МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Кафедра ИИТ

ОТЧЁТ
По лабораторной работе №3
«Атака на алгоритм шифрования RSA посредством метода бесключевого чтения»

Выполнил:
Студент группы ИИ-22
Варицкий М.И.
Проверила:
Хацкевич А.С.

Брест 2024

Цель работы: изучить атаку на алгоритм шифрования RSA посредством метода бесключевого чтения

Ход работы

Вариант	Блок зашифрованного текста			Модуль		
	C_1	C_2	C_3	N_1	N_2	N_3
18	1049662825446164834040003 5357691827658426558873963 3426829664787233746822123 1519559284326799505859750 1585151991221226562414674 5598259048870721529745090 8908900723108219852569451 2413033722102188111593467 9642415679081448055464314 0211156155170686744140635 7628346800729914092699576 8666750985254029495434328 5990883653662587944721980 9774691072261221818682293 5960003584849560298234982 7762479458088052741882724 7506275760840411841276105 1928877757311788890737023 8359582658519705567835856 4024280236768143980395139 2413949075848449266804071 9167459452097604673083482 5022091308791466288411534 7991806619236221580555645 642094551129889193	5325107969226753777805517 0338838367944759407473363 5086102582513518450796058 7711622209173123231563801 6596837141919957491626577 5459125362247298194393630 8893296797013592882870993 1022732342203105985836250 4303960319163414132048920 4198808415027676097788090 0310345161518276137801210 5422517627256378239164337 1592616497819463907472731 0389089079264669684671158 2873172720502996005606560 0794746035633192139002776 0319098938964297015349573 4445954241931140830522671 4424888030367174802416197 9674385697426754291360465 8494959000501405364216678 8386379672634097123305325 7339356304463117830283594 5279098269424191309539065 7769316207093164	1160265419572409133487300 1374230683147976734599074 6692034116945268400650112 9281626946471582569418404 1401745261297411723517696 7078530705812887623443860 2621980184177124713654531 8065176712059193602787019 5929488655174159241227626 1152767383263154417409220 8133069261365772205384455 6671839415037167438048292 7683596614090685826008829 2605911060883463853864224 8729381785808048747366235 3375800634582549084512211 0770148700761078128414024 2471208600543744938256017 8217404625246459978976389 0269795265308848538333841 9397444538150370453356205 7968250539738621219562824 4824008625910960433871964 2975630414480408795906937 24605867166234775	1988823447795522235486025614 9591781307886469688980288373 5007595570828029581036373236 6831897563216696924915585707 2258048697436072769765678313 5986938045847716222396466229 9652248816417371941796429668 8831241165612106403576876704 6697894240504324133884151404 1684952633610963967998965811 8201448601716804672660911458 9913260030681570327952220974 3526941313493450957409519182 4164277158000378191123796831 9662644289161937101454344942 5992347833310185483400955730 6981127433099625774842429719 5732122500031360688053933012 9982067096054289247026443788 0342978927462866388838226438 5750786040731818091262167046 4620518330017695759175571179 1	1646221542648925522589258 4869037735577619763065989 5198608388016370800309275 9167222684479041481522966 3435863181005111799666920 2609103328191214330391507 6375051515476984442070726 6778970735831472810257625 5185358368072566013539993 2125970891606520277659610 5716120454204639632965057 3616878195382915147271268 2744624066982978056568816 8788034697938631005194048 8972947425240617016863702 2279827319684082919849234 6766492702634706034700325 5719013810562950478555865 9979584170787299724163742 3827684066350760466732191 6258401787403961202276146 9029923200187538408573853 6799340026547128897434438 3279997791357294011228618 75632063844321783	1370621589365473186784992 6186816326043180524404836 2110906717098098731978844 4349773767355686067337772 6630586800530419687011700 4622943528918341393351922 8814038399129414806266679 9078243187205290125500344 4757846120480420187838401 4780518978825608154862952 7787032118887683695511417 2783154503350379405404354 1071268943726706017767820 8300217469465255211061531 4529771762648625985405234 0396637716936358070887555 4706717418942792212352735 9347558991525094364581538 6648309778058038386517980 1030968459321891492138603 7803292294238348855709593 2432187305864653050728740 6083683889974262349678955 0392971543110996218888578 49283747864955581

