百万富翁问题 实验报告

王傲

15300240004

保密技术概论

COMP130116.01

指导教师：吴杰

目录：

内容摘要：

本次实验完成了较为简单的 10 以内的百万富翁问题的实现，并同时完成了自主设计的基于 TCP socket 的多重加密安全通信协议。基于该协议的基础上实现了安全通信，并提供了便于操作的图形界面。

关键字：百万富翁 DES RSA 安全通信 SHA 数据库

1. 简介

1.1 实验内容

本次实验，完成了较为简单的多方安全计算的一个应用：10 以内的百万富翁问题，并同时完成了自主设计的基于 TCP socket 的多重加密安全通信协议。基于该协议的基础上，完成了百万富翁问题的代码实现，并提供了便于操作的图形界面。

1.2 实验环境

本次实验平台是 MacBook Pro 15’Mid 2015，操作系统为 macOS High Sierra 10.13.2，基于 Python 2.7.14 完成。

主要使用的 Python 的库包括：

- pyCrypto 2.6.1：用于实现 RSA 加密、签名、SHA256 摘要生成等
- pyDes 2.6.1：用于为数据库实现 DES 对称加密
- PyQt5 5.9.2：用于实现图形界面
- pickle：用于持久化 Python 对象和编码
- socket：用于实现 TCP socket 通信
- 其他 Python 原生库

使用的数据库为 MySQL 5.7.17，具体的概念模式和存储内容会在下面给出（在其他机器上使用时需要严格按照给出的内容存储，否则无法使用）。

已经将服务器端文件放到服务器上运行，服务器 IP 地址为 182.254.130.132，使用时可将 Client.py 的第 21 行 HOST 改为该地址或者直接使用打包好的可执行文件。

2. 基本原理

2.1 百万富翁问题

百万富翁问题由姚期智于 1982 年提出，原始的描述如下：Alice 有 i 百万元，Bob 有 j 百万元($1 <= i, j <= 10$)，如何在不透露自己财富数量的情况下比较出两者财富的大小？

姚期智在论文中给出了这个问题的一个原始解答：

1. Bob 随机选择一个大整数 x，用 Alice 的公钥 E_a 加密，得到 k
2. Bob 将 k-j+1 发送给 Alice
3. Alice 用自己的私钥 D_a，计算 $y_u=D_a(k-j+u)$，$u=1,2,...,10$
4. Alice 随机生成一个适当大小的质数 p，计算 $z_u=y_u \bmod p$
5. Alice 将 $z_{i+1},...,z_{10}$ 修改为 $z_{i+1}+1,...,z_{10}+1$，再对 p 求余数，然后将十个数字和质数 p 发送给 Bob
6. Bob 选出第 j 个数字，查看是否等于 $x \bmod p$；如果是，则 $i >= j$，否则 $i < j$

这个方案还是比较好理解的，注意到这个方案的关键是 $z_j = y_j \bmod p = D_a(k) \bmod p = x \bmod p$。如果不相等的话，说明 j 对应的数字被加了 1，即 i 在 j 之下，否则 i 大于或等于 j。

使用公钥 E_a 和私钥 D_a 的原因是为了保证生成的十个数字 y_u 看起来是完全随机的，使 Bob 无法猜出哪些数字被加了 1（注意加完 1 后一定要再对 p 求余数，否则可能会有原来结果是 p-1 的情况，加完 1 后是 p，会让 Bob 看出来这是加过 1 的，从而获得额外的信息）。

2.2 RSA 加密/签名

RSA 的安全性主要基于大数分解的困难性。使用 RSA 加密/解密的过程如下：

1. 任意选择两个大质数 p 和 q，$N = p*q$
2. 根据欧拉函数，计算 $R = (p-1) * (q-1)$
3. 选择一个小于 R 且与 R 互质的数 E，计算 E 的模反元素 D，使得 $E*D \equiv 1 \bmod n$
4. 得到的公钥是 (N, E)，私钥是 (N, D)

使用公钥加密的方法为：

$$密文 = 明文^{E} \bmod N$$

使用私钥解密的方法是：

$$明文 = 密文^{D} \bmod N$$

如果是使用数字签名，则使用私钥加密，接收方使用公钥解密。

3. 代码实现

3.1 核心代码部分

实现百万富翁问题的核心代码部分，完全按照上面给出的模式，通过客户端和服务器端实现。其中，客户端是 Bob，会发送单个数字并收到 10 个数字；服务器端是 Alice。为了方便测试和比较，服务器端在数据库中存放了 Alice1 到 Alice10，所对应的数额分别是 1 到 10，客户端输入时输入正确的姓名即可比较（一般情况下，姓名和数额毫无关系，Bob 得不到任何其他信息，这里只是为了方便测试）。

