Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу «Криптография»

Студент: И. Т. Батяновский
Преподаватель: А. В. Борисов
Группа: М8О-307Б

Дата:
Оценка:
Подпись:

Москва, 2020

Лабораторная работа №1

Задача:

Разложить каждое из чисел n1 и n2 на нетривиальные сомножители. Ниже представленый вариант.

1. n1=119760639583941053725652803731328419697649739176243841021915621242807618608591,

2. n2=4640559218166094156914109159111519163187841151743307361441882837464585000644675874846465567321539765251799375820937589239985719737640895597242823431300043940827590777924614801336402643706476244259987080742848555919096958592308671421375157228642120372475705421693466178995623443185598183763791363779033266801228117576963439635390847153152534159364312454724658773080761415353568661089057964083886997037671005754984876814452951475039288901199175397709461135789384143.

Вариант 2.

1 Описание решения проблемы

Было очевидно, что грубые алгоритмы (подобные brute force) не смогут решить поставленную задачу за разумное время. Я решил использовать сторонние библиотеки на pypi.org. Был испробован https://pypi.org/project/primefac/ и https://pypi.org/project/primefactors/. К сожалению, эти библиотеки решали проблему слишком долго только для числа n1(больше 2 часов).

После поиска информации на различных форумах, я нашел *https://docs.sympy.org*, который использовал три различных метода:

- Trial division
- Pollard's rho method
- Pollard's p-1 method

Засчет этого, работа факторизации ускорялась, но этого все равно не хватало для решения задачи.

Я решил использовать готовый проект реализующий msieve. Он за пару минут разложил n1 число. Однако, эта программа не принимает числа, содержащие более 300 цифр, следовательно, n2 число разложить через эту программу не получится. Но одногруппники подсказали, что один из множителей этого числа определяется как наибольший общий делиитель с одним из чисел другого варианта. Для поиска этого числа я написал программу на языке python для поиска этого множителя(GCDsearch.py). Поделив найденный НОД я нашел второй множитель числа n2, тем самым решив задачу.

