Министерство образования и науки Российской Федерации

Федеральное Государственное Бюджетное Образовательное Учреждение

Высшего Профессионального Образования

Тамбовский Государственный Технический Университет

Кафедра

Лабораторная работа №3

дисциплина «Защита информации»

Вариант 17

Выполнил студент гр. –41

Д. Ю.

Проверила: И.Л.

Тамбов

**Задание:**

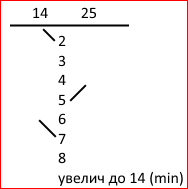
*Реализовать алгоритм Рюкзака.*

**Решение:**

Шифрование

Для начала необходимо сгенерировать сверхвозрастающую последовательность чисел, тоесть закрытый ключ. Правило такое: *каждой член больше суммы всех предыдущих членов*. Например, последовательность {2, 3, 6, 13, 27, 52} является сверхвозрастающей.

Далее есть формула , где i – элемент закрытого ключа, m – число, большее суммы элементов закрытого ключа, 105>(2+3+6+13+27+52) = 103. Выбираем n – взаимно простое к m число, например 31.

Смысл взаимнопростого в делении, оно не должно делиться на такие же числа, что m.

2\*31 mod 105 = 62

3\*31 mod 105 = 93

6\*31 mod 105 = 81

13\*31 mod 105 = 88

27\*31 mod 105 = 102

52\*31 mod 105 = 37

Итого- {62, 93, 81, 88, 102, 37}. Получили нормальную последовательность – это есть открытый ключ.

Входное сообщение переводим в царство нулей и единиц, разбиваем на блоки, равные по длине числу элементов последовательности рюкзака (6 штук). Затем, считая, что единица указывает на присутствие члена последовательности, а ноль - на его отсутствие, вычисляем полные веса рюкзаков - по одному для каждого блока сообщения.

Например, если сообщение в бинарном виде выглядит как 011000110101101110, шифрование, использующее предыдущую последовательность рюкзака, будет происходить следующим образом:

сообщение = 011000 110101 101110

011000 соответствует 93 + 81 = 174

110101 соответствует 62 + 93 + 88 + 37 = 280

101110 соответствует 62 + 81 + 88 + 102 = 333

Шифротекстом будет последовательность 174,280,333.

Дешифрование

Получатель данного сообщения знает закрытый ключ, а также значения n=31 и m=105, использованные для превращения ее в нормальную последовательность рюкзака.

Для дешифрирования сообщения получатель должен сначала определить число X в формуле . Магия обычного цикла for

/// <summary>

/// Подобрать обратное n число

/// </summary>

/// <param name="n">31</param>

/// <param name="m">105</param>

/// <returns></returns>

**private** int ReversN**(**int n**,** int m**)**

**{**

var X **=** **-**1**;**

//подобрать обратное число n

**for** **(**int i **=** 1**;** i **<** m/\*int.MaxValue\*/**;** i**++)**

**{**

**if** **((**n **\*** i**)** **%** m **!=** 1**)** **continue;**

X **=** i**;**

**break;**

**}**

**if** **(**X **==** **-**1**)** **new** Exception**(**"Не подобрано обратное n число!"**);**

**return** X**;**

**}**

Вобщем X=61. Теперь каждое число шифротекста умножаем по схеме:

174\*61 mod 105 = 9 = 3 + 6, что соответствует 011000

280\*61 mod 105 = 70 = 2 + 3 + 13 + 52, что соответствует 110101

333\*61 mod 105 = 48 = 2 + 6 + 13 + 27, что соответствует 101110

Расшифрованным открытым текстом является 011000 110101 101110.

