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

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. **Математические основы криптографии**
2. по дисциплине «Основы информационной безопасности»
3. Выполнил
4. студент гр. 4851003/20001 Федорова А.А.

<*подпись*>

1. Преподаватель
2. асс. преподавателя Климшин И.И.

<*подпись*>

1. **Цель работы**

Ознакомиться с математическими основами криптографии, разработать утилиту шифрования и дешифрования на основе ранцевой криптосистемы Меркля-Хеллмана.

1. **Описание задачи**

* изучение и реализация шифра Цезаря;
* изучение и реализация алгоритма Евклида для определения НОД;
* изучение и реализация алгоритма Миллера для определения простоты числа;
* изучение и реализация алгоритма шифрования RSA;
* моделирование цифровой подписи;
* моделирование сеансового ключа;
* изучение и реализация алгоритма шифрования Меркля-Хеллмана.

1. **Алгоритмы и методы**

## Шифр Цезаря

Суть шифра Цезаря в том, что каждая буква слова сдвигается на k шагов вперёд.

## Алгоритм Евклида

Алгоритм Евклида – это нахождение НОД путём вычитания в паре чисел из большего числа меньшее. Если на очередном шаге числа станут равны, значит, мы нашли их НОД.

## 3) Алгоритм Миллера

Необходимо выяснить, является ли число простым или составным, исходя из правила:

Пусть {\displaystyle n} N – нечетное составное число и {\displaystyle n-1=2^{s}d} N – 1 = 2s \*t, где {\displaystyle d}t — нечётно. Тогда для {\displaystyle a}a, 1 < a < N, {\displaystyle \mathbb {Z} \_{n}}выполняется хотя бы одно из условий:

1. N делится на а
2. {\displaystyle a^{d}\equiv 1{\pmod {n}}}at = 1 (mod N)
3. ∃ целое число {\displaystyle r<s}0 ≤ k < s такое, что а{\displaystyle a^{2^{r}d}\equiv -1{\pmod {n}}}(2^(k-1))\*t = – 1 (mod N)

Если N – простое, то ни одно из трёх условий не выполняется

Можно построить вероятностный алгоритм:

1. а выбирается случайным образом, 1 < a < m. Для него проверяются указанные выше свойства.

2. Если хотя бы одно из свойств нарушается, то число m – составное.

3. Если выполнены оба условия, возвращаемся к шагу 1.

4. Составное число не будет определено как составное после однократного

выполнения шагов 1-3 с вероятностью не большей 1/4. А вероятность не

определить его после k повторений не превосходит 1/4k.

## Алгоритм шифрования RSA

Алгоритм шифрования RSA – криптосистема с открытым ключом. Ключ содержит две составляющие: открытую, которую можно публиковать, и секретную. Открытый ключ содержит число n и показатель шифрования е. Секретный ключ содержит показатель дешифрования d.

Показатели шифрования и дешифрования связаны между собой равенством .

## Шифрование и дешифрование текста

Любой блок текста х, представленный числом, меньшим n и взаимно простым с n, может быть зашифрован любым пользователем с помощью открытого ключа: .

Дешифрование криптограммы y осуществляется только владельцем секретного ключа d: .

## Цифровая подпись

Цифровая подпись выполняется аналогично шифрованию и дешифрованию. Ключ создания подписи является конфиденциальной информацией данного пользователя и содержит показатель d. Соответствующий ему ключ проверки подписи содержит число n и показатель e. Этот ключ является общедоступным.

Для вычисления подписи s для текста х, представленного числом, меньшим n и взаимно простым с n, подписывающий возводит его в степень d: .

Подписанный текст представляет собой пару (x, s).

Для проверки подписи проверяющий вычисляет обратное преобразование и проверяет равенство: .

## Моделирование сеансового ключа

Для моделирования сеансового ключа пользователи договариваются об общем числе n и числе а (меньше n), которое выбирается случайным образом. Далее пользователь А вырабатывает случайное число х < ϕ(n), пользователь Б вырабатывает случайное число у < ϕ(n). Пользователь А вычисляет ах (mod n), а пользователь Б вычисляет ау (mod n), после чего они посылают друг другу результаты. Далее каждый пользователь возводит полученное от другого число в свою степень. Таким образом, имеем (ах)у = (ау)х = аху (mod n). Данное число и будет сеансовым ключом. Если сеансовый ключ у обоих пользователей совпал, значит, он корректен.

