中山大学移动信息工程学院移动网络安全实验报告
(2016学年春季学期)

刘爽
班级：13M2
ID:13354222
专业：移动互联网

2016年3月29日

1 实验题目

大数的质因数分解

1.1 实验要求

给定一个很大的数N，使用相应的求解质因数的方法求解N的两个质因数p和q。在这次实验中，p和q之间的关系满足：

$$|p-q|<2\sqrt[4]{N}$$

1.2 实验目的

学习如何使用第三方开源库。
理解RSA是如何产生密文的。
理解在给定某些条件下如何破解RSA。

2 实验原理

假设A是两个质数p和q的平均值：

$$A=\frac{p+q}{2}$$

因为p和q都是奇数，那么p+q则为偶数，A为一个整数。 根据已知条件我们知道：

$$|p-q|<2\sqrt[4]{N}$$

由Fig.1我们知道，$\sqrt{N}$跟A非常接近。特别的，

$$A>\sqrt{N}$$

并且

$$A-\sqrt{N}<1$$

又因为A是一个整数，我们需要对$\sqrt{N}$进行向上取整来表达A的值：

$$A=\lceil\sqrt{N}\rceil$$

又因为A是在p和q中的中点，所以一定存在一个整数x使得：

$$p=A-x$$

$$q=A+x$$

那么

$$\begin{aligned}N&=p*q\\&=(A-x)*(A+x)\\&=A^2-x^2\end{aligned}$$

因此

$$x=\sqrt{A^2-N}$$

那么，给定x和A的值，你可以用下列公式找到分解N的两个素数p和q：

$$p=A-x$$

$$q=A+x$$

2.1 证明：

给定条件以下三个条件：

$$|p-q| < 2\sqrt[4]{N} \quad (1)$$

$$pq = N \quad (2)$$

$$A = \frac{p+q}{2} \quad (3)$$

证明：
I:

$$A > \sqrt{N}$$

II:

$$A - \sqrt{N} < 1$$

2.1.1 证明1

根据条件(2)和条件(3),要证明$A > \sqrt{N}$,
即需要证明

$$\frac{p+q}{2} > \sqrt{pq}$$

即

$$\frac{(p+q)^2}{4} > pq$$

那么

$$(p+q)^2 > 4pq$$

可得到

$$p^2 + 2pq + q^2 > 4pq$$

因此

$$p^2 + q^2 - 2pq > 0$$

即

$$(p-q)^2 > 0 \quad (4)$$

当N不为一个完全平方数时，即p和q不相等时，不等式(4)恒成立，即证$A > \sqrt{N}$恒成立。

2.1.2 证明2

根据条件(2)和条件(3)，我们可以得到

$$(p-q)^2 < 4\sqrt{N}$$

即

$$\frac{(p-q)^2}{4} < \sqrt{N}$$

两边同时加上pq, 即N，可得到

$$\frac{(p-q)^2+4pq}{4} < \sqrt{N}+N$$

即

$$\frac{(p+q)^2}{4} < \sqrt{N}+N \tag{5}$$

由(5)式可得以下不等式

$$\frac{(p+q)^2}{4} < \sqrt{N}+N < \sqrt{N}+N+1 < \sqrt{N}+N+1+\sqrt{N}$$

可得

$$\frac{(p+q)^2}{4} < 2\sqrt{N}+N+1$$

即

$$\frac{(p+q)^2}{4} < (\sqrt{N}+1)^2$$

两边同时开方，可得

$$\frac{p+q}{2} < \sqrt{N}+1$$

又因为$A = \frac{p+q}{2}$, 可得到

$$A < \sqrt{N}+1$$

即

$$A-\sqrt{N} < 1$$

即证II式。

3 实验内容

分析程序的整个实现过程，讲述每个函数的具体作用并附上关键代码部分。
此段程序使用了第三方库gmp进行大数运算。

3.1 gmp库的下载，安装和使用

3.1.1 gmp库的下载和安装

因为现有电脑中已经装好Mingw,因此只需在命令行中输入以下指令便可自动进行下载和安装:

