Томский Государственный Университет

Институт Прикладной математики и

Компьютерных Наук

**Отчет**

О самостоятельной работе по введению в специальность – криптографию

По теме:

«Knapsack Cryptosystem»

Автор: Студент 1155 группы

Аламов Владимир Александрович

Проверил: Агибалов Геннадий Петрович

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

Томск – 2017

## Содержание

[Введение 3](#_gjdgxs)

[Общее описание шифра 4](#_30j0zll)

[Математическая постановка задачи 6](#_1fob9te)

[Программная реализация 9](#_3znysh7)

[Функция и программа расшифрования 12](#_2et92p0)

[Полный код программы шифрования: 14](#_tyjcwt)

[Полный код программы расшифрования: 18](#_1t3h5sf)

[Литература 22](#_4d34og8)

## Введение

Целью данной работы является самостоятельное изучение и реализация «рюкзачного шифра».

Задачи:

1. Самостоятельно изучить алгоритмы шифрования и расшифрования
2. Написать программы реализации шифрования и расшифрования

Что удалось реализовать:

1. Написаны программы шифрования и расшифрования

Используемый язык программирования : LYaPAS

## Общее описание шифра

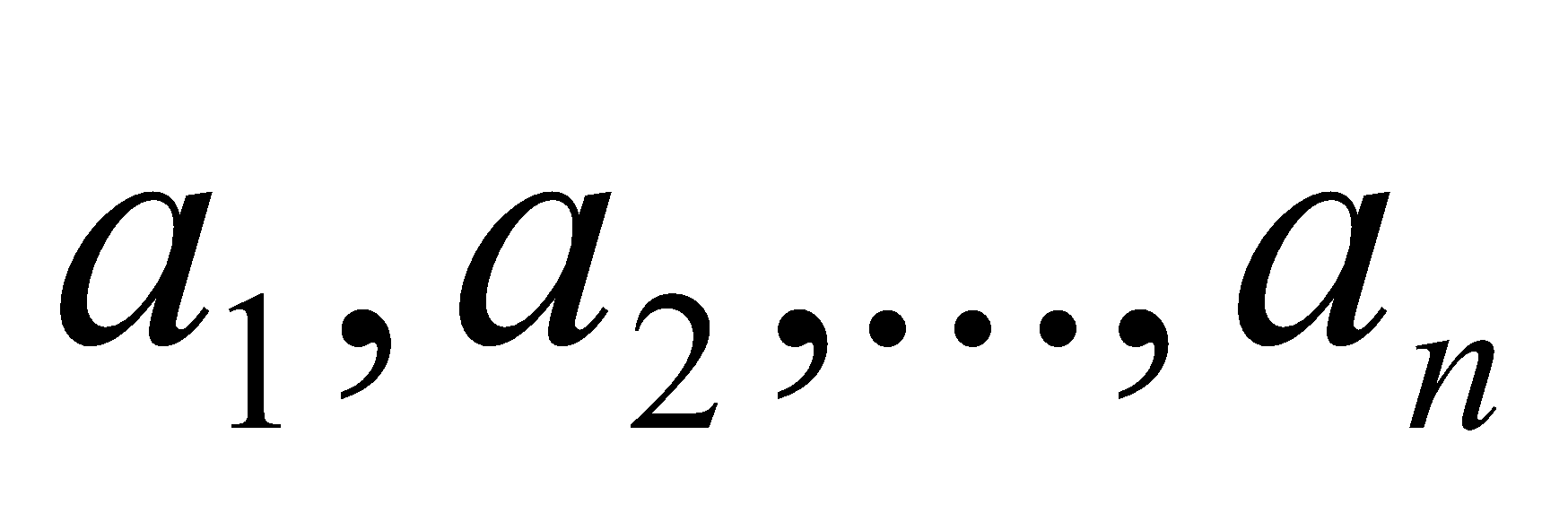
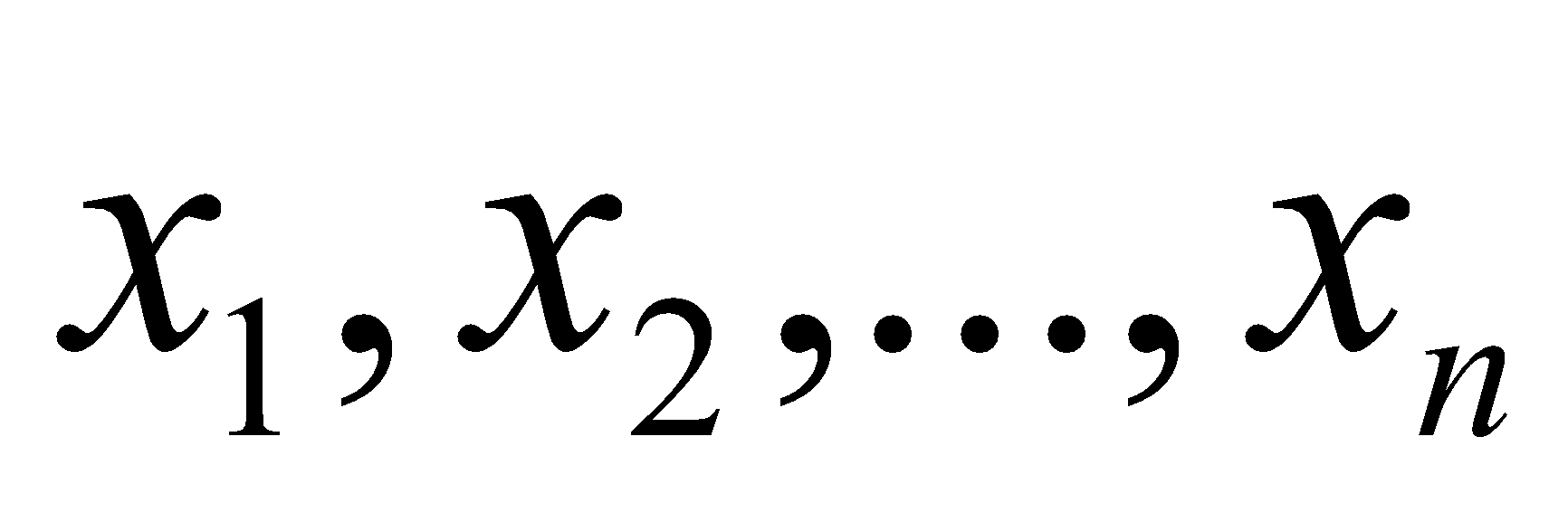
Начало асимметричным шифрам было положено в 1976 г. в работе У. Диффи и М. Хеллмана «Новые направления в современной криптографии». Криптографическая система с открытым ключом (или асимметричное шифрование, асимметричный шифр) — система шифрования, при которой открытый ключ передаётся по открытому каналу и используется для шифрования сообщения. Для расшифровки сообщения используется секретный ключ. Криптографические системы с открытым ключом в настоящее время широко применяются в различных сетевых протоколах и стандартах цифровой подписи. Для построения криптосистемы с открытым ключом выбирается класс задач, для которого в произвольном случае не известен эффективный алгоритм решения, и в этом классе выделяется подзадача, для которой такой алгоритм существует. Выбранную задачу маскируют под задачу общего вида и на основе ее выбирают ключ шифрования. В качестве секретного ключа используется информация, позволяющая перевести выбранную задачу в исходный вид.