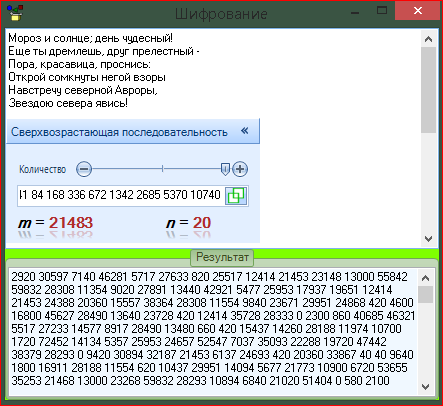


Рисунок Пример шифрования в программе

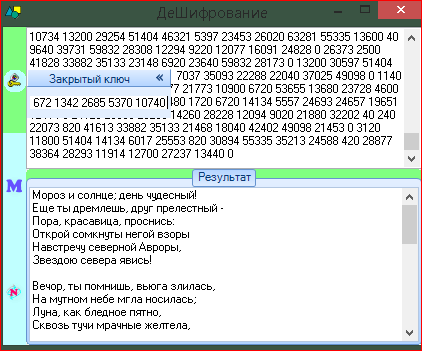


Рисунок Пример дешифрования в программе

Исходный код

А:

class clCripto

**{**

**public** clCripto**(){}**

/// <summary>

/// Сгенерит закрытый ключ

/// </summary>

/// <param name="value"></param>

/// <param name="key"></param>

**public** void Cripto**(**int value**,** **out** string key**,** **out** int m**,** **out** int n**)**

**{**

//вычислить стартовый закрытый ключ

key **=** GenerateCloseKey**(**value**);**

m **=** **this.**m**;**

n **=** **this.**n**;**

//вычислить открытую последовательность ключ

\_openKey **=** GenerateOpenKey**(**key**,** n**,** m**);**

**}**

/// <summary>

/// Открытый ключ, внутренние операции

/// </summary>

**private** string \_openKey **{** get**;** set**;** **}**

/// <summary>

/// Cгенерировать открытый ключ

/// </summary>

/// <param name="closeKey">Закрытый ключ</param>

/// <param name="\_n">Взаимно простое</param>

/// <param name="\_m">Сумма++</param>

/// <returns></returns>

**private** string GenerateOpenKey**(**string closeKey**,** int \_n**,** int \_m**)**

**{**

var resCloseKey **=** ""**;**

**foreach** **(**string val **in** closeKey**.**Split**(**' '**))**

**{**

var tmpKey **=** 0**;**

**if** **(**int**.**TryParse**(**val**,** **out** tmpKey**))**

resCloseKey **+=** **(**tmpKey **\*** \_n **%** \_m**)** **+** " "**;**

**}**

**return** resCloseKey**;**

**}**

/// <summary>

/// Закрытый ключ

/// Cумма m=E++

/// Взаимно простое n

/// </summary>

**private** string GenerateCloseKey**(**int value**)**

**{**

Random rnd **=** **new** Random**();**

var sum **=** 2**;**

var result **=** ""**;**

**for** **(**int i **=** 0**;** i **<** value**;** i**++)**

**{**

var element **=** rnd**.**Next**(**sum**,** sum **+** 3**);**//2..7

sum **+=** element**;**

result **+=** **(**element **+** " "**);**

**if** **(**element **<** 0 **||** sum **+** value **<** 0**)** **throw** **new** ApplicationException**(**"Элемент ключа в минусе! " **+** element**);**

**}**

//Выбрать m>E(res)

m **=** rnd**.**Next**(**sum **+** 2**,** sum **+** value**);**

n **=** rnd**.**Next**(**value**,** value**+**value**);**

//Вычислить взаимно простое n

**for** **(**int i **=** 2**;** i **<** n**;** i**++)**

**{**

**if** **(**n **%** i **==** 0**)**

**if** **(**m **%** i **==** 0**)**/\*найдено взаимно простое\*/

**{**

n **=** rnd**.**Next**(**value**,** value **\*** i**);**

i **=** 1**;**

**}**

**}**

**return** result**;**

**}**

/// <summary>

/// m>E(res)

/// </summary>

**internal** int m **{** get**;** **private** set**;** **}**

/// <summary>

/// n - взаимно простое с m

/// </summary>

**internal** int n **{** get**;** **private** set**;** **}**

/// <summary>

/// Шифруем тексты

/// </summary>

/// <param name="text"></param>

**public** void Cripto**(**string text**,** **out** string result**,** int value**)**

**{**

result **=** ConvertToBin**(**text**,** value**);**

**}**

/// <summary>

/// Перевести текст в двоичный вид

/// </summary>

/// <param name="txtIn">Input string</param>

/// <param name="value">Размер блока</param>

**internal** string ConvertToBin**(**string txtIn**,** int value**)**

**{**

var sum **=** ""**;**

var sb **=** **new** StringBuilder**();**

//текст==>10100011 10010101 ...

