实际上整个通信均由攻击者控制。图 3-3 展示了一个典型的中间人攻击场景,其中,Mallory 是攻击者,他能够截获 Alice 以及 Bob 的通信,实行中间人攻击的步骤如下:

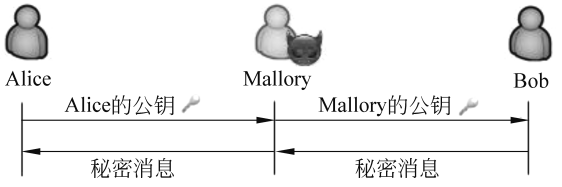

图 3-3 中间人攻击示意图

- Mallory 截获 Alice 送出的公钥,并将自己的公钥转发给 Bob。
- Bob 认为接收到的公钥是 Alice 的,利用接收到的公钥加密一条消息并发出。
- Mallory 截获 Bob 发出的秘密消息,由于该消息使用 Mallory 自己的公钥加密,因此 Mallory 可以解密该消息,得知消息内容,并用 Alice 的公钥再次加密后转发给 Alice。
- Alice 可以解密收到的消息,得知消息内容。

如果这个攻击发生在秘密通信信道建立前双方协商密钥的阶段，那么该场景中的消息就是秘密信道的密钥。因此，Mallory 就可以利用获取到的密钥，解密 Alice 与 Bob 之间的所有通信消息。

从上述的攻击场景可以发现，中间人攻击能够成功的原因是，当 Bob 收到一份声称来自 Alice 的公钥时，他不能判断这个公钥究竟属于谁。因此，要抵御中间人攻击，需要对收到的公钥进行身份认证。

2. 公钥基础设施

PKI 是一种用公钥密码理论和技术实施和提供安全服务，具有普适性的安全基础设施。在实际应用中，PKI 将用户身份与其公钥结合，其核心组件是证书机构（Certificate Authority，CA）。CA 负责验证用户的身份，并签发对应数字证书。数字证书则是证明证书中公钥所有权的文件，由已经验证该公钥所有权的 CA 签发，因此，数字证书的安全性基于对 CA 的信任。

基于 PKI/CA 实现的证书签发以及验证过程如图 3-4 所示。

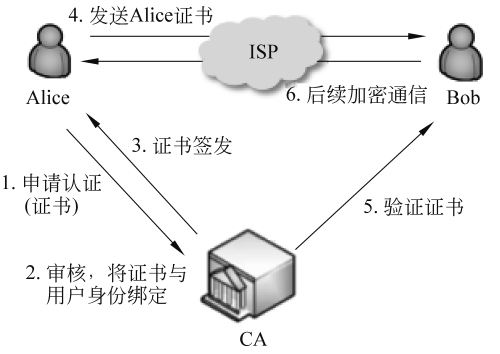

图 3-4　证书签发与验证流程

（1）证书签发。

① 用户 Alice 向 CA 机构提交个人信息、注册密钥信息。

② CA 机构验证与审核用户 Alice 的身份，将其密钥对的公钥、个人信息经过自身私钥加密后封装成一张数字证书。

③ 将生成的数字证书签发给用户 Alice。

（2）证书验证。

① 用户 Alice 将数字证书发送给 Bob。

② 用户 Bob 使用 CA 的公钥检测证书有效性。

③ 如 Bob 证书有效，则 Alice 后续可用 Bob 公钥加密与其通信的信息。

3. X.509 数字证书

数字证书一般包括公钥、拥有者的身份标识以及可信实体的签名。接收者可以通过验证签名来保证证书的完整性。验证成功后，接收者将会确定公钥的拥有者。X.509 标准规定了数字证书的格式，主要包括：

- **Issuer**　该域包含签发该证书的 CA 信息。
- **Subject**　该域包含证书中公钥的拥有者信息。
- **Public key**　该域包含公钥信息，包括公钥算法以及具体公钥等。
- **Signature**　该域包含签发者（Issuer）的数字签名信息，包括签名算法以及具体签名等。
- **Validity**　该域包含该数字证书的有效期。
- **Serial number**　每个证书都有一个独特的序列号，用于与其他证书区分。
- **Extensions**　更新版本的 X.509 证书包含可选的扩展域。

3.2.4 实验步骤

安装实验所需软件:

```
$sudo apt-get install apache2
```

其中,apache2 用于搭建 Web 服务器。

将实验所需文件压缩包 PKI.tar.gz 放入虚拟机~/目录下并解压,进入该目录。

公钥加密是现代加密通信的基础,但是它在共享密钥阶段很容易受到中间人攻击。在实际应用中,PKI 利用 CA 将用户身份与其公钥结合,利用数字证书解决公钥的可信性问题。CA 是一个签发电子证书的可信实体,本实验将建立一个根 CA,并为一个网站签发证书。