2 Исходный код

Поиск НОДа для n2:

```
1  nums =
       [3523581180791504931870993551416295271017491061679972555096190205283337223522 17,
2          1197606395839410537256528037313284196976497391762438410219156212428076186085 91,
3          3448452281301592264881635710704176792350251390158020191525169262027118466601 41,
4          1607693578999756108281995391141095181675311345141909907851446669320766147178 41,
5          2741148223395896290240264954415574797138132280289801178690522789506812411948 19,
6          1087623532924484874412476636855136588931676469306271789461288899676431721541 27,
7          2688873200290900281172144982532040957658841364833661938423612837765006439667 81,
8          1232482689119379231999061412166453636650870454226893581040891853161489114961 03,
9          2849949678058592728534773278622454669783469198065854321335567699592693152711 11,
10         4723795527368714940581432391626228608969652751135434505802724898916670802077 63,
11         3619967274567848718556041810566056720886226662075781608112910608739971517088 87,
```

```
313230894596513941163065516500542159481861849753982064716706926040955753912601,
374456902508739435218273258671224457341348406488533188195528827819627513233269,
611219701749111463195451937544251195208759452152827846401772765239293765019 13,
383456614884902466726252731294544234658015390619372835826246625499154384118189,
242587413455689311805941697582103544343444025737930609728129303011307601823551,
181552877565998943910618543225528579935321447209736978912489118450818545230489,
319373613270896663765954115654922624879359841665992852658124487372881123570003,
374456902508739435218273258671224457341348406488533188195528827819627513233269,
284994967805859272853477327862245466978346919806585432133556769959269315271111,
588334127500298760075185369594470830068532538546945053004260193945595259217406 7
860897709822492909015687256084515086115417713054014938700451765379159402492861 4
184656004214221154240963339228120660363795445043318351770779654197231142468133 6
480578342707287553065652703205564979626673694073622920569912786167694946557747 7
601504089149197050128019045123770929345091698417436985288162503858546975249163 3
038785690560336276219854584839271502720881721227524465277533897866 19,
487382235506648564840107199192413681867587239828653545406477588822435405764775 9
827888672904881758021952112721372792540754490030245608396182923905355014076090 5
023863976659838390092229761027195955475247825918278599180637013523064720193934 4
097195516435438603272584360734684335554669921859642314670438294621187340324196 7
325626024974089074590478580274181015361612573624846467079841362239862486328418 9
543014224877216614751857244348601566526424673501072070269730652152 03,
464055921816609415691410915911151916318784115174330736144188283746458500064467 5
874846465567321539765251799375820937589239985719737640895597242823431300043940 8
275907779246148013364026437064762442599870807428485559190969585923086714213751 5
722864212037247570542169346617899562344318559818376379136377903326680122811757 6
963439635390847153152534159364312454724658773080761415353568661089057964083886 9
970376710057549848768144529514750392889011991753977094611357893841 43,
368515922715559843270802894042079050503316741288998902989988629075750090619884 8
536416211136144358007549202044937915961098909838877993478676055552793938396505 7
567370850592024862587000314182597797508084820751147072432264506835791979388925 2
524605356399543406695994353560240869536789217436246262173965693741483534014589 7
293482734232981975633739612690654329902264503074506941004524051644869226017975 8
061816949897640059513572259917204801107335737467148532100742077067 21,
350878599508444080952853398706611549957096015747528963620304623705617283643398 0
089009328018596525578795975212269369781723178881963030441758416340899464309056 0
729172272922309740696024351318426195621999652644598266620487056556057984210923 2
242348756658721911451558030402267607000629418003334389717856574908521618862072 3
352965399425751141566775855490460717246126766384778691062473224303381227995700 8
624589875316654893714142915368743119865326032445869621567424834833 01,
444846935455998652653477527411780397469905921223109224963400159004018826497115 9
285942473071403768975120070026009479818278660862590363336881868724607727370461 2
502220135161314926799920768560562583724263693922205379208036526084702693979386 1
273604438836830766816473453949582210821960320701492999465327034691730954612782 3
781869069725331428314977499114078997620214362713623954104893108026744745661033 9
818043575151247074356697868419555649375095424436367680314253807082 33,
472136206196364980886332901023461352580773059916025900996788374191838509617335 2
581810523133667792305553466613259404177221188191288539451279983509237773260105 9
788599452596435635291810761654415829846143647752979693595007872571067761870598 5
392465973501702267027564231502043364734252476192423827554141006629665243763066 1
```

```
078222967549760189525938225926355453552756983354285083800596385481859173835256775993355195346250828163678329623000733278409956864037403335179310263,
158968690785896053229304195025980740908911607577490592481192836972930851627507291449244730388234361901478286114627747425663442922357381267982988585772251423678977375807360238275429639874676052862046713568690409185767729868661335316050142125453936421554346233052917382325478595789259674397146933105369462870471989751163449408072638444931191132643054360803184618121059080807310404316851562692251939368391798187363382805306816975035313741234210109232681400128607993114470563577430403187898629612275091047447990814946786123832919869849235193164462877080490779182246565274295436732293643518871833908072627524231172982110419346551522765992254317516715888959815174190264715429324481989444969083616331327076407980393565709505006078950141506587407820420736302617335256351925247739018311504537066619041864399051765841946047321403468580781936233573521469460165494767804910732129539946607701693482114451990193860694698453061853232064399611262485504020168731000842257581537957328326497522478405002465359648875356810280292244547618070727524417592419776792612058732594852983180148665064058817407866064291179