БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №5

по дисциплине «Средства и методы защиты информации в интеллектуальных системах»

Выполнил студент гр. 121701

Мулярчик Д.С.

Проверил

Сальников Д. А.

Минск 2023

Задание:

Разработать программное обеспечение, реализующее функции генерации секретного и открытого ключей, шифрования и цифровой подписи для алгоритма RSA. Обмен входными и выходными данными должен осуществляться через файлы:

открытого ключа;

секретного ключа;

исходного сообщения;

зашифрованного сообщения. Для повышения скорости шифрования использовать метод последовательного возведения в квадрат и умножения. Выполнить тестирование разработанного программного обеспечения на 10 наборах тестовых данных. Длина чисел p и q должна быть не менее 1024 бит.

Проверка, является ли число простым

```python
def is_prime(n, k=5):
    if n <= 3:
        return n == 2 or n == 3
    if n % 2 == 0:
        return False

    r, s = 0, n - 1
    while s % 2 == 0:
        r += 1
        s //= 2

    for _ in range(k):
        a = random.randint(2, n - 2)
        x = pow(a, s, n)
        if x == 1 or x == n - 1:
            continue
        for _ in range(r - 1):
            x = pow(x, 2, n)
            if x == n - 1:
                break
        else:
            return False

    return True
```

Генерация случайных простых чисел

```python
def rabin_miller(num):
    s = num - 1
    t = 0

    while s % 2 == 0:
        s = s // 2
        t += 1
    for _ in range(5):
        a = random.randrange(2, num - 1)
        v = pow(a, s, num)
        if v != 1:
            i = 0
            while v != (num - 1):
                if i == t - 1:
                    return False
                else:
                    i = i + 1
                    v = (v ** 2) % num
        return True
```

```python
def is_prime(num):
    if num < 2:
        return False
    lowPrimes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
                 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151,
                 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241,
                 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349,
                 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449,
                 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569,
                 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661,
                 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787,
                 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907,
                 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]

    if num in lowPrimes:
        return True
    for prime in lowPrimes:
        if (num % prime == 0):
            return False
    return rabin_miller(num)
```

```python
def gcd(a, b):
    while a != 0:
        a, b = b % a, a
    return b


def generate_prime(keysize=1024):
    while True:
        num = random.randrange(2 ** (keysize - 1), 2 ** (keysize))
        if is_prime(num):
            return num


generate_prime()
```

113636889940568476707111782919375034446131754416557475257929055246169134273371613053029907785467438943090239351286162612468599679752263067472198014250648284000872995165087336367097973110007058915803380230013602463056066349750711229969802339184178834770385623217192602054494873657378845367207861564058675937057

Теперь, сгенерируем два случайных простых числа p и q, вычислим их произведение, а также функцию Эйлера. В качестве открытой экспоненты используем одно из простых чисел Ферма (17, 257, 65537).

```python
p, q = generate_prime(KEYSIZE), generate_prime(KEYSIZE)

n = p * q
f = (p - 1) * (q - 1)

while True:
    e = random.randrange(2 ** (KEYSIZE - 1), 2 ** KEYSIZE)
    if gcd(e, f) == 1:
        break
```

Метод find_mod_inverse() вычисляет число d, мультипликативно обратное числу e по модулю n (т.е. удовлетворяет сравнению d*e = 1mod f(n))

```python
def find_mod_inverse(a, m):
    if gcd(a, m) != 1:
        return None
    u1, u2, u3 = 1, 0, a
    v1, v2, v3 = 0, 1, m

    while v3 != 0:
        q = u3 // v3
        v1, v2, v3, u1, u2, u3 = (u1 - q * v1), (u2 - q * v2), (u3 - q * v3), v1, v2, v3
    return u1 % m


d = find_mod_inverse(e, f)
```

Открытый и закрытый ключ

```
public_key = (e, n)
private_key = (d, n)

print(f"Закрытый ключ: {private_key}")
print(f"Открытый ключ: {public_key}")
```