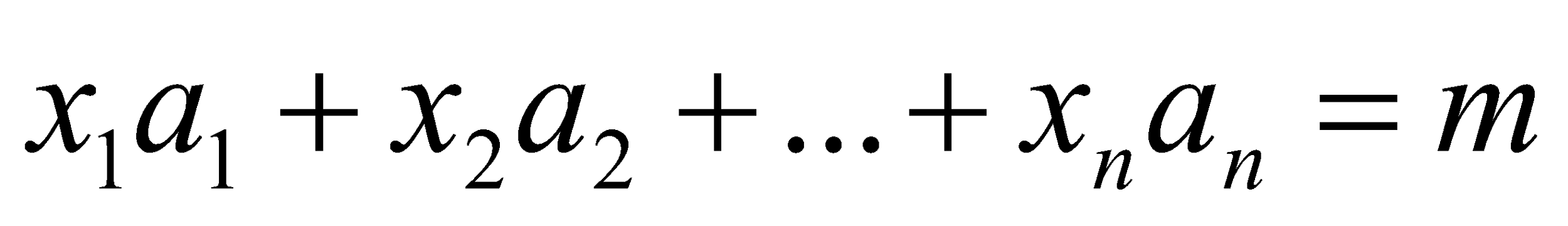
Наиболее распространенными в настоящее время являются криптосистемы, основанные на задаче факторизации (RSA) и задаче нахождения дискретного логарифма (схема Эль-Гамаля). К задачам, не связанным с проблемой факторизации, относится и задача об укладке рюкзака, являющаяся NP-полной. На ее основе был разработан ряд криптосистем, отличающихся простотой реализации. Но в ходе их анализа были выявлены существенные недостатки, делающие эти системы уязвимыми для различного вида криптографических атак. В настоящее время системы этого класса не получили широкого распространения, но ведется работа по их модификации, которая позволит улучшить их надежность. Одним из способов такой модификации является использование специальных порождающих множеств в конечных группах для создания рюкзачной схемы.

Первый алгоритм для шифрования на основе задачи о рюкзаке был разработан Мерклом и Хеллманом в 1978 году и получил название «[Алгоритм Меркла-Хеллмана](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%BD%D1%86%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%9C%D0%B5%D1%80%D0%BA%D0%BB%D1%8F_%E2%80%94_%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0)». Он был опубликован в одностадийном ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) singly-iterated) и мультистадийном вариантах ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) multiply-iterated). Алгоритм мог быть использован только для шифрования, но израильский криптоаналитик [Ади Шамир](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%BC%D0%B8%D1%80,_%D0%90%D0%B4%D0%B8) адаптировал его для использования в [цифровых подписях](https://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%86%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%B4%D0%BF%D0%B8%D1%81%D1%8C). После опубликования схемы Меркл предложил вознаграждение в 100 [долларов](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BB%D0%BB%D0%B0%D1%80_%D0%A1%D0%A8%D0%90) тому, кто удачно осуществит взлом одностадийного алгоритма. В 1982 году Шамир осуществил успешную атаку и получил обещанное вознаграждение. Но даже после его уплаты Меркл был уверен в криптостойкости мультистадийной системы и предложил 1000 [долларов](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BB%D0%BB%D0%B0%D1%80_%D0%A1%D0%A8%D0%90) в случае её удачного взлома. В 1984 году американский математик [Эрнест Брикелл](https://ru.wikipedia.org/w/index.php?title=%D0%AD%D1%80%D0%BD%D0%B5%D1%81%D1%82_%D0%91%D1%80%D0%B8%D0%BA%D0%B5%D0%BB%D0%BB&action=edit&redlink=1) сумел осуществить взлом для сорока-стадийного варианта чуть более чем за 1 час на машине [Cray-1](https://ru.wikipedia.org/wiki/Cray-1). [1]

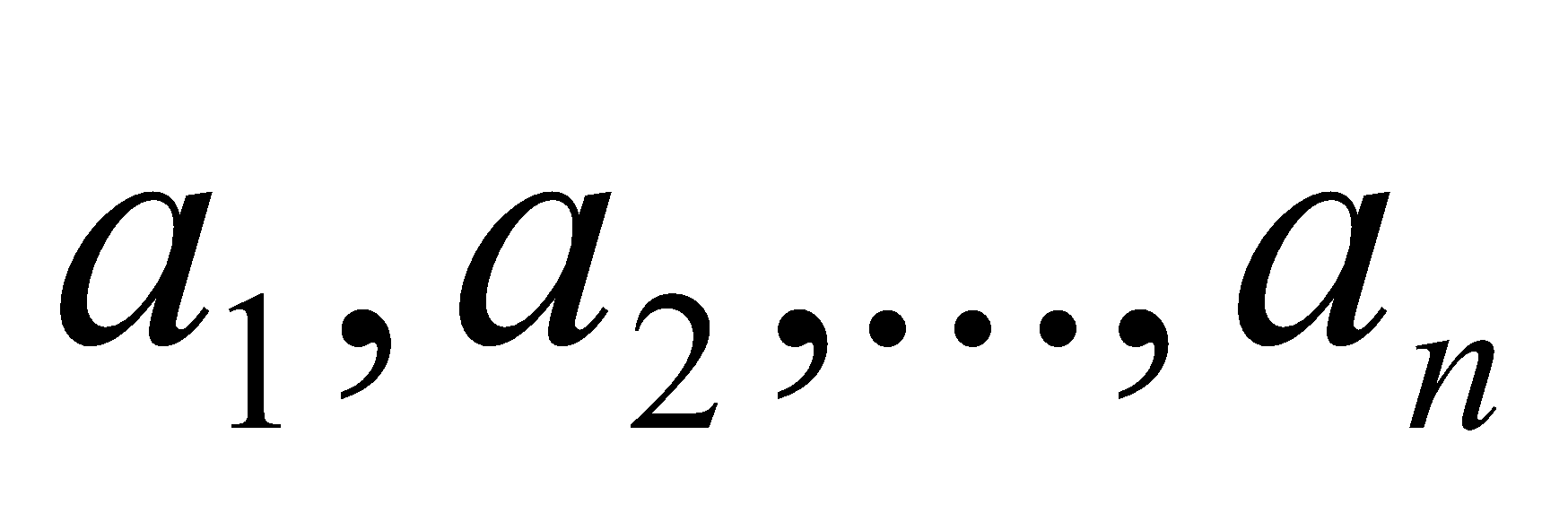
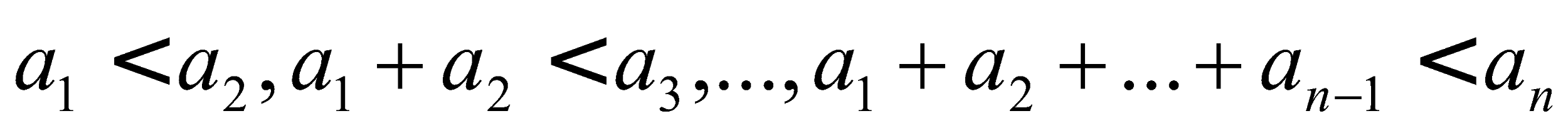
## Математическая постановка задачи

Задача о рюкзаке.

