МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

Специальность Информационные системы и технологии

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9 НА ТЕМУ:

Исследование асимметричных шифров

Выполнила студентка 3 курса 4 группы

Коржова В. С.

Минск 2023

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров.

**Теоретические сведения**

В основу асимметричной криптографии положена идея использовать ключи парами: один – для зашифрования (открытый, или публичный, ключ), другой – для расшифрования (тайный ключ). Отметим, что указанная пара ключей принадлежит получателю зашифрованного сообщения. Все алгоритмы шифрования с открытым ключом основаны на использовании односторонних функций, к числу которых, как известно, относится вычисление дискретного логарифма.

Алгоритмы шифрования с открытым ключом можно использовать для решения следующих задач:

• зашифрования/расшифрования передаваемых и хранимых данных в целях их защиты от несанкционированного доступа;

• формирования цифровой подписи под электронными документами;

• распределения секретных ключей, используемых далее при шифровании документов симметричными методами. В данной работе мы будем работать над аспектами решения первой из указанных задач.

По мнению Диффи и Хеллмана, алгоритм шифрования с открытым ключом должен:

• вычислительно легко создавать пару (открытый ключ *e* – закрытый ключ *d*);

• вычислительно легко зашифровывать сообщение *Mi* открытым ключом;

• вычислительно легко расшифровывать сообщение *Ci*, используя закрытый ключ;

• обеспечивать непреодолимую вычислительную сложность определения соответствующего закрытого ключа при известном открытом ключе;

• обеспечивать непреодолимую вычислительную сложность восстановления исходного (открытого сообщения *Mi*) зная только открытый ключ и зашифрованное сообщение *Ci*.

**Криптоалгоритм на основе задачи об укладке ранца** разработан Р. Мерклом и М. Хеллманом. Это первый алгоритм шифрования с открытым ключом широкого назначения.

Ранцевый (рюкзачный) вектор ***S*** = (*s*1, ..., *sz*) – это упорядоченный набор из *z*, *z* ≥ 3, различных натуральных чисел *si*. Входом задачи о ранце (рюкзаке) называем пару (***S***, *S*), где ***S*** – рюкзачный вектор, а *S* – натуральное число.

Решением для входа (***S***, *S*) будет такое подмножество из ***S***, сумма элементов которого равняется *S*.

В наиболее известном варианте задачи о ранце требуется выяснить, обладает или нет данный вход (***S***, *S*) решением. В варианте, используемом в криптографии, нужно для данного входа (***S***, *S*) построить решение, зная, что такое решение существует. Оба эти варианта являются NP-полными.

Имеются также варианты этой задачи, которые не лежат даже в классе NP.

Как видим, проблема укладки ранца формулируется просто. Дано множество предметов общим числом *z* различного веса. Спрашивается, можно ли положить некоторые из этих предметов в ранец так, чтобы его вес стал равен определенному значению *S*? Более формально задача формулируется так: дан набор значений *k*1, *k*2, …, *kz* и суммарное значение *S*. Требуется вычислить значения *bz* такие, что

*S* = *b*1*k*1 + *b*2*k*2 + ... + *bzkz*. (7.1)

Здесь *bi* может быть либо нулем, либо единицей. Значение *bi* = 1 означает, что предмет *mi* кладут в рюкзак, а *bi*= 0 – не кладут.

Суть метода для шифрования состоит в том, что существуют две различные задачи укладки ранца: одна из них решается легко и характеризуется линейным ростом трудоемкости, а другая решается трудно. Легкий для укладки ранец можно трансформировать в трудный. Трудный для укладки ранец применяется в качестве открытого ключа, который легко использовать для зашифрования, но невозможно – для расшифрования. В качестве закрытого ключа применяется легкий для укладки ранец, который предоставляет простой способ расшифрования сообщения.

В качестве закрытого ключа *d* (легкого для укладки ранца) используется сверхвозрастающая последовательность, состоящая из *z* элементов: *d*1, *d*2, …, *dz*: *d* = {*di*}, *i* = 1, …, *z*.

В основе операции зашифрования сообщения лежит соотношение (7.1). Для зашифрования сообщения (*М*) оно сначала разбивается на блоки, по размерам равные числу (*z*) элементов последовательности в ранце. Затем, считая, что 1 указывает на присутствие элемента последовательности в ранце, а 0 – на его отсутствие, вычисляются полные веса рюкзаков (*Si*, i = 1, …, *z*): по одному ранцу для каждого блока сообщения с использованием открытого ключа получателя ***e.***

**Практическое задание**

1. Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться доступными библиотеками либо программными кодами.

