Міністерство освіти і науки України

Сумський державний університет

Кафедра

Прикладної математики та моделювання складних систем

Звіт з лабораторної роботи № 2

Дисципліна

Криптографія

Студентка: Пороскун Олена Олегівна

Викладач: Козлова Ірина Іванівна

Суми, Сумська область

2021

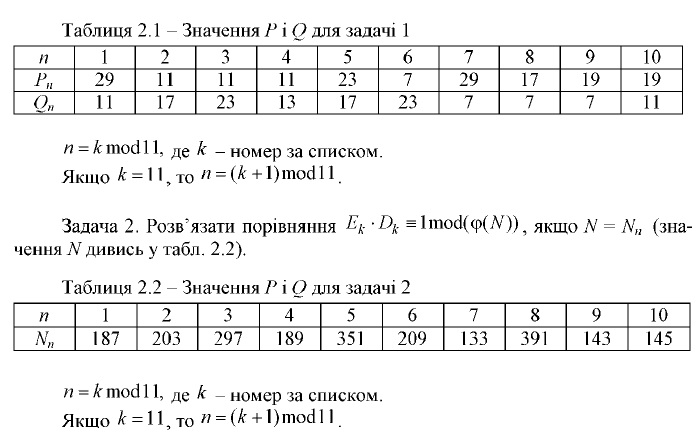
Лабораторна робота №2

**Тема**: Несиметричні криптоперетворення. Алгоритм RSA.

**Мета**: навчитися будувати пару (Ek, Dk) для RSA крипто алгоритму, знаходити модуль перетворення та значення функції Ейлера.

***Зміст задачі:***

1. Побудувати пару (Ek, Dk) для RSA криптоалгоритму, якщо P = Pn, Q = Qn (значення P і Q дивись у табл. 2.1)



***Програмний код(C++):***

#include <string.h>

#include <Windows.h>

#include <iostream>

#include <math.h>

#define MAS\_SIZE 50

using namespace std;

int r[MAS\_SIZE];

int num[MAS\_SIZE];

// НСД(найбільший спільний дільник) двох чисел

int NSD(int a, int b){

if (b == 0) {

return a;

}

int r = a % b;

return NSD(b, r);

}

// функція, що допомогає рекурентно знайти значення масиву а, для знаходженння у = Dк

int recur\_A(int a[], int r[], int mu){

int ans = 0;

cout << "\nЗнаходимо рекурентно значення а" << mu - 1 << "\n";

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

if (i == 0){

a[0] = r[0];

}

if (i == 1){

a[i] = r[0]\*r[1] + 1;

}

if (i > 1){

a[i] = a[i-1] \* r[i] + a[i-2];

}

cout << a[i] << " - a" << i << "\n";

}

ans = a[mu-1];

return ans;

}

int main(){

setlocale (LC\_CTYPE, "ukr");

int N, P, Q;

int phi, E, nsd;

cout << "Введiть P: ";

cin >> P;

cout << "Введiть Q: ";

cin >> Q;

cout << "P: " << P << "\n";

cout << "Q: " << Q << "\n";

N = P\*Q;

cout << "Модуль перетворення N: " << N << "\n";

phi = (P-1)\*(Q-1);

cout << "Функцiя Ейлера phi(N): " << phi << "\n\n";

int key = 0;

while (key != 1){

cout << "Виберiть випадково ключ Ek(так щоб вiн був взаємнопростий з ф. Ейлера): ";

cin >> E;

nsd = NSD(phi, E);

cout << "\nПеревiрка НСД (phi(N), Ek): " << nsd << "\n\n";

key = nsd;

if (nsd == 1){

cout << "Дiафантове рiвняння: ";

cout << phi << "x + " << E << "y = "<< nsd << "\n";

//printf("%dx + %dy = %d\n", phi, E, nsd);

}

else{

cout << "НСД (phi(N), Ek) не дорiвнює 1. \n\n";

}

}

int phi1 = phi, E1 = E;

for(int i = 0; i < MAS\_SIZE; i++){

r[i] = phi1 / E1;

num[i] = phi1 % E1;

phi1 = E1;

E1 = num[i];

if (E1 == 1){

break;

}

}

cout << "\n";

int m = 0;

for (int i = 0; i < MAS\_SIZE; i++){

if (r[i] != 0){

m++;