Имеется упорядоченный набор чисел состоящий из n элементов (), который мы будем называть рюкзаком или рюкзачным вектором, и число m. Нам необходим такой булев вектор () для которого выполняется равенство:

 .

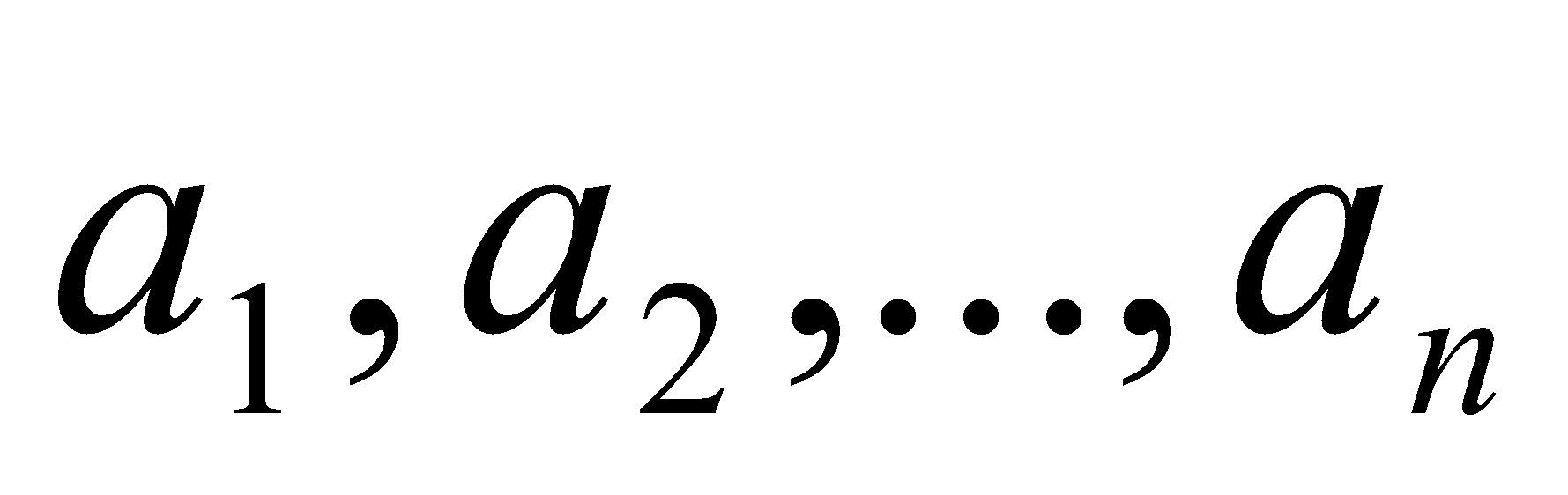
В общем случае для данной задачи нет чёткого решения, и время затрачиваемое на поиск решения растёт в экспоненциальной зависимости от количества элементов. Но эту проблему можно обойти, если рюкзак будет сверхрастущим. Тогда существует решение и оно единственное.

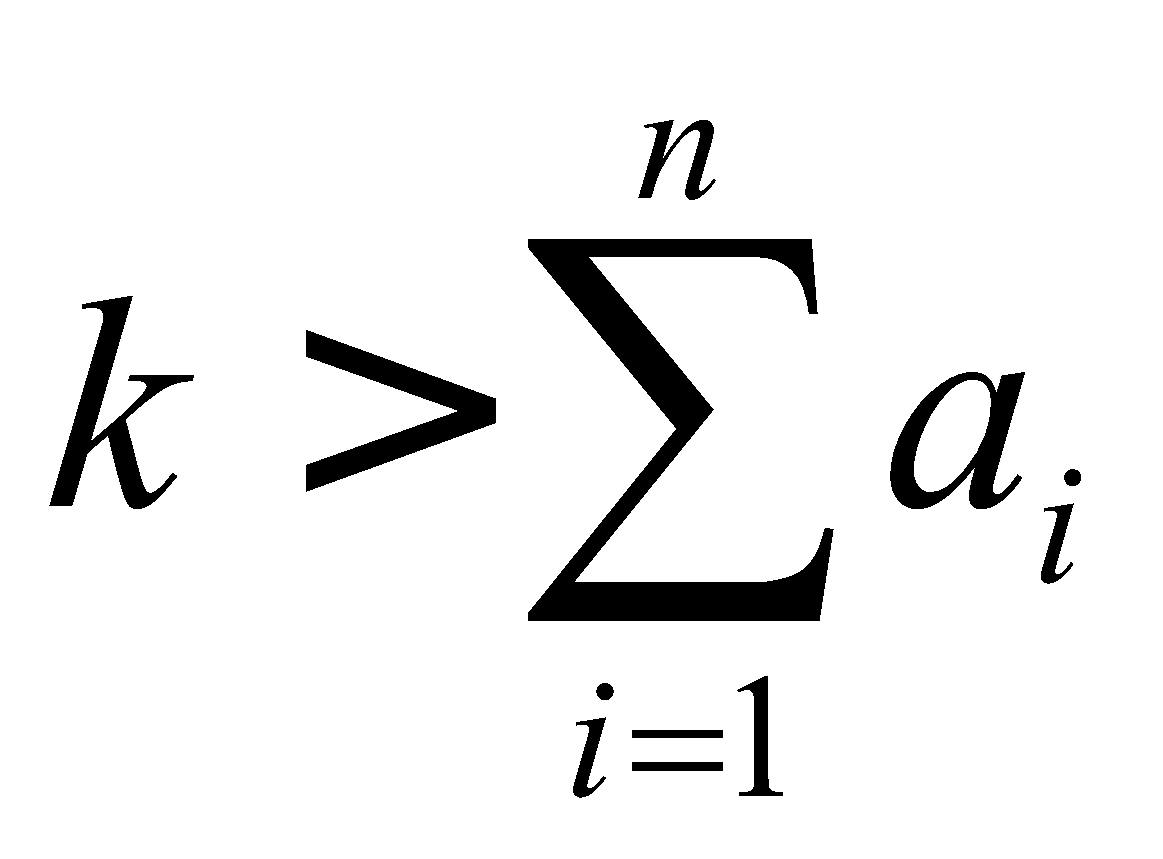
Рюкзак с положительными элементами () будем называть сверхрастущим, если каждый последующий элемент больше суммы предыдущих 

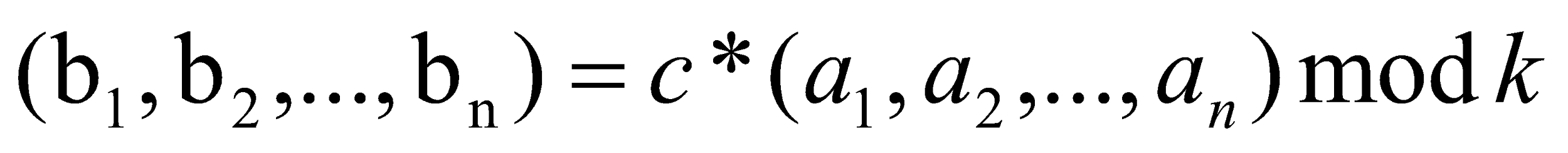
Как известно, отличительным свойством ассиметричных криптосистем является наличие двух ключей-открытого и закрытого. Основным отличием данной криптосистемы является её решение проблемы перехвата злоумышленником открытого ключа. Он получается из закрытого ключа с помощью легких преобразований.