В основе вычислений – кодировочные таблицы Base64 и ASCII. Приложение должно реализовывать следующие операции:

* генерация сверхвозрастающей последовательности (тайного ключа); старший член последовательности – 100-битное число; в простейшем случае принимается z = 6 (для кодировки Base64) и z = 8 (для кодировки ASCII);

Сверхвозрастающей называется последовательность, в которой каждый последующий член больше суммы всех предыдущих.

Мы можем сгенерировать простую сверхвозрастающую последовательность следующим образом: член последовательности образуется путем добавления 1 к сумме ее предыдущих членов.

def generate\_superincreasing\_sequence(z : int) -> list:

    seq = [1] # начинаем с единицы

    while len(seq) < z:

        next\_num = sum(seq) + 1 # следующее число в последовательности

        seq.append(next\_num)

    return seq

Листинг 1 – Генерация простой сверхвозрастающей последовательности

Таким образом мы сгенерируем последовательность {1, 2, 4, 8, 16, 32, 64, 128}.

* вычисление нормальной последовательности (открытого ключа);

Для получения открытого ключа *e* (*e* = {*ei*}, *i* = 1, …, *z*) все значения закрытого ключа умножаются на некоторое число a по модулю n:

*ei*≡ *dia* mod *n.*

Значение модуля *n* должно быть больше суммы всех чисел последовательности; кроме того, НОД (*а*, *n*) = 1.

seq = generate\_superincreasing\_sequence(z)

n = sum(seq) + 1

a = generate\_coprime(n)

public\_key = get\_public\_key(seq, a, n)

def generate\_coprime(phi):

    while True:

        e = random.randrange(2, phi)

        gcd, x, y = extended\_gcd(e, phi)

        if gcd == 1:

            return e

def get\_public\_key(seq : list, a : list, n : list) -> list:

    public\_key = []

    for num in seq:

        public\_key.append((a \* num) % n)

    return public\_key

Листинг 2 – Генерация простой нормальной последовательности

Мы можем взять *n* большее суммы всех членов последовательности на 1, тогда *n* = 256, число *a* обратное ему *a* = 51.

Тогда нормальная последовательность примет вид {51, 102, 204, 152, 48, 96, 192, 128}.

* зашифрование сообщения, состоящего из собственных фамилии, имени и отчества;

Для зашифрования сообщения, состоящего из ФИО, его необходимо перевести в бинарный вид и разбить на блоки, по размерам равные числу (*z*) элементов последовательности в ранце. Затем, считая, что 1 указывает на присутствие элемента последовательности в ранце, а 0 – на его отсутствие, вычисляются полные веса рюкзаков (*Si*, *i* = 1, …, *z*): по одному ранцу для каждого блока сообщения с использованием открытого ключа получателя *e*.

    binary\_str = ''.join(format(ord(x), 'b') for x in message)

    if len(binary\_str) % z != 0:

        binary\_str += '0' \* (z - len(binary\_str) % z)

    binary\_list = [binary\_str[i:i+z] for i in range(0, len(binary\_str), z)]

    print('Сообщение:')

    for bin in binary\_list:

        print(bin)

    print()

    cipher = []

    for bin in binary\_list:

        s = 0

        for i in range(len(bin)):

            if bin[i] == '1':

                s += public\_key[i]

        cipher.append(s)

Листинг 3 – Зашифрование сообщения

Расшифрование остальных блоков шифртекста производится аналогично.

Результат выполнения программы:

