1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт компьютерных наук и кибербезопасности

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «**Математические примитивы криптографии**»
2. по дисциплине «Основы информационной безопасности»
3. Выполнил
4. студент гр. 5151001/40001 Кириллов Д.А

<*подпись*>

1. Преподаватель
2. асс. преподавателя Орёл Е.М

<*подпись*>

1. Санкт-Петербург
2. 2025
3. **Цель работы**

Приобретение расчетных навыков в модульной арифметики, используемой в криптографических алгоритмах и протоколах, ознакомление с математическими вычислениями, используемыми для сокрытия сообщений на примерах алгоритма шифрования RSA и ранцевой криптосистемы Меркля-Хеллмана.

1. **Постановка задачи**

* Изучить использование формул модульной арифметики;
* Изучить метод шифрования алгоритмом Цезаря;
* Научиться находить наибольший общий делитель алгоритмом Евклида;
* Научиться определять является число простое или составное методом Миллера;
* Изучить генерации ключей в алгоритм RSA;
* Научиться зашифровывать и расшифровывать тексты по формулам;
* Научиться подписывать текст цифровой подписью и производить обратную операцию;
* Смоделировать процесс установления сеансового ключа;
* Разработать утилиту шифрования и дешифрования с помощью алгоритма Меркля-Хеллмана.

1. **Описание решения**
2. Нужно вычислить по формуле ((Nгр +Ncп)11 +Ф3)(mod 11).

Nгр.= 1 (номер учебной группы)

Nсп. = 10 (порядковый номер в списке группы)

Ф3 = 19 (порядковый номер в алфавите третьей буквы фамилии)

((Nгр +Ncп)11 +Ф3)(mod 11) = ((1+10)11 +19)(mod 11) = 7.

1. Выбрано и зафиксировано число k=10 для шифрования алгоритмом Цезаря, нужно зашифровать строку, составленную из фамилии, имени и отчества, записанных кириллицей.

* Строка: Кириллов Даниил Андреевич;
* k=10 – зафиксированное число;
* m=33 – размерность алфавита ;
* Зашифрованная строка: Фтътххшл Нйчттх Йчнъоолтб.

1. Вычислить число A= (Nгр.\*(8+Nсп.(mod 7)))²=(1\*(8+10(mod7)))²=121.

Рассчитать число В=ЧЧММГГГГ=16122005, где ЧЧ, ММ, ГГГГ – число, месяц и год рождения.

При нахождении наибольшего общего делителя использован метод Евклида.

* НОД (A, B(mod95)+900) = НОД(121,980) = 1

980 mod 121 = 12

121 mod 12 = 1

12 mod 1 = 0

* НОД = (A, ((B + 50) % 97) + 700) = НОД(121,790) = 1

790 mod 121 = 64

121 mod 64 = 57

64 mod 57 = 7

57 mod 7 = 1

1. mod 1 = 0

* НОД = (A, ((B + 20) % 101) + 1500) = НОД(121, 1543) = 1

1543 mod 121 = 91

121 mod 91 = 30

91 mod 30 = 1

30 mod 1 = 0

* НОД=(A,((B-40)%103) + 2500) = НОД(121, 2596) = 11

2596 mod 121 = 55

121 mod 55 = 11

55 mod 11 = 0

1. Нужно выбрать составное число N, N>10000 и методом Миллера доказать, что число составное:

* Пусть N= 16809

N-1=

16809=

Следовательно, s=3, t=2101.

* + Пусть a=13

Тогда N делится на а (16809/13 =1293) Следовательно, первое условие выполняется, N=16809 – составное.

Нужно выбрать простое число , 100<<1000 и методом Миллера доказать, что оно простое:

* Пусть =109

-1=

108=

Следовательно, s=2, t=25.

* Пусть а=21

1. (109) не делится на а (21) – условие выполняется
2. at = 1 (mod ), или существует целое k: 0≤k<s, такое, что

– условие выполняется

* Пусть а = 9

1. (109) не делится на а (9) – условие выполняется
2. at = 1 (mod N), или существует целое k: 0≤k<s, такое, что

– условие выполняется

* Пусть а=15

1. (109) не делится на а (15) – условие выполняется
2. at = 1 (mod N), или существует целое k: 0≤k<s, такое, что

– условие выполняется

Вывод: Три числа «а» уже являются нехорошими для числа 109, следовательно:

* k=3;
* Вероятность того, что N является составным, не превосходит ;
* 101 – простое число, т.к. вероятность обратного не более .