1. 建立 CA

将 openssl 提供的配置文件/usr/lib/ssl/openssl.cnf 复制到 PKI 目录,按以下形式修改配置文件中的[CA_default]一节,并根据要求创建相应文件夹或文件:

```
dir             =./myCA             #文件夹,存储所有文件
certs           =$dir/certs         #文件夹,存储签发证书
crl_dir         =$dir/crl           #文件夹,存储证书吊销列表
new_certs_dir   =$dir/newcerts      #文件夹,存储新证书
database        =$dir/index.txt     #文件,数据库索引文件
serial          =$dir/serial        #文件,存储当前序列号
```

其中,对于 index.txt 文件,只须创建一个空文件;对于 serial 文件,创建文件后需要输入序列号,以文件形式输入的序列号应为偶数位数的十六进制数字,如 1000,0a 等。

在 PKI 目录下,利用 openssl req 命令为 CA 生成一个自签名证书,该证书证明 CA 可信,并成为根证书:

```
$openssl req -new -x509 -config openssl.cnf -keyout ca.key -out ca.crt
```

其中,参数的具体含义如表 3-3 所示。

表 3-3 openssl req 命令相关参数与含义

参 数	含 义
-new	创建一个证书请求文件,若之后指定了-x509 选项,代表创建自签名证书文件
-keyout [file]	指定自动创建私钥时私钥的输出文件
-out [file]	指定证书请求或自签名证书的输出文件
-config [file]	指定 req 命令的配置文件

在证书生成过程中,需要输入口令(passphrase)与相关信息。**请使用简便易记的口令并牢记**,因为该 CA 在每次签发证书时都会要求输入口令。另外,还需要输入一些相关信息,例如 Country Name, Common Name 等,可以自行输入,也可以按 Enter 键使用默认值。输出文件包括 ca.key 与 ca.crt。具体的 CA 建立流程演示,可参考本书附带的微课视频。

创建自己的 CA

任务 3.4　在实验报告中截图展示这两个文件，写明两个文件分别存储什么内容，并回答：证书中的 Issuer 与 Subject 分别代表什么？为什么 ca.crt 文件中这两个域的数据相同？

注：除了在图形化界面直接查看证书信息外，也可以使用以下命令在命令行查看证书信息：

```
$openssl x509 -in ca.crt -text -noout
```

2. 利用 CA 签发证书

假设有一个网站 PKI.com，需要从 CA 处取得一个电子证书，可以通过以下步骤完成签发流程。

首先，PKI.com 需要生成自己的公私钥对：

```
$openssl genrsa -aes128 -out server.key 2048
```

该命令将生成一对 2048 比特的 RSA 公私钥对，同时将私钥用 AES128 加密，并存储在 server.key 文件中。在生成时，用户需要输入私钥加密所用的口令，同样地，**请使用简便易记的口令并牢记**。可以使用以下命令在命令行查看密钥信息(需要输入私钥加密口令)：

```
$openssl rsa -in server.key -text
```

其次，PKI.com 需要生成证书请求文件(CSR)。该文件包含 PKI.com 的公钥，并会被发送给 CA，请求 CA 对公钥进行签名：

```
$openssl req -new -config openssl.cnf -key server.key -out server.csr
```

在生成证书请求文件时，可在 Common Name 域输入“PKI.com”；对于 extra 中的 attributes 域，可自行输入，或直接按 Enter 键使用默认值；对于其他域，应与 CA 保持一致。

最后，CA 收到证书请求文件(server.csr)后，利用自己的私钥(ca.key)与证书(ca.crt)，签名并生成 PKI.com 所需的 X.509 证书(server.crt)，使用的命令如下所示：

```
$openssl ca -config openssl.cnf -in server.csr \
            -keyfile ca.key -cert ca.crt \
            -out server.crt
```

任务 3.5：在实验报告中截图展示所有生成文件(**server.key**，**server.csr**，**server.crt**)，并指明它们的关系。

3. 验证 CA 签发证书的签名

查看 **PKI.com** 证书(**server.crt**)的相关信息：

```
$openssl x509 -in server.crt -text -noout
```

可以看到证书文件分为 3 部分，Data 域、Signature Algorithm 域与 Signature Value 域。其中，Data 域为证书的基本信息部分，也称为 TBSCertificate(To-Be-Signed Certificate)；而 Signature Algorithm 域是 CA 使用的签名算法；Signature Value 域是 CA 利用指定签名算法对 TBSCertificate 哈希值的签名结果。例如，若签名算法为 sha256WithRSAEncryption，则表示 CA 计算 TBSCertificate 的 SHA256 哈希值，并使用 RSA 算法对该哈希值进行签名。

