Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Кафедра цифрових технологій в енергетиці

ЗВІТ   
з виконання лабораторної роботи №6  
з дисципліни «Безпека інформаційних систем»

«**Шифрування з відкритим ключем на основі задачі рюкзака**»

Виконав: студент групи ТР-12 Каркушевський В.Л.

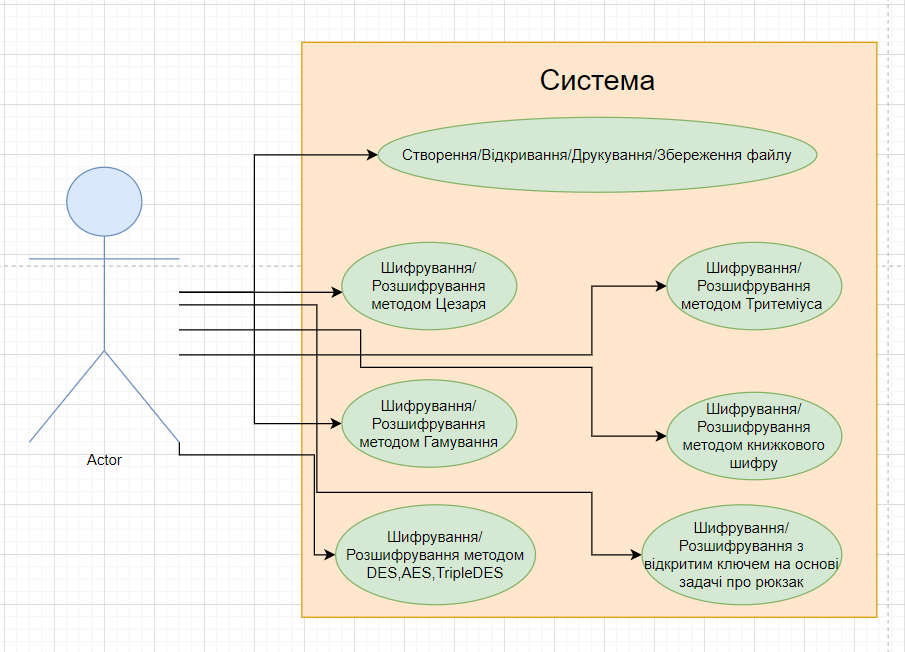
Перевірив: доцент, к.ф.-м.н.  
Тарнавський Ю.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

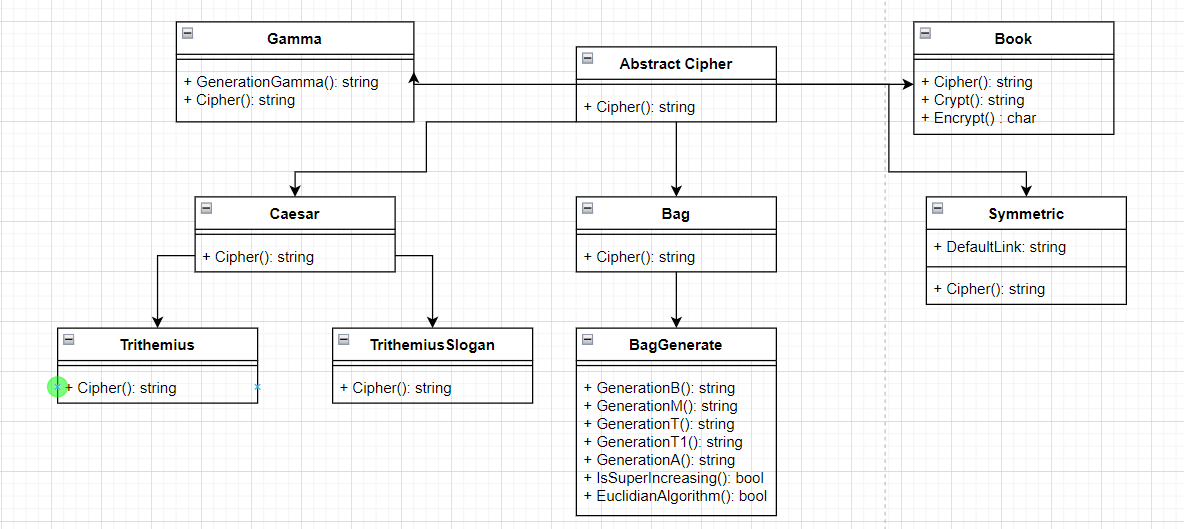
# **Мета роботи**

Ознайомитись з принципами побудови асиметричних криптосистем

# **Діаграма прецедентів**



# **Діаграма класів**



# **Фрагмент коду з реалізацією алгоритму шифрування/розшифрування**

**AbstractCipher**

public abstract class AbstractCipher  
**{** protected const int AlphabetSize = 55295**;** // кількість символів в алфавіті Unicode  
   
