首届（2016）全国高校密码数学挑战赛
赛题三

一、赛题名称：RSA 加密体制破译

二、赛题描述

1.1 问题描述

RSA 密码算法是使用最为广泛的公钥密码体制。该体制简单且易于实现，只需要选择 5 个参数即可（两个素数p和q、模数$N = pq$、加密指数e和解密指数d）。设m为待加密消息，RSA 体制破译相当于已知$m^e \bmod N$，能否还原m的数论问题。目前模数规模为 1024 比特的 RSA 算法一般情况下是安全的，但是如果参数选取不当，同样存在被破译的可能。

有人制作了一个 RSA 加解密软件（采用的 RSA 体制的参数特点描述见密码背景部分）。已知该软件发送某个明文的所有参数和加密过程的全部数据（加密案例文件详见附件 3-1）。Alice 使用该软件发送了一个通关密语，且所有加密数据已经被截获，请问能否仅从加密数据恢复该通关密语及 RSA 体制参数？如能请给出原文和参数，如不能请给出已恢复部分并说明剩余部分不能恢复的理由？

1.2 实例破解

在本次竞赛问题中，我们选取了一个具体加密实例供大家破解，整个算法与加密过程描述如下，截获的加密数据见附件 3-2。

1. RSA 密码算法描述如下，包含体制参数选取和加解密过程。

1）RSA 体制参数选取

Step1. 每个使用者，任意选择两个大素数p和q，并求出其乘积$N = pq$。

Step2. 令$\varphi(N) = (p-1)(q-1)$，选择整数e，使得$\mathrm{GCD}(e, \varphi(N)) = 1$，并求出$e$模$\varphi(N)$的逆元$d$，即$ed \equiv 1 \bmod \varphi(N)$.

Step3. 将数对(e, N)公布为公钥，d保存为私钥。

2）加解密过程

Bob 欲传递明文m给 Alice，则 Bob 首先由公开途径找出 Alice 的公钥(e, N)，Bob 计算加密的信息c为：$c \equiv m^e \bmod N$。

Bob 将密文c传送给 Alice。随后 Alice 利用自己的私钥d解密：

$$c^e \equiv (m^e)^d \equiv m^{ed} \equiv m \bmod N.$$

2. Alice 使用的 RSA 密码体制，有以下事项需要说明：

1） 模数$N = pq$规模为 1024 比特，其中p，q为素数；

2） 素数p由某一随机数发生器生成；

3） 素数q可以随机选择，也可以由 2）中的随机数发生器产生；

4） 可以对文本加密，每次加密最多 8 个明文字符；

5） 明文超过 8 个字符时，对明文分片，每个分片不超过 8 个字符；

6） 分片明文填充为 512 比特消息后再进行加密，填充规则为高位添加 64 比特标志位，随后加上 32 比特通信序号，再添加若干个 0，最后 64 比特为明文分片字符对应的 ASCII 码（注：填充方式参见加密案例，但注意每次通信的标志位可能变化）；

7） 分片加密后发送一个加密帧数据，帧数据文件名称为 FrameXX，其中 XX 表示接收序号，该序号不一定等于通信序号；

8） 帧数据的数据格式如下，其中数据都是 16 进制表示，结构如下

1024bit 模数 N | 1024bit 加密指数 e | 1024bit 密文 $m^e \bmod N$。

9） 由于 Alice 初次使用该软件，可能会重复发送某一明文分片。

1.3 成绩评判

通过数论方法获得的原始明文及 RSA 参数数量，数量多者获胜。

三、国内外研究进展与现状

RSA 的安全性是基于大整数素因子分解的困难性，而大整数因子分解问题是数学上的著名难题。数域筛法是目前 RSA 攻击的首选算法。在 1999 年，一台 Cray 超级电脑用了 5 个月时间分解了 512 比特长的密钥。在 512 比特 RSA 算法破解 10 年之后，即 2009 年 12 月 9 日，768 比特 RSA 算法即 232 数位数字的 RSA-768 被分解。分解一个 768 比特 RSA 密钥所需时间是 512 位的数千倍，而 1024 比特所需时间则是 768 比特的一千多倍，因此在短时间内 1024 比特仍然是安全的。除此之外，目前对于 RSA 算法的攻击主要有以下方式:选择密文攻击、公共模数攻击、低加密密指数攻击、低解密指数攻击、定时攻击等等，详细的 RSA 安全分析参见有关文献。

四、参考文献与可能用到的软件

1. 陈少真，密码学基础，科学出版社，2008-05-30.
2. 任伟，现代密码学，北京邮电大学出版社，2011 年 4 月
3. 冯登国　等译，密码学原理与实践（第三版），电子工业出版社
4. 谢建全，阳春华，RSA 算法中几种可能泄密的参数选择，《计算机工程》 2006 年 16 期
5. Don Coppersmith: Finding a Small Root of a Univariate Modular Equation. EUROCRYPT 1996 (LNCS 1070, Springer): 155-165.
6. GMP package, GNU Multiple Precision Arithmetic Library, https://gmplib.org/
7. Magma Computational Algebra System, http://magma.maths.usyd.edu.au/magma/
8. Pari, http://pari.math.u-bordeaux.fr/
9. NTL: A Library for doing Number Theory, http://www.shoup.net/ntl/