一种基于时间戳的二次 RSA 加密算法

李 莉[1] 朱红霞[2]

（廊坊师范学院 理学院 河北 廊坊 065000）

【摘要】分析了 RSA 加密算法在密码学中的重要作用，并提出了一种结合时间戳和 RSA 算法的混合加密算法，利用 python 语言实现加密解密过程；最后对该算法提出了改进措施。研究内容对读者了解密码学的加密解密原理有一定的价值。

【关键词】时间戳；RSA 加密算法；python 语言

【作者简介】李莉（1980–）女，河北省秦皇岛市卢龙县人，硕士研究生，讲师，主要研究方向：信息安全。

朱红霞，廊坊师范学院理学院；硕士，讲师，主要研究方向：廊坊师范学院信息与计算科学重点建设专业。

【中图分类号】TP393.08 【文献标识码】A 【文章编号】1673-9574（2022）12-00085-03

1. 引言

随着互联网技术的不断发展，人们越来越重视信息的安全问题。密码学是信息安全的基础，而 RSA 加密算法在密码学中有着非常重要的作用。

密码学是由两个分支组成的，即密码编码学和密码分析学。密码体制有两种：对称密码体制和非对称密码体制[1–2]。对称密码体制使用相同的密钥对消息进行加密或解密，系统的保密性主要由密钥的安全性决定，而与算法是否保密无关。对称密码体制可以通过分组密码或流密码来实现，它既可以用于数据加密，又可以用于消息认证。非对称密码体制使用公钥加密消息，使用私钥来解密，这两者并不相同。使用非对称密码体制能够增强通信的安全性[3–5]。

2. RSA 加密算法

RSA 算法是目前最有影响力的公钥加密算法，它能够抵抗到目前为止已知的所有密码攻击，已被 ISO 推荐为公钥数据加密标准[6–8]。这种算法非常可靠，密钥越长，它就越难破解。根据已经披露的文献，目前被破解的最长 RSA 密钥是 232 个十进制位，也就是 768 个二进制位，因此可以认为，1024 位的 RSA 密钥基本安全，2048 位的密钥极其安全，当然量子计算机除外。

RSA 加密过程[9–10]：

（1）密钥的生成如下表

表 2-1 密钥的生成

步骤	说明	描述
1	选择一对不相等且足够大的素数	p,q
2	计算 p,q 的乘积	n=p*q
3	计算 n 的欧拉函数	∅(*n*) =(p–1)*(q–1)
4	选一个与∅(*n*)互素的整数 e	1<e< ∅(*n*)
5	计算出 e 对于∅(*n*)的模反元素 d	de mod ∅(*n*) =1
6	公钥	KU=(e,n)
7	私钥	KR=(d,n)

（2）RSA 加密

通式：

$$密文 = 明文^{E} \bmod N \quad (1)$$

即 RSA 加密就是对明文进行 E 次方后除以 N 后求余数的过程。其中，E 是加密 (Encryption) 的首字母，N 是数字 (Number) 的首字母，mod 是取模运算符。

由此可知，只要知道 E,N 就可以进行 RSA 加密，所以说 E,N 就是 RSA 加密的密钥，E 和 N 的组合就是公钥，记为

$$公钥 =(E,N) \quad (2)$$

（3）RSA 解密

通式：

$$明文 = 密文^{D} \bmod N \quad (3)$$

即 RSA 解密就是对密文进行 D 次方除以 N 后求余数的过程。其中 D 是解密（Decryption）的首字母。由此可知，只要知道 D、N 就可以进行 RSA 解密，所以说 D、N 就是 RSA 解密的密钥，D 和 N 的组合就是私钥，记为