1. Изучение генерации ключей в алгоритме RSA:

* Выбор двух любых больших простых числа p=1553 q=6947;
* НОД(1553, 6947)=1;
* n=1553\*6947=10788691;
* φ(n)=(p−1)(q−1)=1552∗6946=10780192;
* Пусть е =7. НОД(7,1552) =НОД(7,6946) =1;
* Найдем d, такое, что e\*d=1(mod 𝜑(𝑛)) :

7\*d=1(mod 10780192)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Шаг | большее | меньшее | частное | остаток |
| 1 | 10780192 | 7 | 1540027 | 3 |
| 2 | 7 | 3 | 2 | 1 |
| 3 | 3 | 1 | 3 | 0 |
| 4 | 1 | 0 | - | - |

* d=3080055;
* Проверим: (7\*3080055) mod 10780192=1;
* Открытый ключ: (e, n) = (7, 10780192) ;
* Секретный ключ: (d, n) = (3080055, 10780192).

* Выбор произвольного текста x=study;

p=1553, q=6947, n=10780192 , φ(n) = 1078019, e=7, d=3080055;

Вычисление xe mod n:

|  |  |  |
| --- | --- | --- |
| X | ASCII -кодировка | y |
| S | 115 | 3777854 |
| T | 116 | 10692135 |
| U | 117 | 3583045 |
| D | 100 | 725185 |
| Y | 121 | 4505852 |

* Вычисление yd mod n:

|  |  |
| --- | --- |
| Y | X |
| 3777854 | 115 |
| 10692135 | 116 |
| 3583045 | 117 |
| 725185 | 100 |
| 4505852 | 121 |

1. Подписать текст x цифровой подписью s:

* p=1553, q=6947, n=10780192 , e=7, d=3080055, φ(n) = 1078019;
* x=study;
* s=xd mod n

|  |  |  |
| --- | --- | --- |
| X | ASCII -кодировка | S |
| S | 115 | 9430414 |
| T | 116 | 1176963 |
| U | 117 | 3823512 |
| D | 100 | 5836347 |
| Y | 121 | 3817425 |

* Проверить подпись, вычислив обратное преобразование

x=se mod n:

|  |  |
| --- | --- |
| S | x |
| 9430414 | 115 |
| 1176963 | 116 |
| 3823512 | 117 |
| 5836347 | 100 |
| 3817425 | 121 |

* Вывод: Полученный код совпадает с исходным.

1. Результаты моделирования процесса установления сеансового ключа:

* Выбор любого числа a=583;
* Моделирование действий пользователя А: выбор случайным  образом x=1962;
* Нахождение числа A=ax mod n=5831962 mod 10788691=3888432;
* Моделирование действий пользователя В: выбор случайным  образом y=5786, y < φ(n)
* Нахождение числа  B=ay mod n=5835786 mod 10788691=575972;
* Вычисление пользователем А Bx mod n**=**5759721962 mod 10788691=9311245;
* Вычисление пользователем В Ay mod n= 38884325786 mod 10788691=9311245;
* Полученные значения образуют сеансовый ключ;
* Проверка равенства полученных ключей пользователей: (ax)y=(ay)x= axy mod n = 9311245.

1. Разработанная утилита шифрования и дешифрования с помощью алгоритма Меркля-Хелмана находится в приложении А.
2. **Ответы на контрольные вопросы**
3. Вычетом называется остаток деления, то есть r – вычет числа a по модулю b, если разность между a и r делится на b без остатка.

Алгоритм Цезаря основан на замене каждой буквы шифруемой строки на другую букву, которая находится в алфавите на фиксированном сдвиги.

1. Число Кармайкла – это составное число n, которое удовлетворяет сравнению для всех целых b, взаимно простых с n, другими словами – псевдопростое число по каждому основанию b, взаимно простому с n. Такие числа относительно редки, но их бесконечное число, наименьшее из них – 561.
2. Основные свойства мультипликативной группы вычетов по  модулю pq:

* Мультипликативная группа (Z/nZ)\* этого кольца является абелевой и циклической;
* Эта группа состоит из ненулевых чисел, меньших n и взаимно простых с n;
* Остатки от деления образующей группы (Z/nZ)\* на p и q равны соответственно образующим мультипликативных групп полей Fp\* и Fq\*;
* Любой элемент кольца a∈(Z/nZ) может быть единственным образом представлен в виде a(mod p) и a(mod q);
* Если a(mod p) = 0, то a принадлежит идеалу (p) и не является элементом группы (Z/nZ)\*, при этом a образует в (Z/nZ) мультипликативную группу, изоморфную Pq\*;
* Единичным элементом в этой группе является элемент, сравнимый с 1 по модулю q и сравнимый с 0 по модулю p.