}

}

cout << "Подамо а/b = phi(N)/Ek у виглядi ланцювого дробу:";

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

cout << "\n" << r[i] << " - r" << i;

//cout << "\n" << num[i] << " - num " << i;

}

cout << "\nТаким чином, Mu = " << m << "\n\n";

cout << "Розрахуємо значення y = Dk\n"; // y = (-1)^m \* a\_(m-1);

int a[m];

int A = recur\_A(a, r, m);

cout << "Значення а" << m - 1 << " = " << A;

int y = pow(-1, m)\*A;

y = y % phi;

while (y < 0){

y = y + phi;

}

cout << "\n\n" << "y = Dk = y mod(phi(N)) = " << y << "\n\n";

cout << "Перевiрка\n\n";

int multEkDk = E\*y;

printf("Вираз (Ek \* Dk) mod(phi(N)) має дорiвнювати 1. \n", E, y, phi);

printf("Цей вираз дорiвнює (%d) mod(%d) = 1. \n\n", multEkDk, phi);

if ((multEkDk % phi) == 1){

printf("Отже, (Ek, Dk) = (%d, %d) складає RSA ключову пару.\n\n", E, y);

}

else{

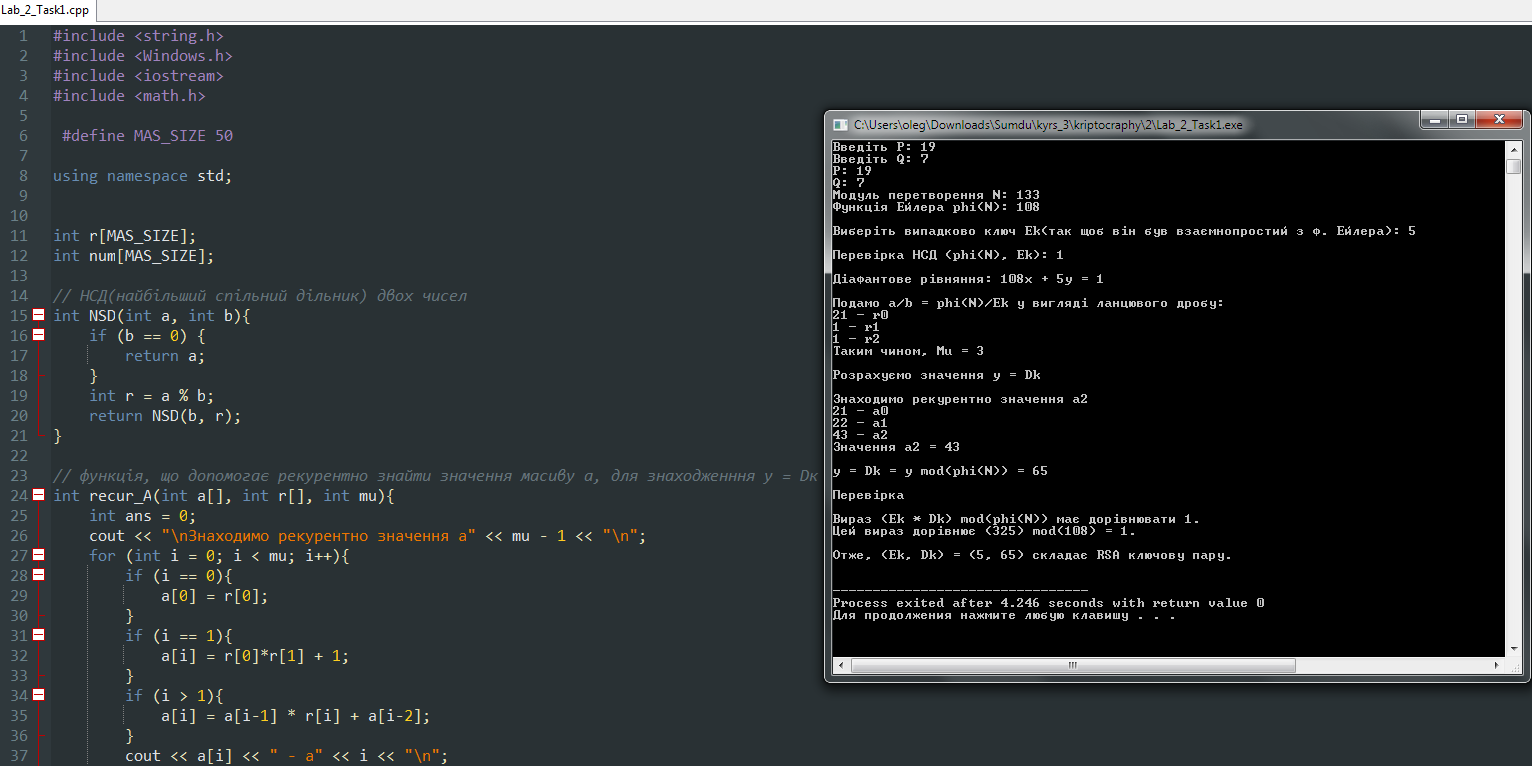
printf("Отже, (Ek, Dk) = (%d, %d) не складає RSA ключову пару.\n\n", E, y);