方案中的加解密通过自己实现的简单 RSA 实现（与通讯加密使用的 RSA 不同），其中 $p = 2147483647, q = 1000000007, N = 2147483662032385529, K = 2147483658884901876$, 选择的 $E = 314159$，通过扩展欧几里得公式计算出 $D=282476754508894175$。

首先，Bob 选择一个较大的随机数 x：

```
random.seed()
x = random.randint(200000,300000)
bigint=x
```

将 x 加密后得到 K，将 K-j+1 发送给 Alice（服务器）：

```
K = simple_RSA_encrypt(x)
num=K-y+1
```

Alice 收到后，利用自己的私钥 D_a 计算 $y_u=D_a(k-j+u)$, $u=1,2, ..., 10$:

```
lis=[simple_RSA_decrypt(num+i) for i in range(10)]
```

之后，Alice 随机生成一个适当大小的质数 p，计算 $z_u=y_u \bmod p$：

```
random.seed()
index=random.randint(0,4)    #随机选择一个作为除数的质数
p=primes[index]

res_dic={}
for i in range(10):
    res_dic[i]=lis[i]%p
```

Alice 将 $z_{i+1}, ..., z_{10}$ 修改为 $z_{i+1}+1, ..., z_{10}+1$, 再对 p 求余数, 然后将十个数字和质数 p 发送给 Bob：

```
for i in range(x,10):
    res_dic[i] = (res_dic[i]+1)%p    #注意加1后要余p否则是错的！！！！！！！！！
res_dic[10]=p    #包含10个数的字典的最后放入选择的作为除数的质数p
```

Bob 选出第 j 个数字，查看是否等于 x mod p；如果是，则 i >= j，否则 i < j：

```
try:
    num=res_dic[y-1]
except:
    return 'error'
p=res_dic[10]    #包含10个数的字典的最后放入对方选择的作为除数的质数p
if num == x%p:
    result = True      #"x >= y"
else:
    result = False     #"x < y"
```

这是百万富翁问题的核心代码部分，比较简单，完全按照姚期智 1982 年的论文给出的形式实现。可以看出，在一次通信过程中，Alice 了解的信息除了自己的数值外只有 K-j+1，而 Bob 了解的信息除了自己的数值外只有十个对 Bob 来说完全随机的数和一个质数 p，所以 Alice 和 Bob 均不能通过额外信息获得对方数额的大小，证明这个方案可以在不泄露两者数额大小的情况下进行比较。

3.2 通讯协议部分

由于涉及到客户端和服务器端的通讯，因此使用自主设计的基于 TCP socket 的多重加密安全通信协议。协议的设计和使用方式会在下面给出。

3.3 数据库部分

为了方便比较，在服务器端使用了数据库，将 Alice1 至 Alice10 和相应的经过 DES 加密的数值放在了数据库里，便于比较。一般情况下名字和数额毫无关系，这里只是为了方便使用和比较。

