Лабораторная работа № 1 по курсу криптографии

Выполнил студент группы М8О-308Б-17 *Иларионов Денис.*

Условие

Разложить каждое из чисел n1 и n2 на нетривиальные сомножители.
Вариант 7.
n1=268887320029090028117214498253204095765884136483366193842361283776500643966781,
n2=14177467869787507650387834432016948376930580071471350079285831921442569467042365
90494758980427157782351530260852126352560893481056955596585856196760851613464821804136259107185547729368883111388512812700339059708262004996928275687558408584407339919174540282553261747449656964703936447130918315087871163722894672660845644433050799802860493503622897613938633077951879747971879859575334614760888258163959225587279203300668232112105942963026762617074322173483051121 87,

Метод решения

Я сначала долго думал над тем, как решить данную задачу, искал различные алгоритмы, но они мне показались довольно сложными. Наиболее эффективными алгоритмами для факторизации больших чисел являются метод квадратичного решета и факторизация с помощью эллиптических кривых. Какое-то время я думал, изучал эти алгоритмы. А позже, узнал от одногруппников, что на самом деле есть кое-какая зависимость у сомножителей второго числа. А именно, что первый его сомножитель находится как НОД с числом из соседнего варианта, а чтобы получить второй - нужно поделить число на этот сомножитель. Также, я узнал, что можно использовать любые средства, и это облегчило мне задачу. Есть много известных и более простых методов факторизации, но они работают за $O(n * \lg n)$ и за $O(\sqrt{n})$. Их вполне хватает для нахождения сомножителей чисел с 6-9 нулями, но тут у нас числа в разы больше. И такие алгоритмы будут выполнять свою работу так долго, что к тому времени Солнце уже успеет превратиться в красного гиганта и поглотить Землю. Поэтому, такие алгоритмы не эффективны для данной ЛР.

Первое число я решил разложить на этом сайте `https://www.alpertron.com.ar/ECM.HTM`
Второе число нашел уже с помощью своей длинной арифметики из 6 лабораторной по дискретному анализу. Но нужен именно самый эффективный алгоритм деления, ибо даже с делением на подобии бинарного поиска, у меня очень долго искала даже НОД с 1 вариантом, а с деление q^ нашло все НОДы за минуту.

Я думаю, что целью данной ЛР было не только узнать побольше об алгоритмах факторизации, но и именно разгадать шифр второго числа, найти взаимосвязь. Именно похожие решения и используются в криптографии, а шифр RSA как раз и основан на факторизации.

Результат работы программы

```
268887 320029 090028 117214 498253 204095 765884 136483 366193 842361 283776
500643 966781 (78 digits) = 414 150068 879409 136107 176764 405542 089303 (39
digits)  649 250936 397607 504492 837402 141095 065227 (39 digits)


 Variant 0 GCD = 1


 Variant 1 GCD = 1


 Variant 2 GCD = 1


 Variant 3 GCD = 1


 Variant 4 GCD = 1


 Variant 5 GCD = 1


 Variant 6 GCD = 1


 Variant 8 GCD = 1304738680325836098271854489803647581810257219791585840585109 6
848067467531800758251299143837940990002563420660299933504725395859764575192257
231951973654902109838773403174196838688508746956481136954975656279172115606067
161186055342248910459683542763941795236487891843320360172073464935554329358004
8106131166649


 Variant 9 GCD = 1


 Variant 10 GCD = 1


 Variant 11 GCD = 1
```

```
 Variant 12 GCD = 1

 Variant 13 GCD = 1

 Variant 14 GCD = 1

 Variant 15 GCD = 1

 Variant 16 GCD = 1

 Variant 17 GCD = 1

 Variant 18 GCD = 1

 Variant 19 GCD = 1

First Divider = 13047386803258360982718544898036475818102572197915858405851096848067467531800758251299143837940990002563420660299933504725395859764575192257231951973654902109838773403174196838688508746956481136954975656279172115606067161186055342248910459683542763941795236487891843320360172073464935554329358004810613116664 9

Second Divider = 10866135942445523506369738829513009804011846137158787994185181350860364600747683379598969094161641001971388218277013900041554993793670838977801293355191763

Vse Okey ;)
```

Листинг программного кода (только файл main.cpp)

```
#include "BigInt.h"
#include <iomanip>
using namespace std;

NBigInt::TBigInt GCD(NBigInt::TBigInt a, NBigInt::TBigInt b) {
        while (a != 0) {
                if (a < b) {
                        NBigInt::TBigInt tmp = a;
                        a = b;
                        b = tmp;
                }
                NBigInt::TBigInt k = a / b;
                a = a - b * k;
        }
        return b;
}

int stoi(const string& str) {
        int tmp = 0;
        for (int i = 0; i < str.size(); ++i) {
                tmp = tmp * 10 + (str[i] - '0');
        }
        return tmp;
}

int main() {

        freopen("input.txt", "r", stdin);
        string stringNum1;
        cin >> stringNum1;
        NBigInt::TBigInt ishodnick(stringNum1);


        for (int i = 0; i < 19; ++i) {
                string stringNum2;
                cin >> stringNum2;
                NBigInt::TBigInt number2(stringNum2);
                NBigInt::TBigInt res = GCD(ishodnick, number2);
                if (i >= 7) {
                        cout << "\n\n Variant " << i + 1;
```

```
                }
                else {
                        cout << "\n\n Variant " << i;
                }
                cout << " GCD = " << res << "\n\n";
        }



        NBigInt::TBigInt numberOne("13047386803258360982718544898036475818102572197915858405851096848067467531800758251299143837940990002563420660299933504725395859764575192257231951973654902109838773403174196838688508746956481136954975656279172115606067161186055342248910459683542763941795236487891843320360172073464935554329358004810613116664 9");
        NBigInt::TBigInt numberTwo = ishodnick / numberOne;
        cout << "\n\nFirst Divider = " << numberOne << "\n";
        cout << "\n\nSecond Divider = " << numberTwo << "\n\n";

        if ((numberOne * numberTwo) == ishodnick) {
                cout << "Vse Okey ;)\n";
        }
        else {
                cout << "Smthng wrong\n";
        }
        cout << "\n\n";

        while (true) {
                int wait;
                cin >> wait;
        }
        return 0;

}
```

Выводы

Факторизация целых чисел - это довольно важный аспект математики. Любое число можно разложить на простые множители, причем, единственным образом. Есть очень много алгоритмов факторизации, но для очень больших чисел еще эффективный алгоритм не придумали. Даже алгоритмы факторизации с помощью эллиптических кривых или квадратичного решета будут работать вечность при факторизации чисел с тысячью/десятью тысячами цифр.
Факторизация больших чисел является задачей большой сложности. Не существует никакого известного способа, чтобы решить эту задачу быстро. Также, после выполнения данной лабораторной работы, я понял, что алгоритм шифрования RSA (который как раз и основан на факторизации) является довольно эффективным и безопасным, и на него можно полагаться.