Московский авиационный институт
(Национальный исследовательский университет)

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

Лабораторная работа № 1

по курсу «Криптография»

Студент: Гаврилов М.С.

Группа: 80-306б

Преподаватель: Борисов А. В.

Оценка:

Москва, 2022

1. Постановка задачи

Разложить каждое из чисел n1 и n2 на нетривиальные сомножители.

Вариант 7.
7)
n1 = 2688873200290900281172144982532040957658841364833661938423612837765006496781,
n2=629078689652619110110463720495678438122734066105165791082866997162673569324602870777401334764597738095098477634387696990987529424753021134530017265150215095178706017790768452750003661306430506440038461778055276250554522875432338747558336595593969116730471713557463123324992221071219860627569543525496421898323106418814202836548716474270002795941557533485168611316063143230252477892493843858478050872063682679331994435823150412240379247670997336786353016381 41,

2. Выполнение работы

Для разложения первого числа была использована функция factor из СКА sage math.

```
In [12]: %time factor(2688873200290900281172144982532040957658841364833661938423612837765006439667 81, int_ = true)

         CPU times: user 13min 32s, sys: 547 ms, total: 13min 32s
         Wall time: 13min 36s

Out[12]: 414150068879409136107176764405542089303 * 649250936397607504492837402141095065227
```

На разложение уходило 13-14 минут (делал где-то четыре раза в разное время), в результате поучилось 2 числа, которые в произведении дают раскладываемое. Проверка:

```
In [89]: print(fd1, fd2)
         number1

         414150068879409136107176764405542089303 649250936397607504492837402141095065227

Out[89]: 2688873200290900281172144982532040957658841364833661938423612837765006439667 81

In [90]: fd1 * fd2

Out[90]: 2688873200290900281172144982532040957658841364833661938423612837765006439667 81

In [92]: fd1 * fd2 == number1

Out[92]: True
```

Второе число не может быть разложено стандартными методами за адекватное время ввиду того, что оно $\approx 10^{464}$, и широкодоступные современные вычислительные мощности не способны разложить число такого порядка.

Для его разложения нужно проверить, нет ли у него и у чисел из других вариантов данный л.р. общих делителей. Эту задачу удобно решать с помощью функции gcd СКА sage. Ввиду небольшого количества вариантов, автоматизация процесса перебора чисел не необходима. Подходящим числом оказалось число n2 из варианта 6:

Возьмем число из варианта 6 и проверим есть ли у него общие делители с исследуемым числом (я исследовал числа из авриантов с 1 по 6, но нетривиальный gcd нашелся только в шестом)

```
numberTest = 64600223543125825727933431006041630872163729412148655690961849128264649129519348448432905480579717691858255710088987
print(numberTest)
```

```
6460022354312582572793343100604163087216372941214865569096184912826464912951934844843290548057971769185825571008898779827285829
1985798882301364315097296445215058206642106717967854087278817870884728585442373238436494567531957863601533387300144173347748607
3778834937717127011778353951714710372986740597476106556688681361968741422855890119823712590155145658957911785029816244348821565
37110578742519717264490892519289690825163455967795361548541359178995038112756777859
```

```
testGcd = gcd(number2, numberTest)
print(testGcd)
```

```
2277976844345517187704850692340641434964115688970801344599764457991678654609850474616650965241941123630876964985949167513237194
8096575712988573236311190389
```

Можно убедиться, что получившееся число просто, так же, как и число, являющееся частным n2 и gcd(n2, n2(шестого варианта). Т.е. вместе они составляют разложение числа n2:

```
is_prime(testGcd)
```

```
True
```

Найденный gcd – простое чило, одна из частей разложения найдена.

```
sd1 = testGcd
```

```
sd2 = int(number2 / sd1)
```

```
is_prime(sd2)
```

```
True
```

Разложение второго числа:

Проверка правильности найденного разложения

```
print(sd1 * sd2 == number2)
```

```
True
```

```
Разложение верно. Вот оно:
```

```
print("{}\n *\n{}\n =\n{}".format(sd1,sd2,number2))
```

```
2277976844345517187704850692340641434964115688970801344599764457991678654609850474616650965241941123630876964985949167513237194
8096575712988573236311190389
 *
2761567533990271787461285898527429494790971185781187221102618127199156918302371470903929398227687961680564855285966673369346240
7039367744712379906230726372704428493862734209337660493798321587743399021935706835460765840849116891975754923300515378221209767
4974763295283164945971953831107964699518682765692561769
 =
6290786896526191101104637204956784381227340661051657910828669971626733569324602870777401334764597738095098477634387696990987529
4247530211345300172651502115095178706017790768452750003661306430506440038461778055276250554522875432338747558332659559396911673
0471713557463123324992221071219860627569543525496421898323106418814200283654871647427000279594155753348516861131606314323025247
78924938438584780508720636882679331994435823150412240379247670997336786353016381 41
```

3. Вывод

В ходе выполнения этой лабораторной работы я на практике ощутил, какие числа можно факторизовать за адекватное время, а какие – нет. Я немножко почитал про алгоритмы факторизации (Прежде чем понять, что все нормальные алгоритмы слишком сложные, чтобы самостоятельно их реализовывать). Также я понял, что порой для решения задачи нужно думать нестандартно, использовать информацию за пределами одного лишь условия задачи. (Самостоятельно додуматься искать НОД чисел из соседних вариантов я бы определенно не смог, потому и пишу это после того как сей секретный прием был раскрыт на лекции)