552422627557683886828462061069447032164569235069818669414169882863307032697282802157247652797734392044016320040859257401114524063142894607111829574025600918893253339517061607978068475589931239014683019592991614837523358909806258991077646147246997493894736434495372693444001308001278879395788963879,191624208718068015686171299450972805253515909112884480565867902529671655940443419162420871806801568617129945097280525351590911288448056586790252967165594044346648117256191866527259013257746490175941447883606374071784769363169152207581445356819643713116570717509704147072181122228045395187521359163973501984457964262201487421259483804145780046492118234512749646088825008417181554035121174581354219296962410856750448190529031735941575253507798593150790972216736431298009983402302302121276710704030134439278341757598100259379669607444268950730119603440006734480101099661237982591387883122230001102854441389846870436820919184377265648736526559593379272139428292838436152529262817891963724717308924224522305311182653859231485873649563920450252677624041195978388744710390172532363083063745412743753556715009911963945245091922784874734290220678484601501149189968384153401644820324493941862061208584686840594025223786924079444262714095490301772077126395790235999836003971290616988894725373002042174148527448991721,168843226853565253697616154422540493335291734846688074164655523608094046836939053377756690137486384608892630271670495825334901346501717168747651434545408082951222809155439069524222622271022327136748075330815779254986868124094373018454530478163301104392732758417509419570206294690430673567335499641590701419555059055047225534596163724964101292801909851819336345841569180224370576850378664988242673976806269467802281352706772727884244675999863931258724609849367757316698120282111148760357415934740218022123400447408847013442712701958320858567973149367256099699198928804324700476844645415672653368067889584026253505220722153568875423450919653644127144161477230078248529404392103475354920799309387153018516635049076332711782159866874962816730597954315002008001123373748886576429320113770107797396319904117148857373617146027153976389826461364816302384194818088644389113718040852129468401985584414791762568326896004766689308652227091416908444771934114327236064335695175033855568724514723276090909238902249450761163116179298370097637736609598746978539681190806175023739437824949790203114195447282762119216205286391137003028125331158247702385902798481867910823926760076341189111357818193897834136876367785553468541342743729023927657307836543731689119550558446364266971611293672837308855338590286435921893375062744052147047677417875
```

```
        934130977543281068768100090834326282132886721944207546209205488511 29,
        151093858430251474606868768035913871208482686953174983381615253610702995669437 8
        228665014484809993284680636465045336584670006512692482057168858805251730522412 4
        355755370476387591838494378611695821743531006167608614420833389111629829780186 5
        460907348745561834472564647434110644877018611946543743680554031457390231514801 0
        605642969399036239279990866481377552631038345038332671300460449150826133047599 4
        029527022204381323242408014804830559968501356093806127730885762649 39,
        150334999063135051279428968431307824504008023479374928828438810281152931865133 4
        186314509247654009172580006457439356152132860210881356046271699329201253058433 0
        361623216743051882118851116282237496594977166868163840331786137975686189271737 5
        280069279523162223543493433550049659931535778659520881621348942909061872472941 6
        131746965336847080815801599020573311110511374951097277607310799592097578622368 4
        234407181645957200918994270361355390957408076391671959950085809104 33,
        154062250949081794905352464916598198236271013859014568010836766059230094210745 5
        572128874985313317596143790569145958434047726221469334726361045510333507710004 1
        993455290528251101324297893844389907069275490156858433293934015201542373720790 9
        866832481701129682530282510285614357458021101462653008089510703033793477187855 4
        891597286701169436904639362768184131040154699128645755841829998901472679565730 2
        590989466425402873370832842632874326160946972589939452327670137815 01,
        162657059238403440123105985940845525481005091143114558077381732038544567859777 6
        695068312796145258618012655448521816316108022278762520239267979899184627816793 6
        565809063790777458251309334207819198020137034051556960335295557939983593891737 5
        588736685732913134320614863250625854639876172558771408300882834772707143477194 4
        964364829767905778186912771898431501076025378480110831840332479020783206206190 4
        051003949822187692691563935316046036041428410910392654850704146722 59,
        134212447269268081486469603983165720134193017053749088894818590919340492696122 3
        536479474040666460885376378742819197105748650798865879482304771084236256654384 2
        737997023364759696474517001465344300528458335539841650829928420258996565679227 7
        448431365676379334764576288837969209413664505934377115504369021966283040157293 1
        468290058814304724439824204259980816710796240835274604462076123461698477384443 7
        137512844829946074307551558342332836812531322809893949894719618171 43]


def computeGCD(x, y):
    while y:
        x, y = y, x % y

    return x


b = 464055921816609415691410915911151916318784115174330736144188283746458500064467587
484646556732153976525179937582093758923998571973764089559724282343130004394082759077 7
924614801336402643706476244259987080742848555919096958592308671421375157228642120372 4
757054216934661789956234431855981837637913637790332668012281175769634396353908471531 5
253415936431245472465877308076141535356866108905796408388699703767100575494848768144 52
951475039288901199175397709461135789384143
for num in nums:
    temp = computeGCD(num, b)
    # print(temp)
```

```
159     if temp != 1:
160         # print(num)
161         break
162 # print("~"*50)
163 # print(b)
164 # print("~"*25)
165 # print(temp)
166 # print("~"*25)
167 print(b // temp, "== first mnozhitel")
168 print(temp, "== gcd second mnozhitel")
```