1. **Ход работы.**

## Расчёт по формуле ((Nгр + Nсп)11 + Ф3)(mod 11)

Nгр = 1;

Nсп = 25;

Ф3 = 5;

((Nгр + Nсп)11 + Ф3)(mod 11) = ((1 + 25)11 + 5)(mod 11) = 9

## Шифр Цезаря

Выберем случайное число k = 5.

Код см. в приложении

Исходная строка: Фёдорова Алёна Александровна

Зашифрованная строка: ЩЅйухузе%ЕрЅте%Еркпцетйхузте

## Алгоритм Евклида

A = (Nгр \* (8 + Nсп (mod 7))2 = (1\*(8+25(mod 7))2 =144

B = 22072004;

НОД(A, (B mod 95) + 900) = НОД(144, 984) = 24

(первое число = 144; второе число = 984)

НОД (A, ((B + 50) mod 97) + 700 )= 72

(первое число =144; второе число = 792)

НОД (A, ((B + 20) mod 101) + 1500, ((B - 40) mod 103) + 2500) = 2

(первое число = 144; второе число = 1590; третье число = 2594)

## Алгоритм Миллера

Для примера рассмотрим составное число N:

N = 25001; (23\*1087), a = 5469, s = 3; t = 3125; N / a ≠0 − условие не нарушается:

25001 / 5469= 4,57…;

at mod N = 1 − условие нарушается:

54693125 mod 25001 = 20258.

С первым же а нам удалось доказать, что число N составное.

Теперь рассмотрим простое число N1:

N1 = 257, a1 = 133; s = 8; t = 1; N / a1 ≠0 − условие не нарушается:

257 / 133= 1,93…;

a1t mod N = 1 − условие не нарушается:

1331 mod 257 = 133;

∃ такое целое число {\displaystyle r<s}0 ≤ k < s, для которого бы выполнялось a2^kt = -1 (mod n) − условие не нарушается:

k = 6; 1336 (mod 257) = 256 = N – 1.

Поскольку вероятность выяснить с первого раза, является ли число составным, составляет 25%, то можем провести ещё 3 теста, чтобы убедится, что число точно составное:

a2 = 65; N / a2 ≠0 − условие не нарушается:

257 / 133= 3,95…; a2 t mod N ≠ 1 − условие не нарушается: 651 mod 257 = 65;

∃ такое целое число {\displaystyle r<s}0 ≤ k < s, для которого бы выполнялось a2^kt = -1 (mod n) − условие не нарушается:

k = 7; 657 (mod 257) = 256 = N – 1. a3 = 201;

N / a3 ≠0 − условие не нарушается:

257 / 201= 1,27…;

a3 t mod N ≠ 1 − условие не нарушается:

2011 mod 257 = 201;

∃ такое целое число {\displaystyle r<s}0 ≤ k < s, для которого бы выполнялось a2^kt = -1 (mod n) − условие не нарушается:

k = 7; 2017 (mod 257) = 256 = N – 1.

a4 = 96;

N / a4 ≠0 − условие не нарушается:

257 / 96= 2,67…;

a2 t mod N ≠ 1 − условие не нарушается:

961 mod 257 = 96;

∃ такое целое число {\displaystyle r<s}0 ≤ k < s, для которого бы выполнялось a2^kt = -1 (mod n) − условие не нарушается:

k = 7; 967 (mod 257) = 256 = N – 1.

Таким образом, доказали, что число 257 простое.

Из проведенных расчетов можно сделать вывод о том, что для проверки составного числа N достаточно сделать проверку одного условия на делимость исходного числа N на число a. При этой проверке мы берем a от 1 до √N – 1 (от 2 до √N с округлением до целой части включительно). Во время проверки простого числа N необходимо соблюсти все условия алгоритма. В первом условии нам необходимо проверить фиксировано от 1 до √N - 1 чисел a, чтобы не пропустить возможные делители числа N. Во втором условии сделать проверки нескольких значений в количестве s (от 0 до s – 1 включительно) для произвольных чисел a. Для того, чтобы с большей вероятностью можно было утверждать, что число N – простое число, нужно найти как можно больше чисел r для конкретных a. Число проверок может многократно увеличиваться в зависимости от того, насколько исходное простое число большое.

## Генерация ключей RSA

Сначала выберем два простых числа p и q: p = 103, q = 127.