$$私钥 =（D,N） \quad (4)$$

3. 加解密实例

3.1 密钥的生成

（1）甲方的公钥和私钥的生成

甲方和乙方约定用相同的算法生成密钥对。

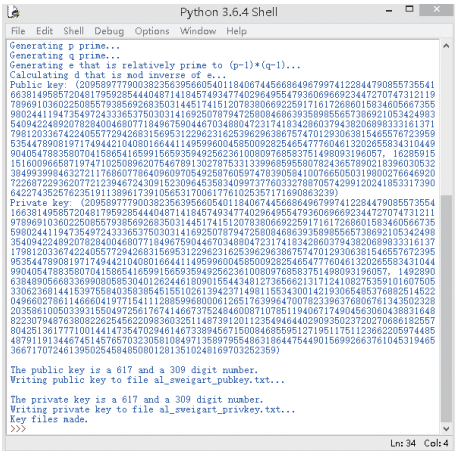

图 3-1 甲方的公钥和私钥的生成

由上面结果图可知，甲方生成了两个非常大的密钥对，它们分别为公钥和私钥。公钥和私钥都是由 617 位和 309 位的数

字组成的。由于它们非常大，所以分别被写入两个文件中，即 al_sweigart_pubkey.txt 和 al_sweigart_privkey.txt。其中公钥会被甲方发送给乙方。

（2）乙方的公钥和私钥的生成

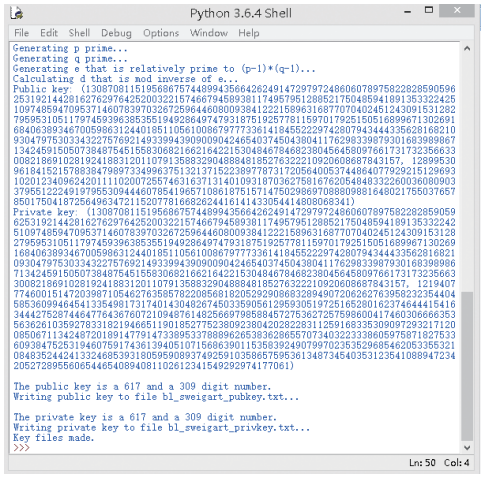

图 3-2　乙方的公钥和私钥的生成

由上面结果图可知，乙方也同样生成了两个非常大的密钥对，它们分别为公钥和私钥。公钥和私钥都是由 617 和 309 位的数字组成的。由于它们非常大，所以分别被写入两个文件中，即 bl_sweigart_pubkey.txt 和 bl_sweigart_privkey.txt。其中公钥会被乙方发送给甲方。

由上面两个密钥对的生成结果可知，双方都生成了符合 RSA 算法要求的密钥对。实际上，在密钥对生成的过程中，还包括了大素数的生成。大素数的生成也是密码学领域研究的一个小方向。由于现在已经有生成大素数的比较成熟的算法，这里就不过多说明了，本文只给出密钥对生成的结果。

3.2 加密过程

(1) 对明文的第一次加密

甲方利用乙方的公钥对明文进行 RSA 加密。我们不妨假设明文为：Made in china。

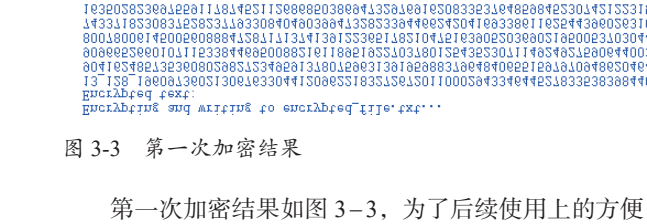

图 3-3　第一次加密结果

第一次加密结果如图 3-3，为了后续使用上的方便，我们把密文保存在 miwen1.txt 里。

（2）加入时间戳

时间戳选择为年、月、日、时和分。为了读者方便的看到加时间戳的结果，甲方选定某一个未来时间，并把它加入到密文的最后。不妨假设甲方选择的时间为 2022 年 6 月 22 日 15 点 30 分，则加入的时间戳为：202206221530。其中年为 4 位数，月、日和时都为 2 位数，时间时间戳为 12 位数字。密文加入时间戳后的结果如下图：