3 Результаты работы программ

Результат msieve по разложению n1:

```
Mon Jun 01 18:22:44 2020 Msieve v. 1.53 (SVN 1005)
Mon Jun 01 18:22:44 2020 random seeds: a103ea20 2a2df016
Mon Jun 01 18:22:44 2020 factoring
119760639583941053725652803731328419697649739176243841021915621242807618608591
(78 digits)
Mon Jun 01 18:22:45 2020 searching for 15-digit factors
Mon Jun 01 18:22:45 2020 commencing quadratic sieve (78-digit input)
Mon Jun 01 18:22:45 2020 using multiplier of 1
Mon Jun 01 18:22:45 2020 using generic 32kb sieve core
Mon Jun 01 18:22:45 2020 sieve interval: 12 blocks of size 32768
Mon Jun 01 18:22:45 2020 processing polynomials in batches of 17
Mon Jun 01 18:22:45 2020 using a sieve bound of 958739 (37824 primes)
Mon Jun 01 18:22:45 2020 using large prime bound of 95873900 (26 bits)
Mon Jun 01 18:22:45 2020 using trial factoring cutoff of 27 bits
Mon Jun 01 18:22:45 2020 polynomial 'A' values have 10 factors
Mon Jun 01 18:25:03 2020 38031 relations (19274 full + 18757 combined from
209696 partial),
need 37920
Mon Jun 01 18:25:03 2020 begin with 228970 relations
Mon Jun 01 18:25:03 2020 reduce to 54400 relations in 2 passes
Mon Jun 01 18:25:03 2020 attempting to read 54400 relations
Mon Jun 01 18:25:04 2020 recovered 54400 relations
Mon Jun 01 18:25:04 2020 recovered 44162 polynomials
Mon Jun 01 18:25:04 2020 attempting to build 38031 cycles
Mon Jun 01 18:25:04 2020 found 38031 cycles in 1 passes
Mon Jun 01 18:25:04 2020 distribution of cycle lengths:
Mon Jun 01 18:25:04 2020 length 1 : 19274
```

```
Mon Jun 01 18:25:04 2020 length 2 : 18757
Mon Jun 01 18:25:04 2020 largest cycle: 2 relations
Mon Jun 01 18:25:04 2020 matrix is 37824 x 38031 (5.5 MB) with weight 1149110
(30.22/col)
Mon Jun 01 18:25:04 2020 sparse part has weight 1149110 (30.22/col)
Mon Jun 01 18:25:04 2020 filtering completed in 3 passes
Mon Jun 01 18:25:04 2020 matrix is 27455 x 27515 (4.4 MB) with weight 921775
(33.50/col)
Mon Jun 01 18:25:04 2020 sparse part has weight 921775 (33.50/col)
Mon Jun 01 18:25:04 2020 saving the first 48 matrix rows for later
Mon Jun 01 18:25:04 2020 matrix includes 64 packed rows
Mon Jun 01 18:25:04 2020 matrix is 27407 x 27515 (2.9 MB) with weight 684845
(24.89/col)
Mon Jun 01 18:25:04 2020 sparse part has weight 494937 (17.99/col)
Mon Jun 01 18:25:04 2020 commencing Lanczos iteration
Mon Jun 01 18:25:04 2020 memory use: 3.0 MB
Mon Jun 01 18:25:08 2020 lanczos halted after 435 iterations (dim = 27403)
Mon Jun 01 18:25:08 2020 recovered 16 nontrivial dependencies
Mon Jun 01 18:25:08 2020 p39 factor: 317975550097572442113430236685690984033
Mon Jun 01 18:25:08 2020 p39 factor: 376634742976910904214589735439587770927
Mon Jun 01 18:25:08 2020 elapsed time 00:02:24
```

Таким образом, n1 можно разложить на 317975550097572442113430236685690984033 и 376634742976910904214589735439587770927.

Разложение n2 через НОД:

```
Process finished with exit code 0C:\Anaconda\envs\sieve\python.exe C:/Users/Ivan/
PycharmProjects/sieve/GCDsearch.py
137164332595449202799493254626046310220717136390527790760363209847449202103911545179132388568097343049996022490023564914875033649313396474861119468644088 59 == first mnozhitel
338321131328863932564818680296074563081532769076856609725860402543148614331548252714137193761514829808776013335385989807884956317300479543405309911233314952878929489767681129617464167512042524219275287655118056649888395988512738187341200216598691518616681800731333973241210979683235278330374664946737167178077 == gcd second mnozhitel

Process finished with exit code 0
```

Получили разложение n2 (first mnozhitel и gcd second mnozhite).

4 Выводы

Эта работа показала насколько процесс факторизации сложен как по ресурсам, так и по времени. У меня был опыт работы с алгоритмом шифрования RSA (летняя практика в прошлом году). Однако, «пощупать руками» сам процесс факторизации и на практике понять стойкость RSA мне не удалось. Как оказалось, факторизация даже во много раз меньших чисел уже занимает огромное время.

Интересно было узнать, из статьи которая мне попалась в ходе работы, что в теории большие корпорации могут получить закрытые ключи через открытые, потратив на это миллионы долларов(хотя даже при такой финансовой поддержке, процесс займет пару месяцев).

Список литературы

[1] *Метод factorint()*
URL: https://docs.sympy.org/latest/modules/ntheory.html?highlight=factorintsympy.ntheory.factor_$factorint$

[2] *Реализация Msieve*
URL: https://sourceforge.net/projects/msieve/

[3] *Метод primefactors*
URL: https://pypi.org/project/primefactors/

[4] *Метод primefac*
URL: https://pypi.org/project/primefac/

[5] *Весь исходный код*
URL: https://github.com/Ivan-Batyanovsky/Cryptography/tree/master/Lab1.