 public abstract string Cipher(bool variant**,** string[] text**,** string startKey)**;**}

public class Symmetric : AbstractCipher  
**{**

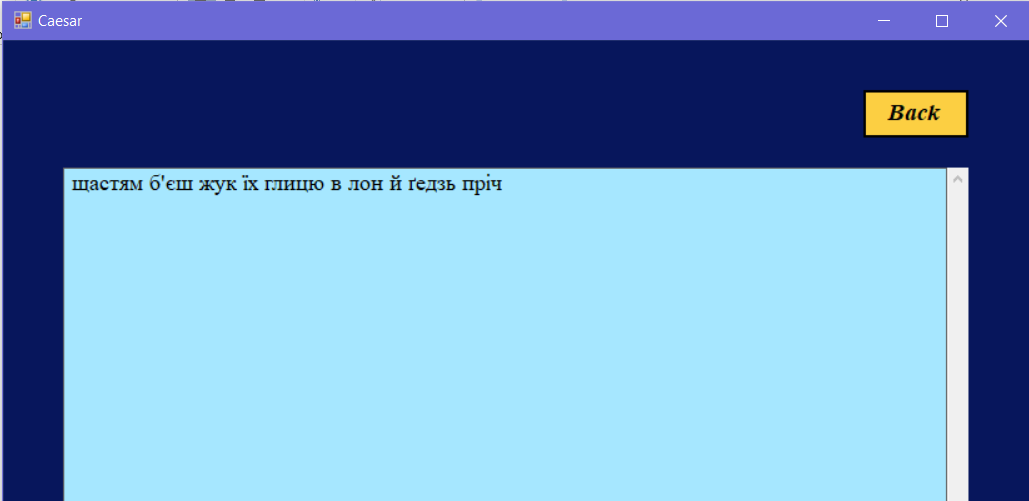
**Bag**public class Bag : AbstractCipher  
{  
 public override string Cipher(bool variant, string[] text, string[] startKey)  
 {  
 var cipherText = new StringBuilder();  
 var b = startKey[0].Split(',').Select(int.*Parse*).ToArray();  
 var m = int.*Parse*(startKey[1]);  
 var t1 = int.*Parse*(startKey[3]);  
 var a = startKey[4].Split(',').Select(int.*Parse*).ToArray();  
 foreach (var oneLine in text)  
 {  
 if (variant)  
 {  
 var temp = new StringBuilder();  
 foreach (var t in oneLine)//кожен символ конвертуємо в 16-бітний код  
 {  
 temp.Append(Convert.*ToString*(Convert.*ToInt32*(t), 2).PadLeft(16, '0'));  
 }  
  
 while (temp.Length % a.Length != 0)// додаємо невистачаючі символи  
 {  
 temp.Append("0");  
 }  
  
 var blocks = new List<string>();  
  
 for (int i = 0; i < temp.Length; i += 5)//ділимо на блоки  
 {  
 blocks.Add(temp.ToString().Substring(i, 5));  
 }  
 var encryptedBlocks = new List<int>();  
 foreach (var block in blocks)//шифруємо  
 {  
 var encryptedBlock = 0;  
 for (int i = 0; i < block.Length; i++)  
 {  
 if (block[i] == '1')  
 {  
 encryptedBlock += a[i];  
 }  
 }  
 encryptedBlocks.Add(encryptedBlock);  
 }  
  
 for (int i = 0; i < encryptedBlocks.Count-1; i++)  
 {  
 cipherText.Append(encryptedBlocks[i] + ",");  
 }  
 cipherText.Append(encryptedBlocks[encryptedBlocks.Count-1]);  
   