E=3

Код программы:

```
from sympy import mod_inverse, integer_nthroot


def chinese_remainder_theorem(c1, c2, c3, n1, n2, n3):
    """

    M ≡ c1 (mod n1)
    M ≡ c2 (mod n2)
    M ≡ c3 (mod n3)

    """
    N = n1 * n2 * n3
    N1 = N // n1
    N2 = N // n2
    N3 = N // n3

    y1 = mod_inverse(N1, n1)
    y2 = mod_inverse(N2, n2)
    y3 = mod_inverse(N3, n3)

    M = (c1 * N1 * y1 + c2 * N2 * y2 + c3 * N3 * y3) % N

    return M


def decrypt_rsa_cubic_root(M):
    """
    Извлекает кубический корень из M.
    """
    low, high = 0, M
    while low <= high:
        mid = (low + high) // 2
        mid_cubed = mid ** 3
        if mid_cubed == M:
            return mid
        elif mid_cubed < M:
            low = mid + 1
        else:
            high = mid - 1
    # Если не нашли точного куба, попробуем найти наименьший приближённый
    return low - 1  # возвращаем приближённое значение



c1 = int(

"1049662825446164834040003535769182765842655887396334268296647872337468221231519559284326799505859750158515199122122656241467455982590488707215297450908908900723108219852569451241303372210218811159346796424156790814
48055464314021115615517068674414063576283468007299140926995768666750985254029495434328599088365366258794472
```

```
19809774691072261221818682293596000358484956029823498277624794580880527418827247506275760840411841276105192
88777573117888907370238359582658519705567835856402428023676814398039513924139490758484492668040719167459452
09760467308348250220913087914662884115347991806619236221580555645642094551129889193")
c2 = int(

"53251079692267537778055170338838367944759407473363508610258251351845079605877116222091731232315638016596837
71419199574916265775459125362247298194393630889329679701359288287099310227323422031059858362504303960319163
41413204892041988084150276760977880900310345161518276137801210542251762725637823916433715926164978194639074
72731038908907926466968467115828731727205029960056065600794746035633192139002776031909893896429701534957344
45954241931140830522671442488803036717480241619796743856974267542913604658494959000501405364216678838637967
26340971233053257339356304463117830283594527909826942419130953906577693162070931641")
c3 = int(

"11602654195724091334873001374230683147976734599074669203411694526840065011292816269464715825694184041401745
26126974117235176967078530705812887623443860262198018417712471365453180651767120591936027870195929488655174
15924122762611527673832631544174092208133069261365772205384455667183941503716743804829276835966140906858260
08829260591106088346385386422487293817858080487473662353375800634582549084512211077014870076107812841402424
71208600543744938256017821740462524645997897638902697952653088485383338419397444538150370453356205796825053
97386212195628244824008625910960433871964297563041448040879590693724605867166234775")
n1 = int(

"19888234477955222354860256149591781307886469688980288373500759557082802958103637323668318975632166969249155
85870722580486974360727697656783135986938045847716222396466229965224881641737194179642966888312411656121064
03576876704669789424050432413388415140416849526336109639679989658118201448601716804672660911458991326003068
15703279522209743526941313493450957409519182416427715800037819112379683119662644289161937101454344942599234
78333101854834009557306981127433099625774842429719573212250003136068805393301299820670960542892470264437880
34297892746286638838226438575078604073181809126216704646205183300176957591755711791")
n2 = int(

"16462215426489255225892584869037735577619763065989519860838801637080030927591672226844790414815229663435863
31810051117996669202609103328191214330391507637505151547698444207072667789707358314728102576255185358368072
56601353999321259708916065202776596105716120454204639632965057361687819538291514727126827446240669829780565
68816878803469793863100519404889729474252406170168637022279827319684082919849234676649270263470603470032557
19013810562950478555865997958417078729972416374238276840663507604667321916258401787403961202276146902992320
01875384085738536799340026547128897434438327999779135729401122861875632063844321783")
n3 = int(

"13706215893654731867849926186816326043180524404836211090671709809873197884443497737673556860673377726630586
68005304196870117004622943528918341393351922881403839912941480626667990782431872052901255003444757846120480
42018783840147805189788256081548629527787032118887683695511417278315450335037940540435410712689437267060177
67820830021746946525521106153145297717626486259854052340396637716936358070887555470671741894279221235273593
47558991525094364581538664830977805803838651798010309684593218914921386037803292294238348855709593243218730
58646530507287406083683889974262349678955039297154311099621888857849283747864955581")

# Восстановление исходного текста
M = chinese_remainder_theorem(c1, c2, c3, n1, n2, n3)
original_message = decrypt_rsa_cubic_root(M)
print("Восстановленное исходное сообщение:", original_message)
```

Результат работы:

```
Восстановленное исходное сообщение: 12297716391223205693139262921852347835386283927572210118724121786639221191336447344226147900884630505438964344706801

Process finished with exit code 0
```

Вывод: изучил атаку на алгоритм шифрования RSA посредством метода бесключевого чтения