1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. [—](https://dl.spbstu.ru/)
4. [Институт компьютерных наук и кибербезопасности](https://dl.spbstu.ru/)

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «**Математические примитивы криптографии**»
2. по дисциплине «Основы информационной безопасности»
3. Выполнили

студенты гр. 5131001/30003 Киселев Е.Д.

<*подпись*>

Губадлы Б.Н.

<*подпись*>

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

<*подпись*>

1. Санкт-Петербург
2. 2024
3. **Цель работы**

Приобретение расчётных навыков в модульной арифметике, 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**
2. Изучить модульную арифметику.
3. Изучить принцип работы шифра Цезаря.
4. Изучить принцип работы алгоритма Евклида.
5. Изучить метод Миллера для простых и составных чисел.
6. Изучить метод генерации ключей.
7. Изучить принцип работы алгоритма шифрования RSA.
8. Научиться работать с цифровой подписью.
9. Разработать утилиту шифрования и дешифрования с помощью алгоритма Меркля-Хеллмана.
10. **Ход работы**

**3.1 Модульная арифметика**

Nгр=5131001; Nсп=10; Ф3=19;

((Nгр.+Nсп.)11+Ф3)(mod 11)=((5131001+10)11+19)(mod 11)=7

**3.2 Шифр Цезаря**

Возьмём число *k*=5 в функции шифрования алгоритма Цезаря. Необходимо определить *m.* В русском алфавите 33 буквы, учитывая наличие пробела при записи ФИО, считаем *m*=33+1=34.

Киселев Евгений Денисович

Пример расчёта:

К=(12+5)(mod34)=17=П

И=(10+5)(mod 34)=15=Н

С=(19+5)(mod 34)=24=Ц

Е=(6+5)(mod 34)=11=Й

Л=(13+5)(mod 34)=18=Р

Ё=(7+5)(mod 34)=12=К

В=(3+5)(mod 34)=8=Ж

Г=(4+5)(mod 34)=9=З

Н=(15+5)(mod 34)=20=Т

Й=(11+5)(mod 34)=16=О

Д=(5+5)(mod 34)=10=И

О=(16+5)(mod 34)=21=У

Ч=(25+5)(mod 34)=30=Ь

Пробел=(34+5)(mod 34)=5=Д

Ответ: киселев евгений денисович = пнцйрйждйжзйтнодийтнцужнь

**3.3 Алгоритм Евклида**

A=(Nгр.(8+Nсп.(mod 7)))2=(5131001(8+10(mod 7)))2=3188390012444121

B=ЧЧММГГГГ=12022005

**3.3.1 НОД(А, В(mod 95)+900)**

В(mod 95)+900 = 12022005(mod 95)+900 = 940

НОД(3188390012444121,940)

А=3188390012444121 mod 940 = 12443121

НОД(12443121, 940)

B = 12443121 mod 940 = 181

НОД(940, 181)

A = 940 mod 181 = 35

НОД(181, 35)

B = 181 mod 35 = 6

НОД(35, 6)

A = 35 mod 6 = 5

НОД(6, 5)

B = 6 mod 5 = 1

НОД(5, 1)

A = 5 mod 1 = 0

Ответ: НОД(3188390012444121, 940) = 1

**3.3.2 НОД(А, (В+50)(mod 97)+700)**

(В+50)(mod 97)+700 = (12022005(+50)(mod 97)+700 = 779

НОД(3188390012444121, 779)

A = 3188390012444121 mod 779 = 300962

НОД(300962, 779)

B = 300962 mod 779 = 168

НОД(779, 168)

A = 779 mod 168 = 107

НОД(168, 107)

B = 168 mod 107 = 61

НОД(107, 61)

A = 107 mod 61 = 46

НОД(61, 46)

B = 61 mod 46 = 15

НОД(46, 15)

A = 46 mod 15 = 1

НОД(15, 1)

B = 15 mod 1 = 0

Ответ: НОД(3188390012444121, 779) = 1

**3.3.3 НОД(А, (В+20)(mod 101)+1500, (В-40)(mod 103)+2500)**

(В+20)(mod 101)+1500 = (12022005+20)(mod 101)+1500 = 1528

(B-40)(mod 103)+2500 = (12022005-40)(mod 103)+2500 = 2569

НОД(3188390012444121, 1528)

A = 3188390012444121 mod 1528 = 2585

НОД(2585, 1528)