Закрытый ключ:
(46642922747170514739578432807000823385994583530908208440915449965279357391212007
78037873820306074806406776720280427444514111846353884331508798456209183672675620
84173924827129555285834355532980469975873310248016479107661191676471337081223991
11138627255392869480848015387884852626503728772996230413387651513203837106855893
11632871185377718064504247425726317859806818547894471040544120961785100372700481
13430691091641081869399634550004354945390111467955256893831391031947009851681179
81018154050998772950424880625880625162907795955124662837805047354722438029789265
80718079103414059835524055189786636591367847835087644975,
18472628441161008285679630582786000930915009087574434325891090251397634933456542
03206274748860806140693172716284154505753319085076601140769939712841715570225905
68727590075574395718285114833810728976039041447937497834761631829062181342966097
51859760012936104013368282876838303767784700815440107315341943171551600877595956
83580693240912955611945650737420108270386671999640679509227095670234520931302411
13912223163761050306067426925387325757064189411368532969553979818623678113577168
47550780969541653999180096591876174817562318538243636674472536162634060187565196
36267426403083415262566566722526290896637681016902644992 7)

Открытый ключ:
(11462650724323269179002208226867723622383930243544083927809828494767204082775462
22127953551129100502810703939770797729806797524347275127367081906299363920076069
11635172637683079609949429819195022666125820567685261795679106685660687600284166
33765836691956832177957458221392925442428853225872240313896345557040 7,
18472628441161008285679630582786000930915009087574434325891090251397634933456542
03206274748860806140693172716284154505753319085076601140769939712841715570225905
68727590075574395718285114833810728976039041447937497834761631829062181342966097
51859760012936104013368282876838303767784700815440107315341943171551600877595956
83580693240912955611945650737420108270386671999640679509227095670234520931302411
13912223163761050306067426925387325757064189411368532969553979818623678113577168
47550780969541653999180096591876174817562318538243636674472536162634060187565196
36267426403083415262566566722526290896637681016902644992 7)

Исходным сообщением будет использоваться целое число от 0 до n-1.

Для зашифрования используем открытый ключ {e, n}, вычислим криптограмму:

```
m = random.randint(1, 100)

def fast_mod_exp(b, exp, m):
    res = 1
    while exp > 1:
        if exp & 1:
            res = (res * b) % m
        b = b ** 2 % m
        exp >>= 1
    return (b * res) % m

print(f"Исходное сообщение: {m}")

c = fast_mod_exp(m, e, n)

print(f"Криптограмма: {c}")

dc = fast_mod_exp(c, d, n)

print(f"Расшифрованная криптограмма: {dc}")
```

Исходное сообщение: 64

Криптограмма:
22639915983328847910818864248913574702108264105133035294340438810431080148828397
45935825442231782350385438228856419826882763800220049335202994904149311064795913
85990559992108341856414877862784432968783523856053899357545709014543395168502084
45580220364379602162847356671789186599564968865786247330258700746656246200407574
72175741159968726912262725560624431919112721552097605343943300872749227279815450
82949964581759058230041996973289392735076098504283482902501780697034418748811249
96168092979767754343153748018206853825713842395372700817619631742395507971667735
10631400623727019895152645576735274798180893491345976941

Расшифрованная криптограмма: 64

Цифровая подпись

Для создания цифровой подписи s с помощью секретного ключа вычисляют s = m^d mod n. Затем формируют пару {m, s} и отправляют получателю.

```
s = fast_mod_exp(m, d, n)

signature = (m, s)

print(f"{signature=}")
```

signature=(64,
430697386962542657249347261550442303229345458724841743809001212076471938316445307950757161637438544448652334028630326429997394693167059092892115594939249711356930264563641678552222606442928884455898530954770392843572444632875402726976781276136123241747031348474152356459661617056482684889057586372427236986349309188019334484284827787570411955987215835299206868544647707923614561676365764508516625287960263363512045299236757066116896971014163764359276075381153547789099310509944134850041114648346307957271001892103611029473172311379725978523707151262423767682150469632816492137947977717120167132545848963989378008836 2)

Теперь проверим цифровую подпись: вычислим прообраз сообщения из подписи.

```
ms = fast_mod_exp(s, e, n)

is_valid = m == ms

print(f"{is_valid=}")
```

is_valid=True