1. Порядок группы — её мощность, то есть количество элементов. Порядок группы равен значению функции Эйлера от n: #(Z/nZ)\*=φ(p)\*φ(q)=(p-1)(q-1).

Так как группа состоит из ненулевых чисел, взаимно простых и меньших n, n=pq, то взаимно простых с n чисел — (p-1)(q-1), так как из натуральных чисел меньше n исключаются все, кратные p и q.

1. Алгоритм расчета кодов символов при декодировании шифрограмм согласно алгоритму Меркля-Хеллмана: Зная открытый и закрытый ключ и текст, который нужно декодировать, необходимо найти число , такое что . Для вычисления обратных чисел по модулю применяется алгоритм Евклида. После вычисления обратного по модулю числа каждое значение шифрограммы умножается на это число и по модулю m (данное число больше, чем сумма чисел открытого числа) и с помощью закрытого ключа определяют биты открытого текста.
2. **Выводы**

В ходе работы были приобретены расчетные навыки в модульной арифметике, используемой в криптографических алгоритмах, и произведено ознакомление с математическими вычислениями, используемыми для сокрытия сообщений на примерах алгоритма шифрования RSA и ранцевой криптосистемы Меркля-Хеллмана.

ПРИЛОЖЕНИЕ А

Листинг программы «Шифрование и дешифрование с помощью алгоритма Меркля-Хеллмана»

import random

from math import gcd

def modgsd(a, m):

    x0, x1, m0 = 0, 1, m

    while a > 1:

        q = a // m

        a, m = m, a % m

        x0, x1 = x1 - q \* x0, x0

    if a != 1:

        raise ValueError("Обратного элемента не существует")

    return x1 % m0

def gsp(n):

    w = []

    max = 0

    for \_ in range(n):

        next\_val = max + random.randint(1, 10)

        w.append(next\_val)

        max += next\_val

    return w

def generate\_keys(numb=8):

    w = gsp(numb)

    m = sum(w) + random.randint(1,100)

    while True:

        n = random.randint(2, m - 1)

        if gcd(n, m) == 1:

            break

    b = [(n \* wi) % m for wi in w]

    return {

        'private': {'w': w, 'm': m, 'n': n},

        'public': b

    }

def encrypt(message, public\_key):

    bits = []

    for char in message:

        bin\_str = format(ord(char), '08b')

        bits\_char = []

        bits\_char.extend([int(bit) for bit in bin\_str])

        bits.append(bits\_char)

    enctext = []

    for bit in bits:

        c = sum([bit \* b for bit, b in zip(bit, public\_key)])

        enctext.append(c)

    return enctext

def decrypt(enctext, private\_key):

    w, m, n = private\_key['w'], private\_key['m'], private\_key['n']

    n\_inv = modgsd(n, m)

    text\_bits = []

    #задача ранца

    for c in enctext:

        c\_prime = (c \* n\_inv) % m

        bits = []

        for wi in reversed(w):

            if wi <= c\_prime:

                bits.insert(0, 1)

                c\_prime -= wi

            else:

                bits.insert(0, 0)

        text\_bits.extend(bits)

    chars = []

    for i in range(0, len(text\_bits), 8):

        byte = text\_bits[i:i+8]

        value = int(''.join(map(str, byte)), 2)

        chars.append(chr(value))

    return ''.join(chars)

# --- Главная программа с вводом ---

if \_\_name\_\_ == "\_\_main\_\_":

    message = input("Введите сообщение для шифрования: ")

    keys = generate\_keys()

    public\_key = keys['public']

    private\_key = keys['private']

    print("Публичный ключ:", public\_key)

    print("Приватный ключ:", private\_key)

    cipher = encrypt(message, public\_key)

    print("\nЗашифрованное сообщение:", cipher)

    decrypted = decrypt(cipher, private\_key)

    print("\nРасшифрованное сообщение:", decrypted)