B = 2585 mod 1528 = 1057

НОД(1528, 1057)

A = 1528 mod 1057 = 471

НОД(1057, 471)

B = 1057 mod 471 = 115

НОД(471, 115)

А = 471 mod 115 = 11

НОД(115, 11)

B = 115 mod 11 = 5

НОД(11, 5)

A = 11 mod 5 = 1

НОД(5, 1)

B = 5 mod 1 = 0

Пусть Р=НОД(А, (В+20)(mod 101)+1500)=1

НОД(P, (B-40)(mod 103)+2500) = НОД(1, 2569)=2569 mod 1 = 0

Ответ: НОД(3188390012444121, 1528, 2569) = 1

**3.4 Метод Миллера**

Выберем любое число *N* : *N* > 10000, например, *N* = 23762. Необходимо методом Миллера доказать, что оно составное. Так же необходимо выбрать простое число *N1*: 100 < *N1*< 1000, например *N1*= 181. Методом Миллера докажем, что оно простое.

**3.4.1 Составное число**

*N* = 23762

Пусть число *a* : 1 < *a* < *N,* например *a = 109.*

Пусть *Р*-остаток от деления числа *N* на *а*, тогда *Р* = *N* mod *а* = 23762 mod 109 = 0, следовательно число *N* делится на число *а* без остатка, следовательно число *N*-составное.

**3.4.2 Простое число**

*N1*=181

*N1*-1=2*s*\**t* => 180=22\*45 => s=2, t=45.

Возьмём *a* равное 109, 23, 3 и проверим выполнение следующих условий:

1) *N1* mod *a* ≠ 0; 2) *at* ≡1(mod *N1*)

Для a = 109:

1) 181 mod 109 = 72

2) 10945≡1(mod 181)

Оба условия выполняются

Для a = 23:

1) 181 mod 23 = 20

2) 2345≡1(mod 181)

Оба условия выполняются

Для a = 3:

1) 181 mod 3 = 1

2)345≡1(mod 181)

Оба условия выполняются

Для числа 181 существует как минимум три “нехороших” числа, вероятность того что число *N1* составное <= 4-3. Следовательно, вероятно, что число *N1*-простое.

**3.5 Генерация ключей и RSA**

Возьмём два простых числа *p* и *q*: 102 < (*p*, *q*) < 106; НОД(*p*, *q*) = 1.

Пусть *p* = 2357, *q* = 1549. Проверим, подходят ли числа под условия:

102 < 2357 < 106

102 < 1549 < 106

НОД(2357, 1549) ≡ 1

Данные числа полностью удовлетворяют условиям.

Вычислим число *n* = *p*\**q* = 2357 \* 1549 = 3653393.

Вычислим порядок группы *φ*(*n*) = (*p*-1)\*(*q*-1) = 2356 \* 1548 = 3646488.

Выберем показатель шифрования *e*, чтобы НОД(*e*, *p*-1) = НОД(*e*, *q*-1) = 1:

Пусть *e* = 11, тогда НОД(11, 2356) ≡ НОД(11, 1548) ≡ 1.

Необходимо найти НОД двух заданных чисел *s* и *t* (*s* > *t*) с помощью алгоритма Евклида:

НОД(*s*, *t*) = [*s* = 3646488; *t* = 11] = НОД(3646488, 11)

*s* = *s0*= *Q1t0* + *t1*= 331498 \* 11 + 10

*t* = *t0* = *Q2t1* + *t2* = 1 \* 10 + 1

*t1* = *Q3t2* + *t3* = 1\*1 + 0

*t3* = 0

Используя матричный способ вычислим показатель d:

Число *d*, обратное по модулю *m* числу *e*. Известна формула: *ed* = 1(mod *φ*(*n*)) ⇒ 11\**d*=1(mod 3646488) ⇒ *d* = 332499

Открытый ключ: (e, n) = (11, 3653393)

Закрытый ключ: (d, n) = (332499, 3653393)

**3.6 Шифрование и дешифрование RSA**

Выберем произвольный текст *x* с использованием ASCII-кодировки. Пусть *x* = light. Необходимо зашифровать *x* по формуле *x*e(mod *n*) и расшифровать по формуле *y*d(mod *n*)*.*

Вычислим *xe*(mod *n*), получив зашифрованный текст *y:*

*e =* 11; *d =* 332499; *n* = 3653393 (данные значения были найдены ранее)

Зашифруем буквы:

*l* = 108 = 10811(mod 3653393) = 1749945968

*i* = 105 = 10511(mod 3653393) = 316882025

*g* = 103 = 10311(mod 3653393) = 351356117

*h* = 104 = 10411(mod 3653393) = 226463024

*t* = 116 = 11611(mod 3653393) = 287118576

Расшифруем буквы:

1749945968 = 1749945968332499(mod 3653393) = 108 = *l*

316882025 = 316882025332499(mod 3653393) = 105 = *i*

351356117 = 351356117332499(mod 3653393) = 103 = *g*

226463024 = 226463024332499(mod 3653393) = 104 = *h*

287118576 = 287118576332499(mod 3653393) = 116 = *t*

Расшифрованный текст полностью совпадает с исходным.

**3.7 Использование цифровой подписи**

Необходимо подписать текст *x* цифровой подписью *s*, где *s* = *xd*(mod *n*).

Пусть *x* = “*light*” = *l*(108) *i*(105) *g(*103) *h*(104) *t*(116); *d* = 332499; *e* = 11; *n* = 3653393 (данные значения были найдены ранее).

Подпись текста цифровой подписью:

*s* = *xd*(mod *n*) = 108332499(mod 3653393) = 3153256

*s* = *xd*(mod *n*) = 105332499(mod 3653393) = 2767900

*s* = *xd*(mod *n*) = 103332499(mod 3653393) = 1143499

*s* = *xd*(mod *n*) = 104332499(mod 3653393) = 1260280

*s* = *xd*(mod *n*) = 116332499(mod 3653393) = 1520576

Проверка подписи:

*x* = *se*(mod *n*) = 315325611(mod 3653393) = 108

*x* = *se*(mod *n*) = 276790011(mod 3653393) = 105

*x* = *se*(mod *n*) = 114349911(mod 3653393) = 103

*x* = *se*(mod *n*) = 126028011(mod 3653393) = 104

*x* = *se*(mod *n*) = 152057611(mod 3653393) = 116

Имеем подписанный текст, представляющий пару (*x*, *s*) для каждого символа.

Так как, после всех преобразований, числовое значение *х* не изменилось, следовательно, преобразования выполнены верно.

**3.8 Моделирование процесса установления сеансового ключа**

Воспользуемся алгоритмом Диффи-Хеллмана для установления сеансового ключа в кольце вычетов по модулю составного ключа.

*n* = 3653393 (данное значение было найдено ранее).

Пусть *а* = 121.

Действия пользователя А: *а* = 121, *х* = 9 (*x* < *φ*(*n*)), тогда А = *ах*(mod *n*) = 1219(mod 3653393) = 321215093.

Действия пользователя В: *у*=15 (*y* < *φ*(*n*)), тогда В = *ау*(mod *n*) = 12115(mod 3653393) = 1200882.

Установка сеансового ключа, действия пользователя А: В*х*(mod *n*) = 12008829(mod 3653393) = 2909958

Установка сеансового ключа, действия пользователя В: А*у*(mod *n*) = 321215093 15(mod 3653393) = 2909958.

Проверка: A*xy*(mod *n*)=321215093135(mod 3653393) = 2909958.

Сеансовые ключи пользователей равны.

1. **Ответы на контрольные вопросы**

1) Вычет - это остаток от деления двух натуральных чисел друг на друга.

Алгоритм шифрования Цезаря основан на применении модульной арифметики, а точнее «сдвиге», шифруемого элемента на некоторое количество алфавитных позиций вперёд/назад.

2) Числа Кармайкла ‒ это составные числа, например составное число *N*, которые обладают свойством *aN-1* = 1(mod *N*), где *а* ‒ целое, с условием НОД(*a*, *N*) = 1.

3)Мультипликативной группа кольца вычета по модулю *p \* q* является абелевой, цикличной и состоит из ненулевых чисел, меньших *n*, и взаимно простых с *n*. Остатки от деления образующей группы на *p* и *q* равны соответственно образующим мультипликативных групп.

4)Порядок группы (*Z\nZ*)\* должен иметь большой простой делитель, так как, кольцо вычетов этой группы *Z\nZ* содержит число *n*-произведение двух больших простых чисел.

5)При декодировании программ согласно алгоритму Меркля-Хеллмана, необходимо найти число *n*-1, такое что *n \* n*-1(mod *m*) = 1. Для вычисления же обратных по модулю чисел применяется алгоритм Евклида. После определения обратного числа, каждое значение шифрограммы умножается на *n*-1 по модулю *m* и, с помощью закрытого ключа, определяются биты открытого текста.