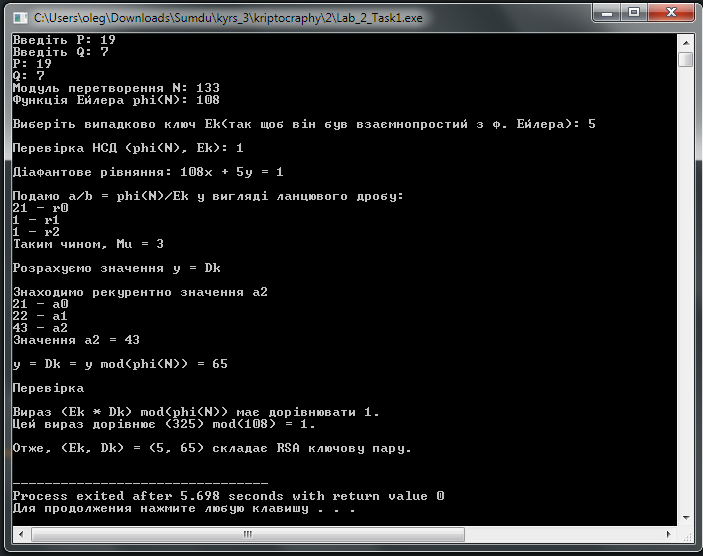
}

return 0;

}

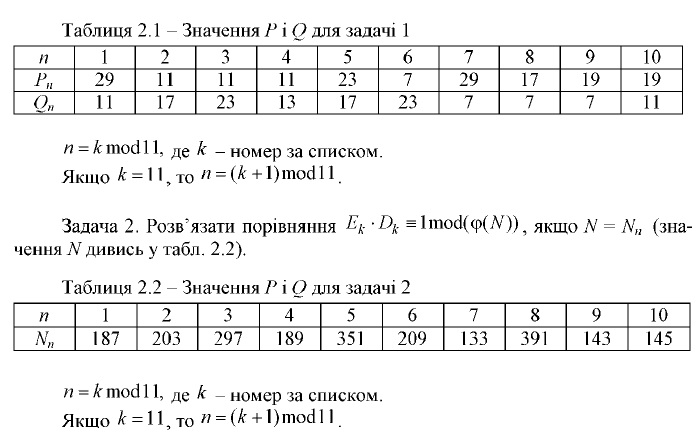
***Прінт-скрін виконання програми:***

******

******

***Зміст задачі:***

2. Розв’язати порівняння Ek · Dk ≡ 1 mod(φ(*N*)), якщо *N* = *Nn*  (значення *N* дивись у табл. 2.2)



***Програмний код(C++):***

#include <string.h>

#include <Windows.h>

#include <iostream>

#include <math.h>

#define MAS\_SIZE 50

using namespace std;

int r[MAS\_SIZE];

int num[MAS\_SIZE];

// НСД(найбільший спільний дільник) двох чисел

int NSD(int a, int b){

if (b == 0) {

return a;

}

int r = a % b;

return NSD(b, r);

}

// функція, що допомогає рекурентно знайти значення масиву а, для знаходженння у = Dк

int recur\_A(int a[], int r[], int mu){

int ans = 0;

cout << "\nЗнаходимо рекурентно значення а" << mu - 1 << "\n";

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

if (i == 0){

a[0] = r[0];

}

if (i == 1){

a[i] = r[0]\*r[1] + 1;

}

if (i > 1){

a[i] = a[i-1] \* r[i] + a[i-2];