*Алгоритм генерации ключей:*

1. Выбирается сверхрастущая последовательность .

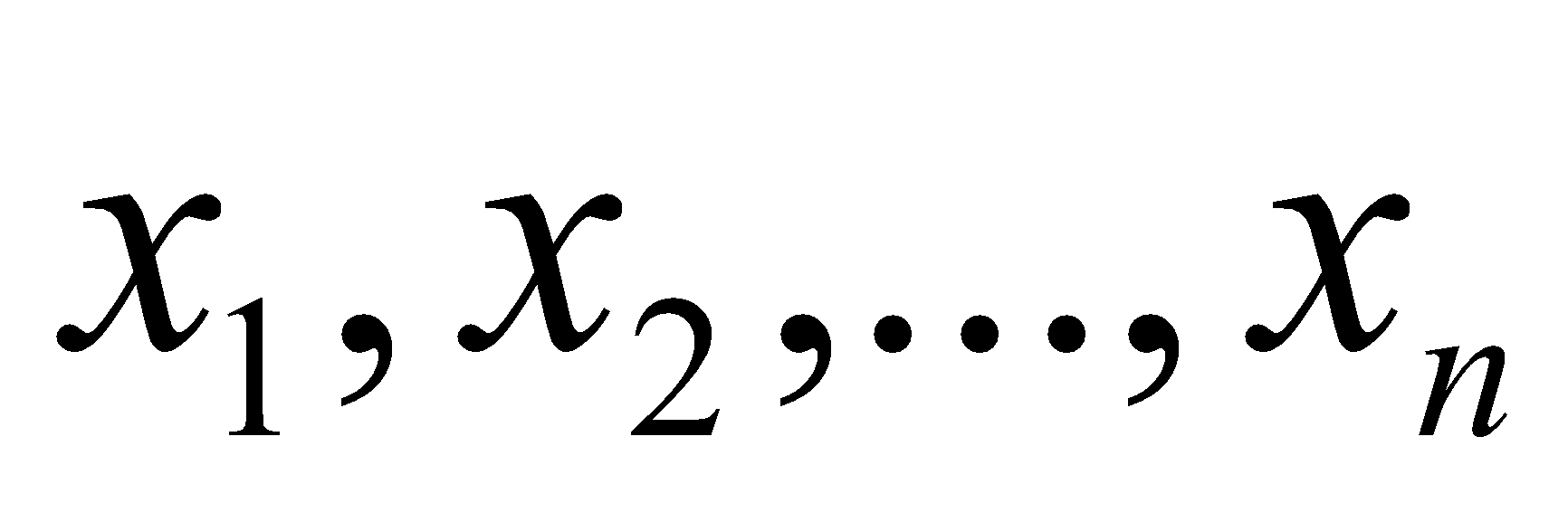
().

1. Выбираем число .
2. Выбираем число с, взаимно простое с k.
3. Формируем открытый ключ почленным умножением по модулю k рюкзачного вектора a на число с