寻找证书中 TBSCertificate 的位置:

```
$openssl x509 -in server.crt -inform pem -outform pem -out server.pem
// 以 ASN.1 格式解析 PEM 文件
$openssl asn1parse -i -in server.pem
```

该命令的输出如图 3-5 所示。根据 ASN.1 标准的定义,第 2 行(图 3-5 中的方框)表示 TBSCertificate 在 PEM 文件中的偏移量(offset)为 4,总长度为 642(头长度为 4,主体长度为 638)。

```
student@buaacst:~/PKI$ openssl asn1parse -i -in server.pem
    0:d=0  hl=4 l= 918 cons: SEQUENCE
    4:d=1  hl=4 l= 638 cons:  SEQUENCE
    8:d=2  hl=2 l=   3 cons:   cont [ 0 ]
   10:d=3  hl=2 l=   1 prim:    INTEGER           :02
   13:d=2  hl=2 l=   2 prim:   INTEGER           :1000
   17:d=2  hl=2 l=  13 cons:   SEQUENCE
   19:d=3  hl=2 l=   9 prim:    OBJECT            :sha256WithRSAEncryption
   30:d=3  hl=2 l=   0 prim:    NULL
   32:d=2  hl=2 l=  69 cons:   SEQUENCE
   34:d=3  hl=2 l=  11 cons:    SET
   36:d=4  hl=2 l=   9 cons:     SEQUENCE
   38:d=5  hl=2 l=   3 prim:      OBJECT            :countryName
```

图 3-5　证书 ASN.1 解析结果

根据解析的信息提取 TBSCertificate,保存为 server.tbs 文件,并计算该文件的哈希值(哈希算法以证书指定版本为准,此处以 SHA256 算法为例):

```
$openssl asn1parse -in server.pem -strparse 4 -out server.tbs
$sha256sum server.tbs
```

任务 3.6　从 server.crt 中提取该证书的签名,从 ca.crt 中提取 CA 的公钥(Exponent 与 Modulus),填入实验给定的 Python 代码中,并补充代码文件,使其完成利用 CA 公钥对该签名的验证过程,即对比 Python 程序输出与上述计算的哈希值。

注:

(1) 在 Python 中,可以用 pow(x,y,z)计算$x^y \bmod z$。

(2) Python 程序的输出由 3 部分组成,分别为 RSA 算法的填充("1fff…"),ASN.1 SHA256 算法指定前缀("3031300d060960864801650304020105000420")与哈希值。

4. 将证书用于 HTTPS Web 服务器

CA 签发的证书可以用于建立 HTTPS 连接。本实验将利用 openssl 为 PKI.com 网站建立一个简单的 HTTPS Web 服务端。

首先,配置 DNS 服务。为了让虚拟机能够解析该域名,需要将以下字段填入虚拟机/etc/hosts 文件中;该字段将主机域名 PKI.com 映射到虚拟机本地机(127.0.0.1 为回送地址)。

```
127.0.0.1 PKI.com
```

然后,基于之前生成的证书,使用 openssl s_server 命令建立一个简单的 Web 服务器。

```
// 将密钥与证书合成为一个文件
```

```
$cp server.key server.pem
$cat server.crt >>server.pem
// 利用 server.pem 建立一个 Web 服务器
$openssl s_server -www -cert server.pem
// 使用 CTRL+C 键可以关闭该服务器进程
```

openssl s_server 命令建立的服务器默认监听 4433 端口，因此，可以通过以下 URL 访问该服务器：https://PKI.com:4433/。

任务 3.7　访问设立的 Web 服务器，截图记录访问结果，描述观察到的情况，并解释原因。

注：此时浏览器应出现报警页面，如图 3-6 所示。可以单击“Advanced...”按钮查看报警信息，但不要单击 Advanced 之后弹出页面上的“Accept the Risk and Continue”按钮。