 }  
 else  
 {  
 var blocks = oneLine.Split(',');//розбиваємо на блоки  
 for (int i = 0; i < blocks.Length; i++)//множимо на t1  
 {  
 blocks[i] = ((int.*Parse*(blocks[i]) \* t1) % m).ToString();  
 }  
  
 var binaryBlocks = new List<string>();  
 foreach (var t in blocks)//перетворюємо число на двійковий код  
 {  
 Stack<int> stack = new Stack<int>();  
 int sum = int.*Parse*(t);  
 for (int j = b.Length-1; j >=0 ; j--)  
 {  
 if (b[j] <= sum)  
 {  
 stack.Push(1);  
 sum -= b[j];  
 }  
 else  
 {  
 stack.Push(0);  
 }  
 }  
  
 binaryBlocks.Add(string.*Join*("",stack));  
 }  
  
 var allBinary = string.*Join*("",binaryBlocks);//створюємо бінарне значення  
 var last = allBinary.Length % 16;  
 for (int i = 0; i < allBinary.Length-last; i+= 16)// перетворюємо на символи  
 {  
 cipherText.Append(Convert.*ToChar*(Convert.*ToUInt16*(allBinary.Substring(i, 16), 2)));  
 }  
  
 }  
  
 cipherText.AppendLine();  
 }  
  
 return cipherText.ToString();  
 }  
}

**BagGenerate**

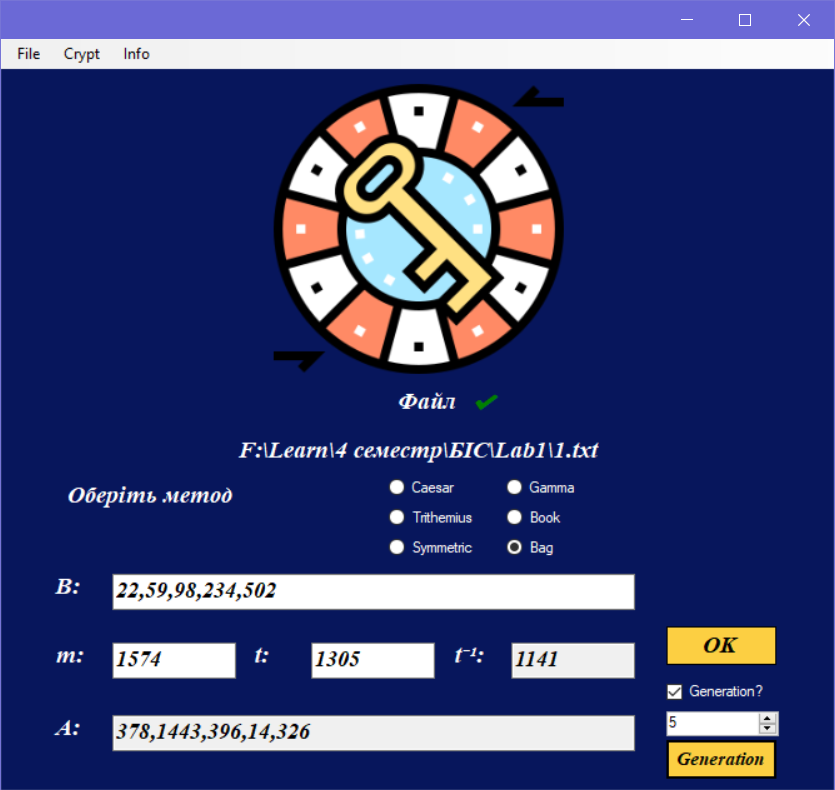
public class BagGenerate : Bag  
**{** public static string *GenerationB*(int number)  
 {  
 var b = new StringBuilder()**;** var random = new Random()**;** for (int i = 1**;** i <= number**;** i++)  
 {  
 var low = (int) ((Math.*Pow*(2**,** i - 1) - 1) \* Math.*Pow*(2**,** number) + 1)**;** var max = (int) (Math.*Pow*(2**,** i - 1) \* Math.*Pow*(2**,** number))**;** b.Append(random.Next(low**,** max + 1) + ",") **;** }  
 b.Remove(b.Length - 1**,** 1)**;** return string.*Join*(","**,**b)**;** }  
  
 public static string *GenerationM*(string b)  
 {  
 var listB = b.Split(',').Select(int.*Parse*).ToArray()**;** return new Random().Next(listB.Sum() + 1**,** listB.Sum() + 1000).ToString()**;** }  
   
 public static string *GenerationT*(string m)  
 {  
 int t**;** var random = new Random()**;** var numberM = int.*Parse*(m)**;** do  
 {  
 t = random.Next(7**,** numberM)**;** } while (!*EuclideanAlgorithm*(t**,**numberM))**;** return t.ToString()**;** }  
 public static int *GenerationT1*(int t**,** int m)  
 {  
 var m0 = m**;** int x = 0**,** y = 1**;** if (m == 1)  
 {  
 return 0**;** }  
  