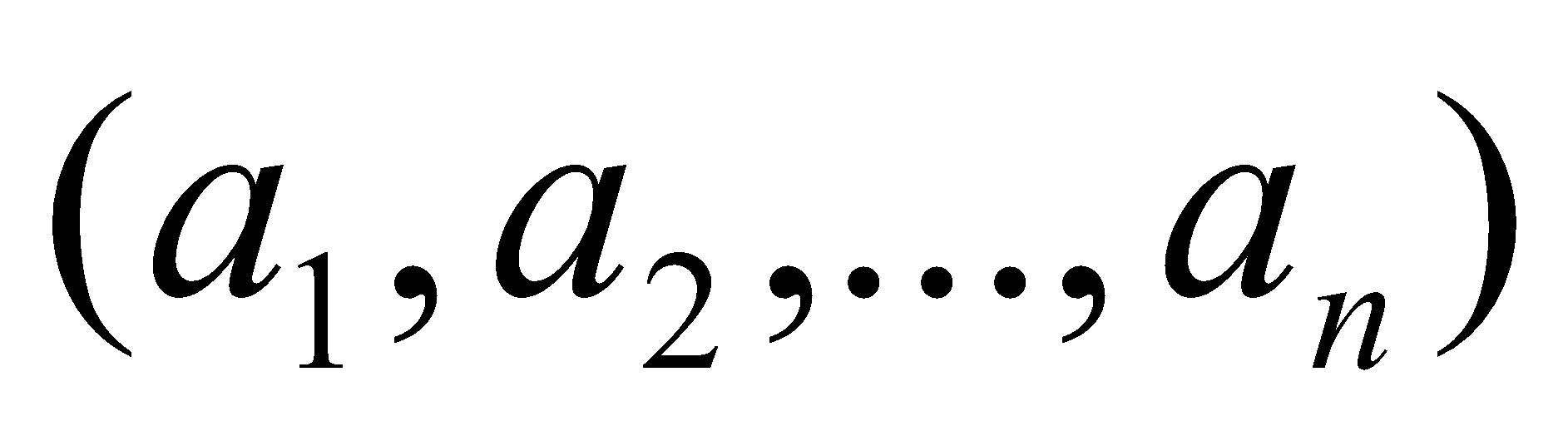
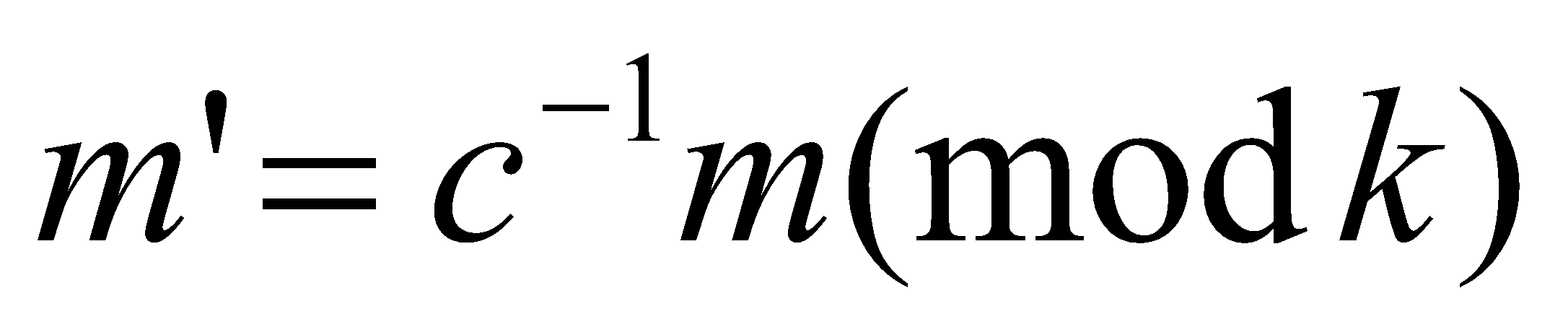
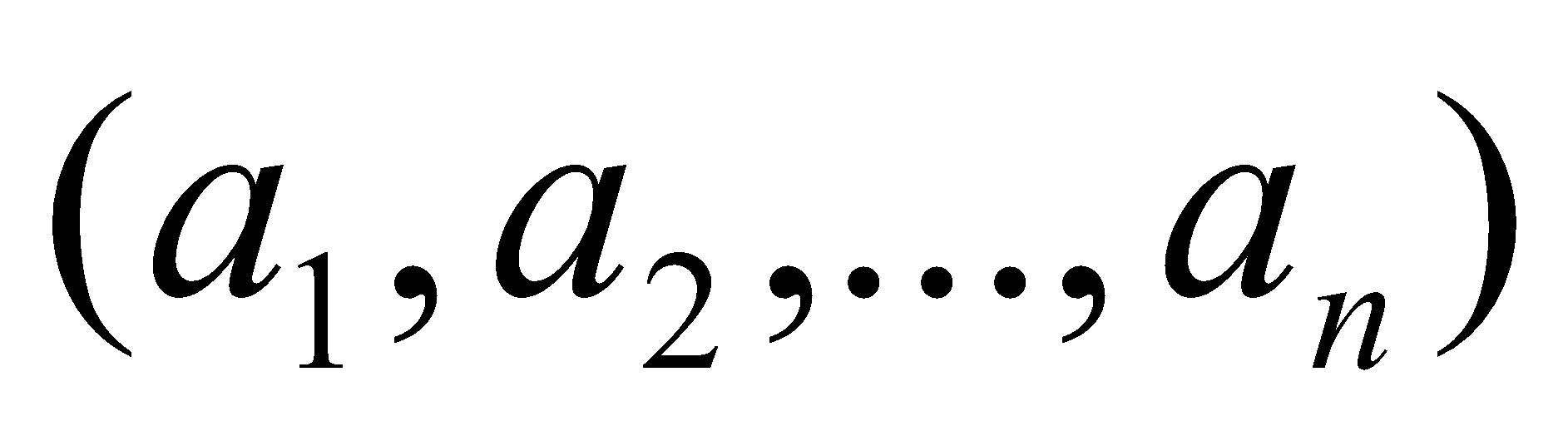
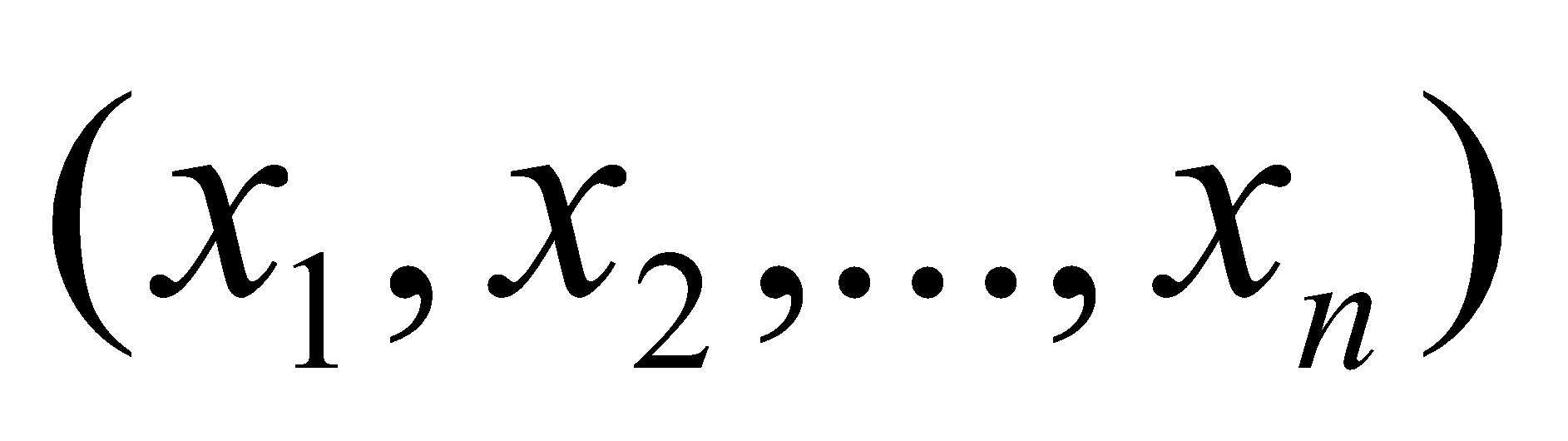


1. Числа c и k являются секретными ключами.

