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 примитивы криптографии»
2. по дисциплине «Основы информационной безопасности»
3. Выполнил
4. студент гр. 4851003/20001 Павлов И.А.

<*подпись*>

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

<*подпись*>

**Цель работы**

Приобретение расчетных 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.

**Описание задачи**

1. Приобрести расчётные навыки в модульной арифметике
2. Разработать утилиту шифрования Меркля-Хеллмана.

**Ход работы.**

Пункт 1.

Вычислим число по формуле:

Пункт 2.

Зашифруем строку, состоящую из фамилии, имени и отчества:

|  |  |  |
| --- | --- | --- |
| k | Исходная строка | Зашифрованная строка |
| 4 | ПавловИльяАлександрович | УдёптёМпагДпиохдсзфтёмы |

Пункт 3.

Вычислим число A по формуле:

Рассчитаем число B по формуле: ЧЧММГГГГ, где ЧЧ, ММ, ГГГГ – число, месяц и год рождения:

B = 30012004

НОД(A, B(mod 95) + 900) = НОД(, 979) = НОД(1506062726783597, 979) = … = НОД(895, 979) = НОД(895, 84) = … = НОД(55, 84) = НОД(55, 29) = НОД(26, 29) = НОД(26, 3) = … = НОД(2, 3) = НОД(2, 1) = НОД(1, 1) = НОД(0, 1) = 1

НОД(A, (B+50)(mod 97)+700) = НОД(, 760) = НОД(1506062726783816, 760) = … = НОД(216, 760) = … = НОД(216, 112) = НОД(104, 112) = НОД(104, 8) = … = НОД(0, 8) = 8

НОД(A, (B+20)(mod 101)+1500, (B–40)(mod 103)+2500) = НОД(, 1576, 2530) =…= НОД(1346, 1576, 2530) = НОД(1346, 1576, 954) = НОД(1346, 230, 954) = НОД(392, 230, 954) =…= НОД(392, 230, 170) = НОД(162, 230, 170) = НОД(162, 60, 170) = НОД(162, 60, 8) = …= НОД(42, 60, 8) = НОД(42, 18, 8) = …=НОД(6, 18, 8) = … = НОД(6, 2, 8) = … = НОД(0, 2, 0) = 2

Пункт 4.

Выберем составное число N при N > 10000: N = 10251. Выберем простое число N1 при 100<N1<1000: N = 727.

Докажем, что число N – составное методом Миллера:

10251 mod 2 = 1

10251 mod 3 = 0

10251 – составное число.

Докажем, что число N1 – простое методом Миллера:

727 mod 2 = 1

…

2813 mod 727 = 1 => 727 – простое число.

Пункт 5.

Изучим генерацию ключей в алгоритме RSA:

1. Выберем числа p и q: 199 и 211 соответственно.
2. Число n = pq = 199 \* 211 = 41989.
3. Вычислим порядок группы ф(n) = (p – 1)(q – 1) = 41580.
4. Выберем показатель e = 13.
5. Вычислим d = 6397.

Пункт 6.

Зашифруем и расшифруем текст по алгоритму RSA:

1. Выберем произвольный текст x = "RainingSummerDay".
2. Зашифруем текст x по формуле: y = xe(mod n) = 8626 38813 25080 39396 21683 28786 9614 12261 24208 18414 9261 25201 19955 12288.
3. Расшифруем текст y по формуле: x = yd(mod n) = "abcdefghijklmn".

Как можно заметить, тексты совпадают, значит алгоритм RSA работает исправно.

Пункт 7.

Подпишем текст цифровой подписью:

1. Подпишем текст x цифровой подписью s, где s=xd(mod n) = 38892 22830 10190 5929 33288 17226 19627 29485 23677 41100 28962 7592 26240 17985.
2. Проверим подпись по формуле: x = se(mod n) = "abcdefghijklmn".

Тексты совпадают.

Пункт 8.

Смоделируем процесс установления сеансового ключа:

1. Выберем число a = 274.
2. Смоделируем действия пользователя А: Выберем число x = 84, найдем число A = ax(mod n) = 13505.
3. Смоделируем действия пользователя В: выберем число y = 56, вычислим число B = ay(mod n) = 20257.
4. Вычислим сеансовый ключ: Key = Bx(mod n) = Ay(mod n) = 37137.

Пункт 9.

См. в Приложении 1.

**Ответы на контрольные вопросы.**

1. Что такое вычет? На чем основан алгоритм шифрования Цезаря?

Вычет числа а – это число сравнимое с а по модулю b. Шифр Цезаря основан на использовании взаимно однозначного отображения колец вычетов со сдвигом.

1. Каковы особенности чисел Кармайкла?

Число Кармайкла – составное число N, которое удовлетворяет сравнению aN-1=1(mod N) для любых целых a, взаимно простых с N.