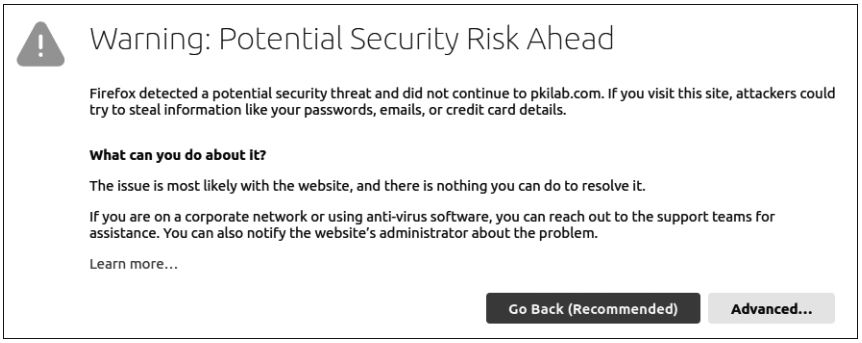

图 3-6　浏览器报警页面

任务 3.8　为了能够顺利访问 Web 服务器，需要在 Firefox 的 CA 列表中加入之前建立的 CA 证书，添加方式参考**附录 G**。再次访问 Web 服务器 https://PKI.com:4433/，截图展示并描述观察到的情况(注意地址栏的图标)。

任务 3.9　本实验在设置域名解析时，将 https://PKI.com:4433/指向本地机，若访问 https://localhost:4433/，会出现什么情况？截图展示访问情况与报警信息，并解释原因。

5. 建立基于 Apache 的 HTTPS 网站

利用 Apache 服务建立一个 HTTPS 网站，需要修改 Apache 配置文件，指定网站文件以及网站密钥与证书的存储位置。例如，要建立一个名为 example.com 的网站，需要在 /var/www 文件夹中新建一个名为 example 的文件夹，在其中新建一个名为 index_https.html 的文件，并填入以下内容(对于熟悉 HTML 的读者，也可以自行编写一个简单的网页文件)：

```
<html>
<head>
<title>example</title>
</head>
<body>
<p>This is the page from HTTPS server.</p>
</body>
```

```
</html>
```

利用以下命令修改/etc/apache2/sites-available/default-ssl.conf 文件：

```
<VirtualHost _default_:443>

ServerName example.com
DocumentRoot /var/www/example
DirectoryIndex index_https.html

SSLEngine on
SSLCertificateFile              #填入网站证书文件的绝对路径
SSLCertificateKeyFile           #填入网站密钥文件的绝对路径

</VirtualHost>
SSLCipherSuite AES256-SHA       #指定加密组件
```

配置文件修改完毕后，需要利用以下命令打开 Apache 的 SSL 模块：

```
// 测试 Apache 配置文件
$sudo apachectl configtest
// 打开 SSL 模块
$sudo a2enmod ssl
// 使用编辑好的网站配置
$sudo a2ensite default-ssl
// 启动或者重启 Apache,取决于当前 Apache 服务的状态
$sudo service apache2 start 或者
$sudo service apache2 restart
```

任务 3.10 利用 CA 为 PKI.com 签发的 **RSA 证书**，配置 Apache 服务，建立一个基于 Apache 的 HTTPS 网站。

(1) 截图展示建立过程。

(2) 截图记录网站访问结果。

3.3 PGP 加解密技术

3.3.1 实验目的

了解 PGP 加解密过程涉及的密钥和 PGP 的工作原理；了解 PGP 信任网络的基本结构；掌握使用 PGP 软件对邮件进行加解密以及签名的方法。

3.3.2 实验内容

熟悉 PGP 软件的命令行操作，熟悉生成 PGP 密钥、导入公钥、证明公钥有效性等基本操作，并使用 PGP 密钥对邮件进行加解密与签名。

3.3.3　实验原理

PGP 是 Phil Zimmermann 于 1991 年开发的一种可为数据通信提供加密与身份认证的程序，常用于电子邮件的加解密与签名，提高了电子邮件通信的安全性。PGP 本身是商业应用程序，开源并具有同类功能的工具是由自由软件基金会开发的 GnuPG(GPG)。PGP 及其同类产品均遵守 OpenPGP 数据加解密标准(RFC 4880)。

