

Лабораторная работа № 1 по курсу: криптография

Выполнил студент группы М8О-308Б-17 МАИ *Милько Павел.*

Задача

Разложить каждое из чисел n_1 и n_2 на нетривиальные сомножители.

Вариант 12.

n_1 :

313230894596513941163065516500542159481861849753982064716706926040955753912601

n_2 :

69353225269381995525384724904375752918032449407382349454759184358213680514846622
76472561273504107781906830075394328076966980564652731377557949869817515416041455
75912570331718107924355720576988204493545005230906033105693432028755478062000534
78435263826499026879430301175011004508837392137334848183373483925857964547900042
45485859363365552338601237203357663127741515653465954857842494487929040401882928
024088822429726706628046686893973455185139755966946831836312393

Решение

Для поиска простых делителей я пытался использовать алгоритм факторизации Ферма. Метод основан на поиске чисел x и y , которые удовлетворяют соотношению $x^2 - y^2 = n$, что приводится к разложению $(x - y)(x + y) = n$. Также уравнение равносильно следующему: $x^2 - n = y^2$, то есть $x^2 - n$ – квадрат. Для начала алгоритма выбирается наименьшее x , такое что $x^2 > n$. Для каждого значения последующего значения вычисляют $(x + k)^2 - n$ и проверяют, не является ли это число точным квадратом. Если оно является точным квадратом, то получено разложение: $n = x^2 - y^2 = (x - y)(x + y)$. Так же можно проверить число на простоту: если один из сомножителей единица, то исходное число – простое. Метод работает быстро, если n является произведением двух близких к друг другу сомножителей.

Программа обрабатывала первое число более 12 часов, но так и не завершила вычисления. Поэтому я воспользовался готовой реализацией метода решета числового поля – msieve.

```
Tue Feb 25 01:27:23 2020
Tue Feb 25 01:27:23 2020
Tue Feb 25 01:27:23 2020 Msieve v. 1.53 (SVN unknown)
Tue Feb 25 01:27:23 2020 random seeds: 31d94c7b 069b18a6
Tue Feb 25 01:27:23 2020 factoring 2041 (4 digits)
```

```

Tue Feb 25 01:27:23 2020 p2 factor: 13
Tue Feb 25 01:27:23 2020 p3 factor: 157
Tue Feb 25 01:27:23 2020 elapsed time 00:00:00
Tue Feb 25 01:27:30 2020
Tue Feb 25 01:27:30 2020
Tue Feb 25 01:27:30 2020 Msieve v. 1.53 (SVN unknown)
Tue Feb 25 01:27:30 2020 random seeds: d45d5401 c30b9837
Tue Feb 25 01:27:30 2020 factoring
313230894596513941163065516500542159481861849753982064716706926040955753912601 (78
digits)
Tue Feb 25 01:27:30 2020 no P-1/P+1/ECM available , skipping
Tue Feb 25 01:27:30 2020 commencing quadratic sieve (78-digit input)
Tue Feb 25 01:27:30 2020 using multiplier of 1
Tue Feb 25 01:27:30 2020 using generic 32kb sieve core
Tue Feb 25 01:27:30 2020 sieve interval: 12 blocks of size 32768
Tue Feb 25 01:27:30 2020 processing polynomials in batches of 17
Tue Feb 25 01:27:30 2020 using a sieve bound of 999269 (39162 primes)
Tue Feb 25 01:27:30 2020 using large prime bound of 99926900 (26 bits)
Tue Feb 25 01:27:30 2020 using trial factoring cutoff of 27 bits
Tue Feb 25 01:27:30 2020 polynomial 'A' values have 10 factors
Tue Feb 25 01:29:17 2020 39510 relations (20731 full + 18779 combined from 211459 partial),
need 39258
Tue Feb 25 01:29:17 2020 begin with 232190 relations
Tue Feb 25 01:29:17 2020 reduce to 55927 relations in 2 passes
Tue Feb 25 01:29:17 2020 attempting to read 55927 relations
Tue Feb 25 01:29:17 2020 recovered 55927 relations
Tue Feb 25 01:29:17 2020 recovered 42152 polynomials
Tue Feb 25 01:29:17 2020 attempting to build 39510 cycles
Tue Feb 25 01:29:17 2020 found 39510 cycles in 1 passes
Tue Feb 25 01:29:17 2020 distribution of cycle lengths:
Tue Feb 25 01:29:17 2020 length 1 : 20731
Tue Feb 25 01:29:17 2020 length 2 : 18779
Tue Feb 25 01:29:17 2020 largest cycle: 2 relations
Tue Feb 25 01:29:17 2020 matrix is 39162 x 39510 (5.7 MB) with weight 1174885 (29.74/col)
Tue Feb 25 01:29:17 2020 sparse part has weight 1174885 (29.74/col)
Tue Feb 25 01:29:17 2020 filtering completed in 3 passes
Tue Feb 25 01:29:17 2020 matrix is 26985 x 27049 (4.3 MB) with weight 901416 (33.33/col)
Tue Feb 25 01:29:17 2020 sparse part has weight 901416 (33.33/col)
Tue Feb 25 01:29:17 2020 saving the first 48 matrix rows for later
Tue Feb 25 01:29:17 2020 matrix includes 64 packed rows
Tue Feb 25 01:29:17 2020 matrix is 26937 x 27049 (2.8 MB) with weight 662930 (24.51/col)
Tue Feb 25 01:29:17 2020 sparse part has weight 476210 (17.61/col)
Tue Feb 25 01:29:17 2020 commencing Lanczos iteration
Tue Feb 25 01:29:17 2020 memory use: 2.9 MB
Tue Feb 25 01:29:23 2020 lanczos halted after 428 iterations (dim = 26933)
Tue Feb 25 01:29:23 2020 recovered 13 nontrivial dependencies
Tue Feb 25 01:29:23 2020 p39 factor: 537228079155448813380781027030896715807
Tue Feb 25 01:29:23 2020 p39 factor: 583050117352260532679885280778162124743
Tue Feb 25 01:29:23 2020 elapsed time 00:01:53