$$mingw-get\ install\ mingw32-gmp$$

3.1.2 gmp库的使用

在C++文件中使用库，只需要用include语句进行库的引用即可：

$$\#include < gmp.h >$$

相关的函数和输入输出都可以在gmp大数据_手册上进行查阅和使用。

3.1.3 使用gmp库的测试代码

如Fig.1所示。

```c
#include <gmp.h>

int main() {
    mpz_t a;
    mpz_init_set_str(a, "12345678901234567890", 0);
    gmp_printf("%Zd\n", a);
    return 0;
}
```

Figure 1: 调用bmp库的测试代码

3.2 Fermat's Method

在Fermat Method中，我们需要找到一个t，使得$\sqrt{t^2-n}$是一个完全平方数。当n不是一个完全平方数时，$t=\frac{p+q}{2}$, $t>\sqrt{n}$。因此，我们需要不断令$t=\sqrt{n}+K(K=1,2,3...)$, 我们需要找到一个K使得 $\sqrt{t^2-n}$是一个完全平方数。找到之后，我们可以得到s，$s=\sqrt{t^2-n}$。最后，我们可以找到通过$p=t+s$和$q=t-s$ 得到p和q。伪代码如下所示：

Algorithm 1 Fermat Method

$t=\sqrt{n}$
while $t<n$ **do**
    $w=t^2-n$
    if $w\ is\ perfect\ square$ **then**
        $s=\sqrt{w}$
        $p=t+s$
        $q=t-s$
        $return\ p\ and\ q$
    end if
    $t=t+1$
end while

3.3 变量初始化

在此程序中，需要初始化的数只有给定的数N，因为它是一个大数，我们利用gmp库中的相关初始化函数进行赋值:

$$mpz\_init\_set\_str()$$

3.4 Fermat()函数

这个函数主要用于质因数分解大数N。
具体过程包括：
-获得N的平方根root(t)。

-判断N是否是一个完全平方数。
-若N不是一个完全平方数，则将root向上取整。
-在每次循环时，root的值都加一，同时判断$root^2 - n$是否为一个完全平方数，并保存$\sqrt{root^2 - n}$的值为s。
-如果$root^2 - n$是一个完全平方数，则通过当前的s和root得到p 和q的值，跳出循环。 关键代码如Fig.2所示。

```
if(ifExact == 0){//Judge whether t^2 < n or not
    mpz_add_ui(root, root, 1);//make t^2 > n
}
while(mpz_cmp(root, n) < 0){
    mpz_t w;
    mpz_init(w);
    mpz_mul (w, root, root);//w = t*t
    mpz_sub (w, w, n);      //w = t*t - n

    mpz_t s;
    mpz_init(s);
    int ifPerfectSquare = mpz_root(s, w, 2);//s = sqrt(w)
    if(ifPerfectSquare != 0){//Judge whether w is a perfect square
        mpz_t p;
        mpz_init(p);
        mpz_t q;
        mpz_init(q);

        mpz_add (p, root, s);//p = t + s
        mpz_sub (q, root, s);//q = t - s
        //Obtain the final results
        mpz_set (p1, p);//p1 = p
        mpz_set (p2, q);//p2 = q

        cout << "Factorization done" << endl;
        return;
    }

    mpz_add_ui(root, root, 1);//t++
}
```