建立的数据库名为 SecureComputation，表名为 account，描述如下：

```
mysql> describe account;
+--------------+---------------+------+-----+---------+-------+
| Field        | Type          | Null | Key | Default | Extra |
+--------------+---------------+------+-----+---------+-------+
| name         | varchar(200)  | NO   | PRI | NULL    |       |
| money_value | varchar(1000) | NO   |     | NULL    |       |
+--------------+---------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

图 1 account 表的概念模式

数据库的内容如下：

name	money_value
Alice1	z9Am2mUr4B0=
Alice10	1hL3e3YBz/4=
Alice2	VlFH/Ox72DU=
Alice3	n4ksYeqLXw0=
Alice4	sfFAVZZCMv0=
Alice5	SPQHkrLV3Po=
Alice6	LNBp59WCS9s=
Alice7	SNoIGfXrSGY=
Alice8	/GflQzE1c/o=
Alice9	AWbyVL//LJY=

图 2 account 表中的内容

其中，name 为姓名，money_value 为经过 DES 加密过的数额，分别为 1 到 10。DES 加密/解密函数和密钥在源文件中给出。

3.4 图形界面部分

图形界面通过 PyQt5 实现，效果如下：

图 3 图形界面示例

图形界面的源文件已经打包为 Windows 下的可执行文件，并且服务器源文件已经放在服务器上，可以点击 exe 文件直接运行。

从图形界面，我们可以看出 Bob 随机选择的大整数、Alice 发送的十个数字和最终结果等内容，方便使用。

4. 安全通信协议

4.1 基本介绍

本次实验是基于 TCP socket 的可靠连接的，然而传送的数据并没有保密。为了使整个系统的安全得到保护，综合本学期所学内容，使用了如下的加密保护措施：

- 基于 RSA 的传输过程加密
- 基于 RSA 的数字签名认证
- 基于 DES 的数据库加密
- 基于时间戳的防止重放攻击
- 基于加盐的 SHA256 的防止修改认证
- 简单的防止 SQL 注入过滤

客户端和服务器端的协议格式相同。

其中，传输所用的 RSA 密钥为 1024 位，客户端和服务器端的公钥和私钥分别被放在了相应的位置，模拟已经从 CA 获得了正确的公钥。

由于 RSA 的密钥是 1024 位，最多只能加密 117 字节，因此将数据分成 100 字节的块，加密时每个块单独加密，解密时每个块单独解密，然后进行拼接。

时间戳通过 DES 加密，而签名和 SHA256 使用了未加密的时间戳来增加随机性。

4.2 具体架构

协议的具体实现分为两层。

真正的数据部分，客户端是用户姓名、要查询的姓名和 K-j+1 组成的字典，服务器端是十个数字和质数 p 组成的字典。

时间戳通过 DES 加密，因为签名用到了时间戳，尽量不用同一种加密方式。

签名内容为字符串“This is from the Client/Server with timestamp: ”加上时间戳，其中时间戳增加了随机性，防止固定的字符串导致固定的加密后的内容。

安全协议的第一层为一个字典，其中第一项是数据部分（通过相应的 RSA 公钥加密），第二项是时间戳（通过 DES 加密），第三项是数字签名（通过相应的 RSA 私钥加密）：

图 4 安全协议的第一层

安全协议的第二层为第一层的摘要。

摘要的目的是为了防止有人截获包后对内容进行修改。但是，只生成第一层的摘要的话，第一层是裸露的，攻击者可以在修改第一层后针对修改后的内容生成摘要。为了确保安全，需要加盐。

选择的加盐内容为时间戳（未加密的部分，而不是裸露在外面的已经加密的部分），通过时间戳增加随机性，而时间戳已经经过 DES 加密，攻击者是无法获得真正的时间戳的，从而无法生成正确的摘要。

摘要内容为协议第一层的内容加上未加密的时间戳：

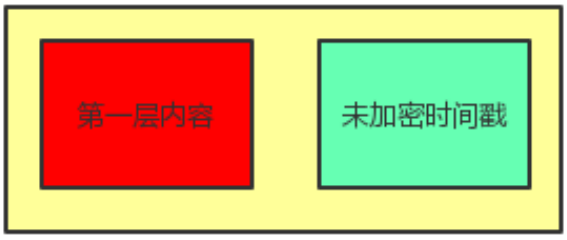

图 5 安全协议的第二层

所以，协议的整体内容如下（摘要与第一层相连）：

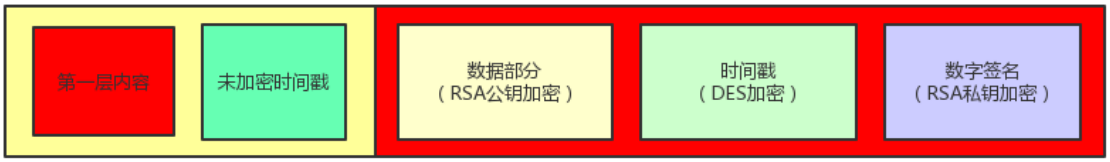

图 6 安全协议的整体

无论是客户端还是服务器端，发送时均使用 send_preprocessing 函数，依照不同的密钥和内容将数据按照协议格式打包。接收时，利用 receive_pre_check 函数进行检查，包内时间戳与当前时间超过 5 秒则拒绝本次操作；SHA256 校验错误则拒绝本次操作；签名错误则拒绝本次操作。如此，尽最大可能的保证了通信的安全。

除了通信安全，本次实验在设计中还考虑了其他安全事项，比如因为使用了数据库，所以不同的 Alice 的金额在数据库中不是明文存储而是使用了 DES 加密，防止数据库被攻击导致信息丢失；为了防止数据库被 SQL 注入攻击，进行了防 SQL 注入过滤（非常简单的过滤，没有更深入的涉及）等。

5. 总结

本次实验，完成了较为简单的多方安全计算的一个应用：10 以内的百万富翁问题，并同时完成了自主设计的基于 TCP socket 的多重加密安全通信协议。基于该协议的基础上，完成了百万富翁问题的代码实现，并提供了便于操作的图形界面。

6. 参考资料

[1] Yao, Andrew C. "Protocols for secure computations." *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE, 1982.

[2] 《保密技术概论》教科书

[3] 保密技术概论 课程资料

[4] CSDN 技术博客