МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра програмних систем і технологій

**Практична робота №3**

з дисципліни

**«Захист програм та даних»**

**Тема :** **«Шифри гамування»**

Виконали:

Ємець Євгенія Ярославівна

Олійник Анастасія Миколаївна

Група ІПЗ-33

Перевірила:

Супрун Ольга Миколаївна

Дата перевірки

2018

**Хід роботи**

**Завдання №1**

В практичній роботі необхідно зашифрувати своє прізвище за допомогою шифру гамування по модулю N та модулю 2.

При оформленні звіту необхідно привести ісходне повідомлення (прізвище), гаму та таблицю шифрування/дешифрування.

Гамування - метод симетричного шифрування, що полягає в «накладенні» послідовності, що складається з випадкових чисел, на відкритий текст. Послідовність випадкових чисел називається гамма-послідовністю і використовується для зашифровування і розшифрування даних. Підсумовування, зазвичай, виконується в будь-якому кінцевому полі. Наприклад, в поле Галуа GF (2) підсумовування набуває вигляду операції «виключне АБО (xor)».

Для шифрування кожного нового повідомлення потрібно використовувати нову гаму. Повторне використання гами неприпустимо з огляду на властивостей операції «xor». Розглянемо приклад: за допомогою однакової гами Y зашифровані два відкритих тексту X₁ і X₂, отримано дві шифрограми і :

Виконаємо додавання двох шифрограму, використовуючи операцію «xor»:

Результат залежить від відкритих текстів і і не залежить від гами . З огляду на надмірності природних мов результат піддається частотному аналізу, тобто відкриті тексти можна підібрати, не знаючи гаму .

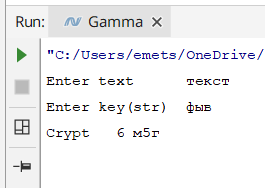
Для формування гами (послідовності псевдовипадкових чисел) потрібно використовувати апаратні генератори випадкових чисел, засновані на фізичних процесах. Якщо кольорів не буде випадкової, для отримання відкритого тексту потрібно підібрати тільки початковий стан генератора псевдовипадкових чисел.

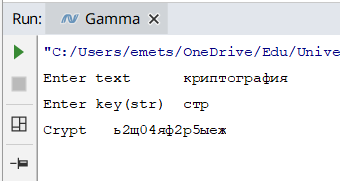
Довжина гами повинна бути не менше довжини захищається повідомлення (відкритого тексту). В іншому випадку для отримання відкритого тексту потрібно підібрати довжину гами, проаналізувати блоки шифротексту вгаданої довжини, підібрати біти гами.

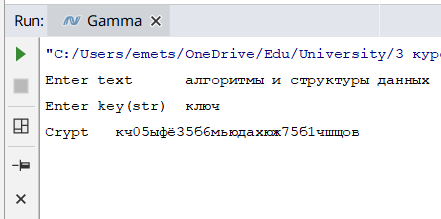
Лістинг програми шифрування по модулю N

**using** System;  
  
**namespace** Gamma{  
 **internal static class** Xor{  
 **private const string *Alf*** = **"абвгдеёжзийклмнопрстуфхцчшщъыьэюя 0123456789"**;  
 **private static int \_k**, **\_x**, **\_z**;  
 **private static string \_res**;  
  
 **public static string** Encryption(**string** source, **string** key){  
 **\_res** = **string**.**Empty**;  
  
 **while** (key.**Length** < source.**Length**){  
 key += key;  
 **if** (key.**Length** > source.**Length**) key = key.Remove(source.**Length**);  
 }  
  
 **for** (**var** i = 0; i < source.**Length**; i++){  
 **for** (**var** id = 0; id < ***Alf***.**Length**; id++){  
 **if** (key[i] == ***Alf***[id]) **\_k** = id;  
 **if** (source[i] == ***Alf***[id]) **\_x** = id;  
 **\_z** = (**\_x** + **\_k**) % ***Alf***.**Length**;  
 }  
  
 **\_res** += ***Alf***[**\_z**];  
 }  
 **return \_res**;  
 }  
  
 **public static string** Decryption(**string** source, **string** key){  
 **\_res** = **string**.**Empty**;  
  
 **while** (key.**Length** < source.**Length**){  
 key += key;  
 **if** (key.**Length** > source.**Length**) key = key.Remove(source.**Length**);  
 }  
  
 **for** (**var** i = 0; i < source.**Length**; i++){  
 **for** (**var** id = 0; id < ***Alf***.**Length**; id++){  
 **if** (key[i] == ***Alf***[id]) **\_k** = id;  
 **if** (source[i] == ***Alf***[id]) **\_x** = id;  
 **\_z** = ((source[i] - key[i]) + ***Alf***.**Length**) % ***Alf***.**Length**;  
 }  
 **\_res** += ***Alf***[**\_z**];  
 }  
 **return \_res**;  
 }  
 }  
  
 **internal static class** Main1{  
 **public static void** Main(**string**[] args){  
 Console.Write(**"Enter text\t"**);  
 **var** text = Console.ReadLine();  
 Console.Write(**"Enter key(str)\t"**);  
 **var** key = Console.ReadLine();  
  
 **var** crypt = Xor.Encryption(text, key);  
 Console.WriteLine(**"Crypt\t{0}"**,crypt); *//шифрование. вернет "ыкык"* **var** decrypt = Xor.Decryption(crypt, key); *// дешифрование. вернет "мама"* Console.WriteLine(**"Decrypt\t{0}"**,decrypt);  
 }  
 }  
}

Приклади роботи







Лістинг програми шифрування по модулю 2

**using** System;  
**using** System.Linq;  
  
**namespace** Gamma2{  
 **internal static class** Program{  
 **private const string *alphabet*** = **"qwertyasdfghjklzxcvbnm 0123456789"**;  
  
 **private static string** Crypt(**string** key, **string** text){  
 **var** alphaindex = **new int**[***alphabet***.**Length**];  
 **for**(**var** i = 0;i< ***alphabet***.**Length**;i++)*//расставляем индексы для букв в алфавите* alphaindex[i] = i;  
   
 **var** newtext = **string**.**Empty**;*//выходящий текст* **var** ikey = 1 + key.Sum(t => alphaindex[***alphabet***.IndexOf(***alphabet***, t)]);*//Значение для перевода словесного ключа в число* **var** random = **new** Random();  
 **var** lol = **new int**[text.**Length**];*//массив случайных значений по длине текста* **for** (**var** i = 0; i < text.**Length**; i++)  
 lol[i] = random.Next(ikey);  
 **for** (**var** i = 0;i<text.**Length**;i++)  
 newtext += ***alphabet***[(alphaindex[***alphabet***.IndexOf(***alphabet***, text[i])] ^ lol[i])%***alphabet***.**Length**];  
 **return** newtext;  
 }  
   
 **public static void** Main(**string**[] args){  
 Console.Write(**"Enter key(str)\t"**);  
 **var** key = (Console.ReadLine());  
   
 Console.Write(**"Enter text\t"**);  
 **var** text = Console.ReadLine();  
   
 Console.WriteLine(**"Coded text\t{0}"**,Crypt(key, text));  
 }  
 }  
}