}

cout << a[i] << " - a" << i << "\n";

}

ans = a[mu-1];

return ans;

}

int main(){

setlocale (LC\_CTYPE, "ukr");

int N, P, Q;

int phi, E, nsd;

cout << "Введiть P: ";

cin >> P;

cout << "Введiть Q: ";

cin >> Q;

cout << "P: " << P << "\n";

cout << "Q: " << Q << "\n";

//N = P\*Q;

cout << "Введiть модуль перетворення N: ";

cin >> N;

cout << "N: " << N << "\n";

phi = (P-1)\*(Q-1);

cout << "Функцiя Ейлера phi(N): " << phi << "\n\n";

int key = 0;

while (key != 1){

cout << "Виберiть випадково ключ Ek(так щоб вiн був взаємнопростий з ф. Ейлера): ";

cin >> E;

nsd = NSD(phi, E);

cout << "\nПеревiрка НСД (phi(N), Ek): " << nsd << "\n\n";

key = nsd;

if (nsd == 1){

cout << "Дiафантове рiвняння: ";

cout << phi << "x + " << E << "y = "<< nsd << "\n";

//printf("%dx + %dy = %d\n", phi, E, nsd);

}

else{

cout << "НСД (phi(N), Ek) не дорiвнює 1. \n\n";

}

}

int phi1 = phi, E1 = E;

for(int i = 0; i < MAS\_SIZE; i++){

r[i] = phi1 / E1;

num[i] = phi1 % E1;

phi1 = E1;

E1 = num[i];

if (E1 == 1){

break;

}

}

cout << "\n";

int m = 0;

for (int i = 0; i < MAS\_SIZE; i++){

if (r[i] != 0){

m++;

}

}

cout << "Подамо а/b = phi(N)/Ek у виглядi ланцювого дробу:";

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

cout << "\n" << r[i] << " - r" << i;

//cout << "\n" << num[i] << " - num " << i;

}

cout << "\nТаким чином, Mu = " << m << "\n\n";

cout << "Розрахуємо значення y = Dk\n"; // y = (-1)^m \* a\_(m-1);

int a[m];

int A = recur\_A(a, r, m);

cout << "Значення а" << m - 1 << " = " << A;

int y = pow(-1, m)\*A;

y = y % phi;

while (y < 0){

y = y + phi;

}

cout << "\n\n" << "y = Dk = y mod(phi(N)) = " << y << "\n\n";

cout << "Перевiрка\n\n";

int multEkDk = E\*y;

printf("Вираз (Ek \* Dk) mod(phi(N)) має дорiвнювати 1. \n", E, y, phi);

printf("Цей вираз дорiвнює (%d) mod(%d) = 1. \n\n", multEkDk, phi);

if ((multEkDk % phi) == 1){

printf("Отже, (Ek, Dk) = (%d, %d) складає RSA ключову пару.\n\n", E, y);

}

else{

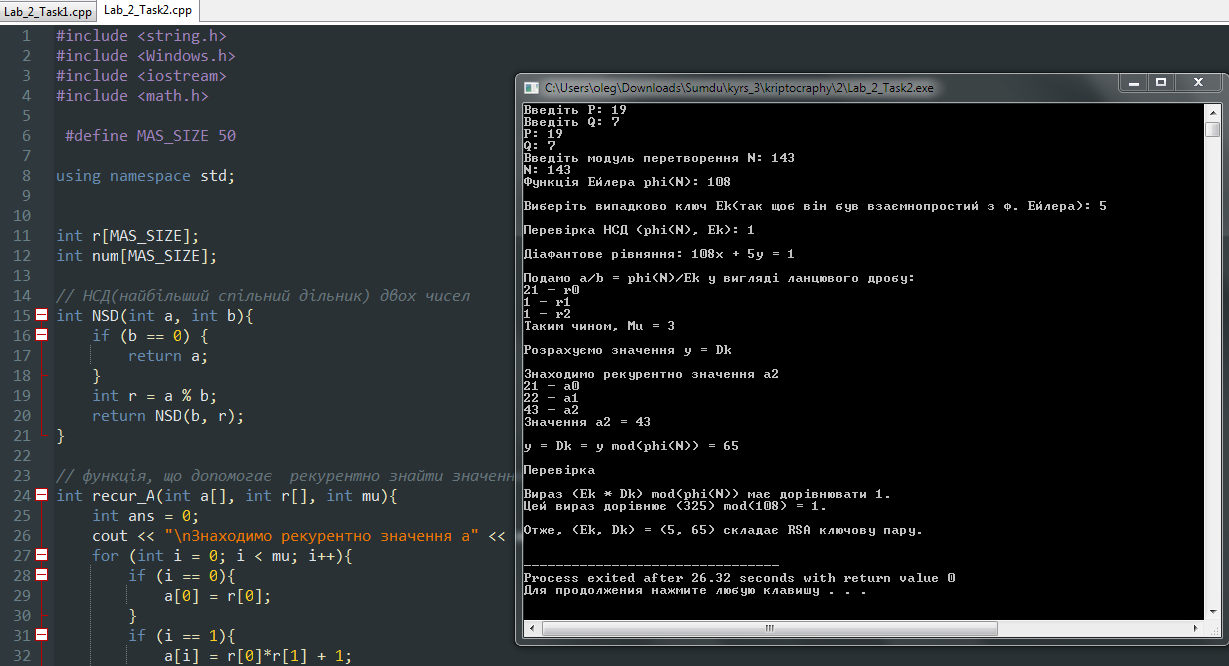
printf("Отже, (Ek, Dk) = (%d, %d) не складає RSA ключову пару.\n\n", E, y);