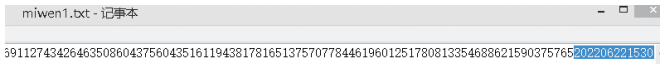

图 3-4　加入时间戳后的部分密文

上图的最后 12 位数字（即图中蓝色字）就是加入的时间戳。为了读者方便看到时间戳加入后的结果，我们把时间戳加入到了密文的最后。

（3）对明文的第二次加密

甲方利用自己的私钥对加入时间戳的密文进行第二次 RSA 加密。由于密文数量较大，所以只截取了一部分，并把密文保存在 miwen2.txt 文件里。

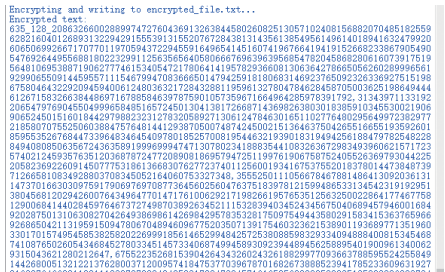

图 3-5　第二次加密结果

在进行了 2 次 RSA 加密和 1 次添加时间戳后，甲方得到了最终的加密结果，即图 3-5。然后甲方利用各种互联网平台把密文发送给乙方。

3.3 解密过程

3.3.1 带时间戳的解密

甲方按照自己选定的时间把最终的密文通过互联网（QQ、微信、邮件等）方式发给乙方。假设乙方收到的密文的时间为 2022 年 6 月 22 日 15 点 30 分 35 秒。则根据双方的约定，乙方知道甲方加的时间戳为：202206221530。

（1）乙方用甲方的公钥对收到的密文进行第一次 RSA 解密。

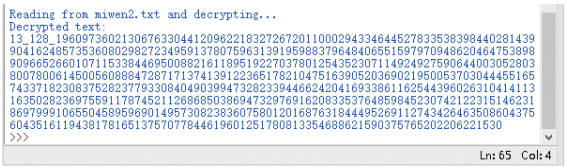

图 3-6　第一次解密结果

由图 3-6 可知，解密结果的最后 12 位正好是乙方按照约定推算出来的时间戳。由此可以说明乙方第一次解密成功。不

仅如此，由于解密得到的时间戳和乙方按照密文接收到的时间推测的时间戳相同，使得乙方能够确定自己收到的密文没有被篡改，并且乙方还能够确定密文确实是甲方发送的。乙方把解密的结果保存在 jiemi1.txt 文件里。

（2）去掉时间戳。

乙方把第一次解密的结果去掉时间戳“202206221530”，并把结果保存在 jiemi2.txt 里。图 3－6 的结果去掉最后的 12 位数字就是去掉时间戳的结果。

（3）第二次解密

乙方利用自己的私钥对去掉时间戳的第一次解密结果（jiemi2.txt）进行解密。

```
Reading from jiemi2.txt and decrypting...
Decrypted text:
Made in china
>>>
                                                    Ln: 70  Col: 4
```

图 3-7　第二次解密结果

由图 3－7 可知，第二次解密的结果是：“Made in china”。由此可见解密成功。

3.3.2 不带时间戳的解密

（1）乙方用甲方的公钥对收到的密文进行第一次 RSA 解密。解密结果和带时间戳的解密的第一次解密结果相同。

（2）第二次解密。

乙方利用自己的私钥对第一次解密结果（jiemi1.txt）进行解密。

```
Reading from jiemi1.txt and decrypting...
Traceback (most recent call last):
  File "F:\教案\密码学实践\第1版\hackingciphers代码\rsadecrypt.py", line 160, in
<module>
    main()
  File "F:\教案\密码学实践\第1版\hackingciphers代码\rsadecrypt.py", line 30, in
main
    decryptedText = readFromFileAndDecrypt(filename, privKeyFilename)
  File "F:\教案\密码学实践\第1版\hackingciphers代码\rsadecrypt.py", line 154, in
readFromFileAndDecrypt
    return decryptMessage(encryptedBlocks, messageLength, (n, d), blockSize)
  File "F:\教案\密码学实践\第1版\hackingciphers代码\rsadecrypt.py", line 91, in
decryptMessage
    return getTextFromBlocks(decryptedBlocks, messageLength, blockSize)
  File "F:\教案\密码学实践\第1版\hackingciphers代码\rsadecrypt.py", line 65, in
getTextFromBlocks
    blockMessage.insert(0, chr(asciiNumber))
OverflowError: Python int too large to convert to C long
>>>
```