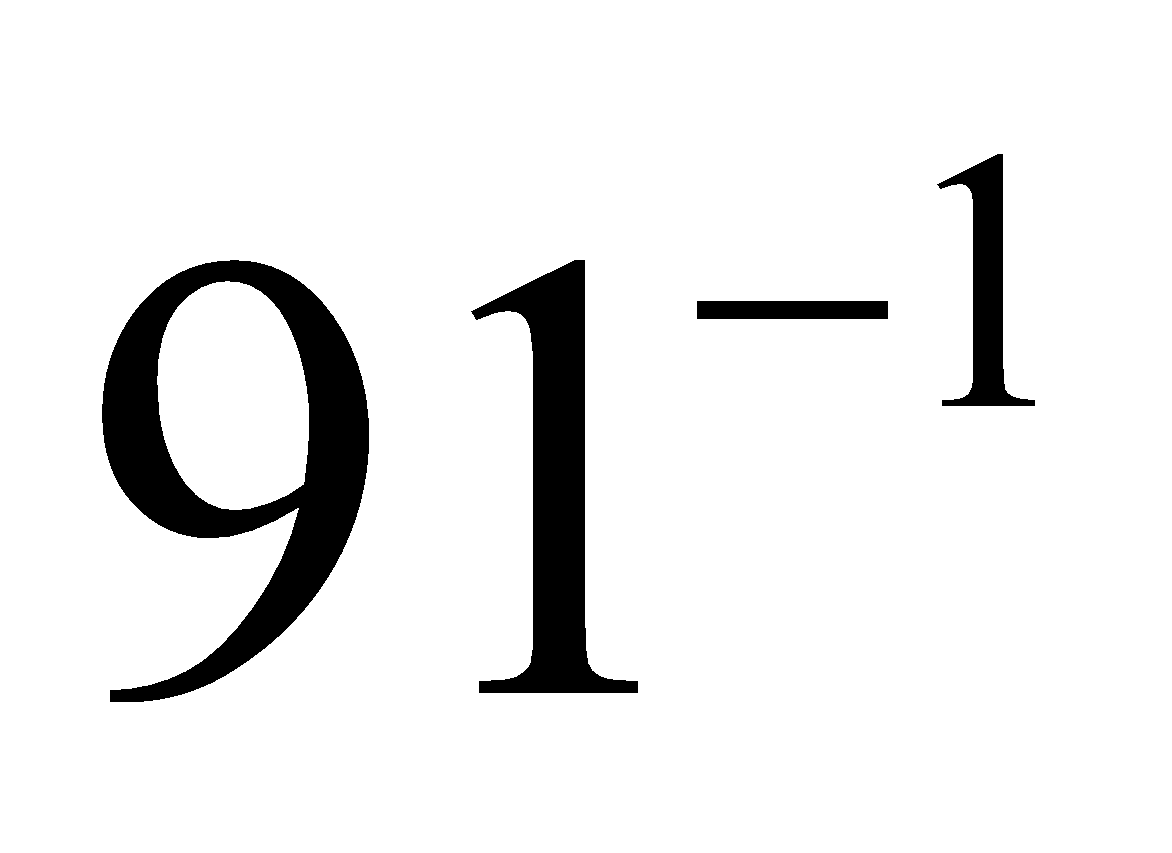
*Алгоритм шифрования:*

1. Открытый текст представляется в виде двоичной последовательности.
2. Последовательность разбивается на блоки длины n, так как это блочный шифр.
3. Каждый блок () заменяется на число m вычисленное по правилу

*Алгоритм расшифрования:*

1. Необходимо найти начальный сверхрастущий вектор: 
2. Для каждого элемента m шифрованного текста вычисляется элемент 
3. Для m’ решается задача о рюкзаке для рюкзака  и находится блок открытого текста 

Пример 1. Используется латинский алфавит, в котором каждая буква представлена пятиразрядной двоичной записью своего номера. Рюкзак-ловушка B=(182,128,192,175,50,100) получен из сверхрастущего рюкзака A путём умножения на c=91 и приведением по модулю n=300. Сообщение Y=(232,178,502) получено шифрованием на основе рюкзака B. Восстановить исходный рюкзак A и, используя его, расшифровать сообщение Y. Решение:

1) Найдем (mod300): ==211.

2) Восстановим исходный рюкзак: A=211B (mod300)= =211(182,128,192,175,50,100)(mod300)=(2,8,12,25,50,100).

3) Преобразуем сообщение Y:Y→Z=211Y(mod300)=211(232,178,502)(mod300)=(52,58,22).

4) Решим задачу о рюкзаке для каждого элемента сообщения Z: 52=50+2→(1, 0, 0, 0, 1, 0), 58=50+8→(0, 1, 0, 0, 1, 0), 22=12+10=12+8+2→(1, 1, 1, 0, 0, 0).

5) Запишем полученные двоичные векторы в единую последовательность, которую разобьём на блоки длины 5:

(1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0) →(1, 0, 0, 0, 1), (0, 0, 1, 0, 0), (1, 0, 1, 1, 1), (0, 0, 0). Последний блок из трёх элементов исключим из рассмотрения.

6) Сопоставим каждому полученному блоку число, для которого этот блок является двоичной записью, и найдём соответствующую букву латинского алфавита:

(1, 0, 0, 0, 1)17→R,

(0, 0, 1, 0, 0) 4→E,

(1, 0, 1, 1, 1) 23→X.

## Программная реализация

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*head(/)*

*T⇒X*

*/'Введите ограничение разности между соседними элементами: '>C*

*@+F1(10) OQ1*

*/F1<C*

*\*s2n(F1/m)*

*OQ1 @-F1*

*@+L1(8) OQ1*

*\*gensrp(8,m/L1,s)*

*@+L2(8) 8⇒Q2*

*\*genkey(L1,s/L2,q,r)*

*/'Открытый ключ: '>C*

*\*skobki(L2/)*

*/'Закрытый ключ: '>C*

*\*skobki(L1/)*

*/'q='>C \*print(q/) /'\nr='>C \*print(r/)*

*@+F3(32768) OQ3*

*/'\nВведите имя файла для шифрования: \n'>C /F3<C*

*\*fopen(F3,0/n)*

*\*freadf(n,F3,0,32768/k)*

*\*fclose(n/)*

*k⇒Q3*

*@+L4(k) Q3⇒Q4*

*\*encrypt(L2,F3/L4)*

*/'Зашифрованное сообщение:\n'>C*

*\*skobki(L4/)*

*\*\**

Особенности:

- Мы вводим разность между элементами для того, чтобы разброс в случайно сгенерированной последовательности не был слишком большим

- Шифруемый текст должен представлять из себя буквы латинского алфавита

- Мы используем 4-х байтовые массивы для больших чисел

- Значение модуля и числа выбранного нами для шифрование должны быть взаимно простыми

**Вспомогательная функция для перевода строки цифр в единое число:**

*s2n(F1/a)*

*Oa⇒i*

*§1 a\*10+F1i-'0'⇒a ∆i ↑(i<Q1)1*

*\*\**

**Функция генерации СРП:**

*gensrp(n,m/L1,s)*

*Os⇒i*

*§1 X;m+1+s⇒L1i ∆Q1 L1i+s⇒s ∆i ↑(i<n)1*

*\*\**

*На вход программы подается: n-число элементов последовательности и m-максимальная разность между элементами.*

*На выходе мы получаем: L1-массив 4-х байтовых элементов, который является сверхрастущей последовательностью или рюкзачным вектором, и s-сумма всех элементов рюкзачного вектора.*

Для наглядности я буду писать результаты, которые выдала программа во время выполнения.

Пример:

m=23, n=8

Функция сгенерировала (12 15 41 81 155 305 610 1227), сумма всех элементов равна 2446.

**Функция генерации открытого ключа:**

*genkey(L1,s/L2,q,r)*

*/'Сумма элементов равна '>C \*print(s/)*

*/'\nВведите число, большее суммы всех элементов: q='>C*

*@+F3(10) OQ3 /F3<C \*s2n(F3/q)*

*§1/'\nВведите число, взаимнопростое с q, r= '>C*

*OQ3 /F3<C \*s2n(F3/r)*

*\*nod(q,r/d) ↑(d≠1)1*

*OQ3 @-F3 Oi*

*§2 L1i\*r;q⇒L2i ∆i ↑(i<Q1)2*

*\*\**

*На вход программы генерации открытого ключа подается L1-закрытый ключ и s-в данном случае число, большее суммы всех элементов закрытого ключа.*

*На выходе мы получаем: L2-открытый ключ, q-число большее суммы всех элементов закрытого ключа и r-число взаимно простое с q.*

В нашем примере мы подаем L1=(12 15 41 81 155 305 610 1227) и s=2446. Теперь нам необходимо ввести q, возьмем его равным 3591. И выберем с равный 331, в самой функции есть проверка,реализуемая через Алгоритм Евклида.