 while (t > 1)  
 {  
 var q = t / m**;** var t0 = m**;** m = t % m**;** t = t0**;** t0 = x**;** x = y - q \* x**;** y = t0**;** }  
  
 if (y < 0)  
 {  
 y += m0**;** }  
  
 return y**;** }  
   
 public static string *GenerationA*(int[] b**,** int t**,** int m)  
 {  
 var arrayA = new StringBuilder()**;** for (int i = 0**;** i < b.Length-1**;** i++)  
 {  
 arrayA.Append(b[i] \* t % m + ",")**;** }  
 arrayA.Append(b[b.Length-1] \* t % m )**;** return arrayA.ToString()**;** }  
  
   
  
   
 public static bool *IsSuperIncreasing*(int[] sequence)  
 {  
 var sum = 0**;** foreach (var t in sequence)  
 {  
 if (t <= sum)  
 {  
 return false**;** }  
   
 sum += t**;** }  
  
 return true**;** }  
   
 public static bool *EuclideanAlgorithm*(int a**,** int b)  
 {  
   
 a = Math.*Abs*(a)**;** b = Math.*Abs*(b)**;** while (b != 0)  
 {  
 int temp = b**;** b = a % b**;** a = temp**;** }  
  
   
 return a==1**;** }  
  
**}**

# **Скріншоти програми**

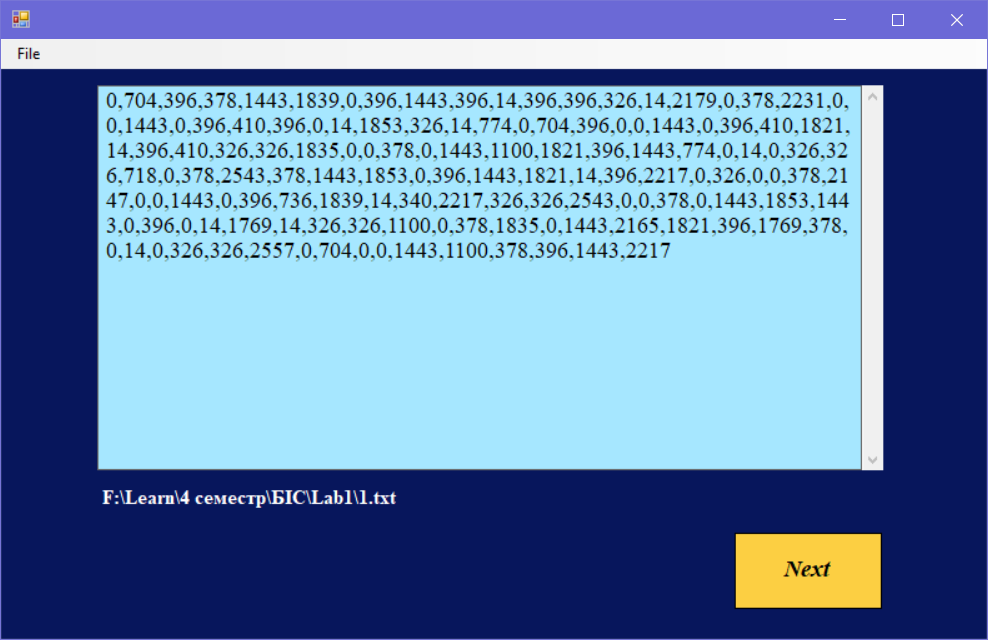
Оберемо панграму



Оберемо метод шифрування з відкритим ключем на основі задачі рюкзака та згенеруємо необхідні поля



Зашифруємо текст. У результаті отримуємо



Збережемо результат. Для розшифрування необхідно ввести B, t, m та натиснути кнопку ОК для заповнення t⁻¹ та A. У результаті отримаємо



# **Висновки**

На цій лабораторній роботі було ознайомлено з теоретичними відомостями про принципи побудови асиметричних систем . Модифіковано програму, розроблену у 5 лабораторній роботі, з графічним інтерфейсом на мові програмування C# з використанням Windows Forms. У програму доповнено новий метод шифрування з відкритим ключем на основі задачі рюкзака. Виконано тестування роботи шифрування та розшифрування.