Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Информационная безопасность

Лабораторная работа №4
Атака на алгоритм шифрования RSA, основанная на Китайской теореме об остатках

Вариант 14

Студент: Мокров Семён Андреевич

P34121

Преподаватель: Маркина Татьяна Анатольевна

Санкт-Петербург 2023

Содержание

Цель работы

изучить атаку на алгоритм шифрования RSA посредством Китайской теоремы об остатках.

Задание

– ознакомьтесь с теорией в [3], в подразделе («Атака на основе Китайской теоремы об остатках»);
39
– получите вариант задания у преподавателя. Экспонента для всех вариантов e = 3;
– используя Китайскую теорему об остатках, получите исходный текст;
– результаты и промежуточные вычисления значений для любых трех блоков шифрованного текста оформите в виде отчета.

Вариант задания

14	494980336813	495019868347	496510218943	405186643929	298462743436	372083067441
				264588538265	26894204289	354383414943
				58896941920	266800308083	31782553847
				424470122024	469634672912	213067042090
				445830333875	423565503334	22742161466
				98276685134	418775305332	313919341914
				210238595626	112405305103	71514328634
				176058872641	302129659337	117790204322
				185715938214	323850375295	268549130622
				418034348683	438598232992	409153352258
				52552730024	10359943018	316714994539
				481876348312	298111389169	270152277750
				438600466605	277384894755	128472385009

Листинг разработанной программы

```
from decimal import Decimal

from numpy.core.defchararray import isnumeric
from termcolor import colored

N_1 = 494980336813

N_2 = 495019868347

N_3 = 496510218943

C_1 = '''
405186643929
264588538265
58896941920
424470122024
445830333875
98276685134
210238595626
176058872641
185715938214
418034348683
52552730024
481876348312
438600466605
'''

C_2 = '''
298462743436
26894204289
266800308083
469634672912
423565503334
418775305332
112405305103
302129659337
```

```
323850375295
438598232992
10359943018
298111389169
277384894755
'''

C_3 = '''
372083067441
354383414943
31782553847
213067042090
22742161466
313919341914
71514328634
117790204322
268549130622
409153352258
316714994539
270152277750
128472385009
'''


def get_int_list(source_list):
    result = []
    for curr_str in source_list.split():
        if isnumeric(curr_str):
            result.append(int(curr_str))
    return result


print(colored("Стартовые данные:", "green"))
print("C1: " + C_1)
print("C2: " + C_2)
print("C3: " + C_3)
print("N1: " + str(N_1))
```

```
print("N2: " + str(N_2))
print("N3: " + str(N_3), "\n")

C_1 = get_int_list(C_1)
C_2 = get_int_list(C_2)
C_3 = get_int_list(C_3)

result = ""

M_0 = N_1 * N_2 * N_3

m_1 = N_2 * N_3
m_2 = N_1 * N_3
m_3 = N_1 * N_2

n_1 = pow(m_1, -1, N_1)
n_2 = pow(m_2, -1, N_2)
n_3 = pow(m_3, -1, N_3)

print(colored("Промежуточные вычисления", "green"))
print("M0: " + str(M_0))
print("m1: " + str(m_1))
print("m2: " + str(m_2))
print("m3: " + str(m_3))
print("n1: " + str(n_1))
print("n2: " + str(n_2))
print("n3: " + str(n_3), "\n")

for i in range(len(C_1)):
    y = C_1[i] * n_1 * m_1 + C_2[i] * n_2 * m_2 + C_3[i] * n_3 * m_3
    print("y[" + str(i) + "]: " + str(y))

    current_bytes = y % M_0
    print("y[" + str(i) + "] mod M0: " + str(current_bytes))

    current_result = round(current_bytes ** (Decimal(1 / 3)))
    print("y[" + str(i) + "] mod M0 ** (1 / 3): " + str(current_result))
```

```
    current_part = current_result.to_bytes(length=4,
byteorder='big').decode('cp1251')
    print("Текущий текст: " + current_part, "\n")

    result += current_part

print(colored("Полученное сообщение: ", "green"))
print(result)
```