Таким образом, L2=(L1\*331)mod(3591)=(381 1374 2798 1674 1031 407 814 354), q=3591,c=331.

**Функция шифрования и вспомогательная функция для вычисления суммы, которая является зашифрованным элементом:**

*calcsum(L1,a/s)*

*Q1-1⇒j Os*

*§1 a&1\*L1j+s⇒s a>1⇒a ∇ j a↦1*

*\*\**

*encrypt(L1,F2/L3)*

*Oi*

*§1 \*calcsum(L1,F2i/s) s⇒L3i ∆i ↑(i<Q2)1*

*\*\**

*На вход подпрограммы encrypt поступает: L1-открытый ключ, F2-бинарное представление текста. Внутри подпрограммы вызывается функция calcsum, которой на вход идет: L1-открытый ключ, F2i-бинарное представление символа. На выходе мы получаем s-сумму элементов открытого ключа, на не нулевых позициях бинарного представления символа.*

*Функция encrypt возвращает нам L3-зашифрованный текст.*

L1=(381 1374 2798 1674 1031 407 814 354),

Текст: Welcome to Warhammer.

Зашифрованное сообщение: (4623 4933 5610)

**Алгоритм Евклида, реализованный вычитанием:**

*nod(a,b/d)*

*§1 ↑(a>b)2 a⇒c b⇒a c⇒b*

*§2 a-b⇒a ↑(a=b)3 →1*

*§3 a⇒d*

*\*\**

## 

## Функция и программа расшифрования

*head(/)*

*@+L1(8) OQ1*

*@+F2(131072)*

*/'Введите закрытый ключ через пробел:\n'>C*

*OQ2 /F2<C \*F2L(F2/L1)*

*\*skobki(L1/)*

*/'Введите множитель: r='>C*

*OQ2 /F2<C \*s2n(F2/r)*

*/'Введите модуль: q= '>C*

*OQ2 /F2<C \*s2n(F2/q)*

*\*obratnoe(r,q/b)*

*/'Обратный по умножению к '>C \*print\_value(r/) /' по модулю '>C \*print\_value(q/) /' равен: b='>C \*print\_value(b/) /'\n'>C*

*/'Введите зашифрованное сообщение \n'>C*

*OQ2 /F2<C*

*@+L3(131072) OQ3*

*\*F2L(F2/L3)*

*\*skobki(L3/)*

*OQ2*

*\*decrypt(L1,L3,b,q/F2)*

*@+F4(32768)*

*/'Введите имя файла для записи расшифрованного текста: \n'>C OQ4 /F4<C*

*\*fopen(F4,1/n)*

*\*fwritef(n,F2,0,Q2/)*

*\*fclose(n/)*

*/F2>C /'\n'>C*

*\*\**

**Функция расшифрования и вспомогательные функции: нахождения обратного элемента и булевого вектора, соответствующего расшифрованному символу**

*decrypt(L1,L2,r,q/F3)*

*Oi*

*§1 L2i\*r;q⇒a \*findvect(a,L1/b) b@>F3 ∆i ↑(i<Q2)1*

*\*\**

*obratnoe(a,b/c)*

*1⇒c*

*§1 ∆c a\*c;b⇒d ↑(d≠1)1 \*\**

*findvect(a,L1/b)*

*Ob*

*§1 Os Q1-1⇒j ∆b⇒c*

*§2 c&1\*L1j+s⇒s c>1⇒c ∇ j c↦2 ↑(s≠a)1*

*\*\**

Так как мы используем попеременно 4-х байтовые массивы, необходима функция перевода строки цифр в элементы 4-х байтового массива

*F2L(F1/L2)*

*Oi Oj*

*§1 ↑(i≥Q1)3 ↑(F1i≠' ')2 ∆i →1*

*§2 \*fins(F1,i/a,i) a⇒L2j ∆Q2 ∆j ∆i →1*

*§3 \*\**

**Вспомогательная функция для забора очередной строки цифр до пробела и перевод их в 4-х байтовую переменную**

*fins(F1,i/a,i)*

*Oa*

*§1 a\*10+F1i-'0'⇒a ∆i ↑(i=Q1)2 ↑(F1i=' ')2 →1*

*§2 \*\**

## 

## ***Полный код программы шифрования***:

head(/)

T⇒X

/'Введите ограничение разности между соседними элементами: '>C

@+F1(10) OQ1

/F1<C

\*s2n(F1/m)

OQ1 @-F1

@+L1(8) OQ1

\*gensrp(8,m/L1,s)

@+L2(8) 8⇒Q2

\*genkey(L1,s/L2,q,r)

/'Открытый ключ: '>C

\*skobki(L2/)

/'Закрытый ключ: '>C

\*skobki(L1/)

/'q='>C \*print(q/) /'\nr='>C \*print(r/)

@+F3(32768) OQ3

/'\nВведите имя файла для шифрования: \n'>C /F3<C

\*fopen(F3,0/n)

\*freadf(n,F3,0,32768/k)

\*fclose(n/)

k⇒Q3

\*\*\* /'\nВведите текст:\n'>C

\*\*\*@+F3(32768) OQ3

\*\*\* /F3<C Q3⇒m

@+L4(m) Q3⇒Q4

\*encrypt(L2,F3/L4)

/'Зашифрованное сообщение:\n'>C

\*skobki(L4/)

\*\*

s2n(F1/a)

Oa⇒i

§1 a\*10+F1i-'0'⇒a ∆i ↑(i<Q1)1

\*\*

gensrp(n,m/L1,s)

Os⇒i

§1 X;m+1+s⇒L1i ∆Q1 L1i+s⇒s ∆i ↑(i<n)1

\*\*

print(a/)

@+F1(10) OQ1

§1 a;10+'0'@>F1.0 a/10⇒a↦1 /F1>C OQ1 @-F1

\*\*

skobki(L1/)

Oi

/'('>C

§1 \*print(L1i/) ∆i ↑(i=Q1)2 /' '>C →1

§2 /')\n'>C

\*\*

nod(a,b/d)

§1 ↑(a>b)2 a⇒c b⇒a c⇒b

§2 a-b⇒a ↑(a=b)3 →1

§3 a⇒d

\*\*

genkey(L1,s/L2,q,r)

/'Сумма элементов равна '>C \*print(s/)

/'\nВведите число, большее суммы всех элементов: q='>C

@+F3(10) OQ3 /F3<C \*s2n(F3/q)

§1/'\nВведите число, взаимнопростое с q, r= '>C

OQ3 /F3<C \*s2n(F3/r)

\*nod(q,r/d) ↑(d≠1)1

OQ3 @-F3 Oi

§2 L1i\*r;q⇒L2i ∆i ↑(i<Q1)2

\*\*

calcsum(L1,a/s)