**5 Вывод**

В данной лабораторной работе были получены навыки работы в модульной арифметике, используемой в криптографических алгоритмах, разобраны принципы работы таких пунктов, как: алгоритм RSA, шифр Цезаря, алгоритм Евклида и ранцевая крипто система Меркля-Хеллмана.

ПРИЛОЖЕНИЕ А

Листинг программы «Утилита шифрования и дешифрования с помощью алгоритма Меркля-Хеллмана»

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

int len\_of\_bits;

int bin2dec(int n)

{

int num = n;

int dec\_value = 0;

int base = 1;

int temp = num;

while (temp) {

int last\_digit = temp % 10;

temp = temp / 10;

dec\_value += last\_digit \* base;

base = base \* 2;

}

return dec\_value;

}

int dec2bin(int num)

{

int bin = 0, k = 1;

if (num >= 0) {

while (num)

{

bin += (num % 2) \* k;

k \*= 10;

num /= 2;

}

return bin;

}

}

int encrypt(char text, int open\_key[10]) {

int letters = dec2bin(int(text));

char string\_of\_bins[100];

string\_of\_bins[0] = '\0';

\_itoa\_s(letters, string\_of\_bins, 10);

int message = 0;

len\_of\_bits = strlen(string\_of\_bins);

int j = 0;

for (int i = 0; i < len\_of\_bits; i++) {

if (string\_of\_bins[i] == '1') {

message += open\_key[j];

if (j == 9) {

j = 0;

}

else {

j++;

}

}

else {

if (j == 9) {

j = 0;

}

else {

j++;

}

}

}

return message;

}

char decrypt(int message, int close\_key[10], int r, int q) {

int n\_1;

for (n\_1 = 0; n\_1 < 1000; n\_1++) {

if (((n\_1 \* r) % q) == 1) {

break;

}

}

int\* almost\_done = (int\*)malloc(sizeof(int) \* len\_of\_bits);

for (int i = 0; i < len\_of\_bits; i++) {

almost\_done[i] = 0;

}

int decoded\_message = (message \* n\_1) % q;

int curr\_max = 0;

int index = 0;

while (decoded\_message > 0) {

for (int i = 0; i < len\_of\_bits; i++) {

if (close\_key[i] <= decoded\_message && close\_key[i] > curr\_max) {

curr\_max = close\_key[i];

index = i;

}

}

decoded\_message -= curr\_max;

curr\_max = 0;

almost\_done[index] = 1;

}

char\* sdecoded\_message = (char\*)malloc(sizeof(char) \* len\_of\_bits);

sdecoded\_message[0] = '\0';

for (int i = 0; i < len\_of\_bits; i++) {

char buffer[10];

\_itoa\_s(almost\_done[i], buffer, 10);

strcat(sdecoded\_message, buffer);

}

int dec\_decoded\_message = 0;

dec\_decoded\_message = bin2dec(atoi(sdecoded\_message));

char final\_decoded\_message = char(dec\_decoded\_message);

return char(final\_decoded\_message);

}

void open\_key\_generation(int close\_key[10], int q, int r, int open\_key[10]) {

int tmp = 0;

for (int i = 0; i < 10; i++) {

tmp = (r \* close\_key[i]) % q;

open\_key[i] = tmp;

}

}

int main(int argc, const char\* argv[]) {

int open\_key[10] = {};

int close\_key[10] = { 1, 2, 7, 13, 67, 91, 182, 370, 734, 1468 }; // Самостоятельно задаем закрытый ключ

int sum\_key = 2935; // Сумма элементов закрытого ключа

int q = 3329; // Произвольное простое число большее суммы

int r = 4139; // Произвольное простое число, взаимно простое с q

char text[100]; // Сообщение для шифровки

printf("Enter text to decode: ");

gets\_s(text);

open\_key\_generation(close\_key, q, r, open\_key); // Генерируем открытый ключ

printf("Encrypted message: ");

for (int i = 0; i < strlen(text); i++) {

int message = encrypt(text[i], open\_key);

printf("%d ", message);

}

printf("\n");

printf("Decrypted message: ");

for (int i = 0; i < strlen(text); i++) {

int message = encrypt(text[i], open\_key);

printf("%c", decrypt(message, close\_key, r, q));

}

printf("\n");

return 0;

}