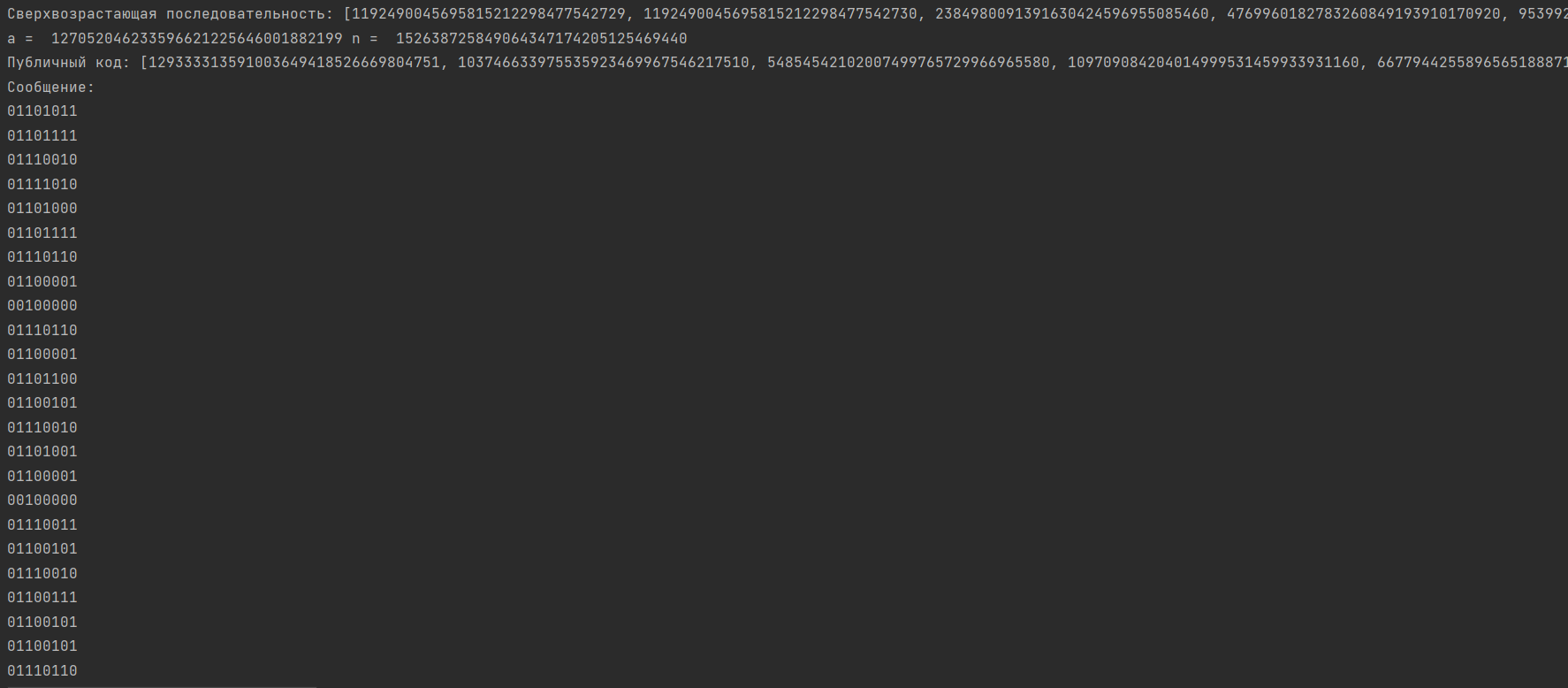


Рисунок 1 – Результат выполнения программы

* оценка времени выполнения операций зашифрования и расшифрования.

Время выполнения операций зависит от длины сообщения, размеров последовательности, длины блока. Операция дешифрования занимает больше времени, чем шифрования.

**Вывод:** в ходе лабораторной работы мы ознакомились с криптоалгоритмом на основе задачи об укладке ранца, научились формировать сверхвозрастающие последовательности, отличая их нормальных последовательностей; разработали приложение, выполняющее зашифрование и расшифрование сообщения и проанализировали время выполнения данных процессов.

**Контрольные вопросы**

1. Что такое «ранцевый (рюкзачный) вектор»? Дать определение.

"Ранцевый вектор" или "рюкзачный вектор" - это вектор, состоящий из целочисленных значений, которые представляют вес каждого из предметов в задаче о рюкзаке.

1. Сформулировать задачу укладки ранца.

Задача укладки ранца (или задача о рюкзаке) заключается в выборе оптимального набора предметов из заданного множества, чтобы заполнить рюкзак до максимально возможного веса, не превышая его грузоподъемность.

1. Если вектор рюкзака имеет вид (14, 28, 56, 82, 90, 132, 197, 284, 341, 455), то какими следует принять коэффициенты bi из выражения (7.1), чтобы получить S = 517? Каким будет решение задачи для S = 516?