Число n = p \* q: n = 103 \* 127 = 13081.

Порядок группы ϕ(n) = (p – 1)(q – 1): ϕ(n) = 102 \* 126 = 12852.

Показатель шифрования е должен быть простым числом, таким что НОД (e, p – 1) = НОД (е, q – 1) = 1, то есть эти пары чисел взаимно-простые. Возьмём e = 19.

Показатель дешифрования d, такой что е \* d = 1 (mod ϕ(n)) рассчитаем матричным способом. Рассмотрим где , e = 19

=> => =>

=> => =>

это матрица где

e и d – показатели шифрования и дешифрования

с – коэффициент уравнения d\*e - φ(n)\*c = 1

Получим d = 4735.

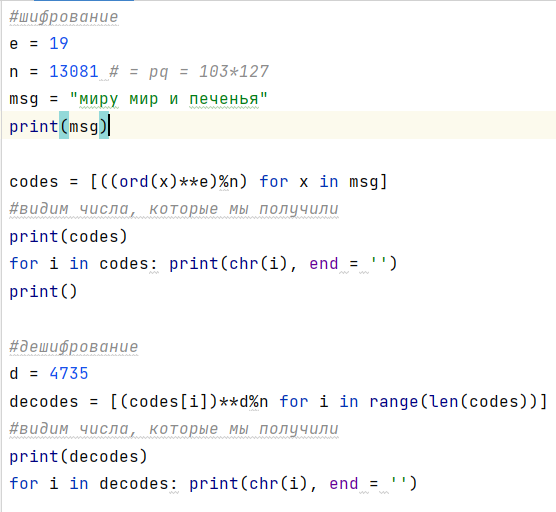
## Шифрование и дешифрование текста

Кодирование и декодирование происходит следующим образом. Чтобы закодировать текст х необходимо возвести в степень ключа шифрования.

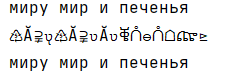
xe = y (mod n). Где x – код каждого из символов сообщения.

Для дешифрования аналогично текст у необходимо возвести в степень ключа дешифрования. yd = x (mod n).

В ходе выполнения данного задания возникла проблема. Те числа, которые получались после шифрования, не имеют символов в таблице ANSII. То есть зашифрованное сообщение нельзя было представить в символьном виде. Решено было проделать шифрование и дешифрование для кодировки Unicode, способ представления Unicode – UTF-8.



Закодированное сообщение получилось очень интересным.



Коды символов исходной строки: [1084, 1080, 1088, 1091, 32, 1084, 1080, 1088, 32, 1080, 32, 1087, 1077, 1095, 1077, 1085, 1100, 1103]

Коды символов зашифрованной строки: [9847, 258, 10956, 837, 651, 9847, 258, 10956, 651, 258, 651, 4692, 5197, 629, 5197, 9750, 4908, 8885]

Коды символов расшифрованной строки: [1084, 1080, 1088, 1091, 32, 1084, 1080, 1088, 32, 1080, 32, 1087, 1077, 1095, 1077, 1085, 1100, 1103]

* 1. **Цифровая подпись**

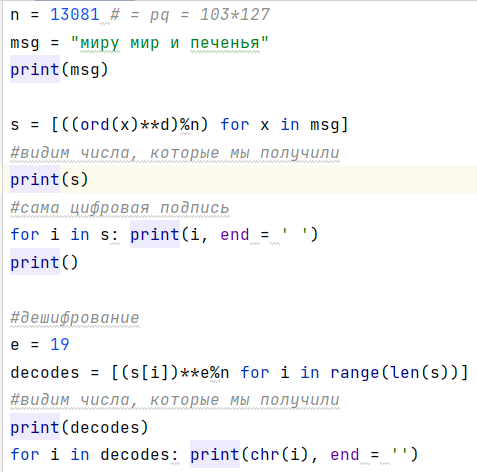
Цифровая подпись s выполняется аналогично алгоритму шифрования и дешифрования. Ключ подписи d является секретным, ключ проверки е является открытым.

s = xd (mod n).

Для проверки подписи выполняется обратное действие x = sе (mod n).

Если равенство выполняется, то подпись корректна.

Подпишем текст х из пункта 1.6.