图 3-8　不带时间戳的第二次解密结果

由图 3－8 可见，解密出错。即不能解密成功。由此可见，假设敌手截获了全部密文，因为敌手不知道密文添加了时间戳，所以即使敌手通过各种手段知道了甲方和乙方的公钥和私钥，敌手也不能破解密文。这说明，添加时间戳大大提高了密文的安全性。

4. 算法的缺陷和改进

(1) 时间戳加入位置的缺陷。时间戳加入到密文的最后，容易被概率统计攻击。信息收发的双方可以约定把时间戳的 12 位数字以某种复杂一些的方式分散到密文中来避免概率统计攻击。当然，时间戳也可以按照双方约定的方法进行确定。比如时、分或月、时、分或时、分、月或时、分、月、年等等。

(2) 第一次解密用的是发送方的公钥。由于公钥可能被黑客截获，会导致黑客很容易进行第一次解密。发送方可以通过改变二次加密的顺序来解决这个问题。即发送方首先用自己的私钥进行加密，然后加入时间戳，最后用接收方的公钥进行加密。这样，第一次的解密就需要接收方自己的私钥进行解密，而黑客不知道双方的私钥，就很难进行第一次解密。这样能提高密文的安全性。

(3) 本文使用 python 语言进行加密解密。为了操作简单、方便，读者也可以使用网上已有的加密软件进行加密和解密。用软件进行加密解密，不仅简单、方便，而且也不影响安全性。只要双方对一些关键的事情约定好即可。

5. 结论

RSA 加密算法应用很广，安全性很高，但是仍然有被破解的风险。为了提高 RSA 加密算法的安全性，本文对明文进行了 2 次 RSA 加密，不仅如此，本文在 2 次 RSA 加密的过程中创新的加入了时间戳（本文实例中的时间戳是“202206221530”这 12 个数字）。由前面的加解密实例可知，加入时间戳不仅提高了明文的安全性，还使得接收方能够在第一次解密结果中判断出密文是否被篡改。这也是本文选择 RSA 公钥加密算法的原因，公钥加密体制不仅能进行安全加密，还能够用于数字签名。

参考文献

[1] 黄恒一，付三丽 . 数学在计算机密码学中的应用研究 [J]. 电脑与电信 .2022（4）

[2] 胡大涛 . 密码学研究与探讨 [J]. 微电子学与计算机，2020(9).

[3] 刘亚敏，薛海洋，张道德 . 公钥密码的实际安全性发展研究 [J]. 信息安全研究，2019，5(01): 29- 38.

[4] 孙瑞，黄佳文，孙学贵 . 基于可满足性问题的非对称密码方案设计 [J]. 无线互联科技，2022,19(08):67-68.

[5] 包伟 . 对称密码体制与非对称密码体制比较与分析 [J]. 硅谷，2014,7(10):138-139.

[6] 朱怀宇，姜群兴 .FPGA 与上位机通信的数据加密方法 [J]. 工业控制计算机，

2018(05): 25-26.

[7] 平伟，贾文丽，孙月驰等 . 基于 DES 算法与 RSA 算法的数据加密技术在电子商务中的应用 [J]. 软件导刊，2018(05): 45- 47.

[8] 张文文，炳勋 . 基于 RSA 与 DES 的多重加密可信加密算法 [J]. 电脑迷，2016(09):81.

[9] 王保仓，贾文娟，陈艳格 . 密码学现状、应用及发展趋势 [J]. 无线电通信技术，2019，45(01):1-8.

[10] 赵彩，丁凰 . 网络信息安全中 DES 数据加密技术研究 [J]. 计算机测量与控制，2017(08): 20-21.