```

Получанные числа действительно являются сомножителями n_1 и при перемножении получается исходное число.

Со вторым числом такой фокус не прошёл, потому что *msieve* не обрабатывает числа длиннее 311 символов (Второе число состоит из 464 символов).

По совету старшекурсников, сомножители второго числа я искал как НОД с числом другого варианта. Тут мой код заработал как надо. Я записал все варианты в

текстовый файл и простейшим алгоритмом Евклида нашёл число, которое имеет НОД больше единицы с искомым числом

find.py

```
1  #!/usr/bin/env python
2
3  import sys
4
5  if len(sys.argv) != 2:
6      print("expected filename")
7      sys.exit(1)
8
9  FILE = open(sys.argv[1], "r")
10
11
12  def read_one(file=FILE):
13      try:
14          n1 = int(file.readline()[3:])
15          n2 = int(file.readline()[3:])
16          return (n1, n2)
17      except:
18          return (0, 0)
19
20
21  def NOD(a, b):
22      while a != 0 and b != 0:
23          if a > b:
24              a %= b
25          else:
26              b %= a
27
28      return a+b
29
30
31  def main():
32      nums = []
33      num_variant = 12
34
35      while True:
36          n1, n2 = read_one()
37          if n1 == 0:
38              break
39
40          nums.append((n1, n2))
41
42      most = nums[num_variant]
43
```

```

44     for idx, num in enumerate(nums):
45         n1 = NOD(most[1], num[0])
46         n2 = NOD(most[1], num[1])
47         for id_res, num_nod in enumerate((n1, n2)):
48             if num_nod != 1:
49                 print(
50                     f"variant={idx}\nNOD:\n{num_nod}\norig:\n{most
51                         [1]}\nsecond:\n{num[id_res]}\n")
52
53                 return
54
55 if __name__ == "__main__":
56     main()

```

variant=8

NOD:

21499260310007687082066022284323057805839606673771229907076105795697304310081244
681783792143016969416752991877866429762049990504811441880293995264057694113

orig:

69353225269381995525384724904375752918032449407382349454759184358213680514846622
76472561273504107781906830075394328076966980564652731377557949869817515416041455
75912570331718107924355720576988204493545005230906033105693432028755478062000534
78435263826499026879430301175011004508837392137334848183373483925857964547900042
45485859363365552338601237203357663127741515653465954857842494487929040401882928
024088822429726706628046686893973455185139755966946831836312393

second:

42048597057914559795863063537336568313877042896313376165304172535228718621411206
31932433990091987949406971150143346271597491500705645949178093262134307318015307
8541946929337080958121008766495807246629051017536955212049514499180340005448669
59007173708778456158379658575505176706837646024069169819799100293687360677414134
30743475571851231332449065079917756805910750971996188143277035497059914480484135
755951043475401544587782005555278233737918896409814525174854107

Найденные делители второго числа:

x=21499260310007687082066022284323057805839606673771229907076105795697304310081244
681783792143016969416752991877866429762049990504811441880293995264057694113
y=32258423903588336156202381949516699594152400391309434660371157805122444218011017
84293724040955769716835194317346127820253149140844825528872564200901961634252917
70243285267671019113906630335645220486106241971033974999602452593327576481109191
429902602150345188892764823492265186634808527223692880107101784955561