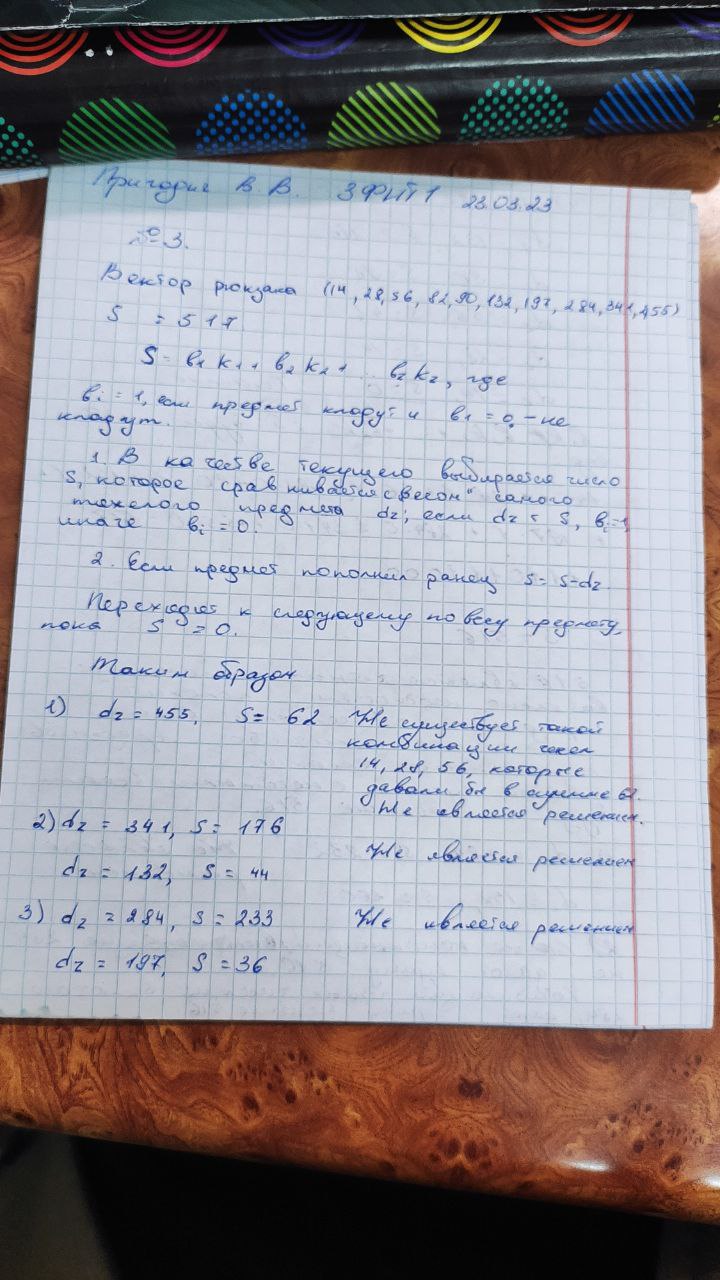


Рисунок 2 – Решение задачи

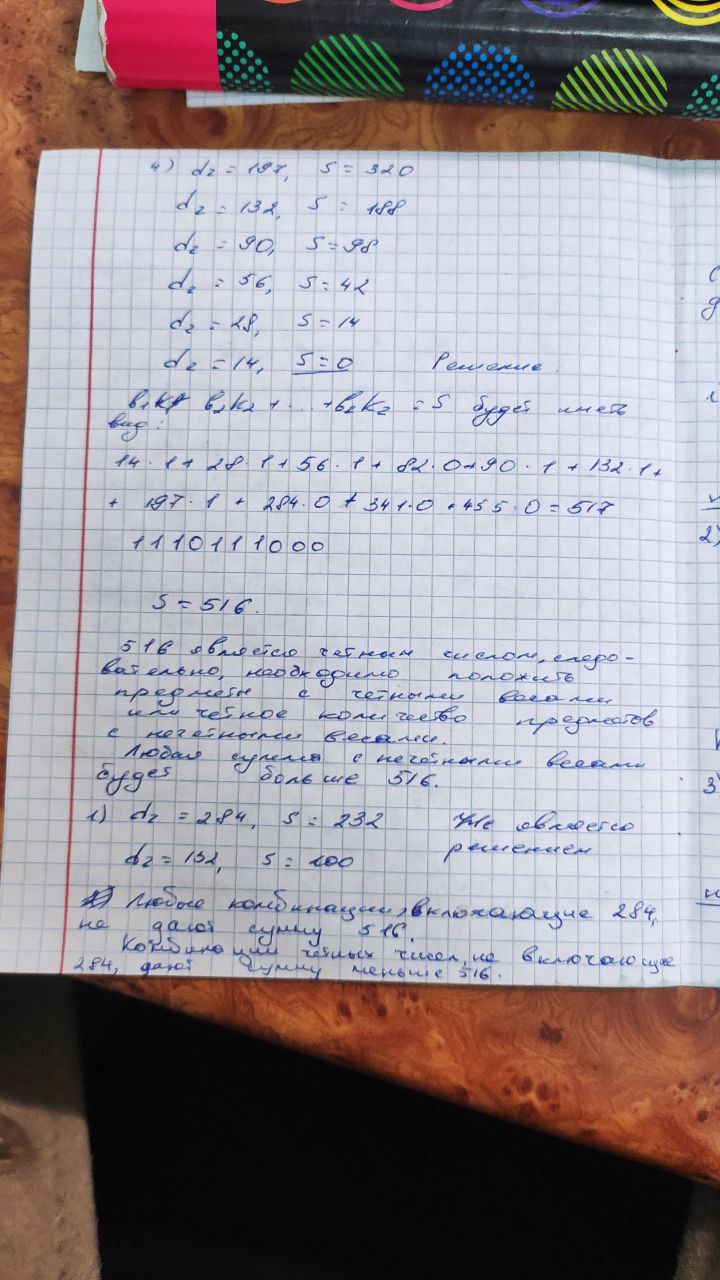


Рисунок 3 – Продолжение решение задачи

1. Что такое сверхвозрастающая последовательность? Привести примеры.

Сверхвозрастающей называется последовательность, в которой каждый последующий член больше суммы всех предыдущих.

Последовательность {2, 3, 6, 13, 27, 52, 105, 210} является сверхвозрастающей,