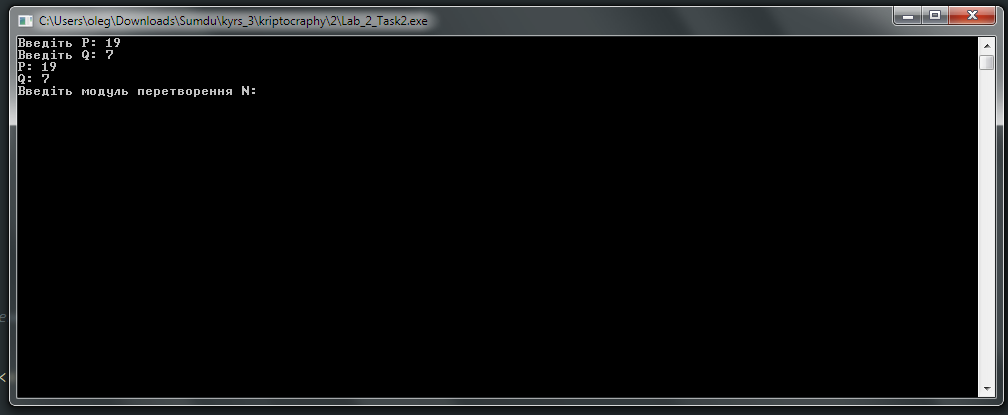
}

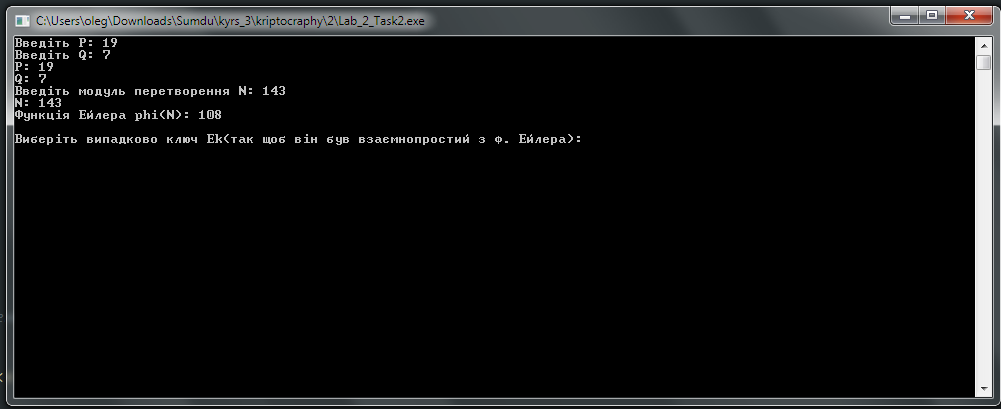
return 0;