PGP 加密由一系列散列、数据压缩、对称密钥加密和公钥加密的算法组合而成。每个步骤均支持几种算法，用户可选，其工作原理如图 3-7 所示。

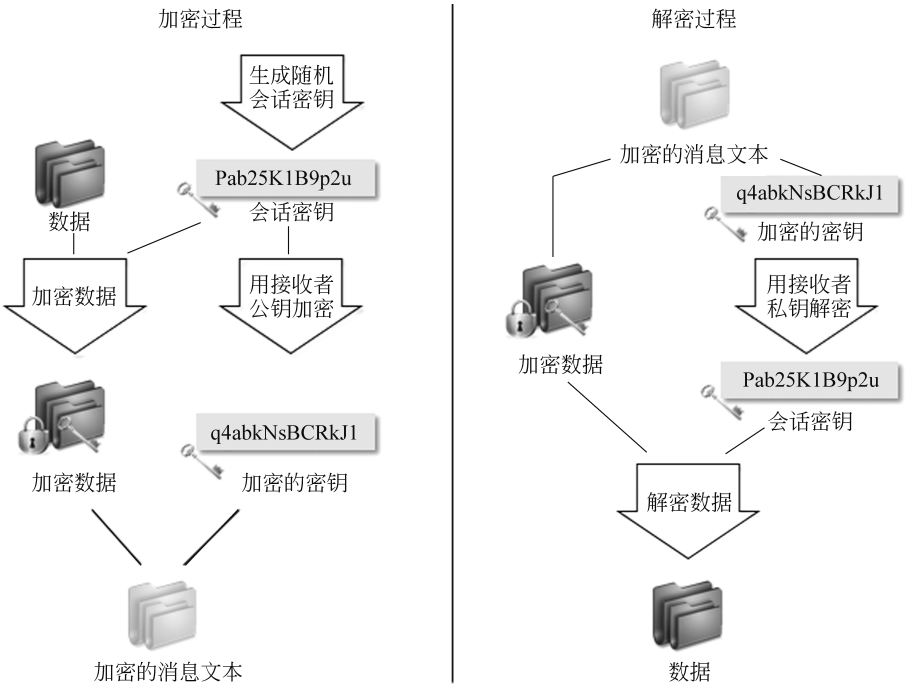

图 3-7　PGP 加解密工作原理示意图

在 PGP 加解密过程中会用到不同的密钥，其用途总结如表 3-4 所示。

表 3-4　PGP 加解密过程所用密钥及其用途

密　钥　名	用　　途
会话密钥	对传送消息的加解密，随机生成，一次性使用
公钥	对会话密钥加密，收发双方共享
私钥	对会话密钥解密，接收者专用
口令	对私钥加密，存储于接收端
会话密钥	对传送消息的加解密，随机生成，一次性使用
公钥	对会话密钥加密，收发双方共享
私钥	对会话密钥解密，接收者专用

每个公钥均绑定一个用户名和/或 E-mail 地址。PGP 公钥会被发布到其公钥服务器上，供所有人下载使用。PGP 协议中，用户需要建立一个公钥环形存储器(PKR)，以存储其

他用户的公钥。在使用公钥前,用户需要保证该公钥确实是所指定用户的有效公钥。PGP使用一个名为信任网络(Web of Trust)的信任模型来实现对公钥身份的认证。在信任网络模型中,有效的公钥可以是来自所信任的人的公钥,也可以是由所信任的人为其他人签名的公钥。因此,在得到一个PGP公钥后,用户需要检验其签名,通过对签名用户的信任程度(信任决策的参数可调)确定该公钥的有效性。

图3-8给出一个PGP信任模型的示例。在一个公开密钥环结构中,用户已获得一组公钥,其中部分公钥直接来自于密钥的拥有者,部分来自于第三方,如密钥服务器。节点中的标记代表公钥环中相对于该用户的实体。图3-8中,节点的阴影或空白显示YOU用户指派的信任程度,树形结构显示了哪些密钥被哪些用户签名。在该示例中,YOU用户完全信任用户D对于其他密钥的签名,部分信任用户A、B对其他密钥的签名。假设用户设置两个部分可信的签名可以证明一个公钥,那么用户F的密钥被PGP认定为是有效的,因为它有部分信任用户A、B的签名。即使一个密钥被认定为有效的,但其拥有者对其他密钥的签名不一定可信。例如,用户I的密钥是有效的,因为其有用户D的签名,且用户D为YOU可信任的;但用户I对其他用户密钥的签名不可信任,因为用户没有指派对用户I的信任程度。因此尽管用户J密钥拥有用户I签名,但PGP仍不认为用户J密钥是有效的。

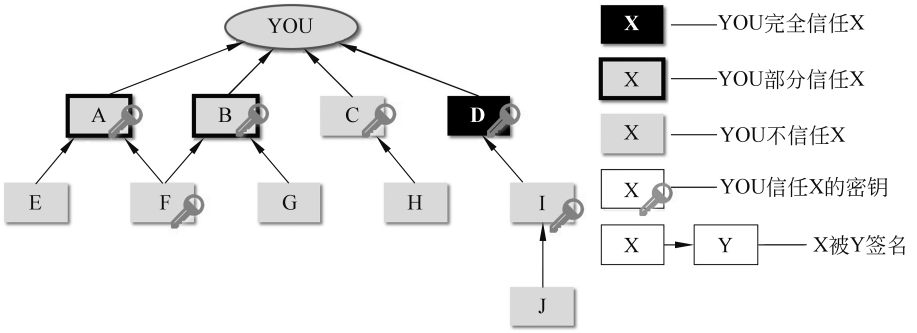

图3-8 一个PGP信任模型的示例

3.3.4 实验步骤

安装实验所需软件:

```
$sudo apt-get install gnupg thunderbird=1:68.7.0+build1-0ubuntu2
```