Коды символов исходной строки: [1084, 1080, 1088, 1091, 32, 1084, 1080, 1088, 32, 1080, 32, 1087, 1077, 1095, 1077, 1085, 1100, 1103]

Коды символов, которые получились: [9847, 258, 10956, 837, 651, 9847, 258, 10956, 651, 258, 651, 4692, 5197, 629, 5197, 9750, 4908, 8885]

Тогда s = 12895 3826 7891 10997 1018 12895 3826 7891 1018 3826 1018 1808 11171 12508 11171 3219 3470 7968.

Теперь проверим подпись, зная формулу: x = se (mod n).

Коды символов расшифрованной строки: [1084, 1080, 1088, 1091, 32, 1084, 1080, 1088, 32, 1080, 32, 1087, 1077, 1095, 1077, 1085, 1100, 1103]

Расшифрованная строка совпадает с исходной. Подпись корректна.

## Моделирование сеансового ключа

Для моделирования сеансового ключа пользователи договариваются об общем числе n и числе а (меньше n), которое выбирается случайным образом. Пусть а = 257, n = 1563.

Далее пользователь А вырабатывает случайное число х < ϕ(n), пользователь Б вырабатывает случайное число у < ϕ(n). Пусть х = 19, у = 23.

Пользователь А вычисляет ах (mod n), а пользователь Б вычисляет

ау (mod n), после чего они посылают друг другу результаты.

Результат А: 905 – отправляет Б

Результат Б: 314 – отправляет А

Далее каждый пользователь возводит полученное от другого число в свою степень, делит по модулю.

Результат А: 31419 mod(n) = 518

Результат Б: 90523 mod(n) = 518

Таким образом, имеем axy (mod n) = ((ax (mod n))y mod n) = ((ay (mod n))x mod n),

т.е. axy (mod n) = Bx (mod n) = Ay (mod n).

Данное число axy (mod n) и будет сеансовым ключом. Для выбранных нами чисел сеансовый ключ = 25719\*23 = 518 (mod 1563). Сеансовый ключ у обоих пользователей совпал c рассчитанным, значит, он корректен.

* 1. **Утилита шифрования и дешифрования**

План действий для разработки утилиты

1. Создадим закрытый ключ – сверхвозрастающую последовательность а, то есть каждый её член больше суммы всех предыдущих.
2. Выберем модуль m, такой что он больше суммы элементов а.
3. Выберем множитель n, такой что m и n взаимно простые.
4. Создадим открытый ключ – это некоторая последовательность б, такая что б1 = а1 \* n (mod m)

Зашифруем сообщение.

1. Каждый элемент сообщения представим в двоичном виде
2. Получим шифр

Расшифруем сообщение

1. Найдём k: (n\*k % m = 1) – то есть k – обратное к n по модулю m
2. Расшифруем поэлементно сообщение
3. **Ответы на контрольные вопросы**
4. Что такое вычет? На чём основан алгоритм Цезаря?

Вычет – это остаток от деления одного числа на другое. Алгоритм Цезаря основан на замене текущего элемента на элемент с фиксированным числом позиций k в текущем алфавите.

1. Каковы особенности чисел Кармайкла?

Это составные (а не простые!) числа n, обладающие свойством:

an-1 = 1 (mod n) для любых целых а, не делящихся на n.

1. Перечислите основные свойства мультипликативной группы кольца вычетов по модулю pq. Такая группа является абелевой, циклической, состоит из ненулевых элементов, меньших n, и взаимно простых с ним.
2. Почему порядок группы (Z/nZ)\* должен иметь большой простой делитель?

Потому что он равен значению функции Эйлера ϕ(n) = (p – 1)(q – 1).

1. Опишите алгоритм расчёта кодов символов при декодировании шифрограмм согласно алгоритму Меркла-Хеллмана.

Получить рассчитанный вес элемента.

Получить число по формуле .

Получить бинарный код элемента, поочередно вычитая из полученного числа элементы закрытого ключа.

Повторять, пока не будут получены коды всех символов.

1. **Выводы**

В данной работе были изучены базовые принципы криптографии: был рассмотрены и опробованы на практике алгоритм шифрования Цезаря, алгоритм определения НОД Евклида, условия проверки числа на простоту, некоторые основы модульной арифметики, простейшие способы шифрования передаваемых сообщений, алгоритм шифрования RSA. Была разработана утилита шифрования и дешифрования на основе ранцевой криптосистемы Меркля-Хеллмана.