1. Перечислите основные свойства мультипликативной группы кольца вычетов по модулю pq?

Мультипликативная группа кольца вычетов по модулю pq является абелевой и циклической, состоит из ненулевых чисел, меньших n = pq и взаимно простых с n.

1. Почему порядок группы (Z/nZ)\* должен иметь большой простой делитель?

Порядок группы (Z/nZ)\* должен иметь большой простой делитель, потому что сложность нахождения порядка группы полиноминальна эквивалентна сложности разложения числа n.

1. Опишите алгоритм расчета кодов символов при декодировании шифрограмм согласно алгоритму Меркля-Хеллмана.

Код символа вычисляется с помощью цикла, на каждом шаге которого от ASCII-кода символа находится остаток от деления на 2 и записывается в последнюю пустую ячейку массива с бинарным кодом символа. Если число отрицательное, то в нулевую ячейку массива записывается 1.

**Вывод**

Изучив алгоритмы шифрования, можно сделать вывод о том, что шифр Цезаря не может быть использован для шифрования сообщений, так как его легко можно расшифровать. Для алгоритма RSA следует грамотно выбирать параметры, так как его сложность зависит от сложности разложения модуля. Алгоритм Меркля-Хеллмана может быть использован для шифрования сообщений, но данный алгоритм уже был скомпрометирован, и поэтому эта криптосистема в нынешнее время непопулярна.

.

**ПРИЛОЖЕНИЕ 1**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <math.h>

#include <locale.h>

#include <wchar.h>

#include <Windows.h>

#define N 33

#define Col 200

int k = 4;

int m = N;

void Caesar(char\* word) {

int i = 0;

while (word[i]) {

if (word[i] == -30) {

word[i] = -72;

}

else if (word[i] > -30 && word[i] < -26) {

word[i] += k - 1;

}

else if (word[i] > -5 && word[i] < 0) {

word[i] = word[i] - 32 + k;

}

else if (word[i] >= -32 && word[i] <= -1) {

word[i] += k;

}

else if (word[i] == -62) {

word[i] = -88;

}

else if (word[i] > -62 && word[i] < -58) {

word[i] += k - 1;

}

else if (word[i] > -37 && word[i] < -32) {

word[i] = word[i] - 32 + k;

}

else if (word[i] >= -64 && word[i] <= -33) {

word[i] = word[i] + k;

}

i++;

}

}

unsigned long long NOD(unsigned long long A, unsigned long long B) {

while (A > 0 && B > 0) {

if (A > B) {

A = A - (B \* (A / B));

}

else {

B = B - (A \* (B / A));

}

}

unsigned long long C = 0;

if (A > 0) C = A;

else C = B;

return C;

}

void SimpleCheck(unsigned long long number) {

int s = (int)log2((number - 1) / 3);

int flag = 0;

int k = 1;

for (int i = 2; i < number; i++) {

if (number % i == 0) {

printf("%llu - Составное число\nШаг - %d\nВероятность - %f\n", number, k, 1.0 / pow(4.0, k));

break;

}

else if ((unsigned long long)pow(i, 3) % number == 1) {

printf("%llu - Простое число\nШаг - %d\nВероятность - %f\n", number, k, 1.0 / pow(4.0, k));

break;

}

for (int j = 0; j < s; j++) {

if ((unsigned long long)pow(i, pow(2, j - 1)) % number == number - 1) {

printf("%llu - Простое число\nШаг - %d\nВероятность - %f\n", number, k, 1.0 / pow(4.0, k));

flag = 1;

break;

}

}

if (flag) {

flag = 0;

break;

}

if (i == number - 1) {

printf("%llu - Простое число\nШаг - %d\nВероятность - %f\n", number, k, 1.0 / pow(4.0, k));

}

k++;

}

}

unsigned long long DOR(unsigned long long number, unsigned long long degree, unsigned long long divider) {

unsigned long long a = number;

for (int i = 0; i < degree-1; i++) {

number = ((number % divider) \* (a % divider));

number %= divider;

}

return number;

}

void Model(unsigned long long a, unsigned long long x, unsigned long long y, unsigned long long n) {

unsigned long long A = DOR(a, x, n);

unsigned long long B = DOR(a, y, n);

unsigned long long messageA = DOR(B, x, n);

unsigned long long messageB = DOR(A, y, n);

unsigned long long axy = (a \* x \* y) % n;

printf("A - %llu\nB - %llu\nmessageA - %llu\nmessageB - %llu\naxy - %llu\n", A, B, messageA, messageB, axy);