其中,thunderbird用于搭建邮件客户端(本实验选用旧版本进行,方便练习GnuPG操作),gnupg用于进行PGP实验。

创建实验目录并进入:

```
$mkdir PGP/
$cd PGP
```

1. 利用Thunderbird搭建邮箱客户端

在图形界面打开Thunderbird,或在命令行输入:

```
$thunderbird
```

在“Set Up Your Existing Email Address”下填入个人邮箱，例如 QQ 邮箱或者 Gmail 邮箱。对于各类邮箱，都需要在网页版邮箱设置中打开 SMTP 与 IMAP。对于 QQ 邮箱，需要额外设置一个授权码，并且在 Password 处填入设置的授权码。对于 Gmail 邮箱，除了打开邮件协议外，还需要在 Google 账号设置中启用“安全性较低的应用的访问权限”；然后单击 Continue 进行配置，显示“Configuration found in ...”之后，需要核对 Incoming 的协议选择 IMAP，而不是 POP3 协议（虚拟机的硬盘空间可能无法容纳邮箱内所有邮件文件），如果需要修改请单击“Configure manually”。一切完成后单击 Done 按钮。然后，在 Thunderbird 上安装支持 PGP 加密和签名邮件的插件 Enigmail，安装方式可以参考**附录 H**。完整的 Thunderbird 搭建与 Enigmail 安装流程可参考本书附带的微课视频。

Thunderbird 搭建与 Enigmail

2. GnuPG 命令行练习

Enigmail 可以帮助用户管理 PGP 密钥，包括生成、删除、发布、导出等。另外，这些功能也可以在命令行通过 GnuPG 的命令实现，下面介绍 GnuPG 的部分命令，还可以使用 man gpg 查看更多命令功能。

```
// 生成 PGP 密钥
$gpg --gen-key

// 查看公钥
$gpg --list-keys
// 查看私钥
$gpg --list-secret-keys

// 导出公钥，--armor 参数代表将其转换为 ASCII 码格式，可以缩写成-a
$gpg --armor --output [output_file] --export [KeyID]
// 导出私钥
$gpg --armor --output [output_file] --export-secret-keys [KeyID]

// 加密[input_file]，并将结果输出到[output_file]中
$gpg --output [output_file] --encrypt [input_file]
// 解密[input_file]，并将结果输出到[output_file]中
$gpg --output [output_file] --decrypt [input_file]

// 签名[input_file]，生成[input_file].gpg 文件，默认采用二进制存储
$gpg --sign [input_file]
// 验证[input_file]签名是否为真
$gpg --verify [input_file]

// 以下参数添加在 gpg 命令后：
// --recipient [UserID] 指定用于加密、验证签名的公钥
// --local-user [UserID] 指定用于解密、生成签名的私钥
```

GnuPG 密钥在生成过程中会要求填入个人信息用于生成密钥的 UserID。需要注意，在 Email Address 字段需填入**之前用于设置 Thunderbird 邮箱客户端的邮箱地址**，而 Real

Name 域可自行定义。完成后输入大写字母 O 结束配置,操作示例如图 3-9 所示。

```
parallels@ubuntu-linux-20-04-desktop:~$ gpg --gen-key
gpg (GnuPG) 2.2.19; Copyright (C) 2019 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: testtest
Email address:           @qq.com
You selected this USER-ID:
    "testtest <          @qq.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? O
```

图 3-9　PGP 密钥生成示例

此时,终端会弹出要求填写密钥加密口令的对话框,如图 3-10 所示,**请使用简便易记的口令并牢记**,便于后期邮件的收发实验。

图 3-10　密钥加密口令填写对话框

密钥生成过程的参数会默认选定,包括使用 3072 比特密钥的 RSA 算法;密钥有效期为两年等。

注:在密钥的生成过程中,终端会提示用户能够通过尝试敲击键盘、移动鼠标等随机举动来增大随机数生成器的熵值。

最终生成示例如图 3-11 所示,每个密钥会由软件随机生成的字符串作为其指纹。同时,每个密钥拥有 KeyID 与 UserID 两个标识符,UserID 是之前输入的用户个人信息,KeyID 则是指纹字符串的后 16 位字符。注意,部分命令要求输入 KeyID 作为参数时,输入指纹的后 8 位同样可实现目标功能。

```
gpg: key 830F237F9AE069FF marked as ultimately trusted
gpg: revocation certificate stored as '/home/parallels/.gnupg/openpgp-revocs.d/1
AD7DCDC7CA747EB149B8FA2830F237F9AE069FF.rev'
public and secret key created and signed.

pub   rsa3072 2022-03-13 [SC] [expires: 2024-03-12]
      1AD7DCDC7CA747EB149B8FA2830F237F9AE069FF
uid                      testtest <          @qq.com>
sub   rsa3072 2022-03-13 [E] [expires: 2024-03-12]
```