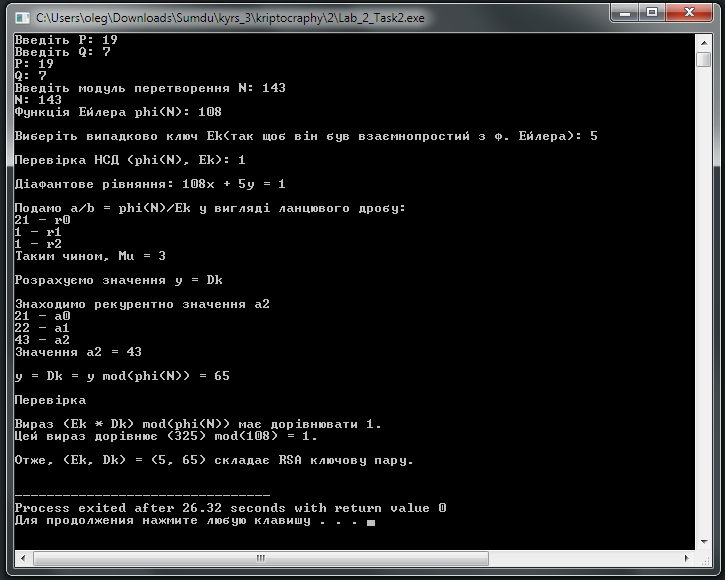
}

***Прінт-скрін виконання програми:***



******

******



***Контрольне питання***

9. Порівняйте складність різних методів RSA крипто перетворень.

Опис модифікацій алгоритму RSA

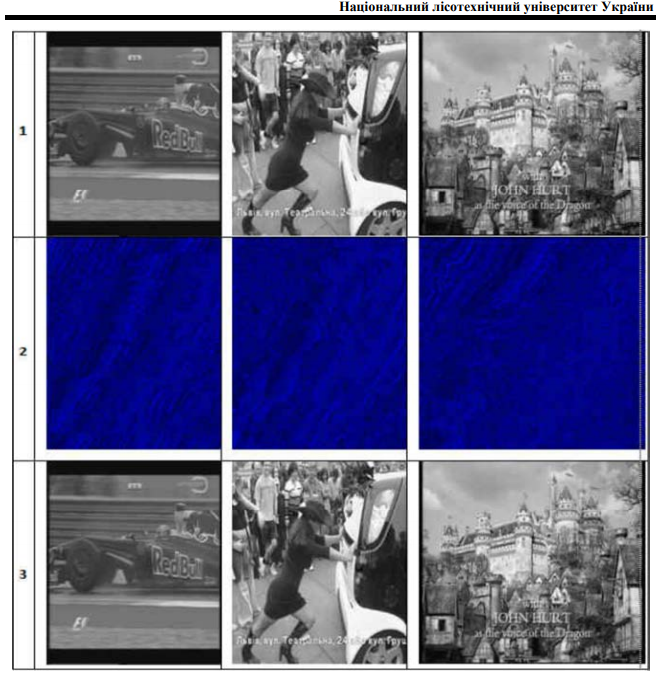
**Шифрування і дешифрування по одному рядку матриці зображення** [1]. Нехай P, Q – пара довільних простих чисел і N = P·Q. Шифрування відбувається поелементно з використанням такого перетворення елементів матриці зображення С:

1. Випадково вибирають натуральне число e < ϕ(N) і знаходять таке натуральне d, що виконується конгруенція ed ≡ 1(mod ϕ(N)).

2. Будують число A = сij + Q + P + i + j – d.

3. Зашифрованим значенням інтенсивності i-го піксела, i = 1, 2, …, m, m – кількість елементів у рядку, вибирають число B ≡ Ae (mod N).

Дешифрування проводять в порядку, протилежному до шифрування після отримання числа Bd ≡ (Ae)d (mod N), виконанням протилежних операції до змісту пунктів 3), 2), 1). Результати наведено на рис. 1.