**foreach** **(**byte b **in** Encoding**.**Unicode**.**GetBytes**(**txtIn**))**

sb**.**Append**(**Convert**.**ToString**(**b**,** 2**).**PadLeft**(**8**,** '0'**));**//.Append(' ');//0000\_100\_' '

//готовые бинарные блоки

var binaryStrArr **=** sb**.**ToString**();**

**for** **(**int i **=** 0**,** indx **=** 0**;** i **<** Math**.**Ceiling**((**double**)**binaryStrArr**.**Length **/** value**);** i**++,** indx**+=**value**)**

**{**

//дополнить блок до длины value

**if** **(**indx **+** value **>** binaryStrArr**.**Length**)** binaryStrArr **=** binaryStrArr**.**PadRight**(**indx **+** value**,** '0'**);**

var part **=** binaryStrArr**.**Substring**(**indx**,** value**);**

var index **=** 0**;**

var sumBlock **=** 0**;**

**foreach** **(**var ch **in** part**)**

**{**

**if** **(**ch **==** '1'**)**//присутствие

sumBlock **+=** int**.**Parse**(**\_openKey**.**Split**(**' '**)[**index**]);**

index**++;**

**}**

sum **+=** sumBlock **+** " "**;**

**}**

**return** sum**;**

**}**

**}**

В:

class clDeCropto

**{**

**public** clDeCropto**(){}**

**internal** string Decript**(**int n**,** int m**,** string codText**,** string CloseKey**)**

**{**

**if** **(**CloseKey**.**Length **<** 1 **||** n **==** **-**1 **||** m **==** **-**1 **||** codText**.**Length **<** 1**)** **return** ""**;**

var x **=** ReversN**(**n**,** m**);**

var bitRes **=** ""**;**

**foreach** **(**var s **in** codText**.**Split**(**' '**))**

**{**

var val **=** 0**;**

**if** **(!**int**.**TryParse**(**s**,** **out** val**))** **continue;**

val **=** **(**val**\***x**)%**m**;**//итоговое 174\*61 mod 105 = 9

var arrInt **=** **new** List**<**int**>();**

//берем числа закрытого ключа меньше val

**foreach** **(**var elemK **in** CloseKey**.**Split**(**' '**))**

**{**

var elem **=** 0**;**

**if** **(!**int**.**TryParse**(**elemK**,** **out** elem**))** **continue;**

**if(**elem **>** val**)** **break;**

arrInt**.**Add**(**elem**);**

**}**

//сумма чисел из массива должна равняться val

//val 9 = 3 + 6 arrInt

//если val = 0 то 0000.len = CloseKey.Split(' ').Len

var resListLinks **=** **new** List**<**string**>();**

var indx **=** arrInt**.**BinarySearch**(**val**);**

//2 5 10

**if(**indx**<**0**)** resListLinks **=** SearchSum**(**arrInt**,** val**);**

var des **=** resListLinks**.**Aggregate**(**""**,** **(**current**,** resListLink**)** **=>** current **+** **(**resListLink **+** " "**));**

bitRes **+=** BinBlocks**(**CloseKey**,** **(**indx **<** 0**)** **?** des **:** arrInt**[**indx**].**ToString**())** **+** " "**;**

**}**

**return** ConvertToText**(**bitRes**);**

**}**

/// <summary>

/// Перевести текст в двоичный вид

/// </summary>

/// <param name="txtIn"></param>

**internal** string ConvertToText**(**string binaryStr**)**

**{**

//готовые бинарные блоки "000100 100000 010000 "