图 3-11　PGP 密钥标识符示意

利用 GnuPG 命令行工具进行密钥配置的详细过程，可参考本书附带的微课视频。

任务 3.11　进入 PGP 实验目录，生成自己的 PGP 密钥，并实验加解密、签名验证过程，截图记录实验命令与过程结果。


配置密钥

3. PGP 加解密和签名邮件

SMTP，IMAP，POP3 等邮件协议都是明文协议。尽管现在的电子邮件服务都会在这些协议之上再加一层 SSL/TLS 协议，保证邮件从客户端到服务器的机密性，但是对于服务器而言，所有邮件都是明文的，而且在电子邮件邮箱的服务器之间依旧以明文方式传输。因此，要是想通过邮件传递秘密消息，可以使用 PGP 技术对邮件内容进行加密。另外，PGP 还提供对邮件的数字签名功能。在上述实验步骤中，已经在 Thunderbird 中安装了支持 PGP 加密和签名邮件的插件 Enigmail，同时生成了对应邮箱的密钥，以下实验将练习如何在 Thunderbird 中利用 PGP 密钥加解密、签名邮件。

首先，打开 Thunderbird，单击工具栏中 Enigmail 下的“Setup Wizard”，弹出窗口提示已经装有 GnuPG，同时也生成了对应邮箱的密钥，单击“Apply my key”按钮后，可以在“Key Management”中找到生成的密钥。

然后，对于加密与签名技术，**收发信人之间必须共享各自的公钥**，这可以通过上传公钥到公钥服务器共享实现，也可以通过导出公钥文件并复制/传递实现。由于公钥服务器需要时间同步，上传的公钥可能不能及时在公钥服务器上搜索到，所以本实验推荐使用第 2 种方法实现，命令行使用如下命令（以收信人 A 将公钥传给发信人 B 为例）：

```
对于收信人 A:
// 导出公钥文件
$gpg -a --output [key_file] --export [KeyID_A]
// 收信人将公钥文件传输给发信人(可以通过邮箱等方式)

对于发信人 B:
// 导入对方的公钥文件
$gpg --import [key_file]
// 查看该公钥文件的有效性
$gpg --edit-key [KeyID_A]
// 输入 quit 退出查看
```

在 gpg --edit-key [KeyID]的命令行输出中，trust 域表示当前用户对该公钥持有者的信任程度，而 validity 域表示该公钥的有效性。图 3-12 给出了一个查看 PGP 密钥信任程度（方框标识的 trust 域）与有效性（方框标识的 validity 域）的示意图。

此时导入的公钥文件的 validity 域应为 unknown，因为用户没有指派对该公钥持有者的信任程度，该公钥也没有签名信息。若要使用该公钥，则需要用户用自己的 PGP 密钥对其签名，证明其有效性。

```
对于发信人 B:
// 查看接收公钥指纹值并与收信人 A 确认
$gpg --fingerprint
// 确定对自己密钥的信任程度
```

```
buaaf610@buaaf610-VirtualBox:~/Lab3/PGP$ gpg --edit-key D72E5F0C
gpg (GnuPG) 1.4.20; Copyright (C) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

pub  2048R/D72E5F0C  created: 2020-04-06  expires: never       usage: SC
                     trust: unknown       validity: full
sub  2048R/94E59CAD  created: 2020-04-06  expires: never       usage: E
[  full  ] (1).           <           .com>
```

图 3-12　查看 PGP 密钥信任程度与有效性示意图

```
$gpg --edit-key [KeyID_B]
// 若此时自己密钥的 trust 域为 unknown,需要在 GnuPG 命令行中输入 trust 命令,选择"I trust ultimately"并保存,完成对当前密钥的信任更新
// 利用自己的密钥签名接收公钥
$gpg --sign-key [UserID_A]
// 再次查看该公钥文件的有效性
$gpg --edit-key [KeyID_A]
```

再次查看 Enigmail 下的"Key Management",会发现导入的公钥也在其中。

在 Thunderbird 主页面,左侧选中设置好的邮箱,在主页面上端单击 Write,会出现写信窗口。在收件人地址栏填入已经导入过公钥的邮箱地址,窗口上方工具栏的加密与签名图标是活动的,可以单击图标切换是否加密与是否签名。例如,图 3-13 所示的操作是加密该邮件,但不签名。