Figure 2: Partial Content of Fermat() Function

4 实验结果与分析

4.1 实验结果I

当p和q之间的关系限制为：

$$|p - q| < 2\sqrt[4]{N}$$

给定N的值，实验结果如Fig.3所示。

4.2 实验结果II

当p和q之间的关系限制为：

$$|p - q| < 2^{11}\sqrt[4]{N}$$

给定N的值，实验结果如Fig.4所示。

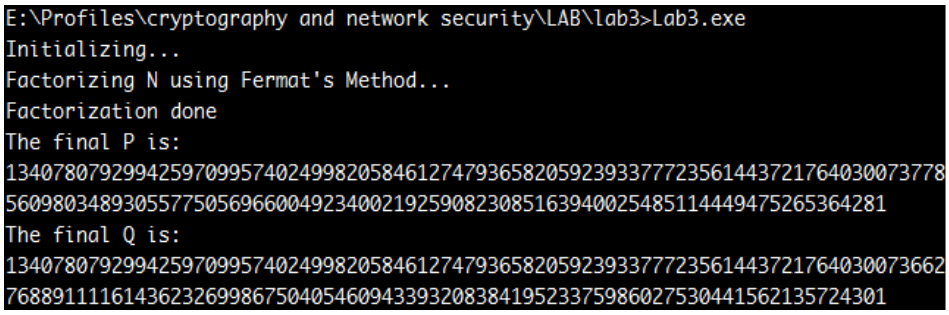

Figure 3: Experimental Result where $|p-q| < 2\sqrt[4]{N}$

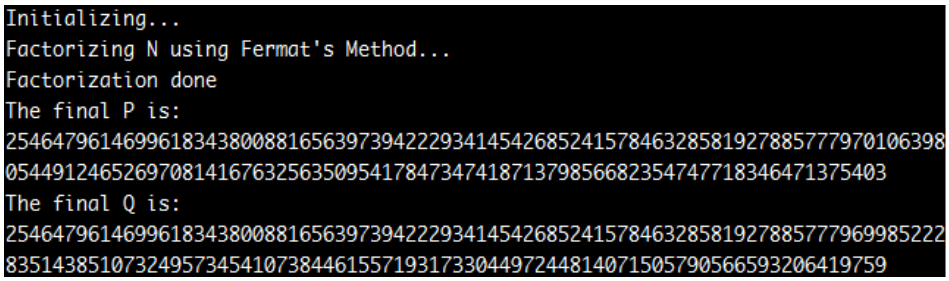

Figure 4: Experimental Result where $|p-q| < 2^{11}\sqrt[4]{N}$

4.3 结果分析

两个实验条件下，使用的都是Fermat Method进行质因数分解，因而算法上没有更改。
由Fig.3和Fig.4可知，在不同的条件下，Fermat Method都可以得到正确的结果。
另外，我们可以看到，当p和q之间的距离限制没有那么严格的时候，如同Fig.4所示，最终的p和q的值差距也会相应的增大。

5 实验感想

此次实验需要灵活运用gmp库，难度适中，有以下感想与收获。

1. 理解算法原理很重要，即使它不需要自己实现。在分解一个大数N 时，我们必须要知道自己使用的方法背后的原理，这样在我们在求满足不同条件的解时，能够根据自己的理解对算法进行适当的修改，从而使得算法可以用于更多的场合。

2.学会尽可能利用第三方资源。如果我们要不借助任何第三方资源而实现大数的分解，所需的代码量是相当惊人的。并且随着这个大数的位数的增加，算法的复杂度会急剧上升。适当的运用第三方资源，尤其是在那些不需要我们全权实现所有过程的工程中，可以极大的节省不必要的时间以及提升效率。

3. 在使用第三方库时，命令行操作往往是最简单的。大部分码代码时都比较倾向于使用IDE，如VS或者Eclipse等。IDE固然集各种功能与一身，但同时

这也让它本身变得臃肿。对于没有太多操作要求的人来说，似乎一个好的编辑器和命令行编译运行的模式更加方便一点。同时，对于第三库的引用，IDE需要的各种设置，往往令人失去耐心，不同于命令行的便捷。

4. 学会用一个方法不代表能够理解透彻其背后的逻辑。正如需要我们证明的两个不等式一样，只有能够通过手动证明出了这两个不等式才能说明你真正理解了这个方法。很多时候我们会屈服于"好像是这样的"的想法而从不会手动去证明，但是往往到自己真的需要去证明的时候我们发现是有困难的，从而成为了我们的弱点。

参考文献

[1] The GNU Multiple Precision Arithmetic Library， Edition 5.1.1， 11 February 2013