1. Можно ли последовательности чисел: {89, 3, 11, 2, 45, 6, 22}, {3, 41, 5, 1, 21, 10}, {2, 3, 11, 29, 45, 6, 39} преобразовать в сверхвозрастающие?

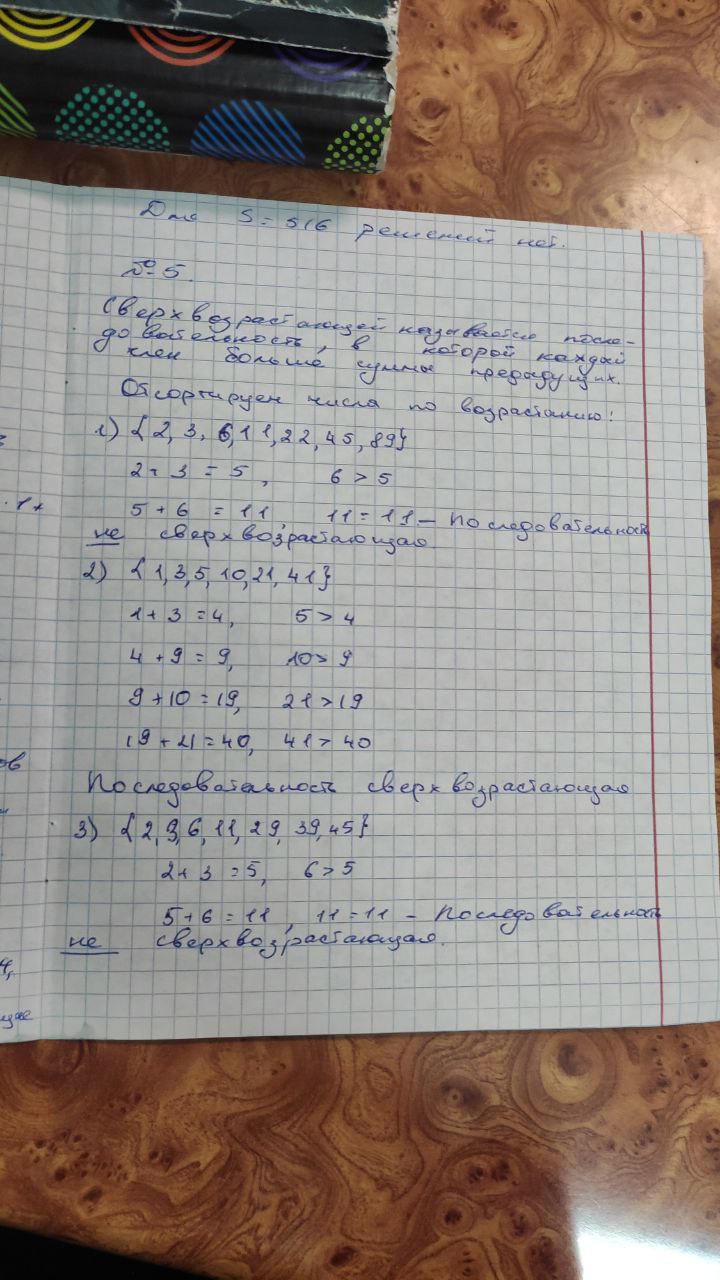


Рисунок 4 – Решение задачи

1. Записать в виде псевдокода алгоритм зашифрования и алгоритм расшифрования сообщения на основе задачи об укладке ранца.