}

void CryptoSystem(char text[]) {

unsigned int i = 0;

unsigned int closeKey[8] = { 2, 3, 6, 13, 27, 52, 105, 210 };

unsigned int m = 2 + 3 + 6 + 13 + 27 + 52 + 105 + 210 + 100;

unsigned int n = 19;

unsigned int revN = 2;

while ((revN \* n) % m != 1 || revN == n) revN++;

unsigned int openKey[8] = {(2\*n)%m, (3\*n)%m, (6\*n)%m, (13\*n)%m, (27\*n)%m, (52\*n)%m, (105\*n)%m, (210\*n)%m};

unsigned int cipherText[Col] = { 0 };

while (text[i]) {

int binaryView[8] = { 0 };

if (text[i] < 0) binaryView[0] = 1;

int h = 7;

char symbol = text[i];

while (symbol && h) {

binaryView[h] = symbol % 2;

symbol /= 2;

h--;

}

for (int j = 0; j < 8; j++) {

if (binaryView[j]) {

cipherText[i] += openKey[j];

}

}

i++;

}

i = 0;

while (cipherText[i]) {

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

i++;

}

printf("\n");

i = 0;

unsigned int cipherBinaryView[8] = {0};

while (cipherText[i]) {

int sum = (cipherText[i] \* revN) % m;

for (int j = 0; j < 8; j++) {

cipherBinaryView[7 - j] = 0;

if (sum >= closeKey[7 - j]) {

cipherBinaryView[7 - j] = 1;

sum -= closeKey[7 - j];

}

}

char newSymbol = 0;

for (int j = 1; j < 8; j++) {

newSymbol += cipherBinaryView[j] \* (int)pow(2, 7 - j);

}

if (cipherBinaryView[0]) newSymbol = -newSymbol;

printf("%c", newSymbol);

i++;

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

unsigned long long number1 = ((unsigned long long)pow((4851003 + 21) % 11, 11) + 3) % 11;

printf("Пункт 1\nNumber1 = %d\n\n", number1);

printf("Пункт 2\nВведите ФИО без пробелов для шифрования по Цезарю:\n");

char name[Col] = {0};

scanf("%s", name);

Caesar(name);

printf("%s\n\n", name);

unsigned long long A = pow((4851003 \* (8 + 21 % 7)), 2);

printf("Пункт 3\nA = %llu\n", A);

unsigned long long B = 30012004;

printf("B = %llu\n", B);

unsigned long long B1 = B % 95 + 900;

printf("NOD(A, B(mod 95) + 900) = %llu\n", NOD(A, B1));

unsigned long long B2 = (B + 50) % 97 + 700;

printf("NOD(A, (B + 50)(mod 97) + 700) = %llu\n", NOD(A, B2));

unsigned long long B3 = (B + 20) % 101 + 1500;

unsigned long long B4 = (B - 40) % 103 + 2500;

printf("NOD(A, (B + 20)(mod 101) + 1500, (B - 40)(mod 103) + 2500) = %llu\n\n", NOD(NOD(A, B3), B4));

unsigned int compositeNumber = 10251;

unsigned int simpleNumber = 727;

printf("Пункт 4\n");

SimpleCheck(compositeNumber);

SimpleCheck(simpleNumber);

printf("\n");

unsigned int p = 199;

unsigned int q = 211;

unsigned int n = p \* q;

unsigned int fn = (p - 1) \* (q - 1);

unsigned int e = 2;

while (NOD(e, p - 1) != 1 || NOD(e, q - 1) != 1) {

e++;

}

unsigned int d = 2;

while ((e \* d) % fn != 1 || e == d) d++;

printf("Пункт 5\np=%u q=%u n=%u e=%u d=%u\n\n", p, q, n, e, d);

char x[40] = "abcdefghijklmn";

unsigned long long cipher[40] = { 0 };

int i = 0;

printf("Пункт 6\nТекст - %s\nЗашифрованный текст - ", x);

while (x[i]) {

cipher[i] = DOR(x[i], e, n);

printf("%llu ", cipher[i]);

i++;

}

printf("\nРасшифрованный текст - ");

i = 0;

while (cipher[i]) {

unsigned long long y = DOR(cipher[i], d, n);

printf("%c", y);

i++;

}

printf("\n\nПункт 7\nТекст - %s\nПодписанный текст - ", x);

unsigned long long signature[40] = { 0 };

i = 0;

while (x[i]) {

signature[i] = DOR(x[i], d, n);

printf("%llu ", signature[i]);

i++;

}

printf("\nПроверенный текст - ");

i = 0;

while (signature[i]) {

unsigned long long s = DOR(signature[i], e, n);

printf("%c", s);

i++;

}

printf("\n\nПункт 8\n");

Model(274, 84, 56, n);

printf("\nПункт 9\nКриптосистема Меркла-Хеллмана\nВведите текст\n");

char text[Col] = { 0 };

scanf("%s", text);

CryptoSystem(text);

return 0;

}