Q1-1⇒j Os

§1 a&1\*L1j+s⇒s a>1⇒a ∇j a↦1

\*\*

encrypt(L1,F2/L3)

Oi

§1 \*calcsum(L1,F2i/s) s⇒L3i ∆i ↑(i<Q2)1

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* работа с файлами

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Открытие файла

\*\*\* F1 - имя файла (в текущем каталоге)

\*\*\* r - режим открытия: 0 - чтение, 1 - запись, 2 - чтение/запись

\*\*\* n - дескриптор файла (<0 при отказе)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fopen(F1,r/n)

0@>F1

{mov eax,5}

{mov ebx,[ebp+220]}

{mov ecx,[ebp+72]}

{int 80h}

{mov [ebp+56],eax}

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Закрытие файла

\*\*\* n - дескриптор файла

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fclose(n/)

{mov eax,6}

{mov ebx,[ebp+56]}

{int 80h}

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Чтение из файла в символьный комплекс

\*\*\* n - дескриптор файла

\*\*\* F1 - файл

\*\*\* b - адрес в F1 первого считанного байта

\*\*\* l - число запрашиваемых байт

\*\*\* k - число считанных байтов (код ошибки для <0)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

freadf(n,F1,b,l/k)

{mov eax,3}

{mov ebx,[ebp+56]}

{mov ecx,[ebp+220]}

{add ecx,[ebp+8]}

{mov edx,[ebp+48]}

{int 80h}

{mov [ebp+44],eax}

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Запись из символьного комплекса в файл

\*\*\* n - дескриптор файла

\*\*\* F1 - файл

\*\*\* b - адрес в F1 первого записываемого байта

\*\*\* l - число записывемых байт

\*\*\* k - число записанных байтов (код ошибки для <0)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fwritef(n,F1,b,l/)

{mov eax,4}

{mov ebx,[ebp+56]}

{mov ecx,[ebp+220]}

{add ecx,[ebp+8]}

{mov edx,[ebp+48]}

{int 80h}

{mov [ebp+44],eax}

\*\*

## Полный код программы расшифрования:

head(/)

@+L1(8) OQ1

@+F2(131072)

/'Введите закрытый ключ через пробел:\n'>C

OQ2 /F2<C \*F2L(F2/L1)

\*skobki(L1/)

/'Введите множитель: r='>C

OQ2 /F2<C \*s2n(F2/r)

/'Введите модуль: q= '>C

OQ2 /F2<C \*s2n(F2/q)

\*obratnoe(r,q/b)

/'Обратный по умножению к '>C \*print\_value(r/) /' по модулю '>C \*print\_value(q/) /' равен: b='>C \*print\_value(b/) /'\n'>C

/'Введите зашифрованное сообщение \n'>C

OQ2 /F2<C

@+L3(131072) OQ3

\*F2L(F2/L3)

\*skobki(L3/)

OQ2

\*decrypt(L1,L3,b,q/F2)

\*\*\*/'++'>C

@+F4(32768)

/'Введите имя файла для записи расшифрованного текста: \n'>C OQ4 /F4<C

\*fopen(F4,1/n)

\*fwritef(n,F2,0,Q2/)

\*fclose(n/)

/F2>C /'\n'>C

\*\*

F2L(F1/L2)

Oi Oj

§1 ↑(i≥Q1)3 ↑(F1i≠' ')2 ∆i →1

§2 \*fins(F1,i/a,i) a⇒L2j ∆Q2 ∆j ∆i →1

§3 \*\*

fins(F1,i/a,i)

Oa

§1 a\*10+F1i-'0'⇒a ∆i ↑(i=Q1)2 ↑(F1i=' ')2 →1

§2 \*\*

skobki(L1/)

Oi

/'('>C

§1 \*print\_value(L1i/) ∆i ↑(i=Q1)2 /' '>C →1

§2 /')\n'>C

\*\*

s2n(F1/a)

Oa⇒i

§1 a\*10+F1i-'0'⇒a ∆i ↑(i<Q1)1

\*\*

print\_value(a/)

@+F1(10) OQ1

§1 a;10+'0'@>F1.0 a/10⇒a ↑(a≠0)1 /F1>C OQ1 @-F1

\*\*

findvect(a,L1/b)

Ob

§1 Os Q1-1⇒j ∆b⇒c

§2 c&1\*L1j+s⇒s c>1⇒c ∇j c↦2 ↑(s≠a)1

\*\*

obratnoe(a,b/c)

1⇒c

§1 ∆c a\*c;b⇒d ↑(d≠1)1

\*\*

decrypt(L1,L2,r,q/F3)

Oi

§1 L2i\*r;q⇒a \*findvect(a,L1/b) b@>F3 ∆i ↑(i<Q2)1

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* работа с файлами

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Открытие файла

\*\*\* F1 - имя файла (в текущем каталоге)

\*\*\* r - режим открытия: 0 - чтение, 1 - запись, 2 - чтение/запись

\*\*\* n - дескриптор файла (<0 при отказе)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fopen(F1,r/n)

0@>F1

{mov eax,5}

{mov ebx,[ebp+220]}

{mov ecx,[ebp+72]}

{int 80h}

{mov [ebp+56],eax}

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Закрытие файла

\*\*\* n - дескриптор файла

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fclose(n/)

{mov eax,6}

{mov ebx,[ebp+56]}

{int 80h}

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Чтение из файла в символьный комплекс

\*\*\* n - дескриптор файла

\*\*\* F1 - файл

\*\*\* b - адрес в F1 первого считанного байта

\*\*\* l - число запрашиваемых байт

\*\*\* k - число считанных байтов (код ошибки для <0)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

freadf(n,F1,b,l/k)

{mov eax,3}

{mov ebx,[ebp+56]}

{mov ecx,[ebp+220]}

{add ecx,[ebp+8]}

{mov edx,[ebp+48]}

{int 80h}

{mov [ebp+44],eax}

\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\* Запись из символьного комплекса в файл

\*\*\* n - дескриптор файла

\*\*\* F1 - файл

\*\*\* b - адрес в F1 первого записываемого байта

\*\*\* l - число записывемых байт

\*\*\* k - число записанных байтов (код ошибки для <0)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

fwritef(n,F1,b,l/)

{mov eax,4}

{mov ebx,[ebp+56]}

{mov ecx,[ebp+220]}

{add ecx,[ebp+8]}

{mov edx,[ebp+48]}

{int 80h}

{mov [ebp+44],eax}

\*\*

## Литература

1. [https://ru.wikipedia.org/wiki/Задача\_о\_ранце\_в\_криптографии](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE_%D1%80%D0%B0%D0%BD%D1%86%D0%B5_%D0%B2_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%B8)
2. An INTRODUCTION to CRYPTOGRAPHY Second Edition. RICHARD A. MOLLIN. 2007 by Taylor & Francis Group, LLC
3. Mark Stamp, Richard M. Low(Eds.). Applied Cryptanalysis: Breaking Ciphers in the Real World. Wiley-IEEE Press, May 2007