**ПРИЛОЖЕНИЕ 1**

Листинг программы «Шифр Цезаря»

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

//русские буквы кодируются отрицательными числами

int main() {

char a[30] = "Фёдорова Алёна Александровна";

for (int i = 0; i < 28; i++) {

a[i] = (char(int(a[i])+5));

printf("%c", a[i]);

}

}

**ПРИЛОЖЕНИЕ 2**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#include <math.h>

#define SIZE 100

#define SEQUENCE 8 //каждая буква кодируется 8 битами

//план действий для выполнения шиф и дешиф по алгоритму Меркля-Хеллмана

/\*

1 Создадим закрытый ключ – это некая сверхвозрастающая последовательность а, то есть каждый её член больше суммы всех предыдущих.

2 Выберем модуль m, такой что он больше суммы элементов а.

3 Выберем множитель n, такой что m и n взаимно простые.

4 Создадим открытый ключ – это некоторая последовательность б, такая что б1 = а1 \* n(mod m)

Зашифруем сообщение.

5 Каждый элемент сообщения представим в двоичном виде

6 Получим шифр

Расшифруем сообщение

7 Найдём k : (n \* k % m = 1) – то есть k – обратное к n по модулю m

8 Расшифруем поэлементно сообщение

\*/

//Функция генерирует закрытый ключ случайным образом и возвращает значение суммы элементов последовательности

int cr\_pr\_key(int\* priv\_key,int sum) {

// 30 - просто число

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

priv\_key[i] = rand() % 30 + (sum + 1);

sum += priv\_key[i];

}

return sum;

}

//Вычисление взаимно простого n для m путём перебора (перебор реализован ниже)

int rel\_prime(int n, int m) {

while (n != m)

{

if (n > m) n -= m;

else m -= n;

}

if (n == 1) return 1; // если нод == 1

else return 0;

}

//Генерация открытого ключа

int cr\_pub\_key(int\* priv\_key, int\* pub\_key, int n, int m) {

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

pub\_key[i] = (priv\_key[i] \* n) % m;

}

return 0;

}

//Поиск длины введённого сообщения

int len(char\* msg) {

int s = 0;

while (\*msg++) s++;

return s;

}

int main()

{

srand(time(NULL));

//1 step

int\* priv\_key = (int\*)calloc(SEQUENCE, sizeof(int));

int sum = cr\_pr\_key(priv\_key, 0);

printf("Your private key: ");

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

printf("%d ", priv\_key[i]);

}

printf("\n");

//2 step

int m = (sum + 1) + rand() % 30;

//3 step

int n = 2;

while (!rel\_prime(n, m)) n++;

//4 step

int\* pub\_key = (int\*)calloc(SEQUENCE, sizeof(int));

cr\_pub\_key(priv\_key, pub\_key, n, m);

printf("Your public key: ");

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

printf("%d ", pub\_key[i]);

}

printf("\n");

//5 step

char\* msg = (char\*)calloc(SIZE, sizeof(char));

printf("Enter your message: ");

gets(msg);

int l\_msg = len(msg);

int\* crypt = (int\*)calloc(l\_msg, sizeof(int));

for (int i = 0; i < l\_msg; i++) {

int t = msg[i];

int shifr = 0;

int j = 0;

while (t) {

shifr += (t % 2) \* pub\_key[SEQUENCE - 1 - j];

t /= 2;

j++;

}

crypt[i] = shifr;

}

//6 step

printf("Encrypted message: ");

for (int i = 0; i < l\_msg; i++) {

printf("%d ", crypt[i]);

}

//7 step

int k = 0, t = 0;

while (!k) {

if (n \* t % m == 1) k = t; t++;

}

//8 step

int\* decrypt = (int\*)calloc(l\_msg, sizeof(int));

for (int i = 0; i < l\_msg; i++) {

decrypt[i] = (crypt[i] \* k)%m;

//printf("%d ", decrypt[i]);

}

printf("\n");

for (int x = 0; x < l\_msg; x++) {

double s = 0;

for (int i = SEQUENCE - 1; i >= 0; i--) {

if (priv\_key[i] <= decrypt[x]) {

s += pow(2, SEQUENCE - i-1);

decrypt[x] -= priv\_key[i];

}

}

decrypt[x] = s;

}

printf("Decrypted message: ");

for (int i = 0; i < l\_msg; i++) {

printf("%c", decrypt[i]);

}

free(priv\_key);

free(pub\_key);

free(msg);

free(crypt);

free(decrypt);

}