Заширофание:

FOR (блоки m размером z сообщения M)

{

FOR (каждый символ ch в блоке)

{

IF (ch == ‘1’)

{

s += e;

}

}

}

Расшифрование:

FOR (значение шифрограммы c)

{

S = c \* a\_r % n;

}

FOR (значение веса d в последовательности S)

{

IF (d < S)

{

res = ‘1’ + res

S -= d

}

ELSE

{

res = ‘0’ + res

}

}

FOR (блоки m размером z сообщения M)

{

m = S

}

1. Используя некоторый вектор S = (103, 107, 211, 430, 863, 1716, 3449, 6907, 13807, 27610), вычислить ключи для зашифрования и расшифрования сообщений.

Найдем сумму элементов в последовательности:

103 + 107 + 211 + 430 + 863 + 1716 + 3449 + 6907 + 13807 + 27610 = 55203

Предположим, что *n* = 55204, с учетом того, что *n* > 55203, *a* = 3001. НОД (*a*, *n*) = 1.

Найдем члены открытого ключа по формуле:

*ei* = *di \* a* mod *n*

*e1* = 103 \* 3001 mod 55204 = 33083

*e2* = 107 \* 3001 mod 55204 = 45087

*e3* = 211 \* 3001 mod 55204 = 25967

*e4* = 430 \* 3001 mod 55204 = 20738

*e5* = 863 \* 3001 mod 55204 = 50478

*e6* = 1716 \* 3001 mod 55204 = 15744

*e7* = 3449 \* 3001 mod 55204 = 27301

*e8* = 6907 \* 3001 mod 55204 = 26407

*e9* = 13807 \* 3001 mod 55204 = 31807

*e10* = 27610 \* 3001 mod 55204 = 51610

Закрытый ключ: 103, 107, 211, 430, 863, 1716, 3449, 6907, 13807, 27610.

Открытый ключ: 33083, 45087, 25967, 20738, 50478, 15744, 27301, 26407, 31807, 51610.

1. Можно ли, с Вашей точки зрения, одновременно зашифровывать (и, соответственно, одновременно расшифровывать) более, чем по одному символу текста? Обосновать решение. Привести примеры решений.

В алгоритме на основе задачи об укладке рюкзака нельзя одновременно зашифровывать и расшифровывать более, чем по одному символу текста. Это связано с тем, что каждый символ перед шифрованием должен быть преобразован в некоторое числовое значение, которое затем будет умножено на соответствующий элемент вектора. Если мы попытаемся зашифровать несколько символов одновременно, мы не сможем однозначно определить, какому символу соответствует каждое число в векторе.

1. Что такое «секретная лазейка»?

Ранцевые криптосистемы не являются криптостойкими. А. Шамир и Р. Циппел обнаружили, что, зная числа *а*, *a*–1 и *n* («секретную лазейку»), можно восстановить сверхвозрастающую последовательность по нормальной последовательност

1. Охарактеризовать криптостойкость алгоритма на основе задачи об укладке ранца.

Криптостойкость алгоритма на основе задачи об укладке ранца зависит от выбора секретного вектора, который определяет ключ шифрования. Если вектор выбран случайно и его элементы являются большими случайными числами, то алгоритм может быть криптостойким.

Однако, алгоритм на основе задачи об укладке ранца имеет некоторые ограничения, которые могут уменьшать его криптостойкость. Например, если вектор является периодическим или имеет другие структурные особенности, то он может быть уязвим для атак, таких как атака методом грубой силы или атака с помощью алгоритмов решения задачи об укладке ранца.

Также алгоритм на основе задачи об укладке ранца не обеспечивает аутентификацию и целостность данных, что может приводить к возможности атак с подменой данных или внедрению вредоносного кода.

Таким образом, криптостойкость алгоритма на основе задачи об укладке ранца зависит от выбора секретного вектора и от соблюдения определенных условий.