//binaryStr = "00010010 00000100 00000000";

var newString **=** ""**;**

var remSpb **=** binaryStr**.**Replace**(**" "**,** ""**);**

const int value **=** 8**;**

**for** **(**int k **=** 0**,** indx **=** 0**;** k **<** Math**.**Ceiling**((**double**)**remSpb**.**Length**)** **/** value**;** k**++,** indx **+=** value**)**

**{**

//дополнить блок до длины value

**if** **(**indx **+** value **>** remSpb**.**Length**)** remSpb **=** remSpb**.**PadRight**(**indx **+** value**,** '0'**);**

var part **=** remSpb**.**Substring**(**indx**,** value**);**//0..8

newString **+=** part **+** " "**;**

**}**

//Процесс восстановление из бинарного в строку:

var sd **=** **new** byte**[**newString**.**Split**(**' '**).**Length**];**

int i **=** 0**;**

**foreach** **(**string s **in** newString**.**Split**(**' '**))**

**{**

**if** **(**s**.**Length **>** 0**)**

sd**[**i**]** **=** Convert**.**ToByte**(**s**,** 2**);**

i**++;**

**}**

**return** encUnic**.**GetString**(**sd**);**

**}**

**private** static Encoding encUnic **=** Encoding**.**Unicode**;**

/// <summary>

/// Бинарные блоки

/// </summary>

/// <param name="CloseKey"></param>

/// <param name="block"></param>

/// <returns></returns>

**private** string BinBlocks**(**string CloseKey**,** string block**)**

**{**

var sumBlock **=** ""**;**

int index **=** 0**;**

var el **=** ""**;**

**{**//5

//когда indx указывает на ячейку с значением в ключе

//2 4 5 6.. indx=5 ==> 0 0 1 0 ..

**foreach** **(**var ch **in** CloseKey**.**Split**(**' '**))**//4 7 15 28 57 114 229

**{**

**if(**ch**==**""**)continue;**

**if** **(**index **<** block**.**Split**(**' '**).**Length**)** el **=** block**.**Split**(**' '**)[**index**];**

**if** **(**ch **==** el**)**//присутствие

**{**

sumBlock **+=** "1"**;**

index**++;**

**}**

**else** sumBlock **+=** "0"**;**

**}**

**}**

**return** sumBlock**;**

**}**

/// <summary>

/// Найти элементы ключа, дающие в сумме значение value

/// </summary>

/// <param name="arr"></param>

/// /// <param name="value"></param>

/// <returns>Вернет последовательности</returns>

**private** List**<**string**>** SearchSum**(**List**<**int**>** arr**,** int value**)**

**{**

//arr = new List<int>() { 2, 3, 6, 13, 27, 52, 105, 210 };

var links **=** **new** List**<**string**>();**//для элементов

//var result = new List<int>();//для сумм

var razn **=** value**;**

//берем самое крайнее число - оно и самое большое

**for** **(**int i **=** arr**.**Count **-** 1**;** i **>=** 0**;** i**--)**

**{**

var elem **=** arr**[**i**];**

**if** **(**elem **>** razn**)** **continue;**

links**.**Add**(**elem**.**ToString**());**//ложим первым самый большой

razn **=** razn **-** elem**;**//204-113

//result.Add(razn);

**if** **(**razn **==** 0**)**

**{**//Значения с большего к меньшему необходимо переставить

links**.**Reverse**();**

//значения от меньшего к большему

**return** links**;**

**}**

**}**

//Иначе не сходится!На нулях например

**return** links**;**

**}**

/// <summary>

/// Подобрать обратное n число

/// </summary>

/// <param name="n">31</param>

/// <param name="m">105</param>

/// <returns></returns>

**private** int ReversN**(**int n**,** int m**)**

**{**

var X **=** **-**1**;**

//подобрать обратное число n

**for** **(**int i **=** 1**;** i **<** m/\*int.MaxValue\*/**;** i**++)**

**{**

**if** **((**n **\*** i**)** **%** m **!=** 1**)** **continue;**

X **=** i**;**

**break;**

**}**

**if** **(**X **==** **-**1**)** **new** Exception**(**"Не подобрано обратное n число!"**);**

**return** X**;**

**}**

**}**