Исходный код

Исходный код расположен в репозитории:
https://github.com/semwett0301/information-security

Результаты работы программы

Скриншоты вывода программы

```
Стартовые данные:
C1:
405186643929
264588538265
58896941920
424470122024
445830333875
98276685134
210238595626
176058872641
185715938214
418034348683
52552730024
481876348312
438600466605
```

```
C2:
298462743436
26894204289
266800308083
469634672912
423565503334
418775305332
112405305103
302129659337
323850375295
438598232992
10359943018
298111389169
277384894755
```

```
C3:
372083067441
354383414943
31782553847
213067042090
22742161466
313919341914
71514328634
117790204322
268549130622
409153352258
316714994539
270152277750
128472385009
```

```
N1: 494980336813
N2: 495019868347
N3: 496510218943

Промежуточные вычисления
M0: 12165746662523251065302685441549667 3
m1: 245782423214104005497221
m2: 245762795403502512848659
m3: 245025101163524977558111
n1: 113402065412
n2: 99985317028
n3: 282471309470
```

```
y[0]: 443803060019301687185040028640414566559097533 50
y[0] mod M0: 430489907018387718589447166 51
y[0] mod M0 ** (1 / 3): 3504728051
Текущий текст: Резу

y[1]: 3256331998876734047629762051926062717909352671 8
y[1] mod M0: 62062589738442145656666816512
y[1] mod M0 ** (1 / 3): 3959223008
Текущий текст: льта

y[2]: 10397337521668625532696244128996180358332499346
y[2] mod M0: 67744879843573900251181255168
y[2] mod M0 ** (1 / 3): 4076544232
Текущий текст: ты и

y[3]: 38118024750630147908386975960907178634086406772
y[3] mod M0: 66882064909035077326321932061
y[3] mod M0 ** (1 / 3): 4059163621
Текущий текст: ссле

y[4]: 24408466591055905796878368126665550555267653688
y[4] mod M0: 56661195822765796957104603136
y[4] mod M0 ** (1 / 3): 3840860896
Текущий текст: дова

y[5]: 34756780270806548098085417914446304115296476412
y[5] mod M0: 63591425627598018592562774016
y[5] mod M0 ** (1 / 3): 3991469856
Текущий текст: ния
```

```
y[6]: 135716078525009581455879532210681538606663358 88
y[6] mod M0: 65182966059870751523102149661
y[6] mod M0 ** (1 / 3): 4024494821
Текущий текст: паке

y[7]: 204838486172877972049453797861811810613699993 96
y[7] mod M0: 67704879172179277046844260352
y[7] mod M0 ** (1 / 3): 4075741728
Текущий текст: тов

y[8]: 317211600300414953036533285493824608656659232 08
y[8] mod M0: 62811260531439438951667081781
y[8] mod M0 ** (1 / 3): 3975079661
Текущий текст: можн

y[9]: 507476328785660108190487197885030005898649057 60
y[9] mod M0: 63766835441247478218219241784
y[9] mod M0 ** (1 / 3): 3995136494
Текущий текст: о со

y[10]: 2363998941862042438645178641470242635443785601 4
y[10] mod M0: 70250938705055230950755975221
y[10] mod M0 ** (1 / 3): 4126204141
Текущий текст: хран
```

```
y[11]: 3945429469091027646950912485313429051592950911 2
y[11] mod M0: 59695714265107189451369416991
y[11] mod M0 ** (1 / 3): 3908238431
Текущий текст: ить_

y[12]: 279327655980206434320413649632318864565838182 50
y[12] mod M0: 40965350428150879852702269 44
y[12] mod M0 ** (1 / 3): 1600069664
Текущий текст: __

Полученное сообщение:
Результаты исследования пакетов можно сохранить___
```

Полученное сообщение

Результаты исследования пакетов можно сохранить___

Выводы

В данной лабораторной работе я:

- Ознакомился с принципом взлома RSA, основанного на Китайской теореме об остатках
- Реализовал процесс взлома на Python