*Рис. 1.*

*1) початкові зображення;*

*2) зашифровані зображення;*

*3) дешифровані зображення*

**Шифрування і дешифрування по одному рядку матриці з додатковим зашумленням**

Нехай P, Q – пара довільних простих чисел і N = P·Q. Шифрування відбувається поелементно з використанням такого перетворення елементів матриці зображення С:

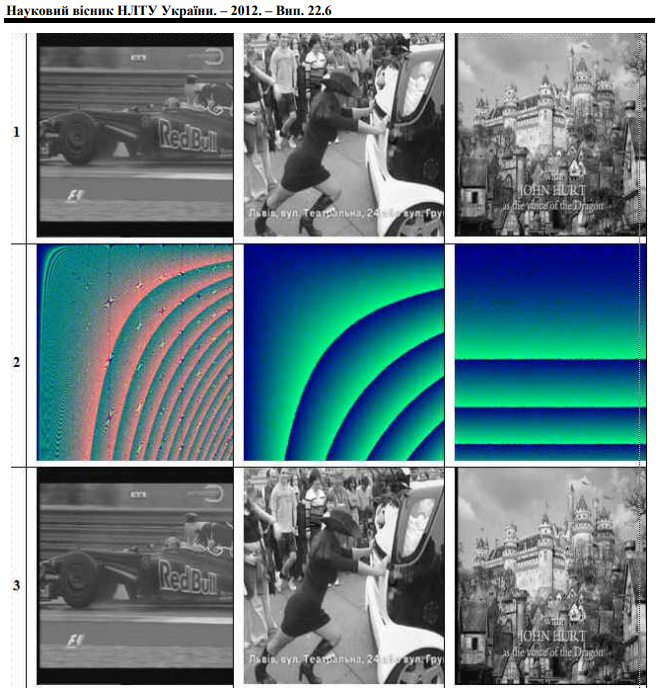
1. Випадково вибирають натуральне число e < ϕ(N) і знаходять таке натуральне d, що виконується конгруенція ed ≡ 1(mod ϕ(N)).

2. Будують число A = сij + Q + P + i + j – d.

3. Зашифрованим значенням інтенсивності i-го піксела, i = 1, 2, …, m, m – кількість елементів у рядку, вибирають число С ≡ Ae (mod N) + f(i, j). Дешифрування проводять в порядку, протилежному до шифрування після отримання числа (С – f(i, j))d ≡ (Ae)d (mod N), виконанням протилежних операції до змісту пунктів 3), 2), 1).

Результати наведено на рис. 2. Для шифрування вибирали такі функції:

f(i, j) = i2 , f(i, j) = i · j, f(i, j) = j2 .



*Рис. 2.*

*1) початкові зображення;*

*2) зашифровані зображення;*

*3) дешифровані зображення*

З порівняння рис. 1, 2) і рис. 2, 2) видно, що шифрування з додатковим зашумленням відрізняється від шифрування без додаткового зашумлення. Контури в обох зашифрованих зображеннях відсутні. Початкові і дешифровані зображення тільки незначно відрізняються рівнем яскравості. Функції додаткової зашумленості f(i, j) можуть бути довільними цілозначними функціями і додатково, до створюваної алгоритмом RSA зашумленості, підвищують криптографічну стійкість вказаних модифікацій.

*Висновки*:

1. Запропоновані модифікації шифрування призначені для шифрування зображень в градаціях сірого кольору і ґрунтуються на використанні ідей базового алгоритму RSA. Їх також можна використати і стосовно кольорових зображень.

2. Стійкість до несанкціонованого дешифрування, запропонованими потоковою модифікацією, забезпечується алгоритмом RSA.

3. Модифіковані методи шифрування побудовані так, що за малих значень ключа також можна досягти якісного шифрування, але за умови вірного підбору параметрів шифрування. При цьому досягається висока швидкість роботи алгоритму.

***Отже****, порівнюючи дві дані модифікації алгоритму RSA, можна сказати про те, що більш складна модифікація – та, у якій використовується шифрування і дешифрування по одному рядку матриці з додатковим зашумленням. Функції додаткової зашумленості підвищують криптографічну стійкість модифікацій.*

***Висновок***

Виконуючи лабораторну роботу, навчилися будувати пару (Ek, Dk) для RSA крипто алгоритму, знаходити модуль перетворення та значення функції Ейлера на практиці, складати програму розв’язування цих задач.

***Список використаної літератури***

1. Ю. О. Борзов, А. М. Ковальчук, Д. Д. Пелешко Модифікація алгоритму RSA: шифрування та дешифрування за одним рядком матриці зображення // Науковий вісник НЛТУ України . 2012. №6. URL: https://cyberleninka.ru/article/n/modifikatsiya-algoritmu-rsa-shifruvannya-ta-deshifruvannya-za-odnim-ryadkom-matritsi-zobrazhennya

***Додаток***