图 3-13　Thunderbird 加密但不签名邮件示意图

任务 3.12　请两两分组,共享公钥并截图记录过程。然后,两两之间**互相**发送:①不加密不签名;②加密但不签名;③加密且签名的邮件,查看并截图记录**收到邮件**的邮件源码与 Enigmail 安全信息,描述观察情况。

注:在 Thunderbird 界面,点开一封邮件后,下方会出现邮件正文。若是想查看邮件源码,可以选择工具栏 View 下的"Message Source"命令,如图 3-14 所示,弹出窗口中的文本内容即是邮件源码。

同时,对于经过 PGP 处理的邮件,在邮件正文上方还能看见 Enigmail 提供的安全信息,选择 Detail 下的"Enigmail Security Info"命令,如图 3-15 所示,就能看到该邮件的详细安全信息。

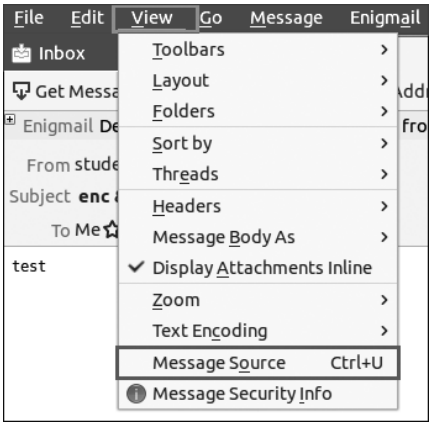

图 3-14　Thunderbird 查看邮件源码

4. PGP 加解密应用

从公钥服务器上搜索并导入[KeyID]为 A60D32AA62CC76AF 的公钥:

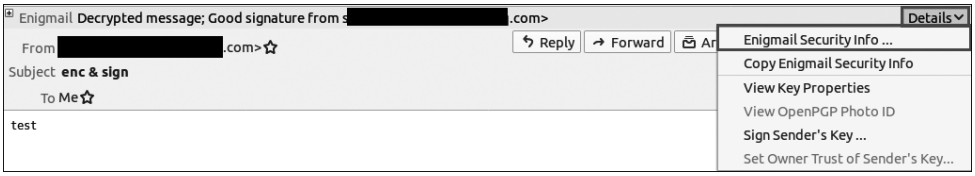

图 3-15　Thunderbird 查看邮件安全信息

```
$gpg --keyserver keyserver.ubuntu.com --search-key [KeyID]
$gpg --keyserver keyserver.ubuntu.com --recv-key [KeyID]
```

* **选做任务 3.1**　利用该公钥加密实验报告，并提交加密后的实验报告文档。

3.4　实验报告要求

(1) 条理清晰，重点突出，排版工整。

(2) 内容要求。

① 实验题目。

② 实验目的与内容。

③ 实验结果与分析(按步骤完成所有实验任务，详细记录并展示实验结果和对实验结果的分析)。

④ 实验思考题：

- Linux 系统为什么要分别设立/etc/passwd 文件与/etc/shadow 文件？它们在文件内容与权限两方面有什么区别？
- 简述 CA 签发证书的过程。
- 在使用 PGP 加解密技术进行邮件加密与签名时，收发信人之间为什么要共享各自的公钥？

⑤ 遇到的问题和思考(实验中遇到了什么问题，是如何解决的，在实验过程中产生了什么思考)。

本章参考文献

[1] Ubuntu manuals. man5 passwd[EB/OL]. (2019-03-30) [2022-08-21]. http://manpages.ubuntu.com/manpages/xenial/man5/passwd.5.html.

[2] Ubuntu manuals. man5 shadow[EB/OL]. (2019-02-24) [2022-08-21]. http://manpages.ubuntu.com/manpages/xenial/man5/shadow.5.html.

[3] Openwall. John the Ripper password cracker[EB/OL]. (2022-06-20) [2022-08-21]. https://www.openwall.com/john/.

[4] Cooper, et al. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile[EB/OL]. (2008-05-01) [2022-08-21]. https://tools.ietf.org/html/rfc5280.

[5] Seed Labs. Public Key Infrastructure(PKI) Lab[EB/OL]. (2010-04-11)[2022-08-31]. https://seedsecuritylabs.org/Labs_16.04/Crypto/Crypto_PKI/.

[6] OpenSSL. openssl-x509 manpage[EB/OL]. (2021-11-01)[2022-08-31]. https://www.openssl.org/docs/man1.0.2/man1/openssl-x509.html.

[7] Callas, et al. OpenPGP Message Format[EB/OL]. (2007-11-21)[2022-08-31]. https://tools.ietf.org/html/rfc4880.